

# 31st International Conference on DNA Computing and Molecular Programming

DNA 31, August 25–29, 2025, Lyon, France

Edited by

Josie Schaeffer

Fei Zhang



*Editors*

**Josie Schaeffer** 

Google  
thiryal+dna31@gmail.com

**Fei Zhang** 

Rutgers University, Newark, NJ, USA  
fz124@newark.rutgers.edu

*ACM Classification 2012*

Theory of computation → Models of computation; Applied computing → Biological networks; Applied computing → Molecular structural biology; Information systems → Information storage systems

**ISBN 978-3-95977-399-7**

*Published online and open access by*

Schloss Dagstuhl – Leibniz-Zentrum für Informatik GmbH, Dagstuhl Publishing, Saarbrücken/Wadern, Germany. Online available at <https://www.dagstuhl.de/dagpub/978-3-95977-399-7>.

*Publication date*

August, 2025

*Bibliographic information published by the Deutsche Nationalbibliothek*

The Deutsche Nationalbibliothek lists this publication in the Deutsche Nationalbibliografie; detailed bibliographic data are available in the Internet at <https://portal.dnb.de>.

*License*

This work is licensed under a Creative Commons Attribution 4.0 International license (CC-BY 4.0):  
<https://creativecommons.org/licenses/by/4.0/legalcode>.



In brief, this license authorizes each and everybody to share (to copy, distribute and transmit) the work under the following conditions, without impairing or restricting the authors' moral rights:

- Attribution: The work must be attributed to its authors.

The copyright is retained by the corresponding authors.

Digital Object Identifier: 10.4230/LIPIcs.DNA.31.0

**ISBN 978-3-95977-399-7**

**ISSN 1868-8969**

**<https://www.dagstuhl.de/lipics>**

## LIPIcs – Leibniz International Proceedings in Informatics

LIPIcs is a series of high-quality conference proceedings across all fields in informatics. LIPIcs volumes are published according to the principle of Open Access, i.e., they are available online and free of charge.

### *Editorial Board*

- Christel Baier (TU Dresden, DE)
- Roberto Di Cosmo (Inria and Université Paris Cité, FR)
- Faith Ellen (University of Toronto, CA)
- Javier Esparza (TU München, DE)
- Holger Hermanns (Universität des Saarlandes, Saarbrücken, DE and Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Wadern, DE)
- Daniel Král' (Leipzig University, DE and Max Planck Institute for Mathematics in the Sciences, Leipzig, DE)
- Sławomir Lasota (University of Warsaw, PL)
- Meena Mahajan (Institute of Mathematical Sciences, Chennai, IN – Chair)
- Chih-Hao Luke Ong (Nanyang Technological University, SG)
- Eva Rotenberg (Technical University of Denmark, Lyngby, DK)
- Pierre Senellart (ENS, Université PSL, Paris, France)
- Alexandra Silva (Cornell University, Ithaca, US)

**ISSN 1868-8969**

**<https://www.dagstuhl.de/lipics>**





In memory of Anthony Genot



## ■ Contents

Preface	
<i>Josie Schaeffer and Fei Zhang</i> .....	0:ix
Organization	
.....	0:xi
List of Authors	
.....	0:xvii

## Papers

Algorithmic Hardness of the Partition Function for Nucleic Acid Strands	
<i>Gwendal Ducloz, Ahmed Shalaby, and Damien Woods</i> .....	1:1–1:23
A Coupled Reconfiguration Mechanism That Enables Powerful, Pseudoknot-Robust DNA Strand Displacement Devices with 2-Stranded Inputs	
<i>Hope Amber Johnson and Anne Condon</i> .....	2:1–2:23
Reachability in Deletion-Only Chemical Reaction Networks	
<i>Bin Fu, Timothy Gomez, Ryan Knobel, Austin Luchsinger, Aiden Massie, Marco Rodriguez, Adrian Salinas, Robert Schweller, and Tim Wylie</i> .....	3:1–3:21
Differentiable Programming of Indexed Chemical Reaction Networks and Reaction-Diffusion Systems	
<i>Inhoo Lee, Salvador Buse, and Erik Winfree</i> .....	4:1–4:23
Programmable Co-Transcriptional Splicing: Realizing Regular Languages via Hairpin Deletion	
<i>Da-Jung Cho, Szilárd Zsolt Fazekas, Shinnosuke Seki, and Max Wiedenhöft</i> .....	5:1–5:22
Secondary Structure Design for Cotranscriptional 3D RNA Origami Wireframes	
<i>Pekka Orponen, Shinnosuke Seki, and Antti Elonen</i> .....	6:1–6:18
Tile Blockers as a Simple Motif to Control Self-Assembly: Kinetics and Thermodynamics	
<i>Constantine G. Evans, Angel Cervera Roldan, Trent Rogers, and Damien Woods</i> .....	7:1–7:19
An Axiomatic Study of Leveraging Blockers to Self-Assemble Arbitrary Shapes via Temperature Programming	
<i>Matthew J. Patitz and Trent A. Rogers</i> .....	8:1–8:20
Synchronous Versus Asynchronous Tile-Based Self-Assembly	
<i>Florent Becker, Phillip Drake, Matthew J. Patitz, and Trent A. Rogers</i> .....	9:1–9:21
Computing and Bounding Equilibrium Concentrations in Athermic Chemical Systems	
<i>Hamidreza Akef, Minki Hhan, and David Soloveichik</i> .....	10:1–10:19
Leakless Polymerase-Dependent Strand Displacement Systems	
<i>Zoë Evelyn Mōhalakealoha Derauf and Chris Thachuk</i> .....	11:1–11:19





## ■ Preface

This LIPIcs volume contains the papers that have been accepted to Track A of the 31st International Conference on DNA Computing and Molecular Programming (DNA31), organized by Nicolas Schabanel and his team at École Normale Supérieure de Lyon, France during August 25-29, 2025, under the auspices of the International Society for Nanoscale Science, Computation, and Engineering (ISNSCE).

The DNA conference series aims at providing researchers from the fields of mathematics, computer science, physics, chemistry, biology, and nanotechnology, among others, with a forum to collaboratively address the analysis, design, and synthesis of information-based molecular systems. Papers and presentations were sought in all areas that relate to biomolecular computing including, but not restricted to, algorithms and models of computation for biomolecular systems, computational processes *in vitro* and *in vivo*, molecular switches, gates, devices, and circuits, molecular folding and self-assembly of nanostructures, analysis and theoretical models of laboratory techniques, molecular motors and molecular robotics, information storage, studies of fault-tolerance and error correction, software tools for analysis, simulation, and design, synthetic biology and *in vitro* evolution, and applications in engineering, physics, chemistry, biology, and medicine.

Authors who wished to orally present their work were asked to select one of the two submission tracks: Track A (full paper) or Track B (one-page abstract accompanied with supplementary document). Track B is primarily for those who plan to submit experimental or theoretical results to a journal rather than publish in the conference proceedings. There were 45 submissions for oral presentations (20 to Track A and 25 to Track B), from China, Czechia, Finland, France, Germany, India, Ireland, Israel, Japan, Poland, South Korea, Taiwan, the UK, and the USA. Each of the 45 qualified submissions was reviewed by at least three reviewers, with most reviewed by four or more, and thoroughly discussed by the Program Committee (PC). The committee decided to accept 11 papers for Track A (55%) and 11 submissions for Track B (44%). We also received 63 submissions for Track C (poster), of which 6 were selected as additional short oral presentations.

We warmly thank Barbara Saccà, Kerstin Göpfrich, Masahiro Takinoue, and Olivier Bournez for their intellectually stimulating invited talks and all the authors of the submissions for making DNA 31 successful. Anthony Genot was to be an invited speaker at the conference, and we were deeply saddened to hear of his unexpected passing in April. It is a terrible loss to his many colleagues and to the field as a whole. We are greatly appreciative of the organizing committee having a memorial keynote in honour of Anthony.

As the PC co-chairs, we would like to express our gratitude towards the PC members and the external reviewers for their hard work in reviewing the papers within a tight schedule and for providing insightful and constructive comments in order to keep high standard of the DNA conferences. We would like to thank the editorial staff of Dagstuhl Publishing Team, and in particular Michael Wagner for his guidance and help during the process of publishing this volume. We also greatly appreciate the guidance from Anne Condon, Shinnosuke Seki and Damien Woods during the process of organizing the program. Last but not the least, we are grateful to the Organizing Committee members: Nicolas Schabanel, Chiraz Benamor, Marie Bozo, Gwendal Ducloz, and Laure Savetier.

We are all looking forward to DNA32 at University of Arkansas in the USA.

Josie Schaeffer and Fei Zhang  
August 2025



## Organization

### Steering Committee

Anne Condon (Chair)	University of British Columbia, Canada
Natasha Jonoska	University of South Florida, USA
Matthew Lakin	University of New Mexico, USA
Thomas Ouldridge	Imperial College London, UK
Satoshi Murata	Tohoku University, Japan
John H. Reif	Duke University, USA
Grzegorz Rozenberg	University of Leiden, The Netherlands
Rebecca Schulman	Johns Hopkins University, USA
Friedrich Simmel	Technical University Munich, Germany
David Soloveichik	University of Texas at Austin, USA
Shelley Wickham	The University of Sydney, Australia
Erik Winfree	California Institute of Technology, USA
Damien Woods	Maynooth University, Ireland
Hao Yan	Arizona State University, USA



## Program Committee

Ebbe S. Andersen	Aarhus University
Wooli Bae	University of Surrey
Florent Becker	Université d'Orléans
Gaëtan Bellot	CNRS, Centre de Biochimie Structurale, Montpellier, France
Luca Cardelli	Microsoft Research
Ho-Lin Chen	National Taiwan University
Yuan-Jyue Chen	Microsoft Research, Redmond
Anne Condon	University of British Columbia
David Doty	University of California, Davis
Hannah Earley	University of Cambridge
Abeer Eshra	Maynooth University
Constantine Evans	Evans Foundation and Maynooth University
Elisa Franco	University of California, Los Angeles
Giuditta Franco	Universita di Verona
Jinglin Fu	Rutgers University - Camden
Cody Geary	Heidelberg ZMBH
Leopold Green	Purdue University
Ibuki Kawamata	Kyoto University
Yonggang Ke	Emory University
James Lathrop	Iowa State University
Chenxiang Lin	Yale University
Dongsheng Liu	The Hong Kong Polytechnic University
Tracy Mallette	University of Washington
Chengde Mao	Purdue University
Satoshi Murata	Tohoku University
Pekka Orponen	Aalto University
Tom Ouldrige	Imperial College London
Yann Ponty	LIX, École Polytechnique
Felix Rizzuto	University of New South Wales
Trent Rogers	University of Arkansas
Lorenzo Rovigatti	Sapienza University of Rome
Josie Schaeffer	Google (co-chair)
Shinnosuke Seki	University of Electro-Communications
David Soloveichik	University of Texas, Austin
Rebecca Taylor	Carnegie Mellon University
Chris Thachuk	University of Washington
Guillaume Theyssier	Aix Marseille Université
Emanuela Torelli	Newcastle University
Shelley Wickham	University of Sydney
Erik Winfree	California Institute of Technology
Sungwook Woo	Pohang University of Science and Technology
Damien Woods	Maynooth University
Fei Zhang	Rutgers University (co-chair)



**Additional Reviewers for Tracks A and B**

Salvador Buse	California Institute of Technology
Tiernan Kennedy	University of Washington
Inhoo Lee	California Institute of Technology
Andrew Miner	Iowa State University
Dawn Nye	
Chandler Petersen	University of Washington
Hugh Potter	Iowa State University
Romeo Rizzi	Università di Verona
Fei Wang	Shanghai Jiao Tong University
Yongzheng Xing	Shandong University

## **Organizing Committee for DNA 31**

Nicolas Schabanel	CNRS, École Normale Supérieure de Lyon (OC Chair)
Chiraz Benamor	École Normale Supérieure de Lyon
Marie Bozo	École Normale Supérieure de Lyon
Gwendal Ducloz	École Normale Supérieure de Lyon
Laure Savetier	École Normale Supérieure de Lyon

## **Sponsors**

International Society for Nanoscale Science (ISNSCE)

National Science Foundation (NSF)

École Normale Supérieure de Lyon


Institut Rhônalpin des Systèmes Complexes


Laboratoire de l'Informatique du Parallélisme


CNRS Sciences informatiques (INS2I)





## ■ List of Authors


Hamidreza Akef  (10)  
The University of Texas at Austin, TX, USA


Florent Becker  (9)  
Laboratoire d'Informatique Fondamentale  
d'Orléans (UR4022), Université d'Orléans,  
France


Salvador Buse  (4)  
California Institute of Technology, Pasadena,  
CA, USA


Angel Cervera Roldan  (7)  
Hamilton Institute and Department of Computer  
Science, Maynooth University, Ireland


Da-Jung Cho  (5)  
Department of Software and Computer  
Engineering, Ajou University, Suwon,  
Republic of Korea

Anne Condon  (2)  
The University of British Columbia,  
Vancouver, Canada


Zoë Evelyn Mōhalakealoha Derauf  (11)  
Paul G. Allen School of Computer Science &  
Engineering, University of Washington, Seattle,  
WA, USA

Phillip Drake  (9)  
University of Arkansas, Fayetteville, AR, USA

Gwendal Ducloz  (1)  
Hamilton Institute and Department of Computer  
Science, Maynooth University, Ireland;  
École Normale Supérieure de Lyon, France


Antti Elonen  (6)  
Department of Computer Science,  
Aalto University, Finland

Constantine G. Evans  (7)  
Hamilton Institute and Department of Computer  
Science, Maynooth University, Ireland

Szilárd Zsolt Fazekas  (5)  
Graduate School of Engineering Science,  
Akita University, Japan


Bin Fu (3)  
University of Texas Rio Grande Valley,  
Edinburg, TX, USA

Timothy Gomez (3)  
Massachusetts Institute of Technology,  
Cambridge, MA, USA

Minki Hhan  (10)  
The University of Texas at Austin, TX, USA

Hope Amber Johnson  (2)  
The University of British Columbia,  
Vancouver, Canada


Ryan Knobel (3)  
University of Texas Rio Grande Valley,  
Edinburg, TX, USA

Inhoo Lee  (4)  
California Institute of Technology,  
Pasadena, CA, USA


Austin Luchsinger (3)  
University of Texas Rio Grande Valley,  
Edinburg, TX, USA


Aiden Massie (3)  
University of Texas Rio Grande Valley,  
Edinburg, TX, USA

Pekka Orponen  (6)  
Department of Computer Science,  
Aalto University, Finland

Matthew J. Patitz  (8, 9)  
University of Arkansas, Fayetteville, AR, USA


Marco Rodriguez (3)  
Massachusetts Institute of Technology,  
Cambridge, MA, USA

Trent Rogers  (7)  
Hamilton Institute and Department of Computer  
Science, Maynooth University, Ireland;  
Computer Science & Computer Engineering,  
University of Arkansas, Fayetteville, AR, USA

Trent A. Rogers  (8, 9)  
University of Arkansas, Fayetteville, AR, USA

Adrian Salinas (3)  
University of Texas Rio Grande Valley,  
Edinburg, TX, USA

Robert Schweller (3)  
University of Texas Rio Grande Valley,  
Edinburg, TX, USA


Shinnosuke Seki  (5, 6)  
University of Electro-Communications,  
Tokyo, Japan


Ahmed Shalaby  (1)  
Hamilton Institute and Department of Computer  
Science, Maynooth University, Ireland

31st International Conference on DNA Computing and Molecular Programming (DNA 31).  
Editors: Josie Schaeffer and Fei Zhang




Leibniz International Proceedings in Informatics  
Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

David Soloveichik  (10)  
The University of Texas at Austin, TX, USA

Chris Thachuk  (11)  
Paul G. Allen School of Computer Science &  
Engineering, University of Washington, Seattle,  
WA, USA

Max Wiedenhöft  (5)  
Department of Computer Science,  
Kiel University, Germany

Erik Winfree  (4)  
California Institute of Technology,  
Pasadena, CA, USA

Damien Woods  (1, 7)  
Hamilton Institute and Department of Computer  
Science, Maynooth University, Ireland

Tim Wylie (3)  
University of Texas Rio Grande Valley,  
Edinburg, TX, USA

# Algorithmic Hardness of the Partition Function for Nucleic Acid Strands

Gwendal Ducloz ✉ 

Hamilton Institute and Department of Computer Science, Maynooth University, Ireland  
École Normale Supérieure de Lyon, France

Ahmed Shalaby ✉ 

Hamilton Institute and Department of Computer Science, Maynooth University, Ireland

Damien Woods ✉ 

Hamilton Institute and Department of Computer Science, Maynooth University, Ireland

---

## Abstract

To understand and engineer biological and artificial nucleic acid systems, algorithms are employed for prediction of secondary structures at thermodynamic equilibrium. Dynamic programming algorithms are used to compute the most favoured, or Minimum Free Energy (MFE), structure, and the Partition Function (PF) – a tool for assigning a probability to any structure. However, in some situations, such as when there are large numbers of strands, or pseudoknotted systems, NP-hardness results show that such algorithms are unlikely, but only for MFE. Curiously, algorithmic hardness results were not shown for PF, leaving two open questions on the complexity of PF for multiple strands and single strands with pseudoknots. The challenge is that while the MFE problem cares only about one, or a few structures, PF is a summation over the entire secondary structure space, giving theorists the vibe that computing PF should not only be as hard as MFE, but should be even harder.

We answer both questions. First, we show that computing PF is  $\#P$ -hard for systems with an unbounded number of strands, answering a question of Condon Hajiaghayi, and Thachuk [DNA27]. Second, for even a single strand, but allowing pseudoknots, we find that PF is  $\#P$ -hard. Our proof relies on a novel *magnification trick* that leads to a tightly-woven set of reductions between five key thermodynamic problems: MFE, PF, their decision versions, and  $\#SSEL$  that counts structures of a given energy. Our reductions show these five problems are fundamentally related for any energy model amenable to magnification. That general classification clarifies the mathematical landscape of nucleic acid energy models and yields several open questions.

**2012 ACM Subject Classification** Theory of computation → Problems, reductions and completeness

**Keywords and phrases** Partition function, minimum free energy, nucleic acid, DNA, RNA, secondary structure, computational complexity,  $\#P$ -hardness

**Digital Object Identifier** 10.4230/LIPIcs.DNA.31.1

**Related Version** *Full Version:* <https://arxiv.org/abs/2506.19756> [11]

**Funding** Work carried out while GD was on internship at Maynooth University. Supported by Science Foundation Ireland (SFI) under grant number 20/FFP-P/8843, European Research Council (ERC) under the European Union’s Horizon 2020 research and innovation programme (grant agreement No 772766, Active-DNA project), and European Innovation Council (EIC), No 101115422, DISCO project. Views and opinions expressed are however those of the author(s) only and do not necessarily reflect those of the European Union, ERC, EIC or SFI. Neither the European Union nor the granting authority can be held responsible for them.

**Acknowledgements** We thank Doan Dai Nguyen, Constantine Evans, Dave Doty, Sergiu Ivanov and Cai Wood for stimulating discussions.



© Gwendal Ducloz, Ahmed Shalaby, and Damien Woods;  
licensed under Creative Commons License CC-BY 4.0

31st International Conference on DNA Computing and Molecular Programming (DNA 31).

Editors: Josie Schaeffer and Fei Zhang; Article No. 1; pp. 1:1–1:23

Leibniz International Proceedings in Informatics



LIPICs Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

## 1 Introduction

The information encoding abilities of RNA is harnessed by biology to encode myriad complex behaviours, and both DNA and RNA have been used by scientists and engineers to create custom nanostructures and molecular computers. Consequently, predicting the structures formed by DNA and RNA strands is crucial both for understanding molecular biology and advancing molecular programming.

The *primary structure* of a DNA strand is simply a word over the alphabet  $\{A, C, G, T\}$ , with U instead of T for RNA. Bases bond in pairs, A-T and C-G, and a set of such pairings for one or more strands is called a *secondary structure*. Typically, the set of all secondary structures  $\Omega$  has size exponential in the total number of bases. Assuming an energy model over secondary structures, each secondary structure  $S$  has an associated real-valued free energy  $\Delta G(S)$ , where more negative means more favourable [27].

The Boltzman distribution is used to model the probability distribution of secondary structures at equilibrium [21, 10, 8]: the probability of  $S$  is given by  $p(S) = \frac{1}{Z} e^{-\Delta G(S)/k_B T}$ , where  $Z$  is a normalisation factor called the partition function (PF):

$$Z = \sum_{S \in \Omega} e^{-\Delta G(S)/k_B T} \quad (1)$$

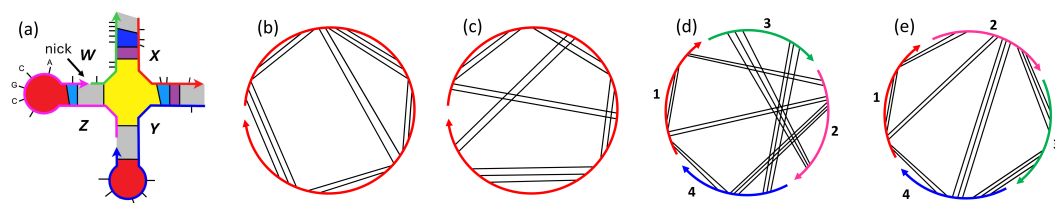
Intuitively,  $Z$  is an exponentially weighted sum of the free energies over  $\Omega$ , where  $k_B$  is Boltzmann's constant and  $T$  is temperature in Kelvin. The algorithmic complexity of computing the PF is the main object of study in this paper.

Predicting the free energy of the most favoured secondary structure(s) at thermodynamic equilibrium is called the Minimum Free Energy (MFE) problem [32, 23, 33]. A third problem of interest, called #SSEL (or number of Secondary Structures at a specified Energy Level), asks a more fine-grained question [7]: Given a value  $k \in \mathbb{R}$  how many secondary structures have free energy  $k$ ? Here, we give a mathematical relation between all of these models (Figure 2) that is somewhat energy-model agnostic and settle the computational complexity of PF in two settings (Table 1).

**Table 1** Results on the computational complexity of MFE and PF. P: problem is solvable in time polynomial in  $n$ , the total number of DNA/RNA bases; NP-hard: as hard as any problem in nondeterministic polynomial time (NP); #P-hard: as hard as counting the accepting paths of an NP Turing machine. All positive results showing a problem in P hold for all 3 models studied here: base pair matching (BPM), base pair stacking (BPS), and nearest neighbour (NN). Negative, or hardness, results are proven in the simple BPM or BPS models and are conjectured [5] to hold in the more complex NN model. Specifically: Condon et al. [5] use the BPM model, and Lyngsø [19] uses the BPS model to prove the NP-completeness of the decision version of MFE. The two #P-hardness<sup>3</sup> results are contributions of this paper, along with the reductions of Figure 2.

Structure	Num. Strands	MFE	PF
Unpseudoknotted	1	P [32, 23, 33]	P [21]
	Bounded ( $\mathcal{O}(1)$ )	P [26]	P [8]
	Unbounded ( $\mathcal{O}(n)$ )	NP-hard [5]; BPM model	#P-hard [Theorem 38]; BPM model
Pseudoknotted	1	NP-hard [1, 20, 19]; including BPS model	#P-hard [Theorem 33]; BPS model





**Figure 1** The nearest neighbour, or Turner, model of DNA/RNA multistranded secondary structure. (a) One of the many possible secondary structures for four DNA (or RNA) strands  $W, X, Y, Z$ . Short black lines represent DNA bases (a few are shown ... C, G, C, A ...), and long lines represent base pairs (not to scale). Loops are colour-coded: stack (purple), hairpin (red), bulge (light blue), internal (dark blue), multiloop (yellow), external (grey). Black arrow: *nick*, the gap between two strands. (b–c) Polymer graphs<sup>1</sup> for two single-stranded secondary structures: (b) unknotted (no crossings), (c) pseudoknotted (crossings). (d) Polymer graph for a secondary structure  $S$  over the strand set  $\{1, 2, 3, 4\}$  with strand ordering 1324 showing crossings. (e) Simply by reordering the strands to 1234 gives a polymer graph without crossings, proving  $S$  is unknotted.

## 1.1 Background and related work

**Efficient algorithms.** Decades ago, the relationship between secondary structure prediction and dynamic programming algorithms was well established. For a single strand of length  $n$ , dynamic programming techniques were used to solve the MFE and PF problems efficiently, meaning in polynomial time in the number of bases, but ignoring so-called pseudoknotted structures. Early algorithms were designed for simple energy models that count the number of base pairs, the base pair matching (BPM) model [32, 23]. Later algorithms accounted for more complex secondary structure features, including stacks, hairpin loops, internal loops and other features that comprise the *nearest neighbour*, or *Turner*, model [21, 8], Figure 1, extensively used by practitioners [30, 17, 31].

**Hardness results for single strands and an open problem.** Somewhat frustratingly, dynamic programming algorithms for MFE prediction do not handle all secondary structures: as noted, pseudoknots (Definition 2) throw a spanner in the works. Pseudoknotted structures are ubiquitous in both biological RNA and DNA nanotech and computing systems, so why ignore them? In 2000, prediction in the presence of pseudoknots was shown to be NP-hard, even for a single strand and under simple energy models like the base pair stacking model (BPS; counts stacks) [1, 20, 19]. NP-hardness implies we are unlikely to see efficient algorithms for the full class of pseudoknots [13], although progress has been made on specific subclasses [10, 24, 1, 9, 4, 16]. Although it feels as though computing PF should be at least as hard as MFE, for reasons outlined below the complexity of PF in this setting (single strand, allowing pseudoknots) remained tantalisingly open.

**Hardness results for multiple strands and another open problem.** Recently, Condon, Hajiaghayi, and Thachuk [5] proved that computing MFE is also NP-hard in the unknotted BPM model when the number of strands is unbounded, meaning it scales with problem size. They left the complexity of PF in this setting as an open problem.

<sup>1</sup> A secondary structure can be represented as a polymer graph by ordering and depicting the directional (5' to 3') strands around the circumference of a circle, with edges along the circumference representing adjacent bases, and straight line edges connecting paired bases. Each such ordering of  $c$  strands is a circular permutation of the strands, and there are  $(c - 1)!$  possible orderings [5, 26, 8].

**Results on #SSEL.** The counting problem #SSEL is also known in the literature as the density of states problem [18]. Efficient dynamic programming algorithms can be adapted to solve a version of this problem where energies fall into discrete bins (ranges), for unpseudoknotted secondary structures [6]. In addition, probabilistic heuristic methods have been proposed to estimate its value with good accuracy [18]. These heuristics originate from statistical physics, particularly from the study of the Ising model. Interestingly, this model draws a useful parallel, as computing its partition function is known to be either in P or #P-complete, depending on the parameters [14].

**Indirect evidence that PF might be harder than MFE.** In 2007, Dirks et al. [8] gave a polynomial time dynamic programming algorithm for PF for multiple, but  $\mathcal{O}(1)$ , strands in the NN model [8]. Their paper includes a definitional contribution that extends the single-strand NN model to multiple strands by including both a strand association penalty and an entropic penalty for rotational symmetry of multi-stranded secondary structures. They observe if the model is permitted to ignore rotational symmetry, any purely dynamic programming algorithm for PF can be translated into an algorithm for MFE using tropical  $(\min, +)$  algebra instead of the classic  $(+, \cdot)$  algebra. But going the other way is not so obvious: translating MFE algorithms into PF algorithms is challenging due to the risk of overcounting secondary structures, by translating to an overly naive algorithm. This observation provides some intuition that PF might be harder than MFE.

Recently Shalaby and Woods [26] gave an efficient algorithm for computing MFE in the same setting as Dirks et al. [8] ( $\mathcal{O}(1)$  strands, unpseudoknotted). For roughly two decades, this setting had supported an efficient algorithm for PF but none for MFE due to rotational symmetry complications. The rareness of the situation and its positive resolution bolstered our intuition PF is not easier than MFE.

**A note about energy models.** Efficient algorithms seem to work across a range of energy models (BPM, BPS, NN),<sup>2</sup> but NP-hardness results have merely been shown for MFE in simple energy models: BPM and BPS. As Condon, Hajiaghayi, and Thachuk observe [5], it seems unlikely MFE would become easier in more sophisticated models like NN.

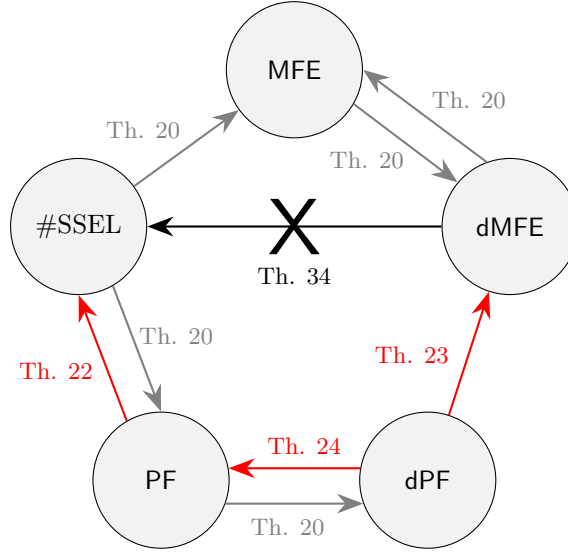
## 1.2 Contributions

We prove hardness results on the complexity of PF in two settings (Table 1), and provide tools for thinking about the complexity of thermodynamic prediction problems (Figure 2).

Section 2 contains definitions of secondary structures, energy models (BPM, BPS and NN), and thermodynamic problems (MFE, PF, #SSEL, and decision problems dMFE and dPF.)

Our first main contribution is to provide a map of reductions between all of these problems, illustrated in Figure 2 with proofs in Section 3. Some of these reductions are rather straight-forward (grey arrows in Figure 2), but the others (red arrows) make use of a new proof strategy we call the energy *magnification trick*. In Section 3.5 we define a property of an energy model called *PF-polynomially magnifiable* (Definition 25) which means that the energy model has a polynomial time magnification adaptable PF algorithm. That general

<sup>2</sup> The BPS model is a special case of the NN model, so any algorithm in the NN model can be modified for the BPS model by easily ignoring all loops except stacks and setting  $\Delta G^{\text{stack}} = -1$ .



■ **Figure 2** Reduction map between the five main problems (Section 2.3) studied in this paper, results shown in Section 3. An arrow from problem A to B means that if there exists an algorithm for A, it can be called to efficiently solve B, or in other words that there is a polynomial-time Turing reduction from B to A. Red arrows use a magnification of the energy model, grey arrows do not. A crossed arrow signifies that no such reduction exists unless  $\#P \subseteq P^{NP}$ , which means the collapse of polynomial hierarchy [2] at level 2. The latter implies the non-existence of arrows from dMFE to dPF and from MFE to #SSEL.

property yields Corollary 26, that all 5 problems in Figure 2 are in P (or FP) whenever the underlying energy model is PF-polynomially magnifiable. Section 3.5 has a full discussion about PF-polynomially magnifiable energy models and magnification adaptable algorithms.

Our second main contribution is to answer two open problems on the complexity of PF: we show that PF is  $\#P$ -hard<sup>3</sup> in (a) the single-stranded case, i.e. with pseudoknots under the BPM model, and (b) in the multi-stranded case even without pseudoknots (Table 1) under the BPS model. These results are proven in Sections 4 and 5, and leverage the reduction map (Figure 2, Section 3). Since dMFE is NP-complete [5, 19], these  $\#P$ -hardness results provide a strong result, showing that PF is strictly harder than dMFE in these two situations, unless  $\#P \subseteq P^{NP}$ , which implies the collapse of the polynomial hierarchy [2] at level 2.

In addition to showing the  $\#P$ -hardness lowerbound on PF, the reductions in Section 3 show that PF and #SSEL have equivalent complexity in the sense that they are polynomial-time Turing reducible [2] to each other, providing an upperbound on the complexity of PF.

Although our  $\#P$ -hardness results are shown for the BPM and BPS models, our reductions apply to the NN model, and Appendix A has some analysis of the NN model.

<sup>3</sup> A first upperbound of all five problems in Figure 2 is the exponential time class EXP, as we can solve any problem just by going through all distinct secondary structures. Therefore, the interesting question is about finding better complexity lowerbounds and upperbounds. The complexity class  $\#P$ , introduced by Leslie Valiant [29], is the class of problems where the goal is to count the number of solutions, with each solution having a polynomial-sized certificate.

### 1.3 Future work

1. MFE hardness in two settings is still an open question: Is MFE NP-hard for NN single-strand pseudoknotted, or NN  $\mathcal{O}(n)$ -strands unpseudoknotted? If so, can the reduction map be leveraged to show hardness results for PF and #SSEL in the NN model?
2. Using the reduction map (Figure 2) for positive results: Is the NN model PF-polynomially magnifiable (see Definition 25)? (The existing algorithm by Dirks et al. [8] does not handle rotational symmetry in a magnification adaptable way.) A positive answer, and the reduction map would immediately imply a polynomial time algorithm for #SSEL. (There already is a polynomial time algorithm for MFE [26], and hence dMFE.)
3. Figure 2 has arrows of two colours: red that means the reduction uses our magnification trick, and grey does not. Can all red arrows be made grey? This involves replacing our magnification trick with a completely different strategy.
4. In Note 13, we assume in the NN model that loops are specified using at most logarithmic precision. Logarithmic precision seems reasonable from a physical perspective, since parameters to the NN free energy model have uncertainty after only a few decimal places. Mathematically, some loop free energies, namely, hairpin, interior and bulge loops, are a logarithmic function of their size and thus may be irrational. Hence we ask: Can this logarithmic precision assumption be dropped?
5. We studied the MFE problem, which aims to compute the minimum free energy value. A natural variant is to ask for its corresponding secondary structures. How does this version relate to our five other problems?

## 2 Definitions

First, we need to review some basic terminology to formulate our main problems in an easy and precise way. We set the scene, in Section 2.1, by the mathematical definitions for single-stranded nucleic acid systems before extending them to the multi-stranded case. In Section 2.2, we provide a brief review of existing energy models and some abstract properties of these models that play a significant role in our reductions. Finally, in Section 2.3, we provide the formal definitions of the main problems of interest in this work.

### 2.1 Single-stranded and multi-stranded nucleic acid systems

Formally, a DNA strand  $s$  is a word over the alphabet of DNA *bases*  $\{A, T, G, C\}$ , indexed from 1 to  $|s| = n$ , where  $n$  denotes the number of bases of  $s$ . Hydrogen bonds, or base pairs, can form between complementary bases, namely C–G and A–T, and formally such a base pair is written as a tuple  $(i, j)$ , where  $i < j$ , of strand indices (indexing from 1).

► **Definition 1** (Single-stranded secondary structure  $S$ ). For any DNA strand  $s$ , a secondary structure  $S$  of  $s$  is a set of base pairs, such that each base appears in at most one pair, i.e. if  $(i, j) \in S$  and  $(k, l) \in S$  then  $i, j, k, l$  are all distinct or  $(i, j) = (k, l)$ .

► **Definition 2** (Unpseudoknotted single-stranded secondary structure). A secondary structure  $S$  of a strand  $s$  is unpseudoknotted if for any two base pairs  $(i, j)$  and  $(k, l) \in S$ ,  $i < k < j$  and only if  $i < l < j$ . Otherwise, we call  $S$  pseudoknotted, see Figure 1.

► **Remark 3.** The maximum number of base pairs in any secondary structure  $S$  of strand  $s$  is  $\lfloor n/2 \rfloor$ . Hence,  $S$  is representable in space polynomial in  $n$ , and verifying whether  $S$  is a valid secondary structure is computable in polynomial time.

We extend these definitions to a multiset  $s$  of  $c \in \{1, 2, 3, \dots\}$  interacting nucleic acid strands, with  $c = 1$  is called **single-stranded** and  $c > 1$  **multi-stranded**. Each strand has a unique identifier  $t \in \{1, \dots, c\}$  [8], and a base  $i_t$  has a strand identifier  $t$  and an index  $1 \leq i \leq l_t$ , where  $l_t$  is the length of the  $t^{\text{th}}$  strand. In this case,  $n$  is the total number of bases:  $n = \sum_{1 \leq t \leq c} l_t$ .

► **Definition 4** (Multi-stranded secondary structure  $S$ ). For  $c \in \mathbb{N}$  interacting strands, a secondary structure  $S$  is a set of base pairs such that each base appears in at most one pair, i.e. if  $(i_n, j_m) \in S$  and  $(k_q, l_r) \in S$  then  $i_n, j_m, k_q, l_r$  are all distinct or  $(i_n, j_m) = (k_q, l_r)$ .

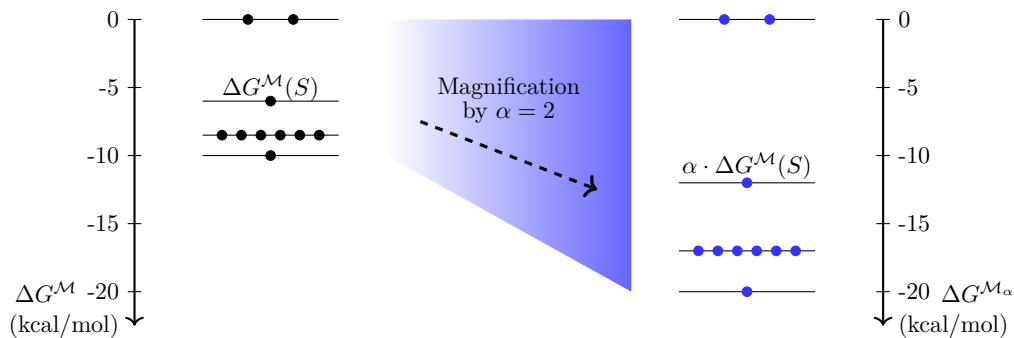
► **Definition 5** (Unpseudoknotted multi-stranded secondary structure). For  $c \in \mathbb{N}$  interacting strands, we call a secondary structure  $S$  unpseudoknotted if there exists at least one ordering  $\pi$  of the  $c$  strands such that if we consider the  $c$  strands in a row as forming one single long strand, with lexicographically ordered bases, then  $S$  is unpseudoknotted according to Definition 2, see Figure 1.

## 2.2 Energy models

We are interested in the computational complexity and relationships between five problems (Section 2.3) with an energy model as part of their input. First, we review three important energy models from the literature. Appendix A provides an analysis of the set of *candidate energy levels* for each model.

► **Definition 6** (Energy model). An **energy model**  $\mathcal{M}$  defines a free energy function  $\Delta G^{\mathcal{M}}$ , such that  $\Delta G^{\mathcal{M}} : \Omega_s \times \mathbb{R}^+ \rightarrow \mathbb{R}$ , assigns a real value  $\Delta G^{\mathcal{M}}(S, T)$  to any secondary structure  $S$  of strand  $s$ , given a temperature  $T$  (in Kelvin), where  $\Omega_s$  is the set of all secondary structures (under interest) of  $s$ .

► **Note 7** (Magnification of an energy model). For a positive real number  $\alpha \in \mathbb{R}^+$  the notation  $\alpha \cdot \Delta G^{\mathcal{M}}$  simply means to multiply the free energy function  $\Delta G^{\mathcal{M}}(S, T)$  by  $\alpha$ . Whenever we apply magnification in this paper, we do so uniformly over all secondary structures  $S$ , which does not change their relative free energy ordering nor distribution per free energy level. This magnification is simple to compute for any given  $S$  (just a multiplication), however it may not be obvious how to modify a sophisticated PF or MFE algorithm to be magnification compatible, as discussed in Section 3.5.



■ **Figure 3** Illustration of the magnification process used in Theorem 22 and Theorem 23. Each node is the free energy of a secondary structure  $S$ , with nodes on the left being  $\Delta G^{\mathcal{M}}(S)$  and the right being  $\Delta G^{\mathcal{M}_\alpha}(S) = \alpha \cdot \Delta G^{\mathcal{M}}(S)$ . Magnification increases the absolute value of all energy levels, without changing the distribution of secondary structures per energy level.

► **Definition 8** (Base pair matching (BPM) model). In the BPM model, the free energy of any secondary structure  $S$ , denoted by  $\Delta G^{\text{BPM}}(S)$ , is the number of base pairs formed in  $S$ , such that each is weighted  $-1$ , hence:  $\Delta G^{\text{BPM}}(S) = -|S|$ .

Despite the simplicity of the BPM model [22], it is still powerful enough to prove hardness results. For example, Condon, Hajiaghayi, and Thachuk [5] used it to prove the NP-hardness of MFE prediction of an unbounded set of strands in the unpseudoknotted case. In 2004, Lyngsø [19] introduced another energy model, called the base pair stacking (BPS) model, and proved NP-hardness of MFE for single-stranded pseudoknotted systems with stacks.

► **Definition 9** (Base pair stacking (BPS) model). The number of base pair stackings (BPS) of any secondary structure  $S$  is defined as  $\text{BPS}(S) = |\{(i, j) \in S \mid (i+1, j-1) \in S\}|$ . In the BPS model, the free energy of a secondary structure  $S$  is  $\Delta G^{\text{BPS}}(S) = -\text{BPS}(S)$ .

► **Note 10.** We say that an energy model  $\mathcal{M}$  is **temperature independent** if its energy function  $\Delta G^{\mathcal{M}}$  is not a function of temperature. Hence,  $\Delta G^{\mathcal{M}}(S, T) = \Delta G^{\mathcal{M}}(S)$  for any  $T$ . The BPM and BPS models are temperature independent, and the NN model (below) is not.

A word of caution. From classical thermodynamics [27, 8]: free energies are typically of the form  $\Delta G = \Delta H - T\Delta S$  meaning they are a function of temperature  $T$  and can be decomposed into enthalpic ( $\Delta H$ ) and entropic ( $\Delta S$ ) contributions. Whether DNA/RNA binding occurs is strongly temperature dependent, which is why free energies are too. Thus, temperature independent energy models are not good representations of typical temperature-varying scenarios, however they are a useful vehicle to show computational complexity results in a fixed-temperature setting.

Beyond temperature dependence, the BPM and BPS models do not account for a number of features of DNA and RNA that provide significant free energy contributions: single-stranded regions and global symmetry. This motivates our third, most realistic, energy model which is called the nearest neighbour (NN) model:

► **Definition 11** (Nearest neighbour (NN) model). Let  $S$  be an unpseudoknotted connected<sup>4</sup> secondary structure over a multiset  $s$  of  $c \geq 1$  strands.  $S$  is decomposed into a multiset of loops, denoted  $\text{loops}(S, s)$ , each being one of the types: hairpin, interior, exterior, stack, bulge, and multiloop, as described in Figure 1. Then, the free energy of  $S$  is the sum:

$$\Delta G^{\text{NN}}(S) = \sum_{l \in \text{loops}(S, s)} \Delta G(l) + (c-1)\Delta G^{\text{assoc}} + k_{\text{B}}T \log R$$

where  $\Delta G : \text{loops}(S, s) \rightarrow \mathbb{R}$  gives a free energy for each loop [25],  $\Delta G^{\text{assoc}} \in \mathbb{R}$  is an association penalty applied  $c-1$  times for a complex of  $c$  strands [8], and  $R$  is the maximum degree of rotational symmetry [26, 8] of  $S$ , details follow.

A few comments are warranted on Definition 11. At fixed temperature, salt concentration, etc., the loop free energy  $\Delta G(l)$  for a stack  $l$  is simply a function of the stack's arrangement of its four constituent DNA/RNA bases [25]. For loops  $l$  with single-stranded regions (e.g. interior, hairpin)  $\Delta G(l)$  is a logarithmic function of loop length [8, 25], although for multiloops a linear approximation is used to facilitate dynamic programming [8]. We write the multiset of loops as a function of both the secondary structure  $S$  and strand  $s$  to emphasise that both base pair indices, and base identities are used to define loops. In this work, we

<sup>4</sup> In the NN model, unlike the BPM and BPS models, multi-stranded secondary structures must be connected [8], meaning the polymer graph of a secondary structure is connected.

consider  $\Delta G^{\text{assoc}}$  to be a constant, however see [8] for more details, including temperature and water-molarity dependence. Versions of the NN model are implemented in software suites like NUPACK [8, 12], ViennaRNA [17], and mfold [31]. For more analysis of the NN model, see Appendix A.

► **Definition 12** (Set of candidate energy levels). Given an energy model  $\mathcal{M}$  and strand  $s$  (or a set of strands  $s$ ), a **set of candidate energy levels**  $\mathcal{G}_s^{\mathcal{M}}$  is a finite superset of the energies of all secondary structures of  $s$ , in other words  $\{\Delta G^{\mathcal{M}}(S) \mid S \in \Omega_s\} \subseteq \mathcal{G}_s^{\mathcal{M}}$ .

► **Note 13.**  $\mathcal{G}_s^{\mathcal{M}}$  is defined as a *superset* so that it is easily computable: specifically, computing  $\mathcal{G}_s^{\mathcal{M}}$  does not require computing the MFE.

In Appendix A we show that there are sets of candidate energy levels of polynomial size and computable in polynomial time for all three models studied. For BPM and BPS the proofs are straightforward, but for NN we rely on an assumption that individual loop free energies are specified using logarithmic precision, by which we mean they can be written down as a rational number using  $O(\log n)$  digits in units of kcal/mol. Physically, this is a reasonable assumption since measuring such free energies beyond a few decimal places is likely quite challenging. Mathematically, the assumption is not a trivial one, since hairpin, interior and bulge loops involve taking logarithms of natural numbers resulting in irrational free energies. However, computationally the assumption seems reasonable as any implemented algorithm will have finite precision.

### 2.3 Definitions of problems: MFE, PF, dMFE, dPF and #SSEL

In this section, we define the main five problems for which we establish computational complexity relationships. For convenience, we present the definitions for a single strand – the multi-stranded case is defined similarly, but the strand  $s$  is replaced by a multiset of strands, and  $n$  denotes the sum of strand lengths, or total number of bases, of the system.

► **Definition 14** (MFE; a function problem).

**Input:** Nucleic acid strand  $s$  of length  $n$ , a temperature  $T \geq 0$ , and an energy model  $\mathcal{M}$ .

**Output:** The minimum free energy  $\text{MFE}(s, T, \mathcal{M}) = \min\{\Delta G^{\mathcal{M}}(S, T) \mid S \in \Omega_s\}$ , where  $\Omega_s$  is the set of all secondary structures (under interest) of  $s$ .

► **Definition 15** (PF; a function problem).

**Input:** Nucleic acid strand  $s$  of length  $n$ , a temperature  $T \geq 0$ , and an energy model  $\mathcal{M}$ .

**Output:** The partition function  $\text{PF}(s, T, \mathcal{M}) = \sum_{S \in \Omega_s} e^{-\Delta G^{\mathcal{M}}(S, T)/k_B T}$ , where  $\Omega_s$  is the set of all secondary structures (under interest) of  $s$ .

► **Definition 16** (dMFE; a decision problem).

**Input:** Nucleic acid strand  $s$  of length  $n$ , a temperature  $T \geq 0$ , an energy model  $\mathcal{M}$  and a value  $k \in \mathbb{R}$ .

**Output:** Is  $\text{MFE}(s, T, \mathcal{M}) \leq k$ ?

► **Definition 17** (dPF; a decision problem).

**Input:** Nucleic acid strand  $s$  of length  $n$ , a temperature  $T \geq 0$ , an energy model  $\mathcal{M}$ , and a value  $k \in \mathbb{R}$ .

**Output:** Is  $\text{PF}(s, T, \mathcal{M}) \geq k$ ?

► **Definition 18** (#SSEL; a counting problem).

**Input:** Nucleic acid strand  $s$  of length  $n$ , a temperature  $T \geq 0$ , an energy model  $\mathcal{M}$ , and a value  $k \in \mathbb{R}$ .

**Output:**  $\#SSEL(s, T, \mathcal{M}, k)$ : number of secondary structures  $S$  of  $s$  such that  $\Delta G^{\mathcal{M}}(S) = k$ .



► **Definition 19** (Polynomial-time Turing reduction). A polynomial-time Turing reduction [2] from a problem  $A$  to a problem  $B$  is an algorithm that solves problem  $A$  using a polynomial number of calls to a subroutine for problem  $B$ , and polynomial time outside of those subroutine calls.

### 3 Reductions between the computational thermodynamic problems

Figure 2 illustrates most of the results of this section. For convenience, all proofs are written for the single-stranded case. However, all results hold in the multi-stranded case, simply by replacing the unique input strand by a set of multiple strands in the proofs.

#### 3.1 Straightforward reductions

In this subsection, we prove all straightforward reductions (gray colored) in our reduction map in Figure 2. In the last three reductions, we use our assumption in Note 13 about the set of candidate energy levels.

► **Theorem 20.** *There exist the following polynomial-time Turing reductions: dMFE to MFE, dPF to PF, MFE to dMFE, MFE to #SSEL, and PF to #SSEL.*

**Proof.**

1. **Reduction from dMFE to MFE:** Let  $(s, T, \mathcal{M}, k)$  be an input for the dMFE problem. After a single call to the  $\text{MFE}(s, T, \mathcal{M})$  function, simply computing the Boolean value  $(\text{MFE}(s, T, \mathcal{M}) \leq k)$  gives the answer to the dMFE problem.
2. **Reduction from dPF to PF:** Let  $(s, T, \mathcal{M}, k)$  be an input for the dPF problem. Similarly, if the partition function  $\text{PF}(s, T, \mathcal{M})$  is known, then the Boolean value  $(\text{PF}(s, T, \mathcal{M}) \geq k)$  is the answer to the dMFE problem.
3. **Turing Reduction from MFE to dMFE:** Let  $(s, T, \mathcal{M})$  be an input for the MFE problem. We know that  $\text{MFE}(s, T, \mathcal{M}) = g_j$  for some  $1 \leq j \leq |\mathcal{G}_s^{\mathcal{M}}|$  by definition of  $\mathcal{G}_s^{\mathcal{M}}$ . Determining the MFE is equivalent to determining the integer  $j$ , which can be achieved through a binary search over  $\mathcal{G}_s^{\mathcal{M}}$  thanks to the dMFE oracle. The complexity of this search is  $\mathcal{O}(\log(|\mathcal{G}_s^{\mathcal{M}}|)) \in \mathcal{O}(\text{poly}(n))$ , as  $\mathcal{G}_s^{\mathcal{M}}$  is of polynomial size. This binary search defines a Turing reduction from MFE to dMFE.
4. **Turing Reduction from MFE to #SSEL:** Let  $(s, T, \mathcal{M})$  be an input for the MFE problem, to determine  $\text{MFE}(s, T, \mathcal{M})$ , we linearly search for  $j = \max_{1 \leq i \leq |\mathcal{G}_s^{\mathcal{M}}|} \{i \mid \#SSEL(s, T, \mathcal{M}, g_i) \neq 0\}$  (i.e. the maximal such  $i$  gives the MFE due to how the indices are ordered). This search requires only a polynomial number of calls to the #SSEL oracle, giving a polynomial time Turing reduction from MFE to #SSEL.
5. **Turing Reduction from PF to #SSEL:** Let  $(s, T, \mathcal{M})$  be an input for the PF problem. The partition function  $\text{PF}(s, T, \mathcal{M}) = \sum_{1 \leq i \leq |\mathcal{G}_s^{\mathcal{M}}|} \#SSEL(s, T, \mathcal{M}, g_i) e^{-g_i/k_B T}$ . This computation is computable in polynomial time as it requires a polynomial number of calls to the #SSEL oracle, hence defining a Turing reduction from PF to #SSEL. ◀

#### 3.2 Polynomial-time Turing Reduction from #SSEL to PF

In this section, we prove one of the red-arrow reductions in Figure 2. The number of secondary structures of a strand  $s$ , denoted  $\#SecStruct(s)$ , plays an important role in our reductions. Note that  $\#SecStruct(s)$  includes pseudoknotted and unpseudoknotted structures. The proof of the following lemma is in Appendix B.1.



► **Lemma 21.** For any strand  $s$  of size  $n > 2$ ,  $\#SecStruct(s) < n!$ .

► **Theorem 22.** There exists a polynomial-time Turing reduction from #SSEL to PF.

**Proof.** For notational convenience, instead of  $\#SSEL(s, T, \mathcal{M}, g_i)$  we write  $\#SSEL(g_i)$  to denote the number of secondary structures with free energy  $g_i$ , and  $\beta = 1/k_B T$ . Let  $N = |\mathcal{G}_s^{\mathcal{M}}|$  denote the size of the set of candidate energy levels, see Definition 12 and Note 13.

Algorithm 1 gives a polynomial time Turing reduction from #SSEL to PF, for the inputs  $s, T, \mathcal{M}, g_k$ , where the main idea behind this algorithm is the following:

- We use our *magnification trick* to magnify the distances between energy levels: this idea imagines another energy model  $\mathcal{M}_j$ , where  $\Delta G^{\mathcal{M}_j} = j \cdot \Delta G^{\mathcal{M}}$  (Definition 6 defines  $\Delta G^{\mathcal{M}}$ ) which we assume a PF oracle can handle (see Section 3.5 for details).
- We compute PF using a call to the oracle  $PF(s, T, \mathcal{M}_j)$  with energy model  $\mathcal{M}_j$ .
- We do this magnification  $N$  times with different magnification factors  $1 \leq j \leq N$ .
- We end up with a system of linear equations that has a unique solution which is the number of secondary structures per energy levels, outputting the correct one  $\#SSEL(g_k)$ .

■ **Algorithm 1** Computing #SSEL by calling the oracle  $PF(s, T, \mathcal{M})$  for PF.

---

**Input:**  $s, T, \mathcal{M}, g_k$

- 1: Compute a set of candidate energy levels  $\mathcal{G}_s^{\mathcal{M}} = \{g_1, \dots, g_N\}$  ▷ see Appendix A
- 2: **for**  $j \leftarrow 1$  to  $N$  **do**
- 3:   Let  $b_j = PF(s, T, \mathcal{M}_j)$ , where  $\Delta G^{\mathcal{M}_j} = j \cdot \Delta G^{\mathcal{M}}$  ▷ see Note 7
- 4: **end for**
- 5: Solve the following system of linear equations with  $N$  unknowns  $\{\#SSEL(g_i), 1 \leq i \leq N\}$  and with  $\beta = 1/k_B T$ :

$$\begin{array}{ccccccc} \#SSEL(g_1)e^{-\beta g_1} & + & \#SSEL(g_2)e^{-\beta g_2} & + & \dots & + & \#SSEL(g_N)e^{-\beta g_N} & = & b_1 \\ \#SSEL(g_1)(e^{-\beta g_1})^2 & + & \#SSEL(g_2)(e^{-\beta g_2})^2 & + & \dots & + & \#SSEL(g_N)(e^{-\beta g_N})^2 & = & b_2 \\ \vdots & & \vdots & & \ddots & & \vdots & & \vdots \\ \#SSEL(g_1)(e^{-\beta g_1})^N & + & \#SSEL(g_2)(e^{-\beta g_2})^N & + & \dots & + & \#SSEL(g_N)(e^{-\beta g_N})^N & = & b_N \end{array}$$

- 6: **Return:**  $\#SSEL(g_k)$
- 

**Correctness.** An algorithm to compute the set  $\mathcal{G}_s^{\mathcal{M}}$  is given in Appendix A. First note that the partition function can be partitioned by energy levels  $g_i$  to give the form:  $PF(s, T, \mathcal{M}) = \sum_{i=1}^N \#SSEL(g_i)e^{-\beta \cdot g_i}$ . Hence, under magnification  $j$  (see Note 7), the general expression for all  $b_j$  is:

$$b_j = PF(s, T, \mathcal{M}_j) = \sum_{i=1}^N \#SSEL(g_i)e^{-\beta \cdot j \cdot g_i} = \sum_{i=1}^N \#SSEL(g_i)(e^{-\beta \cdot g_i})^j$$

which is the form in Algorithm 1. Algorithm 1 solves this system of linear equations; to see this we write it in the standard matrix form  $\mathbf{Ax} = \mathbf{b}$ , where:

$$\mathbf{A} = \begin{pmatrix} e^{-\beta \cdot g_1} & e^{-\beta \cdot g_2} & \dots & e^{-\beta \cdot g_N} \\ (e^{-\beta \cdot g_1})^2 & (e^{-\beta \cdot g_2})^2 & \dots & (e^{-\beta \cdot g_N})^2 \\ \vdots & \vdots & \ddots & \vdots \\ (e^{-\beta \cdot g_1})^N & (e^{-\beta \cdot g_2})^N & \dots & (e^{-\beta \cdot g_N})^N \end{pmatrix}, \quad \mathbf{x} = \begin{pmatrix} \#SSEL(g_1) \\ \#SSEL(g_2) \\ \vdots \\ \#SSEL(g_N) \end{pmatrix}, \quad \mathbf{b} = \begin{pmatrix} b_1 \\ b_2 \\ \vdots \\ b_N \end{pmatrix}$$

Since  $\mathbf{A}$  is a Vandermonde matrix [15], and  $e^{-\beta \cdot g_i} \neq e^{-\beta \cdot g_j}$  if  $i \neq j$ , then matrix  $\mathbf{A}$  is non-singular. Therefore, the system of equations  $\mathbf{Ax} = \mathbf{b}$  has a unique solution. That solution is the list  $[\#SSEL(g_1), \#SSEL(g_k), \dots, \#SSEL(g_N)]$ , the  $k$ th entry being  $g_k$ , which is returned.

**Time analysis of Algorithm 1.** By the results in Appendix A, a candidate set  $\mathcal{G}_s^{\mathcal{M}}$  is computed in polynomial time.  $N = \text{poly}(n)$ , therefore, Algorithm 1 calls the PF oracle  $\text{PF}(s, T, \mathcal{M}_j)$  a polynomial number of times. Moreover, each call has polynomial input size. Solving the resulting system of  $N$  linear equations can be achieved in  $\text{poly}(N)$  time by Gaussian elimination, since the inputs of the system are stored in polynomial space:

- Matrix  $\mathbf{A}$ : the largest entry of  $\mathbf{A}$  has size  $\log((e^{-\beta \cdot g_N})^N) = N \log(e^{-\beta \cdot g_N}) = \text{poly}(n)$ , since  $g_i = \text{poly}(N) = \text{poly}(n)$ , Note 13.
- Vector  $\mathbf{b}$ : the largest entry of  $\mathbf{b}$  has size

$$\begin{aligned} \log(b_N) &= \log(\#SSEL(g_1)(e^{-\beta g_1})^N + \dots + \#SSEL(g_N)(e^{-\beta g_N})^N) \\ &\leq \log(N \#SecStruct(s)(e^{-\beta g_N})^N) \\ &\leq \log(N) + \log(\#SecStruct(s)) + N \log(e^{-\beta g_N}) \\ &\leq \text{poly}(n) \quad \text{As } \#SecStruct(s) < n! \text{ by Lemma 21.} \end{aligned}$$

This shows the existence of a polynomial-time Turing reduction from  $\#SSEL$  to PF. ◀

### 3.3 Reduction from dMFE to dPF

► **Theorem 23.** *There exists a polynomial-time Turing reduction from dMFE to dPF.*

**Proof.** We denote the minimal step between any two energy levels by  $\delta$ , Note 13, and  $\beta = 1/(k_B T)$ . Algorithm 2 gives a polynomial time Turing reduction from dMFE to dPF, where the main idea behind this algorithm is the following:

- We use our magnification trick to make a **huge** magnification of energy levels.
- This magnification is carefully designed so that the contribution to the PF of exactly one secondary structure, the MFE one, overwhelms the contribution to PF of others.
- Our construction guarantees this by handling the worst case scenario:
  1. Only one secondary structure is at the MFE level.
  2.  $\#SecStruct(s)$  secondary structures, see Lemma 21, belong to the closest energy level to the MFE level, which is  $\text{MFE} + \delta$ .
- This huge separation of the contribution of secondary structures is achieved due to the exponential nature of the PF. That, in turn, solves dMFE problem with a dPF oracle.

■ **Algorithm 2** Solve dMFE calling an oracle for dPF.

---

**Input:**  $s, T, \mathcal{M}, k$

- 1: Compute a set of candidate energy levels  $\mathcal{G}_s^{\mathcal{M}}$  ▷ see Appendix A
- 2: Let  $x = \max\{g \in \mathcal{G}_s^{\mathcal{M}} \mid g \leq k\}$  ▷  $x$  is the max value from  $\mathcal{G}_s^{\mathcal{M}}$  that is  $\leq k$ .
- 3: Let  $\Delta G^{\mathcal{M}'} = (\log(n!)/(\beta\delta)) \cdot \Delta G^{\mathcal{M}}$
- 4: Let  $k' = e^{-\log(n!) \cdot x/\delta} = (n!)^{-x/\delta}$

**Return:**  $\text{dPF}(s, T, \mathcal{M}', k')$

---

**Correctness.** We need to prove the following claim:  $\text{dMFE}(s, T, \mathcal{M}, k) \Leftrightarrow \text{dPF}(s, T, \mathcal{M}', k')$ , which is equivalent to  $\text{MFE}(s, T, \mathcal{M}) \leq k \Leftrightarrow \text{PF}(s, T, \mathcal{M}') \geq k'$ . By definition of  $x$ , we have  $\text{MFE}(s, T, \mathcal{M}) \leq k \Leftrightarrow \text{MFE}(s, T, \mathcal{M}) \leq x$ . Therefore, it's equivalent to prove the following:

$$\text{MFE}(s, T, \mathcal{M}) \leq x \Leftrightarrow \text{PF}(s, T, \mathcal{M}') \geq k'.$$

The partition function, after magnification, is the following:

$$\text{PF}(s, T, \mathcal{M}') = \sum_{g \in \mathcal{G}_s^{\mathcal{M}}} \# \text{SSEL}(s, T, \mathcal{M}, g) e^{-\beta \cdot g \cdot \ln(n!) / (\beta \delta)} = \sum_{g \in \mathcal{G}_s^{\mathcal{M}}} \# \text{SSEL}(s, T, \mathcal{M}, g) (n!)^{-g/\delta}$$

$\Rightarrow$  If  $\text{MFE}(s, T, \mathcal{M}) \leq x$ , then a MFE secondary structure contributes to the magnified partition function,  $\text{PF}(s, T, \mathcal{M}')$ , with a coefficient of  $(n!)^{-\text{MFE}(s, T, \mathcal{M})/\delta} \geq (n!)^{-x/\delta} = k'$ , hence  $\text{PF}(s, T, \mathcal{M}') \geq k'$  as required.

$\Leftarrow$  Conversely, if  $\text{MFE}(s, T, \mathcal{M}) > x$ , then by definition of  $\delta$ ,  $\text{MFE}(s, T, \mathcal{M}) \geq x + \delta$ .

$$\begin{aligned} \text{PF}(s, T, \mathcal{M}') &= \sum_{g \in \mathcal{G}_s^{\mathcal{M}}} \# \text{SSEL}(s, T, \mathcal{M}, g) (n!)^{-g/\delta} \\ &\leq \# \text{SecStruct}(s) \cdot (n!)^{-\text{MFE}(s, T, \mathcal{M})/\delta} \\ &< n! \cdot (n!)^{-\text{MFE}(s, T, \mathcal{M})/\delta} \quad \text{As } \# \text{SecStruct}(s) < n! \text{ by Lemma 21.} \\ &< (n!)^{-(\text{MFE}(s, T, \mathcal{M}) - \delta)/\delta} \\ &< (n!)^{-x/\delta} = k' \end{aligned}$$

This proves the main claim that:  $\text{dMFE}(s, T, \mathcal{B}, k) \Leftrightarrow \text{dPF}(s, T, \mathcal{B}', k')$ .

#### Complexity analysis.

- Step 1: is done in polynomial-time, since  $|\mathcal{G}_s^{\mathcal{M}}| = \text{poly}(n)$ , Note 13.
- Step 2 (magnification step): is done in polynomial-time, since  $\log(\frac{\log(n!)}{\beta \cdot \delta}) \leq \log(n^2) + \log(k_B T) + \log(1/\delta) \leq \text{poly}(n) + \text{poly}(T)$ , due to the logarithmic size of  $1/\delta$ , Note 13.
- Step 3:  $k'$  can be computed in  $\text{poly}(n)$  time, and the size of  $k'$  is  $\log((n!)^{-x/\delta}) \leq (-x/\delta) \cdot n^2 = \text{poly}(n)$  bits.

This shows the existence of a polynomial-time Turing reduction from dMFE to dPF.  $\blacktriangleleft$

### 3.4 Polynomial-time Turing Reduction from PF to dPF

This next reduction also exploits our magnification trick, hence is slightly more involved than the previous one from MFE to dMFE. while for PF, the search space is of exponential size. A first approach would be to binary search for the PF value, using the oracle dPF. This approach runs in polynomial time if we are satisfied with linear precision of the PF value (but the PF can have arbitrary precision since it uses exponentials of  $e$ ). However, we can search for the exact value by combining the simple binary search with our magnification trick, efficiently exploiting the dPF oracle to compute the exact PF contribution at each energy level. The proof of the following theorem is in Appendix B.2.

► **Theorem 24.** *There exists a polynomial-time Turing reduction from PF to dPF.*

### 3.5 Discussion for Section 3: reductions and the magnification trick

When we call a specific oracle using our *magnification trick*, we ask the oracle to function under the same energy model  $\mathcal{M}$  but magnified (e.g. by factor  $j$  in Line 3 of Algorithm 1, to give  $\mathcal{M}_j$ ). We believe this kind of magnification of the energy model, combined with our reductions in Section 3, is a useful notion for finding polynomial time algorithms for thermodynamic problems (such as those in Figure 2), hence we propose a general definition:

► **Definition 25** (PF-polynomially magnifiable energy model). An energy model  $\mathcal{M}$  is **PF-polynomially magnifiable** if there is a polynomial time algorithm  $\text{PF}(\cdot, \cdot, \mathcal{M}_j)$  that computes PF under the  $j$ -magnified energy model:  $\Delta G^{\mathcal{M}_j} = j \cdot \Delta G^{\mathcal{M}}$ , for all  $j \in \text{poly}(n)$ , and we call  $\text{PF}(\cdot, \cdot, \mathcal{M})$  a **magnification adaptable** algorithm under  $\mathcal{M}$ .

### 3.5.1 Positive (polynomial time) results

In the unpseudoknotted 1-strand, or even  $\mathcal{O}(1)$ -strand cases, with the BPM and BPS models, there is a polynomial time algorithm for PF. It is not hard to show that PF for BPM and BPS is magnification adaptable, hence BPM and BPS are PF-polynomially magnifiable energy models. Hence, from our reductions and that PF algorithm, we get polynomial time algorithms for free for the other four problems besides PF in Figure 2. The same holds for the NN model *without rotational symmetry* (i.e. single-stranded NN model, or the multi-stranded NN model that ignores rotational symmetry).

► **Corollary 26.** *When  $\mathcal{M}$  is a PF-polynomially magnifiable model, all five problems in Figure 2 are in P.*

In the NN model with multiple strands and unpseudoknotted secondary structures, Dirks et al. [8] gave a polynomial time algorithm for PF, and Shalaby and Woods [26] gave one for MFE. One might ask if Dirks et al. [8], plus the reductions in Figure 2 are sufficient to yield a polynomial time algorithm for MFE, but this is not known since we don't know whether the Dirks et al. PF algorithm is magnification adaptable. The issue is in how Dirks et al. handle rotational symmetry, not by pure dynamic programming, but by computing a naive PF that: (1) Completely ignores rotational symmetry, and (2) Overcounts *indistinguishable* secondary structures (less symmetry, means more overcounting, see [8] for details). Then they use an algebraic argument to bring back rotational symmetry simultaneously with canceling out the overcounting effect, an argument that may not be preserved under our magnification trick. Hence we do not currently know whether the NN model in its full generality (including rotational symmetry) is PF-polynomially magnifiable.

### 3.5.2 Temperature independent energy models

If the energy model is not PF-polynomially magnifiable, but is temperature independent (Note 10), we can achieve our magnification trick simply by instead reducing (i.e. inverse magnification) the partition function temperature as follows (this assumes, as usual, that the partition function *is* temperature dependent):

- Assume we have a polynomial time algorithm  $\text{PF}(\cdot, \cdot, \mathcal{M})$  but want to compute  $\text{PF}(\cdot, \cdot, \mathcal{M}')$ , where  $\Delta G^{\mathcal{M}'} = \alpha \Delta G^{\mathcal{M}}$ .
- Call  $\text{PF}(s, T', \mathcal{M})$ , with  $T' = T/\alpha$ .

$$\begin{aligned}
 \text{PF}(s, T', \mathcal{M}) &= \sum_{S \in \Omega} e^{-\Delta G^{\mathcal{M}}(S)/k_B T'} \\
 &= \sum_{S \in \Omega} e^{-\alpha \Delta G^{\mathcal{M}}(S)/k_B T} && \mathcal{M} \text{ is temperature independent.} \\
 &= \text{PF}(s, T, \mathcal{M}')
 \end{aligned}$$

- Hence,  $\text{PF}(\cdot, \cdot, \mathcal{M})$  is magnification adaptable.

► **Note 27 (NP-hardness results).** As we mentioned before, the dMFE problem is NP-hard in: (1) The BPS model when we allow pseudoknots [19], and (2) The BPM model when we have unbounded number of strands [5] without pseudoknots. Then, #SSEL, PF, and dPF are NP-hard as well in these energy models using our reduction map, Figure 2. However, we provide **sharper** complexity results in the following sections.

#### 4 #P-hardness of PF and #SSEL in BPS model

In this section, we prove #P-hardness for the problems  $\text{PF}(\cdot, \cdot, \text{BPS})$  and  $\text{\#SSEL}(\cdot, \cdot, \text{BPS})$  (see Definition 9 for the BPS model). Our strategy is to construct a reduction chain from #3DM to #4-PARTITION, then from #4-PARTITION to #BPS, where #BPS is the counting problem of the number of secondary structures having exactly  $K$  stacks.

First, we state the definitions of the three problems within the reduction chain, before showing that there exist weakly parsimonious reductions<sup>5</sup> within the chain.

► **Definition 28** (#3DM).

**Input:** Three finite sets  $X$ ,  $Y$ , and  $Z$  of the same size, and a subset  $T \subseteq X \times Y \times Z$ .

**Output:** The number of perfect 3-dimensional matchings  $M$ , where  $M \subseteq T$  is a perfect 3-dimensional matching if any element in  $X \cup Y \cup Z$  belongs to exactly one triple in  $M$ .

► **Definition 29** (#4-PARTITION).

**Input:** A set of  $k$  elements  $\mathcal{A} = \{a_1, a_2, \dots, a_k\}$ , a weight function  $w : \mathcal{A} \rightarrow \mathbb{Z}^+$ , and a bound  $B \in \mathbb{Z}^+$  such that the weight of each element  $w(a_i)$  is strictly between  $B/5$  and  $B/3$ .

**Output:** The number of partitions of  $\mathcal{A}$  into 4-tuples, such that the sum of the weights of the elements in each tuple is equal to  $B$ .

► **Definition 30** (#BPS).

**Input:** A nucleic acid strand  $s$  and a number  $K \in \mathbb{N}$ .

**Output:** The number of secondary structures  $S$  of  $s$  that have  $K$  base pair stackings.

We prove that #BPS is #P-complete through two consecutive weakly parsimonious reductions: from #3DM, to #4-PARTITION, and from #4-PARTITION to #BPS. The proof of the following lemma and theorem are in Appendix B.3.

► **Lemma 31.** *#4-PARTITION is #P-complete.*

► **Theorem 32.** *#BPS is #P-complete.*

► **Theorem 33.** *#SSEL( $\cdot, \cdot, \text{BPS}$ ) is #P-complete and  $\text{PF}(\cdot, \cdot, \text{BPS})$  is #P-hard.*

**Proof.** #SSEL( $\cdot, \cdot, \text{BPS}$ ) belongs to #P and is equivalent to the problem #BPS. Hence, #SSEL( $\cdot, \cdot, \text{BPS}$ ) is #P-complete. The reduction map (Figure 2), specifically Theorem 22, implies that computing the partition function is #P-hard in the BPS model. ◀

► **Theorem 34.** *In the BPS model, there is no polynomial-time Turing reduction from #SSEL to dMFE, unless  $\#P \subseteq P^{\text{NP}}$ .*

**Proof.** dMFE is NP-complete in the BPS model according to Lyngsø [19], and #SSEL is #P-complete according to Theorem 33. Suppose that there exists a polynomial-time Turing reduction from #SSEL to dMFE (in the BPS model):  $\#SSEL \in P^{\text{dMFE}}$ .

As these problems are complete,  $\#SSEL \in P^{\text{dMFE}} \Rightarrow \#P \subseteq P^{\text{NP}}$ . ◀

► **Remark 35.** If  $\#P \subseteq P^{\text{NP}}$ , then  $P^{\#P} \subseteq P^{\text{P}^{\text{NP}}} = P^{\text{NP}} = \Delta_2^P \subseteq \Sigma_2^P$ . According to Toda's theorem [28],  $\text{PH} \subseteq P^{\#P}$ . Therefore,  $\text{PH} \subseteq \Sigma_2^P$ : the Polynomial Hierarchy would collapse at level 2.

<sup>5</sup> A reduction is weakly parsimonious if there is a suitably-computable relation between the number of solutions of the two problems; depending only on the initial problem instance [2].

## 5 #P-hardness of PF and #SSEL for unpseudoknotted secondary structures of an unbounded set of strands in the BPM model

In this section, we prove #P-hardness of the two restricted problems  $\text{PF}(\cdot, \cdot, \text{BPM})$  and  $\text{\#SSEL}(\cdot, \cdot, \text{BPM})$ , see Definition 8 (BPM model), in the scenario of unpseudoknotted structures of an unbounded set of strands. Specifically, we show that the problem #MULTI-PKF-SSP is #P-complete. This main result is proven using lemmas from Appendix B.4.

► **Definition 36** (#MULTI-PKF-SSP).

**Input:**  $c$  nucleic acid strands and a positive integer  $k$ .

**Output:** The number of unpseudoknotted secondary structures containing exactly  $k$  base pairs formed by the  $c$  strands.

► **Theorem 37.** #MULTI-PKF-SSP is #P-complete.

**Proof.** To prove #P-completeness, we first consider the reduction provided by Condon, Hajiaghayi, and Thachuk for the NP-completeness of the associated decision problem [5]. Using the same notation, we show that this reduction is weakly parsimonious. Since this reduction contains many details, we do not recall all of them here.

We first note that the reduction in [5] is from a variation of 3-Dimensional Matching, 3DM(3), in which each element appears in at most 3 triples. Without loss of generality, this reduction can be extended from another version of 3DM, in which all triples are distinct.

Condon, Hajiaghayi, and Thachuk showed (in Lemma 12 of [5]) that any solution of 3DM, can be transformed into a secondary structure of the new instance  $I'$  containing  $P$  base pairs, where  $P$  is the maximum number of possible base pairs ( $P = \min(A, T) + \min(G, C)$ ). It is easy to see that this transformation from 3DM to MULTI-PKF-SSP is injective.

For the reverse transformation, we show an intermediate result in Lemma 43, ensuring the form of any secondary structure of the new instance  $I'$  containing  $P$  base pairs. This lemma is stated and proven in Appendix B.4. According to this lemma, the back transformation is clear: considering the perfect triples provides a solution for 3DM. Let two secondary structures leading to the same 3DM solution. Consequently, they have the same perfect and center-trim-deprived triples. Let us make a little dichotomy here:

- If we consider the strands as indistinguishable, then the two secondary structures are the same and the reduction is parsimonious.
- Else if we consider the strands as distinguishable, then the two secondary structures differ as some of their trim-complement and separator support strands are permuted. Since there are  $2n$  separator strands,  $2n$  separator-complement strands and  $2m + n$  trim-complement strands, there are  $(2n)!^2 \cdot (2m + n)!$  distinct secondary structures mapping to the same 3DM solution.

In both cases, there is a simple relation between the number of solutions of the two problems, and hence the reduction is weakly parsimonious.

#MULTI-PKF-SSP belongs to #P, since MULTI-PKF-SSP is in NP. Since #3DM is #P-complete even in the case of distinct triples [3], #MULTI-PKF-SSP is also #P-complete. ◀

► **Theorem 38.** #SSEL( $\cdot, \cdot, \text{BPM}$ ) is #P-complete and  $\text{PF}(\cdot, \cdot, \text{BPM})$  is #P-hard, in the scenario of unpseudoknotted structures of an unbounded set of strands.

**Proof.** It is clear that #SSEL( $\cdot, \cdot, \text{BPM}$ ) belongs to #P and is equivalent to the problem #MULTI-PKF-SSP in this scenario. Hence, #SSEL( $\cdot, \cdot, \text{BPM}$ ) is #P-complete.

The reduction map (Figure 2) and specifically Theorem 22 implies that computing the partition function is #P-hard in this scenario as well. ◀

## References

- 1 Tatsuya Akutsu. Dynamic programming algorithms for RNA secondary structure prediction with pseudoknots. *Discrete Applied Mathematics*, 104(1):45–62, 2000. doi:10.1016/S0166-218X(00)00186-4.
- 2 Sanjeev Arora and Boaz Barak. *Computational complexity: a modern approach*. Cambridge University Press, 2009.
- 3 Jeffrey Bosboom, Erik D. Demaine, Martin L. Demaine, Adam Hesterberg, Roderick Kimball, and Justin Kopinsky. Path puzzles: Discrete tomography with a path constraint is hard. *Graphs and Combinatorics*, 36:251–267, 2020. doi:10.1007/s00373-019-02092-5.
- 4 Ho-Lin Chen, Anne Condon, and Hosna Jabbari. An  $\mathcal{O}(n^5)$  algorithm for MFE prediction of kissing hairpins and 4-chains in nucleic acids. *Journal of Computational Biology*, 16(6):803–815, 2009. PMID: 19522664. doi:10.1089/cmb.2008.0219.
- 5 Anne Condon, Monir Hajiaghayi, and Chris Thachuk. Predicting minimum free energy structures of multi-stranded nucleic acid complexes is APX-hard. In Matthew R. Lakin and Petr Šulc, editors, *27th International Conference on DNA Computing and Molecular Programming (DNA 27)*, volume 205 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 9:1–9:21, Dagstuhl, Germany, 2021. Schloss Dagstuhl – Leibniz-Zentrum für Informatik. doi:10.4230/LIPIcs.DNA.27.9.
- 6 Jan Cupal, Ivo L. Hofacker, and Peter F. Stadler. Dynamic programming algorithm for the density of states of rna secondary structures. In Reinhard Hofstädt, Thomas Lengauer, Markus Löffler, and Dirk Schomburg, editors, *Computer Science and Biology’96 (German Conference on Bioinformatics)*, pages 184–186, Leipzig, Germany, 1996. University of Leipzig.
- 7 Erik D Demaine, Timothy Gomez, Elise Grizzell, Markus Hecher, Jayson Lynch, Robert Schweller, Ahmed Shalaby, and Damien Woods. Domain-based nucleic-acid minimum free energy: Algorithmic hardness and parameterized bounds. In *30th International Conference on DNA Computing and Molecular Programming (DNA 30)*, volume 30 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 2:1–2:21. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2024. doi:10.4230/LIPIcs.DNA.30.2.
- 8 Robert M Dirks, Justin S Bois, Joseph M Schaeffer, Erik Winfree, and Niles A Pierce. Thermodynamic analysis of interacting nucleic acid strands. *SIAM review*, 49(1):65–88, 2007. doi:10.1137/060651100.
- 9 Robert M Dirks and Niles A Pierce. A partition function algorithm for nucleic acid secondary structure including pseudoknots. *Journal of computational chemistry*, 24(13):1664–1677, 2003. doi:10.1002/JCC.10296.
- 10 Robert M Dirks and Niles A Pierce. An algorithm for computing nucleic acid base-pairing probabilities including pseudoknots. *Journal of computational chemistry*, 25(10):1295–1304, 2004. doi:10.1002/JCC.20057.
- 11 Gwendal Ducloz, Ahmed Shalaby, and Damien Woods. Algorithmic hardness of the partition function for nucleic acid strands, 2025. Extended arXiv version of this paper: arXiv:2506.19756.
- 12 Mark E Fornace, Nicholas J Porubsky, and Niles A Pierce. A unified dynamic programming framework for the analysis of interacting nucleic acid strands: enhanced models, scalability, and speed. *ACS Synthetic Biology*, 9(10):2665–2678, 2020.
- 13 Michael R Garey and David S Johnson. *Computers and Intractability; A Guide to the Theory of NP-Completeness*. W. H. Freeman & Co., USA, 1979.
- 14 Leslie Ann Goldberg, Martin Grohe, Mark Jerrum, and Marc Thurley. A complexity dichotomy for partition functions with mixed signs. *SIAM Journal on Computing*, 39(7):3336–3402, 2010. doi:10.1137/090757496.
- 15 Roger A Horn and Charles R Johnson. *Matrix Analysis*. Cambridge University Press, 2nd edition, 2012.



- 16 Hosna Jabbari, Ian Wark, Carlo Montemagno, and Sebastian Will. Knotty: efficient and accurate prediction of complex RNA pseudoknot structures. *Bioinformatics*, 34(22):3849–3856, 2018. doi:10.1093/BIOINFORMATICS/BTY420.
- 17 Ronny Lorenz, Stephan H Bernhart, Christian Höner zu Siederdissen, Hakim Tafer, Christoph Flamm, Peter F Stadler, and Ivo L Hofacker. ViennaRNA package 2.0. *Algorithms for Molecular Biology*, 6(1):26, 2011. doi:10.1186/1748-7188-6-26.
- 18 Feng Lou and Peter Clote. Thermodynamics of RNA structures by Wang–Landau sampling. *Bioinformatics*, 26(12):i278–i286, 2010.
- 19 Rune B Lyngsø. Complexity of pseudoknot prediction in simple models. In Josep Díaz, Juhani Karhumäki, Arto Lepistö, and Donald Sannella, editors, *Automata, Languages and Programming*, pages 919–931, Berlin, Heidelberg, 2004. Springer Berlin Heidelberg. doi:10.1007/978-3-540-27836-8\_77.
- 20 Rune B Lyngsø and Christian NS Pedersen. RNA pseudoknot prediction in energy-based models. *Journal of computational biology*, 7(3-4):409–427, 2000. doi:10.1089/106652700750050862.
- 21 John S McCaskill. The equilibrium partition function and base pair binding probabilities for RNA secondary structure. *Biopolymers: Original Research on Biomolecules*, 29(6-7):1105–1119, 1990.
- 22 Ruth Nussinov and Ann B Jacobson. Fast algorithm for predicting the secondary structure of single-stranded RNA. *Proceedings of the National Academy of Sciences*, 77(11):6309–6313, 1980.
- 23 Ruth Nussinov, George Pieczenik, Jerrold R Griggs, and Daniel J Kleitman. Algorithms for loop matchings. *SIAM Journal on Applied mathematics*, 35(1):68–82, 1978.
- 24 Elena Rivas and Sean R Eddy. A dynamic programming algorithm for RNA structure prediction including pseudoknots. *Journal of molecular biology*, 285(5):2053–2068, 1999.
- 25 John SantaLucia Jr and Donald Hicks. The thermodynamics of DNA structural motifs. *Annual Review of Biophysics*, 33:415–440, 2004. doi:10.1146/annurev.biophys.32.110601.141800.
- 26 Ahmed Shalaby and Damien Woods. An efficient algorithm to compute the minimum free energy of interacting nucleic acid strands. In *ICALP: The 52nd International Colloquium on Automata, Languages and Programming*, Leibniz International Proceedings in Informatics (LIPIcs), 2025. To appear. arXiv version: arXiv:2407.09676. doi:10.48550/arXiv.2407.09676.
- 27 Ignacio Tinoco, Olke C Uhlenbeck, and Mark D Levine. Estimation of secondary structure in ribonucleic acids. *Nature*, 230(5293):362–367, 1971.
- 28 Seinosuke Toda. On the computational power of PP and  $\oplus P$ . In *FOCS: 54th Annual Symposium on Foundations of Computer Science*, pages 514–519. IEEE Computer Society, 1989. doi:10.1109/SFCS.1989.63527.
- 29 Leslie G. Valiant. The complexity of enumeration and reliability problems. *SIAM Journal on Computing*, 8(3):410–421, 1979. doi:10.1137/0208032.
- 30 John N. Zadeh, Christopher D. Steenberg, Justin S. Bois, Blake R. Wolfe, Matthew B. Pierce, Abbas R. Khan, Raymond M. Dirks, and Niles A. Pierce. NUPACK: Analysis and design of nucleic acid systems. *Journal of Computational Chemistry*, 32(1):170–173, 2011. doi:10.1002/JCC.21596.
- 31 Michael Zuker. Mfold web server for nucleic acid folding and hybridization prediction. *Nucleic acids research*, 31(13):3406–3415, 2003. doi:10.1093/NAR/GKG595.
- 32 Michael Zuker and David Sankoff. RNA secondary structures and their prediction. *Bulletin of mathematical biology*, 46:591–621, 1984.
- 33 Michael Zuker and Patrick Stiegler. Optimal computer folding of large RNA sequences using thermodynamics and auxiliary information. *Nucleic acids research*, 9(1):133–148, 1981. doi:10.1093/NAR/9.1.133.



## A Appendix: Candidate energy levels for three free energy models

We state two lemmas, the proofs of which are in the full version of this work [11]. The first states that for an instance of any of the three energy models, BPM, BPS and NN, there is a set of candidate energy levels of polynomial size and polynomial time computable in number of bases  $n$ . For the NN model, the second lemma implies two more efficient and refined algorithms that better approximate the energy levels occupied by secondary structures.

► **Lemma 39.** *Given a multiset of strands  $s$  and temperature  $T$ , for each of the BPS, BPM and NN models, there is a set of candidate energy levels  $\mathcal{G}_s^M$  of polynomial size and computable in polynomial time in the number of bases  $n$ . More precisely, the size  $|\mathcal{G}_s^M|$  is linear for BPS and BPM, and  $O(n^{O(1)})$  for NN.*

► **Lemma 40.** *Given a multiset of strands  $s$  and temperature  $T$  for the NN model, there is an  $O(n^4|\mathcal{G}'|^2)$  algorithm to compute a set of candidate energy levels  $\mathcal{G}_s^{NN}$ , where  $|\mathcal{G}'| = n^{O(1)}$  is defined in the proof. The computed set is precisely the occupied energy levels (that have at least one structure) if we ignore rotational symmetry. A second algorithm allows for rotational symmetry, but gives a (typically strict) superset of the occupied energy levels.*

## B Appendix: Omitted proofs

### B.1 Proof of Lemma 21, omitted from Section 3

► **Lemma 21.** *For any strand  $s$  of size  $n > 2$ ,  $\#\text{SecStruct}(s) < n!$ .*

**Proof.** By enumerating all secondary structures based on the number of base pairs formed in each. Any secondary structure  $S$  has  $k \leq \lfloor n/2 \rfloor$  base pairs, these  $k$  base pairs are formed between  $2k$  different bases. The first base, among the  $2k$  different bases, has  $2k - 1$  bases to form a base pair with. The next “unpaired” base has  $(2k - 3)$  bases to form a base pair with. At the end, the number of secondary structures which have exactly  $k$  base pairs is upper bounded by  $\binom{n}{2k} \cdot (2k - 1) \cdot (2k - 3) \cdots 3 \cdot 1$ . Leading to

$$\#\text{SecStruct}(s) \leq \sum_{k=0}^{\lfloor n/2 \rfloor} \binom{n}{2k} \cdot (2k - 1) \cdot (2k - 3) \cdots \quad (2)$$

$$\leq \sum_{k=0}^{\lfloor n/2 \rfloor} \frac{n!}{(2k)! (n - 2k)!} \frac{1 \cdot 2 \cdot 3 \cdots 2k}{2 \cdot 4 \cdot 6 \cdots 2k} \quad (3)$$

$$\leq \sum_{k=0}^{\lfloor n/2 \rfloor} \frac{n!}{(2k)! (n - 2k)!} \frac{(2k)!}{k! 2^k} \quad (4)$$

$$\leq n! \sum_{k=0}^{\lfloor n/2 \rfloor} \frac{1}{(n - 2k)! k! 2^k} \quad (5)$$

$$\leq n! \sum_{k=0}^{\lfloor n/2 \rfloor} \frac{1}{\lfloor n/3 \rfloor! 2^k} \quad \text{as } \sum_{k=0}^{\lfloor n/2 \rfloor} \frac{1}{2^k} < 2 \quad (6)$$

$$< \frac{2n!}{\lfloor n/3 \rfloor!} \leq n! \quad \text{for } n \geq 6 \quad (7)$$

Note that,  $\#\text{SecStruct}(s) = n!$  for  $n = 1$  or  $n = 2$ , which are useless cases for any system. To extend the inequality to all  $n > 2$ , we only need to check if it is verified for  $3 \leq n \leq 5$ . By computing the right-hand side of Equation (2), the associated numbers of possible secondary structures are at most 4, 10, and 26 which are upper bounded by its corresponding  $n!$ . Therefore,  $\#\text{SecStruct}(s) < n!$  for all  $n > 2$ .  $\blacktriangleleft$

## B.2 Proof of Theorem 24, omitted from Section 3

► **Theorem 24.** *There exists a polynomial-time Turing reduction from PF to dPF.*

**Proof.** Let  $(s, T, \mathcal{M})$  be the input of the PF problem, and consider Algorithm 2 and notations used in Theorem 23. Since  $\#\text{SSEL}(s, T, \mathcal{M}, g) < n!$  for all  $g \in \mathcal{G}_s^{\mathcal{M}}$  by Lemma 21, the partition function  $\text{PF}(s, T, \mathcal{M}) = \sum_{g \in \mathcal{G}_s^{\mathcal{M}}} \#\text{SSEL}(s, T, \mathcal{M}, g)(n!)^{-g/\delta}$  can be seen as a number written in base  $n!$ . Determining this number in base  $n!$  is equivalent to determining all coefficients of the form  $\#\text{SSEL}(s, T, \mathcal{M}, g)$ . To achieve this, we do the following:

- We perform a binary search for each coefficient  $\#\text{SSEL}(s, T, \mathcal{M}, g)$  in decreasing order over  $\mathcal{G}_s^{\mathcal{M}}$  (from the most to the less favourable).
- First, guess  $\#\text{SSEL}(s, T, \mathcal{M}, g_{|\mathcal{G}_s^{\mathcal{M}}|})$ , by calling  $\text{dPF}(s, T, \mathcal{M}', c_1(n!)^{-g_{|\mathcal{G}_s^{\mathcal{M}}|}/\delta})$ , repeatedly, using binary search with search variable  $c_1$  over the range  $0 \leq c_1 \leq n!$ .
- Inductively, after computing  $\#\text{SSEL}(s, T, \mathcal{M}, g_i)$ , find  $\#\text{SSEL}(s, T, \mathcal{M}, g_{i-1})$  by calling dPF oracle with input  $(s, T, \mathcal{M}', \sum_{j=1}^i \#\text{SSEL}(s, T, \mathcal{M}, g_j)(n!)^{-g_j/\delta} + c_{i-1}(n!)^{-g_{i-1}/\delta})$ , where  $c_{i-1}$  is the search variable:  $0 \leq c_{i-1} \leq n!$ .
- At the end, we know all of the  $\#\text{SSEL}(s, T, \mathcal{M}, g_i)$  values, we then combine them to directly compute the partition function  $\text{PF}(s, T, \mathcal{M}) = \sum_{i=1}^N \#\text{SSEL}(s, T, \mathcal{M}, g_i)e^{-g_i/(k_B T)}$ .

By using binary search, the algorithm requires only  $|\mathcal{G}_s^{\mathcal{M}}| \log(n!)$  calls to dPF oracles, which is  $\text{poly}(n)$ , Note 13. Hence, it is a polynomial-time Turing reduction from PF to dPF.  $\blacktriangleleft$

## B.3 Proof of Lemma 31, omitted from Section 4

► **Lemma 31.** *#4-PARTITION is #P-complete.*

For the sake of completeness, we first include a quick overview of the 4-PARTITION NP-completeness from Garey and Johnson (1979) [13], as we heavily depend on it in our reduction. After that we give the proof of Lemma 31.

### Proof overview of the NP-completeness of 4-PARTITION.

Given an instance of 3DM with three disjoint sets  $X, Y, Z$ , with  $|X| = |Y| = |Z|$ , and a set  $T \subseteq X \times Y \times Z$ , the reduction constructs an instance  $(\mathcal{A}, w, B)$  of 4-PARTITION as follows:

- For each element  $x \in X$ , we introduce  $N(x)$  elements  $(x[i])_{1 \leq i \leq N(x)}$  in  $\mathcal{A}$ , where  $N(x)$  is the number of times  $x$  appears in  $T$ . These elements have two possible weights, depending on whether  $i = 1$  or not. The element  $x[1]$  is called *actual* while the other ones are called *dummy*. Same happens to every  $y \in Y$ , and  $z \in Z$ .
- For each triple  $(x, y, z) \in T$ , an element  $u_{xyz}$ , is introduced in  $\mathcal{A}$ , whose weight depends on  $x, y$  and  $z$ .
- The bound  $B$  and the weight function  $w$  are constructed such that the following equivalence hold: A 4-tuples has weight equaling  $B$  iff it is of the form  $(u_{xyz}, x[i], y[j], z[k])$ , with  $i, j, k$  being all equal to 1, or all greater than 1. In that case, we call the 4-tuple either *dummy* or *actual*.

Given a 3D-matching  $M$ , we can construct the following 4-partition  $P$ . For each triple  $(x, y, z) \in M$ , we add to  $P$  the *actual* 4-tuple  $(u_{xyz}, x[1], y[1], z[1])$ . For each triple  $(x, y, z) \in T \setminus M$ , we add to  $P$  the *dummy* 4-tuple  $(u_{xyz}, x[i], y[j], z[k])$ , with  $i, j$ , and  $k$  all greater than 1. It is possible to construct these triples since there are enough *dummy* elements.  $P$  is a solution of 4-PARTITION since it is a partition (all elements are part of a 4-tuple) and since the weight of each 4-tuple equals  $B$  by construction.

Reciprocally, given a 4-partition  $P$  of  $\mathcal{A}$ , we construct a 3D-matching  $M$ . All 4-tuples of  $P$  have weight equaling  $B$  and are of the expected form. We introduce in  $M$  the triples  $(x, y, z) \in T$  for the ones  $u_{xyz}$  is part of an *actual* 4-tuple in  $P$ .  $M$  is a matching since there is exactly one *actual* element  $a[i]$  per element  $a \in X \cup Y \cup Z$ . Moreover,  $M$  is perfect since every *actual* element belongs to an *actual* 4-tuple. ◀

**Proof.** Now, we prove that in the mentioned reduction from 3DM to 4-PARTITION [13], the number of solutions (4-tuples) equals the number of perfect matchings in the initial instance of 3DM multiplied by a coefficient that depends only on that initial instance.

Let  $M$  and  $M'$  two distinct perfect 3D-matchings. There exists  $(x, y, z) \in M \setminus M'$ . Therefore,  $(u_{xyz}, x[1], y[1], z[1])$  belongs to a 4-partition that corresponds to  $M$ , while the 4-tuple  $(u_{xyz}, x[i], y[j], z[k])$  in a 4-partition corresponding to  $M'$ , must have  $i, j$ , and  $k$  being greater than 1. Therefore, the two 4-partitions corresponding to  $M$  and  $M'$  are distinct.

Reciprocally, let  $P$  and  $P'$  are two 4-partitions corresponds to the same 3D-matching. Each tuple of  $P$  and  $P'$  is of the form  $(u_{xyz}, x[i], y[j], z[k])$  where  $(x, y, z) \in T$  and  $i, j, k$  are equal to 1 or greater than 1. The Garey and Johnson's proof states that these two partitions have the same collections of *actual* 4-tuples. Therefore, they differ in their *dummy* 4-tuples (with  $i, j, k$  greater than 1), which means that some *dummy* elements happens in different order.

For each element  $a \in X \cup Y \cup Z$ , there are  $N(a) - 1$  *dummy* elements. Therefore, there are  $\alpha = \prod_{a \in X \cup Y \cup Z} (N(a) - 1)!$  different ways to arrange these elements in dummy collections. We can conclude that there are  $\alpha$  distinct 4-partitions corresponding to the same 3D-matching.

This implies that the number of solutions of 4-PARTITION equals the number of perfect matchings in the initial instance of 3DM multiplied by multiplied by  $\alpha$ , hence this reduction is weakly parsimonious.

It is straightforward that #4-PARTITION belongs to #P as 4-PARTITION belongs to NP. Since #3DM is #P-hard according to Bosboom et al. [3], then #4-PARTITION is #P-complete. ◀

► **Theorem 32.** *#BPS is #P-complete.*

**Proof.** In 2004, Lyngsø [19] proved that the decision problem BPS is NP-complete, with a reduction from BIN-PACKING. We adapt his reduction from 4-PARTITION to BPS. Let  $(\mathcal{A} = \{a_1, \dots, a_k\}, w, B)$  be an instance of 4-PARTITION. Let us first note that in any instance of 4-PARTITION,  $k$  is a multiple of 4, and  $Bk/4 = \sum_{i=1}^k w(a_i)$ , otherwise, it is straightforward to see that this instance has no solution. We proceed exactly as Lyngsø, constructing the following DNA strand  $s$ :

$$s = C^{w(a_1)} A C^{w(a_2)} A \dots A C^{w(a_k)} A A A \underbrace{G^B A G^B A \dots A G^B}_{k/4 \text{ substrings of } G's}$$

and setting the target  $K = \sum_{i=1}^k w(a_i) - k$ .

As A bases can only form base pairs with T bases, all base pairs in a secondary structure of  $s$  will be C-G base pairs. If a substring  $C^{w(a_i)}$  binds with at least two distinct substrings  $G^B$ , then it accounts for at most  $w(a_i) - 2$  in the BPS score. Since the other substrings  $C^{w(a_j)}$  account for at most  $w(a_j) - 1$ , then  $\text{BPS}(S) \leq K - 1$ . Hence, we can find a structure  $S$  with  $\text{BPS}(S) = K$  iff we can partition the  $k$  substrings  $C^{w(a_i)}$  into  $k/4$  groups that can each be fully base paired using one substring  $G^B$ ; i.e. the total length of the substrings of C's in any group can be at most  $B$ .

It means that  $\text{BPS}(S) = K$  iff we can partition the  $k$  elements  $a_i$  into  $k/4$  groups that each has total weight  $\leq B$ . Since  $B = \frac{4}{k} \sum_{i=1}^k w(a_i)$  and since the weight of each element in  $\mathcal{A}$  is strictly between  $B/5$  and  $B/3$ , each group has a total weight of exactly  $B$  and contains exactly 4 elements. Therefore,  $\text{BPS}(S) = K$  iff 4-PARTITION problem has a solution.

Consider two distinct 4-partitions  $P_1$  and  $P_2$ . Let  $(a_i, a_j, a_l, a_m)$  belong to  $P_1$  but not to  $P_2$ . The secondary structure corresponding to  $P_1$  has the group  $(C^{w(a_i)}, C^{w(a_j)}, C^{w(a_l)}, C^{w(a_m)})$  fully bounded with a substring  $G^B$ . The secondary structure corresponding to  $P_2$  does not have these bindings, as  $(a_i, a_j, a_l, a_m)$  does not belong to  $P_2$ . Therefore, all distinct 4-partitions map to distinct secondary structures.

Conversely, let  $S_1$  and  $S_2$  two secondary structures of  $s$ , having  $K$  base pair stackings and corresponding to the same 4-partition. Consequently, the groups of 4 elements bounded with the substrings  $G^B$  are the same in  $S_1$  and  $S_2$ , up to permutations. Specifically, it means that  $(k/4)! (4!)^{k/4}$  secondary structures of  $s$  having  $K$  BPS map to the same 4-PARTITION solution. The coefficient  $(k/4)!$  accounts for the permutation over the substrings  $G^B$ , while the coefficient  $4!$  accounts for the permutation between the substrings  $C^{w(a_i)}$  within a same group (i.e. paired to a same substring  $G^B$ ).

Therefore, the number of solutions is multiplied by  $(k/4)! (4!)^{k/4}$  and this reduction is weakly parsimonious. Moreover, it is straightforward to see that #BPS is in #P, as BPS is in NP. Since #4-PARTITION is #P-complete by Lemma 31, #BPS is #P-complete also. ◀

## B.4 Omitted lemmas and proofs from Section 5

► **Lemma 41.** *Let  $S$  be an optimal secondary structure of  $I'$  with  $P$  base pairs. In  $S$ , if a trim-complement strand is bound with a center-trim, it cannot also be bound with another trim.*

**Proof.** By absurd, suppose that  $S$  has a trim-complement strand which is bound with a center-trim but also with another trim. Since all C's are paired in  $S$ , two adjacent bases of this trim-complement strand are paired with bases belonging to distinct domains  $\text{Trim}_1$  and  $\text{Trim}_2$ . Since one of these two domains is a center-trim, they are non-adjacent and are separated by the non-empty sequence  $u$ . Since there is no pseudoknot, bases of  $u$  can only be paired with each other. Without loss of generality, we can consider that  $\text{Trim}_2$  is a middle-trim. Therefore,  $u$  ends with a flank  $x \text{ Sep } y \text{ Sep } z$ . Since all A and T bases are paired in  $S$  and since there is no pseudoknot, two adjacent bases of this flank must be paired with two adjacent bases of  $u$ . Therefore,  $u$  must contain the complementary flank  $\bar{z} \text{ Sep } \bar{y} \text{ Sep } \bar{x}$ , which is absurd since all triples are different by hypothesis and since all xyz-support strands and their complement are distinct by construction. ◀

► **Lemma 42.** *Let  $S$  be an optimal secondary structure of  $I'$  with  $P$  base pairs.  $S$  has exactly  $n$  perfectly bound center-trim domains.*

**Proof.** Let us first show that  $S$  has at least  $n$  perfectly bound center-trim domains. Since there are  $2m + n$  trim-complement strands and every C's are paired in  $S$ , at least  $nE$  bases belonging to center-trim domains are paired. According to the previous lemma, if one base of a center-trim is paired, then the entire domain is paired. Therefore, there are at least  $n$  perfectly bound center-trim domains.

Let us now suppose that there are  $n + i$  bound center-trim domains, with  $1 \leq i \leq m - n$ . They are perfectly bound according to the previous lemma. We call a flank isolated if its neighboring trim-domains are both bound with trim-complementary strands. Let us count the number of isolated flanks. There are  $m - n - i$  center-trim deprived triples. All of them are bound to at most  $2(m - n - i)E$  trim-complement bases. Therefore, there are at least  $[2m + n - 2(m - n - i) - (n + i)] \cdot E = (2n + i)E$  end-trim bases that participate in creating isolated flanks with one of the  $n + i$  bound center-trims. Therefore, there are at least  $(2n + i)$  isolated flanks.

For each isolated flank, consider the set of strands to which it is bound. This set is not bound to the rest of the template domain since there is no pseudoknot. According to Lemma 10 of Condon et al. [5], the total number of support strands is  $10n$ . Since there are at least  $2n + i$  isolated flanks, at least one of them is bound to less than 5 support strands. We can apply Lemma 17 of Condon et al. [5] to this flank, since the ACT-unpairedness is 0 in  $S$ . Therefore, this flank must be bound to exactly 5 support strands. Recursively, the lemma assumptions still hold, and one can continue applying it until one has 0 available support strand but still  $i$  isolated flanks respecting the assumptions, which is absurd. Therefore, we know that at most  $n$  center-trim domains are perfectly bound.

Finally, there are exactly  $n$  perfectly bound center-trim domains. ◀

► **Lemma 43.** *Each optimal secondary structure of instance  $I'$  having  $P$  base pairs is of the following form:  $n$  perfect triples,  $m - n$  triples whose both end-trims are bound to trim-complement strands, and whose 5' and 3' flanks are paired together.*

**Proof.** Let  $S$  be an optimal secondary structure of  $I'$  with  $P$  base pairs.



According to Lemma 42,  $S$  has exactly  $n$  fully connected center-trim domains. The remaining trim-complement strands must necessarily be fully bound with trim domains, as there is no alternative configuration for them.

Following the same reasoning as in the previous proof, we recursively apply Lemma 17 from Condon et al. to the isolated flanks. Consequently, all support strands are bound. Since there are no unpaired A or T bases, each xyz-domain is paired with its complement.

To conclude, we note that the 5' and 3' flanks of trim-deprived triples must be bound together. ◀



# A Coupled Reconfiguration Mechanism That Enables Powerful, Pseudoknot-Robust DNA Strand Displacement Devices with 2-Stranded Inputs

Hope Amber Johnson  

The University of British Columbia, Vancouver, Canada

Anne Condon  

The University of British Columbia, Vancouver, Canada

---

## Abstract

DNA strand displacement, a collective name for certain behaviors of short strands of DNA, has been used to build many interesting molecular devices over the past few decades. Among those devices are general implementation schemes for Chemical Reaction Networks, suggesting a place in an abstraction hierarchy for complex molecular programming. However, the possibilities of DNA strand displacement are far from fully explored. On a theoretical level, most DNA strand displacement systems are built out of a few simple motifs, with the space of possible motifs otherwise unexplored. On a practical level, the desire for general, large-scale DNA strand displacement systems is not fulfilled. Those systems that are scalable are not general, and those that are general don't scale up well.

We have recently been exploring the space of possibilities for DNA strand displacement systems where all input complexes are made out of at most two strands of DNA. As a test case, we've had an open question of whether such systems can implement general Chemical Reaction Networks, in a way that has a certain set of other desirable properties – reversible, systematic,  $O(1)$  toeholds, bimolecular reactions, and correct according to CRN bisimulation – that the state-of-the-art implementations with more than 2-stranded inputs have. Until now we've had a few results that have all but one of those desirable properties, including one based on a novel mechanism we called coupled reconfiguration, but that depended on the physically questionable assumption that pseudoknots cannot occur. We wondered whether the same type of mechanism could be done in a pseudoknot-robust way.

In this work we show that in fact, coupled reconfiguration can be done in a pseudoknot-robust way, and this mechanism can implement general Chemical Reaction Networks with all inputs being single strands of DNA. Going further, the same motifs used in this mechanism can implement stacks and surface-based bimolecular reactions. Those have been previously studied as part of polymer extensions of the Chemical Reaction Network model, and on an abstract model level, the resulting extensions are Turing-complete in ways the base Chemical Reaction Network model is not. Our mechanisms are significantly different from previously tested DNA strand displacement systems, which raises questions about their ability to be implemented experimentally, but we have some reasons to believe the challenges are solvable. So we present the pseudoknot-robust coupled reconfiguration mechanism and its use for general Chemical Reaction Network implementations; we present the extensions of the mechanism to stack and surface reactions; and we discuss the possible obstacles and solutions to experimental implementation, as well as the theoretical implications of this mechanism.

**2012 ACM Subject Classification** Theory of computation; Applied computing → Chemistry

**Keywords and phrases** Molecular programming, DNA strand displacement, Chemical Reaction Networks

**Digital Object Identifier** 10.4230/LIPIcs.DNA.31.2

**Funding** *Hope Amber Johnson*: Supported by an NSERC Discovery Grant.

*Anne Condon*: Supported by an NSERC Discovery Grant.



© Hope Amber Johnson and Anne Condon;  
licensed under Creative Commons License CC-BY 4.0

31st International Conference on DNA Computing and Molecular Programming (DNA 31).

Editors: Josie Schaeffer and Fei Zhang; Article No. 2; pp. 2:1–2:23

Leibniz International Proceedings in Informatics



LIPICs Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

## 1 Introduction

One of the more ambitious goals of molecular programming is to come up with a general framework to write programs for biocompatible molecules, possibly up to the complexity of modern silicon computers. Such a framework would likely involve an abstraction hierarchy, where human-understandable programs are written at higher levels and translated through the layers, eventually producing a set of molecules that physically execute the program. Over the past decades, one candidate pair of layers for such a hierarchy has emerged: the Chemical Reaction Network as an abstract model, and DNA strand displacement as a more physical description of molecules.

The Chemical Reaction Network (CRN) model describes abstract chemical species and the arbitrarily specified reactions they undergo. The stochastic CRN model, in particular, can compute semilinear functions in an “always eventually always” errorless sense [1, 2, 7, 13, 22, 27], and can simulate Turing machines with arbitrarily small probability of error [29]. Various extensions of the model to include polymers have been proposed, which can do Turing-universal computation much more robustly [6, 10, 19, 20, 24, 26, 32]. So a reliable and scalable general CRN implementation would be a good candidate for a general-purpose molecular computer, and an implementation of one of the polymer extensions even more so.

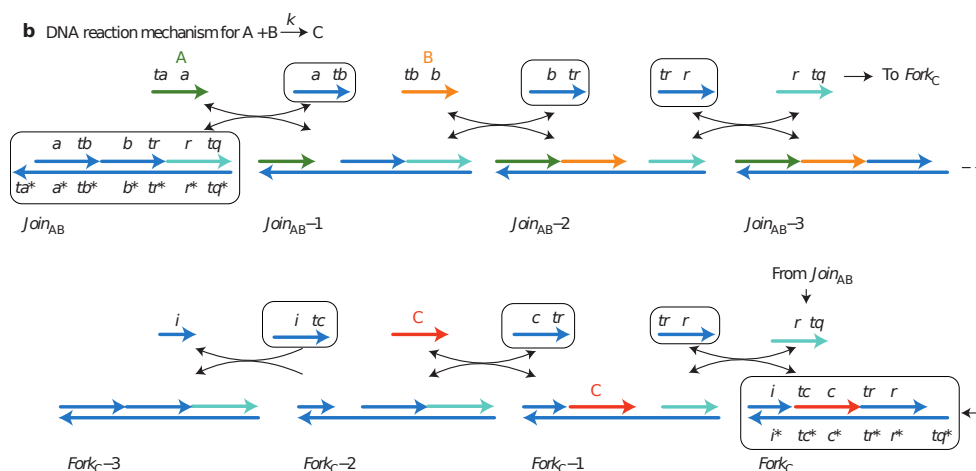
DNA strand displacement (DSD) refers to certain behaviors of DNA, involving parts of strands displacing bonds between other parts, that have been used to build many useful molecular devices [28]. Most such devices have been built out of simple toehold exchange motifs, which in Section 2 we call the  $tx_3$  and  $tx_4$  motifs. To help analyze these systems, reaction enumerators [3, 21] and formal models [16, 23] have been developed, turning DSD from a description of experimental reality into a formal system.

Many general implementation schemes for arbitrary CRNs have been designed in DSD, with various relative advantages and disadvantages [5, 8, 18, 24, 30, 31]. Figure 1 gives an example. Such a system will generally involve *signal* strands, representing the formal species of the CRN, and *fuel* complexes, assumed “always present” and consumed to drive the CRN’s reactions. While some have been implemented experimentally [8, 31], even 3 species and 3 reactions in a formal CRN is enough to have problems scaling up robustly [31]. There have been attempts to address this, including a design that would be “leakless” on a theoretical level [33, 35]. However, these have not yet led to effective scalable implementations.

We have recently been trying to explore the space of possibilities in DSD, beyond the  $tx_3$  and  $tx_4$  motifs that make up most existing DSD systems. This is partly out of theoretical interest, and partly from a more practical motivation: CRN implementation schemes use fuels with at least 3 strands, and have issues scaling up [31]. Meanwhile seesaw gates and some other systems, though they can’t implement arbitrary CRNs, have been made with 2-stranded inputs and scaled up well to larger systems [9, 25, 34]. Therefore we investigated whether arbitrary CRN implementations could be made with 2-stranded inputs (i.e., fuels and signals), with an additional set of desirable conditions: reversible, systematic implementation, uses  $O(1)$  orthogonal toehold domains, correct according to modular CRN bisimulation, and built out of bimolecular reactions [16]. We found two systems that can implement arbitrary CRNs, but each with one undesirable condition: one uses  $O(n)$  toehold domains, while one requires trimolecular reactions [18]. Both of these conditions would cause issues with experimental implementations. This was not satisfying, so we kept searching.

Existing DSD implementations often implement  $A + B \rightarrow \dots$  logic in a way that requires a fuel with at least 3 strands: one strand to bind the reactants  $A$  and  $B$ , one strand to be released in the reaction that binds  $A$ , and one to be released in the reaction that binds  $B$  [30, 31]. The mechanism shown in Fig. 1, for example, represents the binding of  $A$  by





■ **Figure 1** An example CRN implementation with DSD [8]. The reaction  $A + B \rightarrow C$  is implemented using two 4-stranded fuels.

opening a shared toehold that allows the binding of  $B$ , but this required kicking off an  $A$ -specific strand. Recently we found a mechanism we called *coupled reconfiguration*, where instead of binding one strand and releasing another, two strands can reconfigure each other and split apart. Thus the  $A$  signal strand can reconfigure into an “off” state, and the fuel strand can reconfigure into a “bound  $A$ ” state, allowing us to implement arbitrary CRNs with 1-stranded inputs [15]. However, our mechanism depended on a common assumption in DSD models that certain structures called *pseudoknots* don’t form, and would have undesired pathways if pseudoknots could occur. Specifically, it represented the binding of  $A$  by changing which of an  $A_1$  or  $A_2$  domain was bound to its complement, such that the interaction with  $B$  would be non-pseudoknotted only in the “bound  $A$ ” state. While our mechanism was great on paper in the pseudoknot-free model, that model is based not on the belief that pseudoknots actually don’t occur in reality, but the belief that they’re too complex and we don’t want to deal with them. If implemented in reality, it’s likely that the mechanism would simply fail due to pseudoknotted undesired pathways. We again found this unsatisfying, and wondered if coupled reconfiguration could be done without depending on the no-pseudoknots assumption.

Here in Section 3 we present a mechanism we call  $CR_4$ , based on 4-way branch migration, that does coupled reconfiguration without relying on the no-pseudoknots assumption. In particular, the  $CR_4$  mechanism changes which toeholds are open and closed, meaning we can compose  $A$  and  $B$  with the same shared toehold logic used in Fig. 1 and most similar systems; but as a coupled reconfiguration mechanism, it does so without requiring an additional strand. In Section 4 we show how this mechanism can implement arbitrary CRNs with 1-stranded inputs, with all the other desirable conditions as well. The  $CR_4$  mechanism is based on a  $2tx_4$  motif, which turns out to be useful for implementing polymer CRNs, and in Section 5 we give mechanisms for stack push and pop reactions and for bimolecular surface CRN reactions, all with 1- and 2-stranded inputs. These mechanisms could be slotted in to the existing theory for stacks [24] and surface CRNs [26] without changing the higher abstraction layers, if desired. Our mechanisms are sufficiently far from existing experimentally tested mechanisms that there will likely be challenges implementing them, but we suspect it would be possible. In Section 6 we give some speculation on what challenges may arise and how to overcome them, as well as further theoretical implications.

## 2 Formal DSD, motifs, and correctness

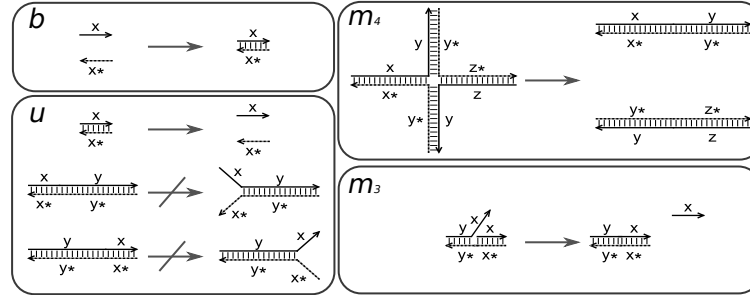
We work with a formal model of DNA strand displacement systems, and are trying to explore the limits of that model. The more abstract Chemical Reaction Network model is both an important use case for DSD systems and a good way to test the computational power of a class of DSD systems. In this section we give some definitions of Chemical Reaction Networks, DSD systems, and the definition of a correct implementation that we use. Some more formal definitions were omitted for space, and can be found in Appendix A.

► **Definition 1.** A Chemical Reaction Network is given by a finite set of (abstract) species, with a finite set of specified reactions between those species. Reactions are pairs  $(R, P)$ , where  $R$  and  $P$  are each a multiset of species. The states of a CRN are then all multisets over the set of species, and transition between each other according to the reactions.

Often reactions will be given as a triple  $(R, P, k)$ , written  $R \xrightarrow{k} P$ , where  $k$  is a rate constant. For our purpose we care only what can happen, not how fast it can happen, so we use only  $R \rightarrow P$ . The reversible reaction  $R \rightleftharpoons P$  means  $R \rightarrow P$  and  $P \rightarrow R$ .

► **Definition 2.** A strand is a sequence of domains, listed from “5’ end” to “3’ end”. Here a domain is simply one of some finite set of domains, such that for each domain  $x$  there is a complement  $x^*$ , where  $(x^*)^* = x$ . In general and in our work  $x^* \neq x$ , but this is not a requirement. Each  $x$  is defined to be either long or short, with  $x^*$  short if and only if  $x$  is. We often refer to short domains as toeholds.

A complex is a finite multiset of strands with a list of bonds between pairs of their domains, such that each domain  $x$  is either unbound or bound to exactly one  $x^*$ , and the resulting graph is connected.



■ **Figure 2** The four basic steps that define our DSD model. Modified from [16].

► **Definition 3.** The anchored, pseudoknot-allowed DSD model is the following four basic steps, illustrated in Figure 2:

- Binding ( $b$ ): Two unbound domains  $x$  and  $x^*$  can bind.
- Toehold unbinding ( $u$ ): Two domains  $(x, x^*)$  bound to each other can unbind if  $x$  is short and the bond  $(x, x^*)$  is not anchored by an adjacent bond as shown in Figure 2.
- Three-way branch migration ( $m_3$ ): Given three domains, one unbound instance of  $x$  and a bound pair  $(x, x^*)$ , the two instances of  $x$  can exchange which is bound to the  $x^*$ , if the resulting bond is anchored by an adjacent bond as shown in Figure 2.
- Four-way branch migration ( $m_4$ ): Given four total domains, two bound pairs of  $(x, x^*)$  for the same  $x$ , the two pairs can exchange which  $x$  is bound to which  $x^*$ . This can happen only if both of the resulting bonds are anchored by adjacent bonds as shown in Figure 2.

Basic steps are meant to be applied to complexes, producing reactions (also called steps) whose reactants and products are complexes. A step between complexes is *reversible* if its reverse is also one of the steps.  $b$  is reversible if the resulting bond meets the conditions for  $u$  (short domain and not anchored);  $u$  is always reversible by  $b$ ;  $m_3$  is reversible by  $m_3$  if the initial  $(x, x^*)$  bond was anchored; and  $m_4$  similarly to  $m_3$ .

Often when working with DSD systems we want to require that all complexes involved are *non-pseudoknotted*. This greatly simplifies analysis without too much loss in potential, as pseudoknots are not well understood and thus not reliable for use in engineered systems.

► **Definition 4.** A complex is non-pseudoknotted if there is some ordering of the strands such that, for any bonds  $(x, x^*)$  and  $(y, y^*)$ , the  $x$  and  $x^*$  are either both between  $y$  and  $y^*$  or both outside them. If such an ordering exists, it is unique up to circular permutation [12].

The anchored, pseudoknot-free DSD model is the anchored, pseudoknot-allowed model with the following modification: the  $b$  step can only happen if the resulting complex is non-pseudoknotted. (The  $u$ ,  $m_3$ , and  $m_4$  steps, by their construction, cannot make a non-pseudoknotted complex pseudoknotted.) This model is equivalent to Peppercorn’s model with the `--reject-remote` option [3].

► **Definition 5.** A DSD system is a DSD model and a set of initial complexes. The enumerated CRN of a DSD system is the (smallest) CRN where every initial complex is a species, every basic step applicable to species is a reaction, and the products of every such step are species.

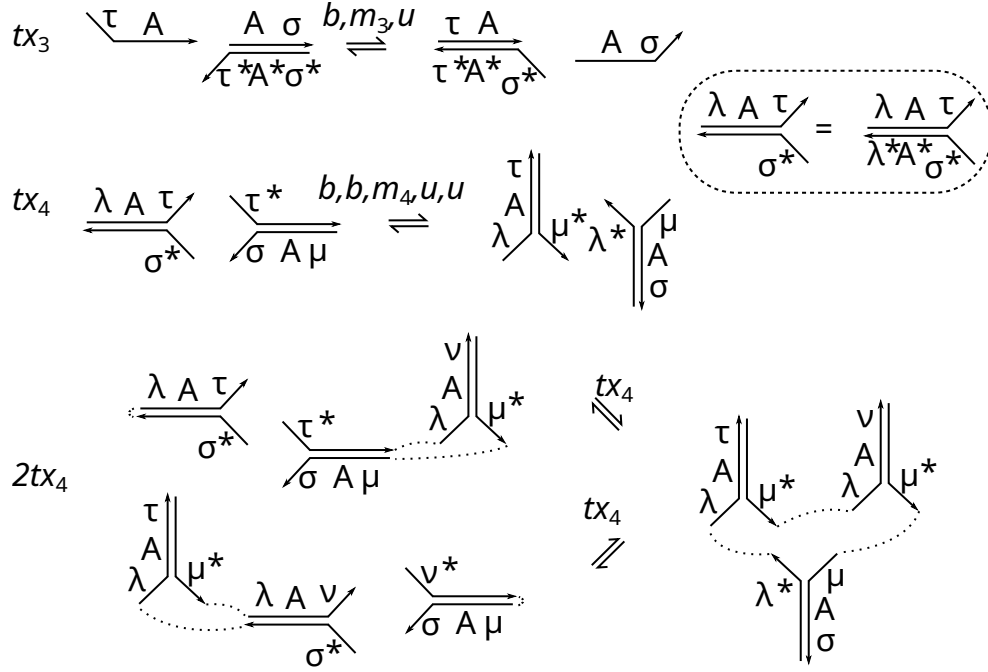
The process of generating the enumerated CRN is often called *reaction enumeration*, and can be done automatically by tools such as Visual DSD [21] and Peppercorn [3].

While many sequences of steps are in theory possible, the designs in this paper are built out of the same sequence of steps used repeatedly. We often call these *motifs* [18]. Figure 3 shows the  $tx_4$  (toehold exchange with 4-way branch migration) and  $2tx_4$  motifs we use.

Following models such as Peppercorn, we often assume that once two complexes combine and start interacting, no third complex can bind to the result until it’s finished all meaningful unimolecular steps. This is based on a low-concentration assumption where bimolecular steps are slow relative to unimolecular steps, making that assumption realistic. Peppercorn calls this a “condensed reaction”, and gives a proper definition [3]. We use this to implicitly exclude certain complicated and unlikely undesired pathways, but don’t mention it much.

When a DSD system is made to implement CRNs, its initial complexes will generally be made of *signals* and *fuels*. Signals represent the formal species of the CRN, while fuels are assumed to be “always present” (left as an exercise to the experimentalist) so they can drive the reactions they’re meant to enable. Once an implementation CRN is enumerated from the DSD system, the fuels are removed (so e.g.  $A + f \rightarrow B$  becomes  $A \rightarrow B$  for  $f$  a fuel), and we consider an implementation correct if the result meets the conditions of CRN bisimulation [17]. These are, approximately, “anything that can happen, should” and “anything that should happen, can”, where “can” refers to the DSD implementation and “should” to the formal CRN. When we discuss the number of strands in the *inputs* of a CRN implementation, we mean the signals and fuels. Experimentally, these are the complexes that have to be created externally, one way or another, to make the system run.

One concept we wish to explore here is *pseudoknot-robustness*, defined in terms of the correctness of CRN implementations. Most DSD systems don’t involve pseudoknots, and many analysis algorithms [3, 4, 12] assume they can’t happen. However, that assumption is based less on a belief that they’re actually physically impossible, and more on a desire not to deal with them. We are therefore interested in systems that are correct whether pseudoknots are possible or not. Formally, this is defined by checking the “anything that can happen,



■ **Figure 3** The  $tx_3$  and  $tx_4$  motifs, and further abstractions. The mechanisms in this paper are built on the sequence of two  $tx_4$  motifs shown, so we give an abstracted notation that is useful to present them. The  $tx_3$  motif used in most existing DSD systems is presented for comparison, but we won't use it further. Our typical use of the  $2tx_4$  motif involves single strands and 2-stranded complexes, where strand connections (dotted lines) make the intermediate state a closed loop as shown. This gives the forks in question nothing to do other than reversing or completing the motif.

including pseudoknots, should" condition in the anchored, pseudoknot-allowed model, and checking the "anything that should happen, can even without using pseudoknots" condition in the anchored, pseudoknot-free model.

### 3 The four-way coupled reconfiguration mechanism

Our (first) goal with this mechanism is to implement generic  $A + B \rightleftharpoons C + D$  logic, which is sufficient to implement arbitrary CRNs. However, the method we use is built out of smaller parts, namely the four-way coupled reconfiguration ( $CR_4$ ) mechanism; and is best explained by first defining the  $CR_4$  mechanism and how that mechanism can be abstracted into inputs and outputs, then showing how the general  $A + B \rightleftharpoons C + D$  implementation is built out of the abstracted  $CR_4$  mechanism. So Fig. 4 shows the configurations that a  $CR_4$  mechanism reconfigures between, and how the toeholds serve as abstractable inputs and outputs; Fig. 5 shows the actual  $CR_4$  mechanism; and Fig. 6 shows how two  $CR_4$  mechanisms can be connected by a shared toehold. Ultimately in Fig. 7 we end up implementing  $A + B \rightleftharpoons C + D$  as  $A + B \rightleftharpoons int_r$ ,  $C + D \rightleftharpoons int_r$  for  $int_r$  an intermediate specific to this reaction; this is the same strategy implemented in e.g. the  $tx_3$ -based CRN implementation [5, 8] in Fig. 1.

In our previous work, we designed a  $tx_3$ -motif-based mechanism where two strands come together, reconfigure each other, and split apart; such that neither strand can reconfigure on its own, and when the two strands come together, they can split apart only if both reconfigure or neither does [15]. We named this property *coupled reconfiguration*, and it allowed us to

design arbitrary CRN implementations using only 1-stranded inputs, as opposed to 3- or more-stranded complexes. To our knowledge, this was the first DSD mechanism designed with this property; others had relied on different combinations of strands to get different configurations. However, it was very reliant on the assumption that pseudoknots couldn't happen, with the strands able to reconfigure on their own via pathways involving pseudoknots. This makes the mechanism theoretically interesting, but likely not to work in reality, and thus unsatisfactory. So the core of the result of this paper is a  $CR_4$  mechanism, based on the  $2tx_4$  motif, that does coupled reconfiguration in a pseudoknot-robust way.

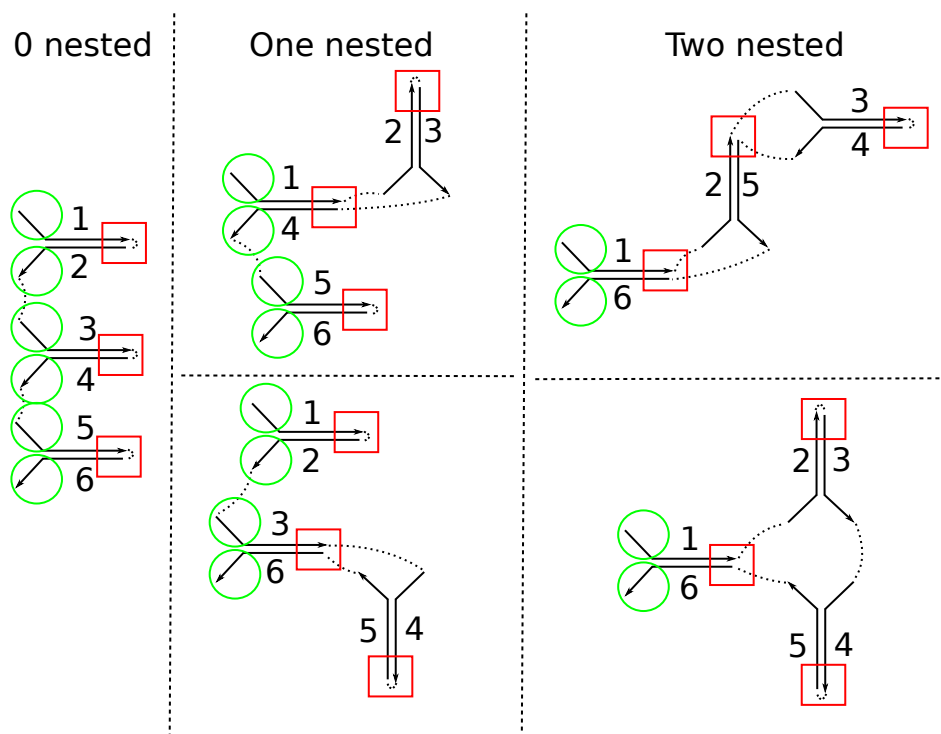
Within the pseudoknot-robust anchored model, there are two ways we're aware of for one  $tx_3$  or  $tx_4$  motif to depend on a previous one. (That is, for two motifs in a sequence to be meaningfully in that order, as opposed to two independent motifs that just happened to happen in that order.) First, specific to  $tx_4$  motifs, one ( $tx_3$  or  $tx_4$ ) motif can put together the combination of two toeholds necessary for a (second)  $tx_4$  motif to happen. For example, in the  $2tx_4$  motif in Figure 3, the first  $tx_4$  motif puts together the  $(\mu, \lambda^*)$  combination necessary for the second to happen. This generally involves both motifs acting on the same long domains, so it's good for creating dependencies within the same long domain identity, but not between independent long domain identities. In the  $CR_4$  mechanism in Figure 5, all the dependencies between the three  $2tx_4$  motifs are of this type.

Second, the two motifs can have a shared toehold. By “shared toehold” here we mean a toehold flanked by two long domains, where a motif acting on one long domain opens the toehold, allowing it to be used in a motif acting on the other long domain. The opening of toeholds in most previous DSD implementations is of this type, as in Figure 1.

The  $CR_4$ -based CRN implementation will use both. Within a single long domain identity associated with a given formal species, a specific arrangement of toehold identities allows a sequence of  $2tx_4$  motifs between a “gate” strand and a “signal” strand in an “on” configuration, after which the signal strand is in the “off” configuration and the gate strand has also reconfigured. The arrangement of toeholds ensures that at any point in the process there are at most two reactions available, the intended forward and reverse reactions, even if pseudoknots are allowed. The gate strand will have different toeholds open and closed before and after the reconfiguration, which allows composition if any of those toeholds is shared with another copy of the  $CR_4$  mechanism. Opening and closing shared toeholds with coupled reconfiguration means we don't need the extra strands that most DSD CRN implementations use. While we use 7 (or 6, or possibly less with optimization) orthogonal toehold identities, we can reuse those same 6-7 sequences throughout, which is what we mean by  $O(1)$  toeholds.

For one motif to enable another motif via a shared toehold, the first motif should transform the toehold from “unavailable” to “available”, from a state where it can't do the second motif to a state where it can. In the pseudoknot-robust paradigm, “can” is measured in the pseudoknot-free model, so a toehold is available if it is both unbound and not nested inside another bond. But “can't” is measured in the model with pseudoknots allowed, so a toehold is unavailable if it is bound. A toehold that is unbound but nested inside another bond can't be pseudoknot-robustly said to be either available or unavailable.

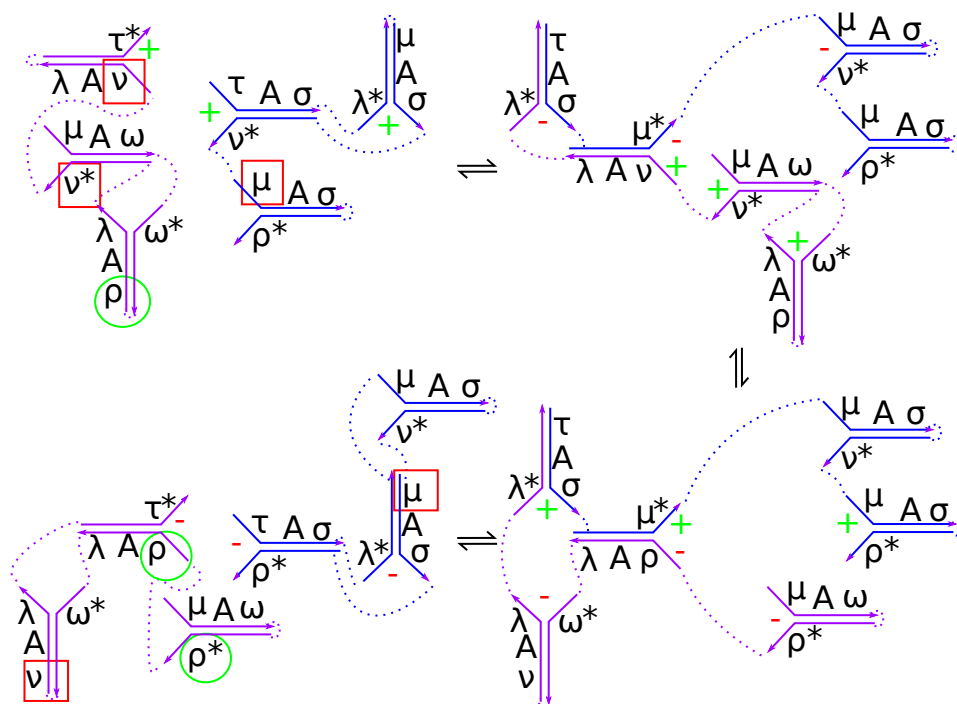
Our mechanisms involve various  $2tx_4$  motifs with the same long domain identity eventually opening up a shared toehold, which will then enable more motifs with a different long domain identity. Figure 4 shows, for 3 pairs of alternating long domains  $A$  and  $A^*$ , the possible configurations, and the availabilities of their toeholds. The “one nested” configurations are particularly useful. A strand that switches between them will have the toehold pair between domains 2 and 3 switch availabilities in one direction, while the pair between 4 and 5 switches in the other. We use 3 pairs because 2 alternating copies of a long domain and its complement have only two configurations, and not enough toeholds that change availability.



■ **Figure 4** Non-pseudoknotted configurations of a strand with 6 long domains alternating between  $A$  and  $A^*$ , each flanked by toeholds with open toeholds facing “out”. For this purpose we number the domains 1 through 6, odd numbers being  $A$ , even numbers  $A^*$ . Of the 6 possible pairings between two sets of 3 elements each, one of them (1-4, 3-6, 2-5) is pseudoknotted; the other five are shown. Note in each configuration which toeholds are both unbound and not nested (green circles), and which are bound (red squares). The  $CR_4$  mechanism will cause each of its strands to switch between two of these configurations, and its meaningful effect is to turn some toeholds “on” and others “off”.

We can then describe the  $CR_4$  mechanism, shown in Figure 5. The mechanism involves two strands, one (“left strand”, purple) reconfiguring between the two one-nested configurations in Figure 4, one (“right strand”, blue) reconfiguring between a one-nested configuration and the linear two-nested configuration. So while the left strand has a pair of toeholds that changes from available to unavailable and a pair that does the reverse, the right strand only has a single toehold that changes from available to unavailable and none that do the reverse. There are multiple regions, indicated by dotted lines, that can contain any sequence without affecting the mechanism unless that sequence somehow interacts with the domains involved in the mechanism. This will be particularly useful for composing copies of the  $CR_4$  mechanism.

The  $CR_4$  mechanism as presented is physically reversible, with each step as well as the overall mechanism being fully reversible in the model. It can be made irreversible in either direction by removing a single toehold. Removing either of the two  $\omega^*$  toeholds on the left strand will make the reaction irreversible in the direction towards which that  $\omega^*$  ends up unbound. (Removing the  $\rho^*$  or  $\nu^*$  toehold can also do so, but those are used for composition.) Similarly on the right strand, in each direction one of the three  $\sigma$  toeholds changes from bound to unbound, and removing it will make the reaction irreversible in that direction.



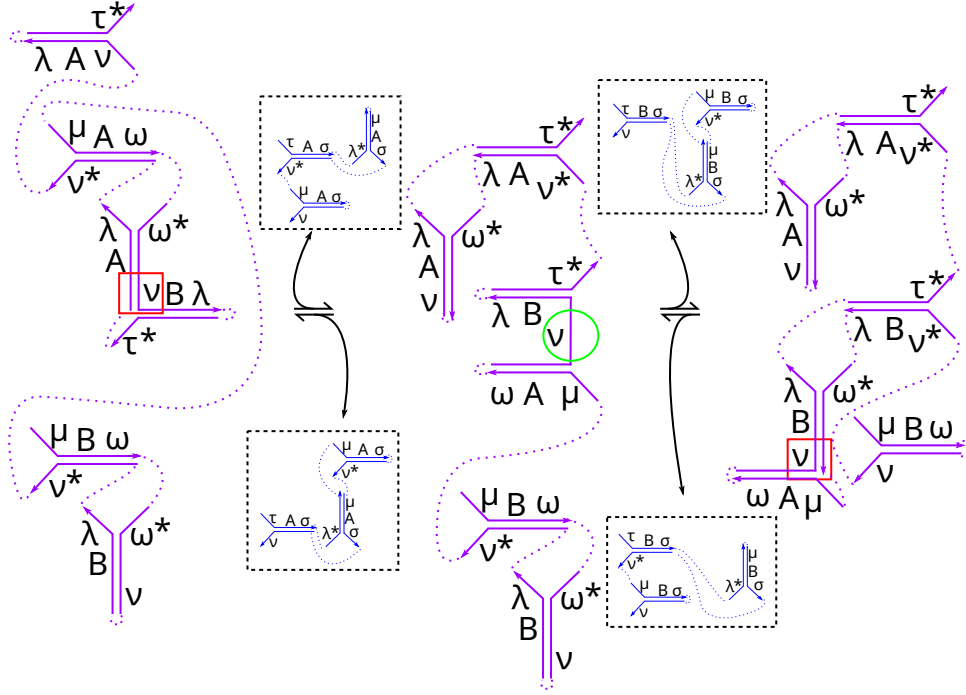
■ **Figure 5** The pseudoknot-robust coupled reconfiguration  $CR_4$  mechanism. Each step is a  $2tx_4$  motif as in Figure 3. Dotted lines indicate strand connections that can contain arbitrary sequences; note that the reaction as drawn starts and ends with two single strands. + and - indicate forks that interact in the forward and reverse directions, respectively. Green circles indicate toeholds that are available at the end and bound at the beginning, for composition with other reactions. Red squares do the same for the opposite direction. Note that only one of the two strands has both.

#### 4 Implementing CRNs with four-way coupled reconfiguration

To implement CRNs, we first need to figure out how to compose modules of the  $CR_4$  mechanism. Ideally, we would like what we call a *systematic*,  $O(1)$  *toeholds* implementation, which we defined in our previous work [16]. This means that the “signal strands” for formal species e.g.  $A$  and  $B$  have the same structure and the same toehold domains, differing only in the identity of their long domain(s), which should be species-specific. To get interaction between independent long domains we want to use a shared toehold, where one of the toeholds that changes availability in one mechanism is the same as one of the ones that changes availability in the other. Given the properties of the mechanism, we need to use the left strand for this, and equate the  $\rho^*$  of one mechanism with the  $\nu$  of the other. So while Figure 5 had  $\rho$  be a unique toehold, for the CRN implementation  $\rho = \nu^*$  and  $\rho^* = \nu$ ; the  $CR_4$  mechanism still works with this replacement. The composition is shown in Figure 6.

Now that we can connect multiple  $CR_4$  modules, we can use this to make CRN implementations. Similar to our previous mechanism [15], we use left strands as gates and right strands as species strands, where one of the two configurations of a species strand is the “on” state and the other is the “off” state. The on state of a species strand represents the actual species, while the off state is a fuel that represents nothing, unless something turns it on. Then a gate strand will in sequence turn each reactant species off then turn each product species on, each time reconfiguring itself in a way that enables the next interaction.



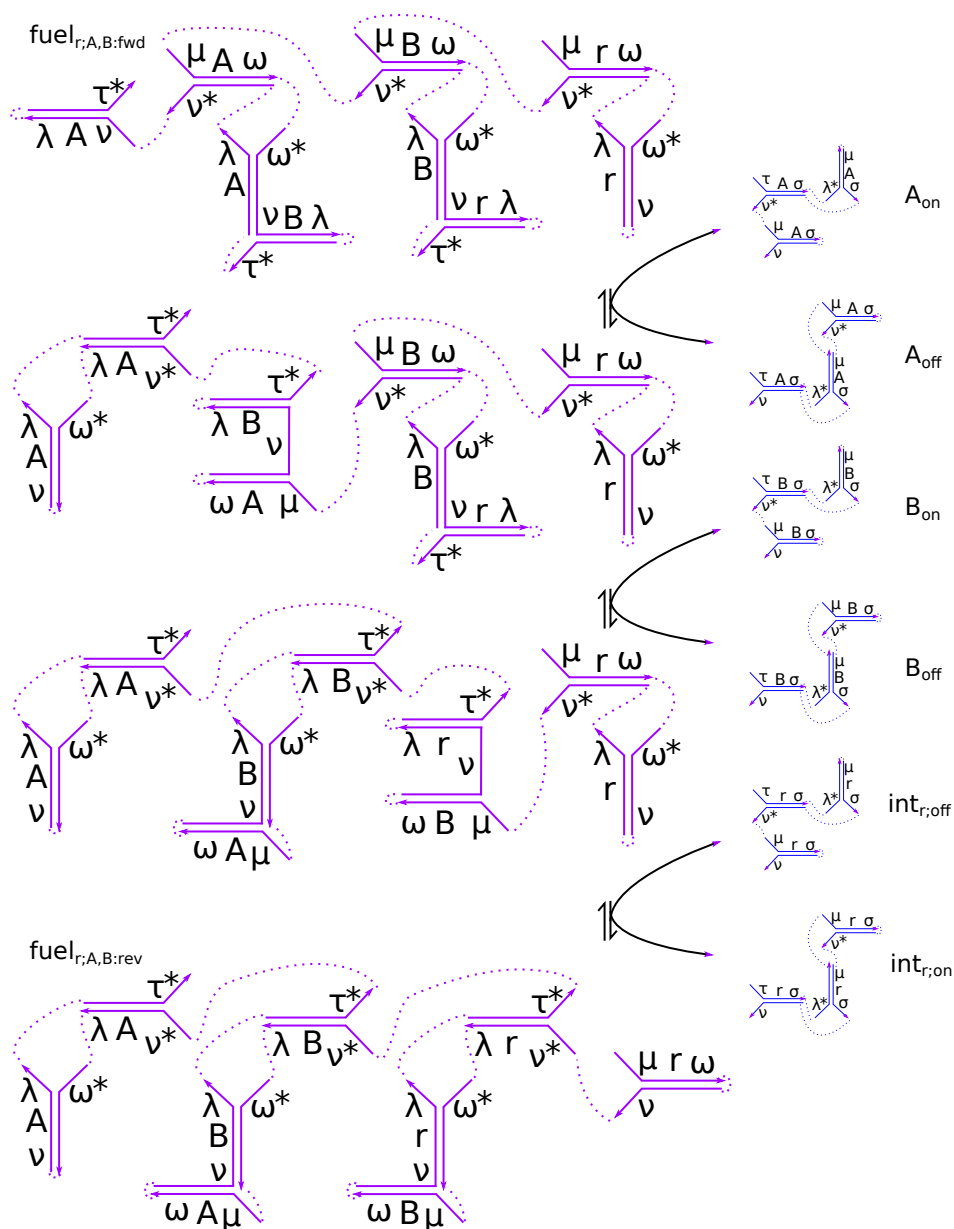


■ **Figure 6** A strand (purple) with two copies of the coupled reconfiguration mechanism from Figure 5 connected by a shared toehold. This strand with the shared toehold open (green circle) can interact with the counterpart (blue) of either mechanism, but only with one of them; either reaction will close off the toehold needed for the other (red squares). The toehold  $\rho$  in Fig. 5 is given the sequence of  $\nu^*$  to make this connection work.

In contrast to the previous mechanism, this one has a polarity issue: one could not, for example, equate the  $\rho$  of one  $CR_4$  mechanism in Figure 5 with the  $\nu$  of another, because the  $\rho$  is 5' of a long domain in its mechanism while the  $\nu$  is 5' of a different long domain in its mechanism. Nor can one connect a  $CR_4$  mechanism that turns a  $B$  off with one that turns a  $C$  on, if signals  $B$  and  $C$  are from the same template. So instead of implementing  $A + B \rightleftharpoons C$  in one gate, we implement  $A + B \rightleftharpoons int_r$ . Here  $int_r$  is an intermediate, specific to reaction  $r$ , and while the species strands use the one-nested state as on and the linearly nested state as off,  $int_r$  uses one-nested as off and linearly nested as on. Figure 7 shows this gate. The intermediate strand would then be shared with a second gate for the products, so  $A + B \rightleftharpoons C + D$  would be implemented as  $A + B \rightleftharpoons int_r$  and  $C + D \rightleftharpoons int_r$ . For irreversible reactions,  $\omega^*$  toeholds should be removed from the gates to make the  $int_r$  module of the reactant gate, and all modules of the product gate, irreversible as described in Section 3.

We coded this implementation as a translation scheme in the Nuskell language [4], and ran it in that program to check whether it works. In fact it does on two simple CRNs we tested. At least, it works in the pseudoknot-free model; Nuskell does not calculate pseudoknots, and thus cannot check whether our system is pseudoknot-robust. In general, our argument for correctness is that the same combination of two open toeholds and a long domain does not appear in our system except in the desired reactions, namely the forward and reverse reactions at any given step. Thus no  $m_4$  step can happen except for the intended  $tx_4$  motifs. This can be observed from the figures.

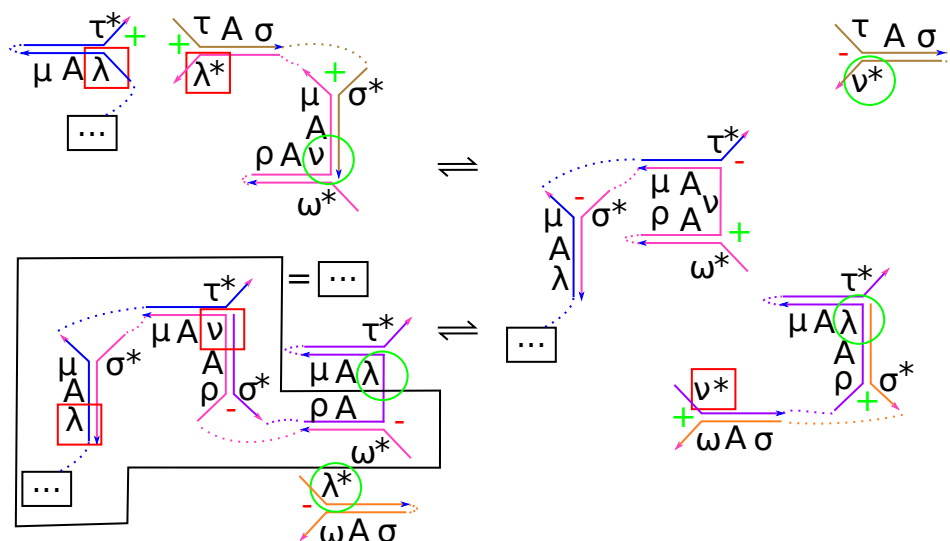




**Figure 7** An implementation of a CRN reaction using shared toeholds as shown in Figures 5 and 6. This figure shows the reactants half of  $A + B \rightleftharpoons \dots$ ; for a reversible reaction  $A + B \rightleftharpoons C + D$ , the products half is an exact copy of the mechanism with  $A$  and  $B$  replaced by  $C$  and  $D$ . A single gate strand (purple) reconfigures the reactant strands (blue) to their “off” states, then converts an intermediate with a reaction-specific long domain  $r$  to its “on” state. That intermediate can then interact with the product gate.

## 5 Polymer extensions: stacks and surface CRNs

While well-mixed CRNs can do many interesting tasks, they are provably less powerful than classical computers [2, 22, 27, 29]. Adding some form of geometry, such as unbounded polymerization or a surface grid, makes up the difference in computational power. To that end, there have been designs for DSD implementations of polymer CRN-like systems, such as stack machines [24] and surface CRNs [10, 26], and some general polymer models [6, 19].

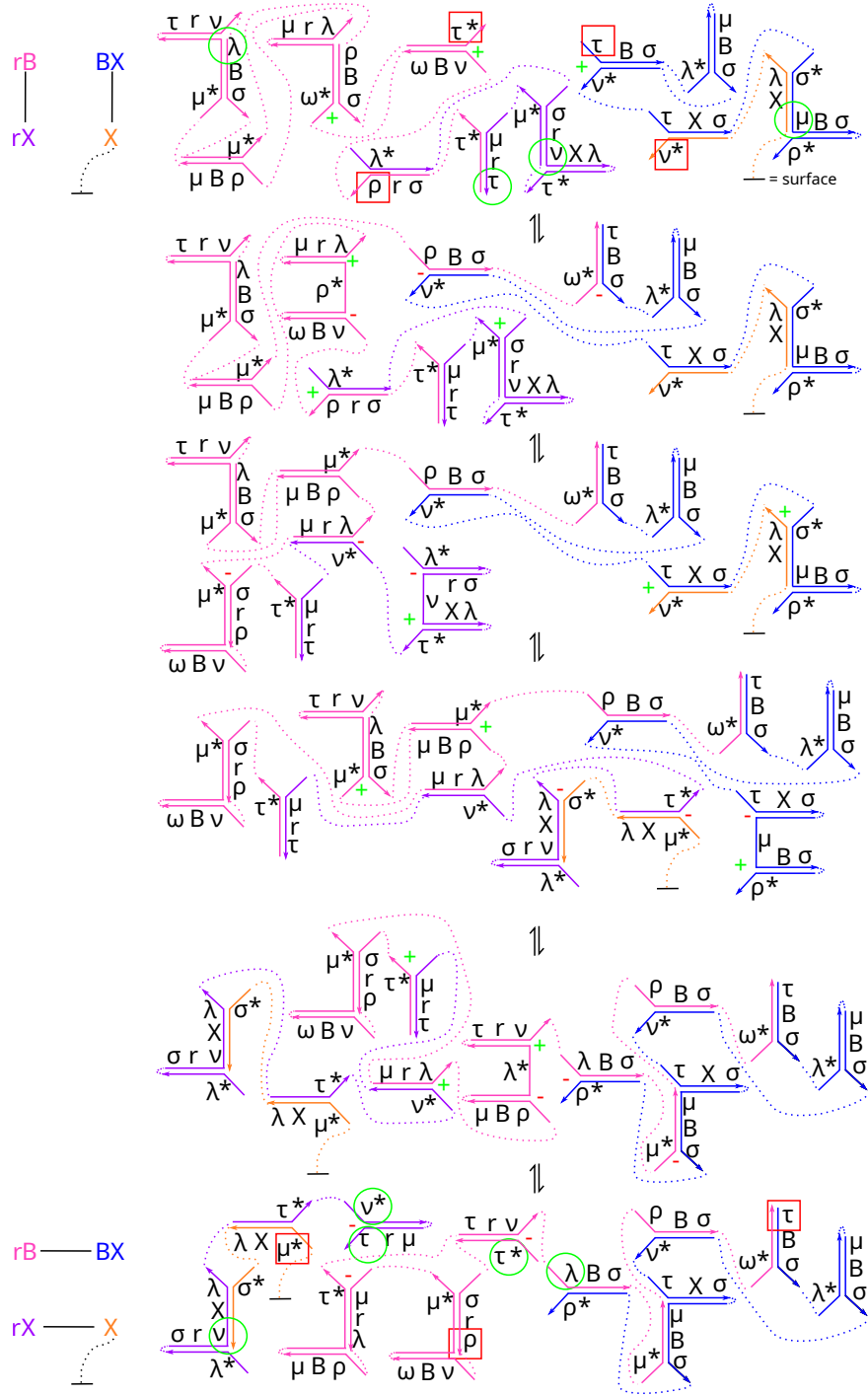


■ **Figure 8** An implementation of stacks (reversible push and pop reactions) using  $2tx_4$  motifs. The ... box represents the rest of the stack; note that the end state has the same pattern as the start state, with an additional unit on the stack, drawn enclosed in the larger box. All four non-stack complexes have toeholds composable with the previous CRN mechanism (red squares, green circles).

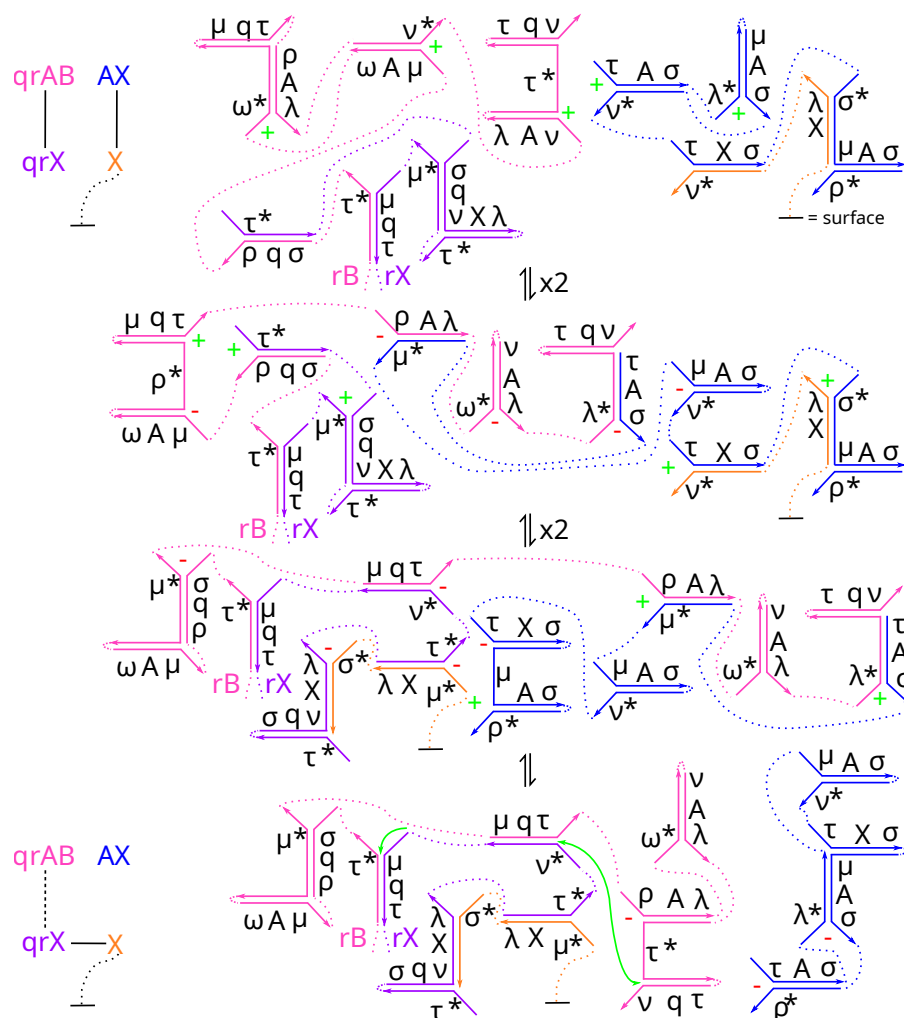
It turns out that the  $2tx_4$  motif is convenient for polymerization mechanisms: of the three sequences of 2 long domains (plus flanking toeholds) involved, the motif takes one of them from bound to the second to bound to the third. This can be used to add or remove strands from a polymer. For example, Figure 8 shows a mechanism for adding an additional unit to a linear stack polymer. If necessary this mechanism can be made irreversible by removing appropriate  $\sigma^*$  toeholds, but this is not usually desirable for stack machines. It is worth noting that this mechanism is not, technically, a coupled reconfiguration mechanism, since there is no strand or complex that has multiple configurations for the same set of strands.

The four non-stack complexes could be considered fuels, but if instead their presence is controlled by some other system, they can control whether or what gets added to the stack. For example, if their composition toeholds are shared with a species strand from the well-mixed CRN mechanism, that would make a CRN augmented by controllable stacks. Note that the toehold identities here have no correlation to those in Figure 7, so they can be renamed as necessary to be compatible with that mechanism. The stack machine implementation from [24] is, on the formal level, a CRN augmented by controllable stacks, and their implementation could be replaced by ours with no change to the formal augmented CRN. Thus our mechanism can make stack machines with only 2-stranded input complexes.

Another type of polymer system is surface CRNs, given in the same notation as CRNs but where the species are understood to occupy points on a grid [26]. Then a reaction  $A + B \rightarrow C + D$  means that an  $A$  and  $B$  at adjacent points to each other interact and become a  $C$  and  $D$  in the respective same points on the surface. Implementing this generally requires a “replacement-in-place” mechanism, where a species strand bound to a generic base strand is identified by an incoming fuel, and then replaced on the same base strand by a different strand. Though complex, this is possible with the  $2tx_4$  motif, as shown in Figure 9. Specifically, that figure shows the  $B \rightleftharpoons I_r$  half of a  $B \rightleftharpoons C$  reaction; the  $C \rightleftharpoons I_r$  half is symmetric. Multiple involved strands have toeholds suitable for composition, so the replacement could be controlled by e.g. an  $A$  species in a well-mixed CRN.



■ **Figure 9** A replacement-in-place mechanism that can be used for surface CRN implementations.  $r$  is a reaction-specific domain,  $X$  is universal. The goal is to replace what's bound to the base strand, connected to some surface. Shared toeholds allow interaction between  $B$ ,  $r$ , and  $X$  domains.



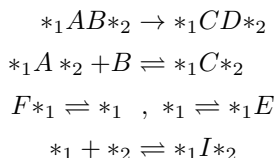
■ **Figure 10** A replacement-in-place mechanism for the first reactant of a bimolecular surface CRN reaction. This mechanism can be best understood as a combination of those from Figures 5 and 9, so some steps were skipped in this figure. The  $rB$  and  $rX \dots$  indicate the left reactant complex shown in Figure 9. A reversible  $2tx_4$  motif within one product (green arrows) can separate the displayed parts of the  $qrAB$  and  $qrX$  strands, allowing that reaction with the second reactant.

Strands in Figure 9 are identified by their long domains: the species-specific domain  $B$ , the reaction-specific domain  $r$ , and the universal attachment domain  $X$ . The  $rX$  strand has no species-specific domains, so it can interact with the reactant  $B$  or the product  $C$ . The  $rB$  and (not shown)  $rC$  strands then mediate between the  $rX$  strand and the species strands.

To implement the bimolecular  $A + B \rightleftharpoons I_r$  surface CRN half-reaction, one could just put together two copies of this mechanism. However, the  $rX$  strand would then either have to be in a 3-stranded fuel complex with the  $rA$  and  $rB$  strands, or start out with the  $rA$  strand and have the  $rB$  strand enter later. But the second case would not let it remember, halfway through a reaction, whether it's interacting with the reactants or the products; and what should be  $A + B \rightleftharpoons C + D$  could become  $A + B \rightleftharpoons A + D$  or any of a few other combinations. Instead, we use a variant mechanism where a  $qrX - qrAB$  2-stranded fuel causes the  $A$  strand to be released on its own, rather than attached to the  $qrAB$  strand. (We use  $q$  as a

second reaction-specific long domain to minimize crosstalk potential.) This mechanism is shown in Figure 10. The  $\tau^*$  domain at the 5' end of the  $qrAB$  strand is a shared toehold, the same as the  $\tau^*$  at the 3' end of the  $rB$  strand as shown in Figure 9. Thus in the full reaction, a  $qrAB - qrX$  fuel will find an  $AX - X$  complex on the surface, displace the  $A$  according to Fig. 10, at which point the newly opened half of those strands will interact with and displace a neighboring  $BX - X$  complex on the surface, releasing a  $qrAB - BX$  reverse fuel complex and leaving a  $qrX - X$  complex on the surface to receive the products.

In our previous work on theoretical polymer systems, we showed that all “single-locus” polymer networks could be implemented by the following five types of reaction schema [19]:



Of those, the stack mechanism in Fig. 8 implements  $*_1 \rightleftharpoons *_1E$  and, with modifications,  $F*_1 \rightleftharpoons *_1$ . The surface mechanism in Fig. 9 implements  $*_1A*_2 + B \rightleftharpoons *_1C*_2$  when combined with the well-mixed CRN mechanism, and with the mechanism in Fig. 10 implements  $*_1AB*_2 \rightarrow *_1CD*_2$  (reversibly or irreversibly). The last of those reactions is only necessary for reactions that look like it, but might be doable by a variant of the stack mechanism.

## 6 Discussion and further work

### Experimental considerations

The mechanisms we presented work in the formal model of DSD we use, but being different from previously tested DSD mechanisms, leaves open the question of whether they would work in physical DNA molecules. There are four specific questions possibly interesting as general questions about DSD, beyond this mechanism: length of the mechanism; fuel synthesis; internal toehold unbinding and binding; and whether  $m_4$  steps require two matching toeholds.

Like our previous work [15], this mechanism involves a much longer sequence of steps between two complexes coming together and the result eventually separating than most experimentally tested DSD systems. The more complex a system is, the more that can go wrong, even if we don't know what. So we'd like to know whether DSD mechanisms are robust enough for a reaction of this length. Relatedly, we argued that 1- or 2-stranded input complexes would be easier to synthesize than 3- or more-stranded complexes, but we gained that at the expense of making those strands much longer, which also raises concern.

In DSD systems in general, the usual way to make the fuel complexes is to put the strands together in solution and anneal them [31]. This requires any particular multiset of strands to have one minimum free energy configuration, the desired fuel. In a coupled reconfiguration mechanism, the same strand has multiple “equal”-energy configurations, only some of which are fuels. This is likely solvable; in our previous coupled reconfiguration work [15], we speculated about co-transcriptional folding mechanisms. In this mechanism, there are regions where it doesn't matter what's in them; it might be possible to add sequences in those regions that slightly bias an annealed strand to form in one configuration over the others, but gently enough that the coupled reconfiguration mechanism can still complete.

In the DSD model we use, any domain is either short (toehold) or long; toehold domains can unbind whenever they're not anchored, then bind again to a different copy of their complement. In experimental use, “toehold” means a domain whose binding energy is

roughly equal to the energy from entropy of complexes dissociating, in the current conditions. Toeholds unbinding and re-binding all within a single complex is, to our knowledge, not studied, but our mechanism depends on it. We suspect that, even if internal toehold unbinding doesn't work with 6-base toeholds, there will be some length of toehold that does behave the way we require. Part of testing this mechanism would involve finding that length.

In the anchored model, we assume an  $(m_4)$  step can only happen if the branch migration domains are flanked by two pairs of bound domains (usually toeholds); if only one side or none are bound, the  $m_4$  step does not happen. Our mechanism relies heavily on this, as we use pairs of toehold identities to carry information; in Figure 5, for example, there are steps where a  $(\rho, \mu^*)$  pair needs to interact with either of two  $(\mu, \rho^*)$  pairs, but not the  $(\mu, \nu^*)$  pair next to them. But again, this isn't well tested. Dabby's experiments show a difference in rates between an  $m_4$  step with both sides bound and an  $m_4$  step with one side bound and the other nonexistent [11], which is promising. But in our system the second side will have two non-complementary toeholds, which in practice will have nonzero interaction. And our system needs 6-7 orthogonal toeholds. Again, we suspect that even if 6-base toeholds don't have the desired behavior, there will be some design of toeholds that will.

### Theoretical implications

We have for a while been wondering if a general CRN implementation with DSD using only 2-stranded inputs, with a certain set of other desirable conditions, was possible. Those conditions were: the system is physically reversible, is a systematic implementation, has  $O(1)$  toeholds, is made of macroscopically bimolecular reactions ("condensed reactions" as defined in [3] or [16]), and is correct according to modular CRN bisimulation. We define those conditions properly in our previous work, where we also prove that with the additional condition of "no remote toehold  $m_3$  steps are ever possible", no such scheme can exist [16]. (The middle of a  $2tx_4$  motif involves toeholds unbinding and binding to a different copy of their complement, which is equivalent to a remote toehold  $m_3$  step. We intended that to define  $m_4$ -only systems, but the  $2tx_4$  motif shows that the condition was too strong.)

We have a number of examples of general CRN implementations that meet all but one of those conditions: the state-of-the-art CRN implementations use 3- or more-stranded fuels [5, 8, 24, 30], while we have come up with a scheme that uses  $O(n)$  toeholds and one that relies on macroscopically trimolecular reactions [18]. Our previous  $m_3$ -based coupled reconfiguration mechanism meets all of those conditions [15], but relies on the assumption that pseudoknots physically don't happen. We found this questionable, and defined a new desired condition, pseudoknot-robustness. This mechanism meets all of those conditions including pseudoknot-robustness, closing the question of whether such a mechanism is possible.

There are still interesting directions to go from this on the theoretical level. Intuitively, there still seems to be a difference between 2-stranded and 3-stranded DSD input complexes: our 2-stranded mechanisms require long sequences of steps for  $A + B \Rightarrow \dots$  logic, which 3-stranded mechanisms can do with two  $tx_3$  [30] or  $tx_4$  [14] motifs. It might be possible to explain why in formal terms, or discover a more efficient 2-stranded mechanism. Relatedly, the coupled reconfiguration mechanism was important for us, and the conditions under which it's possible can be further explored. Our mechanism relies on using pairs of toeholds to enable or disable  $m_4$  steps, while  $m_3$  steps only rely on one toehold, so it's an interesting question whether the same mechanisms can be done with only  $m_3$  steps. And finally, if any experimental testing shows that DNA doesn't behave the way our model assumes, defining the difference might open up further theoretical exploration.

## References

- 1 Dana Angluin, James Aspnes, Zoë Diamadi, Michael J. Fischer, and René Peralta. Computation in networks of passively mobile finite-state sensors. *Distributed Computing*, pages 235–253, March 2006. doi:10.1007/s00446-005-0138-3.
- 2 Dana Angluin, James Aspnes, and David Eisenstat. Stably computable predicates are semilinear. In *Proceedings of the twenty-fifth annual ACM symposium on Principles of distributed computing*, pages 292–299. ACM, 2006. doi:10.1145/1146381.1146425.
- 3 Stefan Badelt, Casey Grun, Karthik V Sarma, Brian Wolfe, Seung Woo Shin, and Erik Winfree. A domain-level DNA strand displacement reaction enumerator allowing arbitrary non-pseudoknotted secondary structures. *Journal of the Royal Society Interface*, 17(167):20190866, 2020. doi:10.1098/rsif.2019.0866.
- 4 Stefan Badelt, Seung Woo Shin, Robert F Johnson, Qing Dong, Chris Thachuk, and Erik Winfree. A general-purpose CRN-to-DSD compiler with formal verification, optimization, and simulation capabilities. In Damien Woods and Yannick Rondelez, editors, *DNA Computing and Molecular Programming*, volume 9818 of *Lecture Notes in Computer Science*, pages 232–248. Springer, 2017. doi:10.1007/978-3-319-66799-7\_15.
- 5 Luca Cardelli. Two-domain DNA strand displacement. *Mathematical Structures in Computer Science*, 23(02):247–271, 2013. doi:10.1017/S0960129512000102.
- 6 Luca Cardelli and Gianluigi Zavattaro. On the computational power of biochemistry. In *International Conference on Algebraic Biology*, pages 65–80. Springer, 2008. doi:10.1007/978-3-540-85101-1\_6.
- 7 Ho-Lin Chen, David Doty, and David Soloveichik. Deterministic function computation with chemical reaction networks. *Natural Computing*, 13(4):517–534, 2014. doi:10.1007/s11047-013-9393-6.
- 8 Yuan-Jyue Chen, Neil Dalchau, Niranjan Srinivas, Andrew Phillips, Luca Cardelli, David Soloveichik, and Georg Seelig. Programmable chemical controllers made from DNA. *Nature Nanotechnology*, 8(10):755–762, 2013. doi:10.1038/nnano.2013.189.
- 9 Kevin M Cherry and Lulu Qian. Scaling up molecular pattern recognition with DNA-based winner-take-all neural networks. *Nature*, 559(7714):370, 2018. doi:10.1038/s41586-018-0289-6.
- 10 Samuel Clamons, Lulu Qian, and Erik Winfree. Programming and simulating chemical reaction networks on a surface. *Journal of the Royal Society Interface*, 17(166):20190790, 2020. doi:10.1098/rsif.2019.0790.
- 11 Nadine L Dabby. *Synthetic molecular machines for active self-assembly: prototype algorithms, designs, and experimental study*. PhD thesis, California Institute of Technology, 2013. doi:10.7907/T0ZG-PA07.
- 12 Robert M Dirks, Justin S Bois, Joseph M Schaeffer, Erik Winfree, and Niles A Pierce. Thermodynamic analysis of interacting nucleic acid strands. *SIAM review*, 49(1):65–88, 2007. doi:10.1137/060651100.
- 13 David Doty and Monir Hajiaghayi. Leaderless deterministic chemical reaction networks. In David Soloveichik and Bernard Yurke, editors, *DNA 2013: Proceedings of The 19th International Meeting on DNA Computing and Molecular Programming*, volume 8141 of *Lecture Notes in Computer Science*, pages 46–60. Springer International Publishing, 2013. doi:10.1007/978-3-319-01928-4\_4.
- 14 Benjamin Groves, Yuan-Jyue Chen, Chiara Zurla, Sergii Pocheikailov, Jonathan L Kirschman, Philip J Santangelo, and Georg Seelig. Computing in mammalian cells with nucleic acid strand exchange. *Nature nanotechnology*, 11(3):287–294, 2016. doi:10.1038/nnano.2015.278.
- 15 Hope Amber Johnson and Anne Condon. A Coupled Reconfiguration Mechanism for Single-Stranded DNA Strand Displacement Systems. In Thomas E. Ouldridge and Shelley F. J. Wickham, editors, *28th International Conference on DNA Computing and Molecular Programming (DNA 28)*, volume 238 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 3:1–3:19, Dagstuhl, Germany, 2022. Schloss Dagstuhl – Leibniz-Zentrum für Informatik. doi:10.4230/LIPIcs.DNA.28.3.



- 16 Robert F. Johnson. Impossibility of sufficiently simple chemical reaction network implementations in DNA strand displacement. In Ian McQuillan and Shinnosuke Seki, editors, *Unconventional Computation and Natural Computation*, pages 136–149. Springer International Publishing, 2019. doi:10.1007/978-3-030-19311-9\_12.
- 17 Robert F Johnson, Qing Dong, and Erik Winfree. Verifying chemical reaction network implementations: A bisimulation approach. *Theoretical Computer Science*, 2018. doi:10.1016/j.tcs.2018.01.002.
- 18 Robert F. Johnson and Lulu Qian. Simplifying Chemical Reaction Network Implementations with Two-Stranded DNA Building Blocks. In Cody Geary and Matthew J. Patitz, editors, *26th International Conference on DNA Computing and Molecular Programming (DNA 26)*, volume 174 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 2:1–2:14, Dagstuhl, Germany, 2020. Schloss Dagstuhl – Leibniz-Zentrum für Informatik. doi:10.4230/LIPIcs.DNA.2020.2.
- 19 Robert F Johnson and Erik Winfree. Verifying polymer reaction networks using bisimulation. *Theoretical Computer Science*, 843:84–114, 2020. doi:10.1016/j.tcs.2020.08.007.
- 20 Matthew R. Lakin and Andrew Phillips. Modelling, simulating and verifying Turing-powerful strand displacement systems. In Luca Cardelli and William Shih, editors, *DNA Computing and Molecular Programming*, pages 130–144, Berlin, Heidelberg, 2011. Springer Berlin Heidelberg. doi:10.1007/978-3-642-23638-9\_12.
- 21 Matthew R. Lakin, Simon Youssef, Filippo Polo, Stephen Emmott, and Andrew Phillips. Visual DSD: a design and analysis tool for DNA strand displacement systems. *Bioinformatics*, 27(22):3211–3213, 2011. doi:10.1093/bioinformatics/btr543.
- 22 Jérôme Leroux. Vector addition systems reachability problem (a simpler solution). In *EPiC*, volume 10, pages 214–228. Andrei Voronkov, 2012. doi:10.29007/BNX2.
- 23 Rasmus L Petersen, Matthew R Lakin, and Andrew Phillips. A strand graph semantics for DNA-based computation. *Theoretical computer science*, 632:43–73, 2016. doi:10.1016/j.tcs.2015.07.041.
- 24 Lulu Qian, David Soloveichik, and Erik Winfree. Efficient Turing-universal computation with DNA polymers. In Yasubumi Sakakibara and Yongli Mi, editors, *DNA Computing and Molecular Programming*, volume 6518 of *Lecture Notes in Computer Science*, pages 123–140. Springer, 2011. doi:10.1007/978-3-642-18305-8\_12.
- 25 Lulu Qian and Erik Winfree. Scaling up digital circuit computation with DNA strand displacement cascades. *Science*, 332(6034):1196–1201, 2011. doi:10.1126/science.1200520.
- 26 Lulu Qian and Erik Winfree. Parallel and scalable computation and spatial dynamics with DNA-based chemical reaction networks on a surface. In Satoshi Murata and Satoshi Kobayashi, editors, *DNA Computing and Molecular Programming*, volume 8727 of *Lecture Notes in Computer Science*, pages 114–131. Springer, 2014. doi:10.1007/978-3-319-11295-4\_8.
- 27 Charles Rackoff. The covering and boundedness problems for vector addition systems. *Theoretical Computer Science*, 6(2):223–231, 1978. doi:10.1016/0304-3975(78)90036-1.
- 28 Friedrich C. Simmel, Bernard Yurke, and Hari R. Singh. Principles and applications of nucleic acid strand displacement reactions. *Chemical Reviews*, 119(10):6326–6369, 2019. PMID: 30714375. doi:10.1021/acs.chemrev.8b00580.
- 29 David Soloveichik, Matthew Cook, Erik Winfree, and Jehoshua Bruck. Computation with finite stochastic chemical reaction networks. *Natural Computing*, 7(4):615–633, 2008. doi:10.1007/s11047-008-9067-y.
- 30 David Soloveichik, Georg Seelig, and Erik Winfree. DNA as a universal substrate for chemical kinetics. *Proceedings of the National Academy of Sciences*, 107(12):5393–5398, 2010. doi:10.1073/pnas.0909380107.
- 31 Niranjana Srinivas, James Parkin, Georg Seelig, Erik Winfree, and David Soloveichik. Enzyme-free nucleic acid dynamical systems. *Science*, 358, 2017. doi:10.1126/science.aal2052.



- 32 Allison Tai and Anne Condon. Error-free stable computation with polymer-supplemented chemical reaction networks. In Chris Thachuk and Yan Liu, editors, *DNA Computing and Molecular Programming*, pages 197–218, Cham, 2019. Springer International Publishing. doi:10.1007/978-3-030-26807-7\_11.
- 33 Chris Thachuk, Erik Winfree, and David Soloveichik. Leakless DNA strand displacement systems. In Andrew Phillips and Peng Yin, editors, *DNA Computing and Molecular Programming*, volume 9211 of Lecture Notes in Computer Science, pages 133–153. Springer, 2015. doi:10.1007/978-3-319-21999-8\_9.
- 34 Anupama J Thubagere, Chris Thachuk, Joseph Berleant, Robert F Johnson, Diana A Ardelean, Kevin M Cherry, and Lulu Qian. Compiler-aided systematic construction of large-scale DNA strand displacement circuits using unpurified components. *Nature Communications*, 8:14373, 2017. doi:10.1038/ncomms14373.
- 35 Boya Wang, Chris Thachuk, Andrew D Ellington, Erik Winfree, and David Soloveichik. Effective design principles for leakless strand displacement systems. *Proceedings of the National Academy of Sciences*, 115(52):E12182–E12191, 2018. doi:10.1073/pnas.1806859115.

## A Formal definitions of DSD

Here we give formal definitions of various aspects of DSD systems that were omitted from Section 2 for lack of space. When discussing the definition of correctness in terms of CRN bisimulation, we also discuss how our CRN mechanism satisfies it.

### Anchored and semi-anchored bonds

In Definition 3 the  $u$ ,  $m_3$ , and  $m_4$  steps had conditions based on whether certain bonds were “anchored by an adjacent bond as shown in Figure 2”. This refers to a formal definition of anchored, which is as follows:

► **Definition 6.** A bond  $(x, x^*)$  is anchored if, for some other bond  $(y, y^*)$ , either  $x y$  and  $y^* x^*$  are each adjacent 5' to 3' on their respective strands, or  $y x$  and  $x^* y^*$  are. We also in that case say  $(x, x^*)$  is anchored by  $(y, y^*)$ , and note that if so, then  $(y, y^*)$  is also anchored by  $(x, x^*)$ .

Fig. 2 contains many examples of anchored bonds. In particular, the resulting bonds of  $m_3$  and  $m_4$  steps, and the cases where a  $u$  step can't happen, are examples of anchored bonds as described in the rules of Definition 3.

One can also define a *semi-anchored* model, as well as a remote toehold model, by relaxing those conditions:

► **Definition 7.** A bond  $(x, x^*)$  is semi-anchored if, for some other bonds  $(y_i, y_i^*)$ ,  $1 \leq i \leq n$ ,  $y_i^* y_{i+1}$  are adjacent 5' to 3', as are either  $x y_1$  and  $y_n^* x^*$  or  $x^* y_1$  and  $y_n^* x$ . Note that an anchored bond is semi-anchored, satisfying this definition for  $n = 1$ .

The semi-anchored pseudoknot-allowed or pseudoknot-free DSD models are defined from the corresponding anchored model by modifying the  $m_3$  and  $m_4$  steps to require the resulting bond(s) to be semi-anchored instead of anchored. The  $u$  step is unchanged, still able to happen if the bond is not fully anchored.

The remote-toehold pseudoknot-allowed DSD model is defined from the anchored pseudoknot-allowed model by removing the requirement in the  $m_3$  and  $m_4$  steps that the resulting bond(s) be anchored at all. The remote-toehold pseudoknot-free DSD model is defined from the anchored pseudoknot-free model by, for  $m_3$  and  $m_4$  steps, removing the requirement that the resulting bond(s) be anchored but adding a requirement that the resulting complex be non-pseudoknotted, since adding a pseudoknot would now otherwise be possible.

It is also possible to mix-and-match conditions, e.g. requiring  $m_4$  steps to be fully anchored while allowing remote-toehold  $m_3$  steps. In fact, there are a lot of different possible DSD models, of which these are just a few; for example, Petersen et al. define a model that forbids  $u$  steps if the bond is semi-anchored (which they refer to as “anchored”) [23]. We disagree with that because such a model would cause difficulties for  $tx_4$  motifs, which seem to be experimentally possible [11, 14].

A good example of the semi-anchored model is the 3-way-initiated 4-way branch migration mechanism used in the original surface CRN paper [26]; the  $m_3$  step that forms the 4-way junction is valid in the semi-anchored model, but not the anchored model. In our work, the semi-anchored model is referenced only for comparison with the anchored model. The remote-toehold model is mentioned in passing related to the conditions in the impossibility proof of our previous work: that work used a mixed model where  $m_3$  was remote-toehold while  $m_4$  was anchored only [16]. Then, for the class of DSD systems it proved could not exist, they were required to implement CRNs in such a way that no remote-toehold  $m_3$  steps would ever be enumerated. In this work, the  $2tx_4$  motif (Figure 3) contains two  $u, b$  sequences each of which is equivalent to a remote-toehold  $m_3$ , so our system would not meet the condition of the impossibility proof; thus this mechanism’s existence suggests those conditions were too strict.

We mentioned that the anchored, pseudoknot-free model is equivalent to Peppercorn’s model with the `--reject-remote` option used [3]. Peppercorn’s standard model, without that flag, is equivalent to the remote-toehold pseudoknot-free model.

## Condensed reactions

One assumption we often make is that, in the (relatively low) concentrations typical for DSD, if two complexes come together and interact, they will have enough time to reach a stable state (possibly involving separating into two or more complexes) before any third complex comes by and interacts with them. Thus the sequence of steps from the first (necessarily)  $b$  step to the eventual separation can be treated as a single reaction, called a *condensed reaction*, following [3]:

► **Definition 8.** *In a given DSD system, a resting set (sometimes resting state) is a set of complexes such that any complex in the set can transform into any other via unimolecular steps, and no complex in the set can transform into any complex not in the set via a unimolecular step. A condensed reaction is a sequence of steps, starting with a bimolecular  $b$  step whose reactants are both elements of some resting sets, followed by 0 or more unimolecular steps, resulting in one or more complexes each of which is an element of some resting set. More precisely, a condensed reaction is an equivalence class of such pathways, where two pathways are equivalent if they are the same reaction when considered as reactions from resting sets to resting sets.*

For example, the entire pathway of Figure 5 is a single condensed reaction. It is worth noting that in a fully reversible system, any condensed reaction must have more than one product, since any single complex formed after an initial  $b$  step could separate, via the reverse of the pathway up to that point, into the original complexes, and thus that intermediate complex cannot be an element of a resting set.

The assumption that only condensed reactions can happen, if it holds, is particularly useful in claiming that no undesired cross-reactions can happen in a given system. Usually there aren’t actually any trimolecular undesired reactions that could happen, but the

assumption lets us avoid checking a large amount of combinations that don't go anywhere. This assumption specifically doesn't hold for cooperative hybridization mechanisms, including but not limited to our cooperative 4-way branch migration-based CRN implementation [18].

### CRN implementations and CRN bisimulation

We use the formal verification method of CRN bisimulation [17] to determine whether our DSD system correctly implements the CRN it claims to implement. CRN bisimulation is defined as a relation between two CRNs, so here we take the abstract CRN as the *formal* CRN, and the enumerated CRN of the DSD system as the *implementation* CRN. CRN bisimulation then says the implementation is correct if there is an *interpretation* of implementation species that satisfies certain conditions, from [17]:

► **Definition 9.** *Given a formal CRN and an implementation CRN, an interpretation is a function from implementation species to multisets of formal species. An interpretation is a CRN bisimulation if it satisfies the following three conditions:*

- Atomic condition: *Any formal CRN species has an implementation species interpreted as the singleton multiset of that formal species.*
- Delimiting condition: *Any implementation reaction, when interpreted, is either a formal reaction or trivial (a “reaction” whose reactants equal its products). (“Anything that can happen, should.”)*
- Permissive condition: *In any implementation state, for any formal reaction that can happen in its interpretation, a reaction interpreted as that reaction can happen after zero or more trivial reactions. (“Anything that should happen, can.”)*

*The implementation CRN is correct according to CRN bisimulation if there exists an interpretation that is a CRN bisimulation.*

In general with DSD systems, CRN bisimulation is done after removing fuel species from the enumerated CRN, and the interpretation is required to interpret each signal species as the singleton multiset of its formal species (thus satisfying the atomic condition automatically). We note that as far as CRN bisimulation is concerned, removing a fuel species  $f$  is equivalent to adding a reaction  $\emptyset \rightarrow f$  and requiring the interpretation of  $f$  to be  $\emptyset$ .

Most CRN implementations are *modular* in a certain sense: the implementation consists of distinct modules, one for each formal reaction, each of which implements that formal reaction starting and ending with a set of common “signal” species. This motivates a definition of *modular CRN bisimulation*, which is often easier to check for both computers and humans, again from [17]:

► **Definition 10.** *Given a formal and implementation CRN divided into a set of pairs of formal and implementation sub-CRNs, a modular bisimulation interpretation is an interpretation of the implementation species such that:*

- *the interpretation, restricted to any pair of sub-CRNs, is a CRN bisimulation on those sub-CRNs (i.e., satisfies the atomic, delimiting, and permissive conditions);*
- *there is a set of signal species, where each signal species is an implementation species interpreted as one copy of one formal species and appears in each implementation sub-CRN for which its formal species appears in the corresponding formal sub-CRN, with at least one signal species for each formal species; and*
- *for each implementation species in any sub-CRN, there is a sequence of trivial reactions that turns that species into a set of signal species corresponding to its interpretation, plus possibly some species whose interpretation is empty.*

In other words, each “module” must implement any formal reaction in it, and also be able to turn any of its implementation species into the “signal species” that can then go on to implement any formal reaction in any other module. In DSD implementations you might have one module for each formal reaction plus one “crosstalk module”, containing no formal reactions and all the implementation reactions that might occur between species belonging to two different modules. The modular bisimulation condition then checks that those crosstalk reactions neither turn any species into something they shouldn’t turn into, nor destroy any species in a way that can’t be turned back into its signal species.

In Figure 7, the  $A_{on}$  and  $A_{off}$  complexes are interpreted as  $A$  and  $\emptyset$ , respectively, with  $A_{on}$  the modular signal species for  $A$ ; and similar for  $B$ . The gate strands are interpreted as, from top to bottom,  $\emptyset$ ,  $A$ ,  $A + B$ , and  $\emptyset$ . Finally,  $int_{r;off}$  is interpreted as  $\emptyset$ , and  $int_{r,on}$  as the products of the formal reaction, e.g.  $C + D$  if the reaction is  $A + B \rightleftharpoons C + D$ . So the first two reactions are trivial, and the third is interpreted as the formal reaction. Any point in the process can reverse itself to produce the signals  $A_{on}$  and  $B_{on}$ , and the (not shown) product half can turn  $int_{r,on}$  into  $C_{on} + D_{on}$ , satisfying modularity. While this looks like a proof of correctness, the actual hard work is in reaction enumeration, proving that the reactions shown in the figure are the only ones that happen. Since CRN bisimulation happens after enumeration, this is beyond its scope.

Pseudoknot-robustness is then defined in terms of CRN bisimulation. First, note that if the same initial complexes are enumerated separately in the pseudoknot-allowed and pseudoknot-free anchored models, the pseudoknot-allowed species and reactions will be supersets of their pseudoknot-free counterparts. Then a DSD implementation of a formal CRN is *pseudoknot-robustly correct* if there is a single interpretation that is a CRN bisimulation on the pseudoknot-allowed enumerated CRN, and its restriction to the pseudoknot-free enumerated species is a CRN bisimulation on that CRN. Equivalently, it is correct if the atomic and permissive conditions hold in the pseudoknot-free model, the delimiting condition holds in the pseudoknot-robust model, and (similar to the modularity condition) any implementation species in the pseudoknot-robust model can, via trivial reactions, turn into only species that exist in the pseudoknot-free model.

## B Further discussion of correctness

We did code the well-mixed CRN implementation scheme (Figure 7) into Nuskell [4] and verified that it satisfies modular bisimulation on two small formal CRNs, a binary counter and an oscillator. However, Nuskell (in particular, Peppercorn, which it uses for reaction enumeration) only handles pseudoknot-free systems, and cannot confirm that our implementation scheme is pseudoknot-robust.

Our argument for correctness is, first, from Figures 5-7 we clearly see that starting from the signal species  $A_{on} + B_{on}$  with appropriate fuel species ( $fuel_{r;A,B;fwd}$ ,  $int_{r;off}$ ,  $fuel_{r;C,D;rev}$ ,  $C_{off}$ , and  $D_{off}$ , with those last three being the products copy of the mechanism that is implied but not shown in Figure 7), the reaction can progress, consuming  $A_{on}$  and  $B_{on}$  and producing  $C_{on}$  and  $D_{on}$ . For CRN bisimulation purposes, the step that produces  $int_{r,on}$  is the “turning point” where the interpretation (as described in the previous appendix) changes from  $A + B$  to  $C + D$ . This argument is effectively the permissive condition of CRN bisimulation.

Second, the modularity condition: given that  $A_{off}$  and  $B_{off}$  are fuels, any intermediate complex can reverse the process, which produces the signal species ( $A_{on}$  and/or  $B_{on}$ ) of its interpretation; and similarly  $int_{r,on}$  or any intermediate of the products half can complete

the process to produce the signal species  $C_{on}$  and/or  $D_{on}$ . This shows that any possible combination of complexes *can* do anything its interpretation should be able to do, leaving only to show that no undesired reactions can happen.

For undesired reactions, our argument comes from checking all combinations of toeholds that appear together in a way that could allow a four-way branch migration step. We are assuming that no  $m_4$  step occurs unless toeholds on both sides match; whether this is true is an experimental question. Since the entire system never has any unbound long domains anywhere, no  $m_3$  steps are possible, and any combination of just  $b$  and  $u$  steps can't make a nontrivial condensed reaction (argument similar to one made in [16]). In the initial states, those toehold combinations are (from Figure 5, with  $\rho = \nu^*$  applied, each given 5' first then 3'):  $(\tau, \nu^*)$ ,  $(\lambda^*, \sigma)$ ,  $(\mu, \nu)$ ,  $(\mu, \nu^*)$ ,  $(\omega^*, \lambda)$ ,  $(\nu, \tau^*)$  on the top left, and  $(\tau, \nu)$ ,  $(\lambda^*, \sigma)$ ,  $(\mu, \nu^*)$ ,  $(\mu, \nu)$ ,  $(\nu^*, \tau^*)$ ,  $(\omega^*, \lambda)$  on the bottom left. In various intermediate steps (including some instances of the intermediate of the  $2tx_4$  motif, not shown in Figure 5), the following additional pairs appear:  $(\sigma^*, \lambda)$ ,  $(\nu, \mu^*)$ ,  $(\lambda^*, \omega)$ ,  $(\nu^*, \mu^*)$ . So for example a  $(\tau, \nu^*)$  can bind and enable an  $m_4$  step with a  $(\nu, \tau^*)$  on the same long domain identity, but not any other combination, in the assumptions of the model we're using.

So, the possible interactions are: the  $(\tau, \nu^*) + (\nu, \tau^*)$  interaction in the initial state, which is the intended first step; the  $(\tau, \nu) + (\nu^*, \tau^*)$  interaction in the final state, which is the intended first step of the reverse pathway; and no other possible interactions between anything in the initial and/or final states. The four pairs that appear only in intermediates will be able to interact with various complementary pairs on the same intermediate complex; it can be seen from Figure 5 that the only such interactions will be the intended forward and reverse  $2tx_4$  motifs. Finally, two copies of the mechanism at the same intermediate step will have complementary pairs of toeholds; however an interaction between them would be a violation of the condensed reaction (i.e. low-concentration) assumption. It's not actually clear whether any such effectively tetramolecular reactions would actually cause problems; two instances of an  $A$  intermediate interacting, if they eventually just disentangle and both complete their respective reactions, might just end up in the same place as each one completing independently. Or possibly there could be issues if an  $A + B$  reactant side interacts with an  $A + C$  reactant side, or other such crosstalk. But in the condensed reaction model, this mechanism unambiguously works.

The argument for correctness of the polymer mechanisms would be along the same lines, confirming that no complementary toehold pairs appear except for the intended forward and reverse steps.



# Reachability in Deletion-Only Chemical Reaction Networks

**Bin Fu** ✉

University of Texas Rio Grande Valley,  
Edinburg, TX, USA

**Ryan Knobel** ✉

University of Texas Rio Grande Valley,  
Edinburg, TX, USA

**Aiden Massie** ✉

University of Texas Rio Grande Valley,  
Edinburg, TX, USA

**Adrian Salinas** ✉

University of Texas Rio Grande Valley,  
Edinburg, TX, USA

**Tim Wylie** ✉

University of Texas Rio Grande Valley,  
Edinburg, TX, USA

**Timothy Gomez** ✉

Massachusetts Institute of Technology,  
Cambridge, MA, USA

**Austin Luchsinger** ✉

University of Texas Rio Grande Valley, Edinburg,  
TX, USA

**Marco Rodriguez** ✉

Massachusetts Institute of Technology,  
Cambridge, MA, USA

**Robert Schweller** ✉

University of Texas Rio Grande Valley,  
Edinburg, TX, USA

---

## Abstract

For general discrete Chemical Reaction Networks (CRNs), the fundamental problem of reachability – the question of whether a target configuration can be produced from a given initial configuration – was recently shown to be Ackermann-complete. However, many open questions remain about which features of the CRN model drive this complexity. We study a restricted class of CRNs with *void rules*, reactions that only decrease species counts. We further examine this regime in the motivated model of step CRNs, which allow additional species to be introduced in discrete stages. With and without steps, we characterize the complexity of the reachability problem for CRNs with void rules. We show that, without steps, reachability remains polynomial-time solvable for bimolecular systems but becomes NP-complete for larger reactions. Conversely, with just a single step, reachability becomes NP-complete even for bimolecular systems. Our results provide a nearly complete classification of void-rule reachability problems into tractable and intractable cases, with only a single exception.

**2012 ACM Subject Classification** Theory of computation → Models of computation; Theory of computation → Problems, reductions and completeness

**Keywords and phrases** CRN, Chemical Reaction Network, Reachability, Void Reactions

**Digital Object Identifier** 10.4230/LIPIcs.DNA.31.3

**Funding** This research was supported in part by National Science Foundation Grant CCF-2329918.

## 1 Introduction

**Background.** In molecular programming, Chemical Reaction Networks [5, 6] have become a staple model for abstracting molecular interactions. The model consists of a set of chemical species (formal symbols) as well as a set of reactions that dictate how these species interact. As an example, the reaction  $A + B \rightarrow C + D$  describes chemical species  $A$  and  $B$  reacting to form species  $C$  and  $D$ . While chemical kinetics are commonly modeled continuously using ordinary differential equations, this approximation breaks down for systems with relatively small volumes where species are present in very low amounts. Such systems are better modeled as discrete Chemical Reaction Networks (which we will hereby be referring to simply as CRNs), where the system state consists of non-negative integer counts of each species and



© Bin Fu, Timothy Gomez, Ryan Knobel, Austin Luchsinger, Aiden Massie, Marco Rodriguez, Adrian Salinas, Robert Schweller, and Tim Wylie;  
licensed under Creative Commons License CC-BY 4.0

31st International Conference on DNA Computing and Molecular Programming (DNA 31).

Editors: Josie Schaeffer and Fei Zhang; Article No. 3; pp. 3:1–3:21



Leibniz International Proceedings in Informatics

Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany



■ **Table 1** Summary of reachability results. The notation  $(k, k-1)^+$  means one or more void rules of the form  $(k, k-1)$  for  $k \geq 2$ . The notation  $(k \geq 3, g \leq k-2)$  means void rules with at least three reactants that consume at least two species. The notation  $\rightarrow$  NPC signifies that the step CRN result follows directly from the basic CRN case.

Reachability Results				
	Basic CRNs (1-step)		2-Step CRNs	
Void Rules	Complexity	Ref.	Complexity	Ref.
$(2, 1)$	NL	Thm. 11	NPC	Thm. 12
$(k, k-1)^+$	$O( \Lambda ^2 \Gamma )$	Thm. 21	NPC	Cor. 22
$(2, 0)$	$O( \Lambda ^2 \Gamma  \log( \Lambda ))$	[1]	NPC	Thm. 9
$(2, 0), (2, 1)$	$O( \Lambda ^2 \Gamma  \log( \Lambda ))$	Thm. 18	NPC	Thm. 9
$(k \geq 3, g \leq k-2)$	NPC	Cor. 24	$\rightarrow$ NPC	-

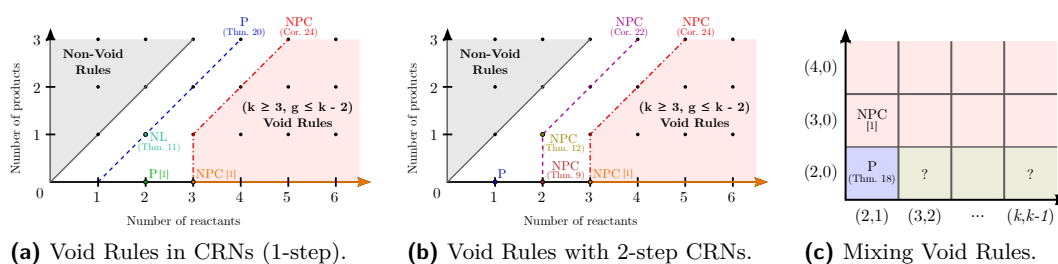
state transitions occur stochastically as a continuous time Markov process [13]. It turns out that this model of chemistry has deep connections to several other well-studied mathematical objects. In fact, CRNs are equivalent [9] to Vector Addition Systems (VASs) [16] and Petri-Nets [23] which were independently introduced to represent and analyze concurrent and distributed processes. This underlying object also appears in the form of commutative semigroups when only fully reversible reactions are considered [17, 22], and as Population Protocols [4] when reactions must have exactly two input and two output elements.

Perhaps the most fundamental problem within these models is that of *reachability* which asks: given an initial configuration and target configuration for a particular system, can the target configuration be reached from the initial configuration by following some sequence of legal transitions [20, 21, 25]? Although interest in this problem dates back to the 1970s and 1980s, it was only recently resolved with a flurry of new results over the past few years [10, 11, 12, 18, 19], which conclude that reachability is Ackermann-complete for these models. Over the course of this 40+ year-long quest to determine the complexity of reachability, scientists also began to discover the centrality of this problem. Several other important problems from seemingly unrelated areas have been reduced to reachability, from system liveness to language emptiness problems and more [14, 26], emphasizing the significance of succinctly classifying reachability.

Despite the closure of this problem for general CRNs, and in fact due to how complicated these systems can be, there is a natural motivation to explore reachability for more restricted systems. Many papers have emerged that investigate reachability in restricted versions of the model [7, 8, 15, 26, 31], but one of the most elementary restrictions is that of [1] and [2]. There, the authors consider deletion-only systems (called *void rules*) where reactions only ever consume species and reduce the size of the system. The interaction rules for these systems can be thought to convert some chemical reactants into inert waste species, or cause system agents to “go offline” and become inactive. Similar work has also investigated reachability in a slightly different version of size-reducing chemical reaction networks whose stoichiometric matrices are totally unimodular [28, 29]. This limited class of systems permits tractable and intractable problems, placing it along an interesting complexity boundary.

**Related Work.** In [1], the authors studied reachability for this restricted class of CRNs that use (deletion-only) *void rules* (e.g.,  $A + B \rightarrow A$  or  $A + B \rightarrow \emptyset$ ). Under this restriction, they prove NP-completeness for reachability using void rules of size  $(3, 0)$  (3 reactants, no





**Figure 1** Visual representations showing how the results from Table 1 fit together. (a) A plot depicting the complete characterization of reachability complexity for basic CRNs with uniform-type void rules. (b) A plot depicting the complete characterization of uniform-type step results with only a single additional step. (c) Void rule systems with mixed-type rules in basic CRNs.

products) and provide a polynomial time algorithm for reachability using void rules of size (2,0) (2 reactants, no products) for unary inputs. In this paper, we continue this line of work by considering reachability in CRNs with void rules of varying numbers of reactants and products. Since void rules arguably constitute the simplest type of reaction, one may wonder what computationally interesting behavior can be achieved with them – and this exact idea is investigated in [2].

The authors of [2] and [3] combine void rules with an experimentally motivated model extension called *step* CRNs. Ideally, experimental scientists use the CRN model to design molecular systems and predict the molecular interactions and possible end-states of those systems. However, contrary to the CRN model in which all the molecules are ready to interact from the start, many experimental designs rely on a progressive addition of molecules that allow some interactions to occur first before moving forward with the experiment. Furthermore, the ability of reactions to add molecules that interact with the rest of the system is precisely what makes experimental applications of theoretical concepts difficult to realize with minimal error at a practical scale [27, 32]. Thus, the step CRN model was introduced which incorporates a sequence of discrete steps, where, at each step, new species are added to the existing CRN and react until no further reactions can occur. This augmentation more closely reflects a laboratory setting in which chemicals are incrementally added in a test tube and left to interact. The work of [2] and [3] shows that, surprisingly, extremely simple void rules are capable of simulating threshold formulas and threshold circuits when steps are added. In this paper, we show that adding even a single step drastically changes the computational complexity of the reachability problem, moving from  $P$  (and even  $NL$ ) to NP-complete in many cases.

**Our Contributions and Paper Organization.** In this paper we present several results for the various types of deletion-only interaction rules. We show that bimolecular deletion-only systems are solvable in polynomial time, with or without catalysts for basic (*single-step*) CRNs, while they are NP-complete even with one additional step. On the other hand, we show that larger void rules (with  $k \geq 3$  reactants and  $g \leq k - 2$  products) are NP-complete even for basic CRNs unless all but one reactant is a catalyst, in which case we show the problem is in  $P$ . These results are outlined in Table 1, and we visualize the complexity landscape in Figures 1a, 1b, and 1c. When taken together, these results yield the following theorems, which provide a nearly complete characterization of reachability with void rules in CRNs with and without steps.

► **Theorem.** *Reachability for basic CRNs uniformly using size  $(k \geq 3, g \leq k - 2)$  void rules is NP-complete, and is in P when uniformly using any other size void rule.*

► **Theorem.** *Reachability for 2-step CRNs with only void rules that use exactly one reactant is in P, and is NP-complete otherwise.*

► **Theorem.** *Reachability for basic CRNs that use a combination of void rule types is in P for combinations of  $(2, 0) + (2, 1)$  rules as well as  $(k, k - 1)^+$  rules, and is NP-complete for any combination that uses  $(k \geq 3, g \leq k - 2)$  rules.*

We begin the paper by defining void rules, Step Chemical Reaction Networks, and the reachability problem in Section 2. Sections 3-6 establish the complexity of reachability based on the size of rules used by a given system: Section 3 shows membership in NP for all deletion-only systems. Section 4 considers bimolecular rule sets that are either all catalytic or non-catalytic, and shows membership in P in either case. It also shows that, in contrast, the problem becomes NP-complete with the inclusion of a second step. Section 5 expands this to bimolecular systems with mixed catalytic and non-catalytic rules, and Section 6 considers larger size rules which are polynomially solvable if all but a single reactant serve as catalysts, and NP-complete otherwise.

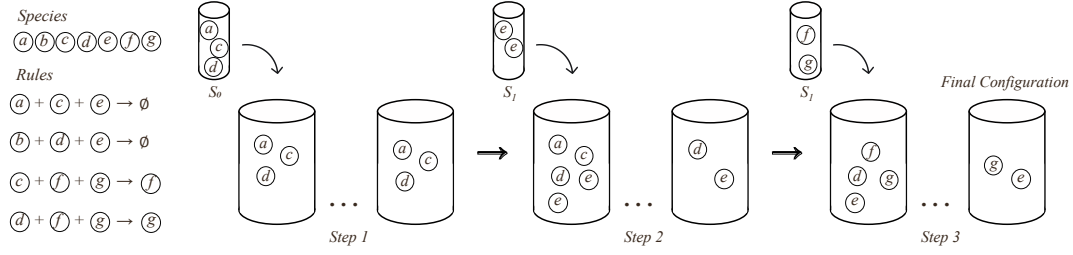
## 2 Preliminaries

### 2.1 Chemical Reaction Networks

**Basics.** Let  $\Lambda = \{\lambda_1, \lambda_2, \dots, \lambda_{|\Lambda|}\}$  denote some ordered alphabet of *species*. A configuration  $\vec{C} \in \mathbb{N}^\Lambda$  is a length- $|\Lambda|$  vector of non-negative integers where  $\vec{C}[i]$  denotes the number of copies of species  $\lambda_i$ . For a species  $\lambda_i \in \Lambda$ , we denote the configuration consisting of a single copy of  $\lambda_i$  and no other species as  $\vec{\lambda}_i$ . A *rule* or *reaction* is represented as an ordered pair  $\gamma = (\vec{R}, \vec{P}) \in \mathbb{N}^\Lambda \times \mathbb{N}^\Lambda$ .  $\vec{R}$  contains the minimum counts of each *reactant* species necessary for reaction  $\gamma$  to occur, where reactant species are either *consumed* by the rule in some count or leveraged as *catalysts* (not consumed); in some cases a combination of the two. The *product* vector  $\vec{P}$  has the count of each species *produced* by the *application* of rule  $\gamma$ , effectively replacing vector  $\vec{R}$ . The species corresponding to the non-zero elements of  $\vec{R}$  and  $\vec{P}$  are termed *reactants* and *products* of  $\gamma$ , respectively.

The *application* vector of  $\gamma$  is  $\vec{P} - \vec{R}$ , which shows the net change in species counts after applying rule  $\gamma$  once. For a configuration  $\vec{C}$  and rule  $\gamma$ , we say  $\gamma$  is *applicable* to  $\vec{C}$  if  $\vec{C}[i] \geq \vec{R}[i]$  for all  $i \in \Lambda$ , and we define the *application* of  $\gamma$  to  $\vec{C}$  as the configuration  $\vec{C}' = \vec{C} + \vec{P} - \vec{R}$ . For a set of rules  $\Gamma$ , a configuration  $\vec{C}$ , and rule  $\gamma \in \Gamma$  applicable to  $\vec{C}$  that produces  $\vec{C}' = \vec{C} + \vec{P} - \vec{R}$ , we say  $\vec{C} \rightarrow_\Gamma^1 \vec{C}'$ , a relation denoting that  $\vec{C}$  can transition to  $\vec{C}'$  by way of a single rule application from  $\Gamma$ . We further use the notation  $\vec{C} \rightarrow_\Gamma^* \vec{C}'$  to signify the transitive closure of  $\rightarrow_\Gamma^1$  and say  $\vec{C}'$  is *reachable* from  $\vec{C}$  under  $\Gamma$ , i.e.,  $\vec{C}'$  can be reached by applying a sequence of applicable rules from  $\Gamma$  to initial configuration  $\vec{C}$ . Here, we use the following notation to depict a rule  $(\vec{R}, \vec{P})$ :  $\vec{R}[1]\lambda_1 + \dots + \vec{R}[|\Lambda|]\lambda_{|\Lambda|} \rightarrow \vec{P}[1]\lambda_1 + \dots + \vec{P}[|\Lambda|]\lambda_{|\Lambda|}$ . For example, a rule turning two copies of species  $H$  and one copy of species  $O$  into one copy of species  $W$  would be written as  $2H + O \rightarrow W$ .

► **Definition 1** (Discrete Chemical Reaction Network). *A discrete chemical reaction network (CRN)  $\mathcal{C}$  is an ordered pair  $(\Lambda, \Gamma)$  where  $\Lambda$  is an ordered alphabet of species, and  $\Gamma$  is a set of rules over  $\Lambda$ .*



■ **Figure 2** An example step CRN system. The test tubes show the species added at each step and the system with those elements added. The CRN species and void rule-set are shown on the left.

A configuration is called *terminal* with respect to a CRN  $(\Lambda, \Gamma)$  if no rule  $\gamma$  can be applied to it. An initial configuration  $\vec{A}$  and CRN  $(\Lambda, \Gamma)$  is said to be *bounded* if a terminal configuration is guaranteed to be reached within some finite number of rule applications starting from  $\vec{A}$ . We denote the set of reachable configurations of a CRN as  $REACH_{\vec{A}, \Lambda, \Gamma}$ . We define the subset of reachable configurations that are terminal as  $TERM_{\vec{A}, \Lambda, \Gamma}$ .

## 2.2 Void Rules

A *void* rule is any rule that does not create any new copies of any species types (only deletes). Thus, a void rule either has no products or has products that are a subset of its reactants (in which case these products are termed *catalysts*). The formal definitions of void rules and rule size are as follows.

► **Definition 2** (Void rules). A rule  $(\vec{R}, \vec{P})$  is a void rule if  $\vec{P} - \vec{R}$  has no positive entries and at least one negative entry. There are two classes of void rules, catalytic and true void. In catalytic void rules, one or more reactants remain and one or more reactant is deleted after the rule is applied. In true void rules, there are no products remaining.

► **Definition 3.** The size/volume of a configuration  $\vec{C}$  is  $\text{volume}(\vec{C}) = \sum \vec{C}[i]$ . When  $\vec{C}$  is an initial configuration of a CRN, we refer to  $\Sigma_{\vec{C}} = \text{volume}(\vec{C})$ .

► **Definition 4** (size- $(i, j)$  rules). A rule  $(\vec{R}, \vec{P})$  is said to be a size- $(i, j)$  rule if  $(i, j) = (\text{volume}(\vec{R}), \text{volume}(\vec{P}))$ . A reaction is trimolecular if  $i = 3$ , bimolecular if  $i = 2$ , and unimolecular if  $i = 1$ .

## 2.3 Step CRNs

A step CRN is an augmentation of a basic CRN in which a sequence of additional copies of some system species are added after a terminal configuration is reached. Formally, a step CRN of  $k$  steps is an ordered pair  $((\Lambda, \Gamma), (\vec{S}_0, \vec{S}_1, \vec{S}_2, \dots, \vec{S}_{k-1}))$ , where the first element of the pair is a normal CRN  $(\Lambda, \Gamma)$ , and the second is a sequence of length- $|\Lambda|$  vectors of non-negative integers denoting how many copies of each species type to add after each step. We define a *step-configuration*  $\vec{C}_i$  for a step CRN as a valid configuration  $\vec{C}$  over  $(\Lambda, \Gamma)$  along with an integer  $i \in \{0, \dots, k-1\}$  that denotes the configuration's step. We denote the initial volume of step CRN as  $\Sigma_{\vec{S}_0} = \text{volume}(\vec{S}_0)$  and total volume as  $\Sigma_T = \sum_{i=0}^{k-1} \text{volume}(\Sigma_{\vec{S}_i})$ . Figure 2 illustrates a simple step CRN system.

Given a step CRN, we define the set of reachable configurations after each sequential step. To start off, let  $REACH_1$  be the set of reachable configurations of  $(\Lambda, \Gamma)$  with initial configuration  $\vec{S}_0$ , which we refer to as the set of configurations reachable *after step 1*. Let

$\text{TERM}_1$  be the subset of configurations in  $\text{REACH}_1$  that are terminal. Note that after a single step we have a normal CRN, i.e., 1-step CRNs are just normal CRNs with initial configuration  $\vec{S}_0$ . For the second step, we consider any configuration in  $\text{TERM}_1$  combined with  $\vec{S}_1$  as a possible starting configuration and define  $\text{REACH}_2$  to be the union of all reachable configurations from each possible starting configuration attained by adding  $\vec{S}_1$  to a configuration in  $\text{TERM}_1$ . We then define  $\text{TERM}_2$  as the subset of configurations in  $\text{REACH}_2$  that are terminal. Similarly, define  $\text{REACH}_i$  to be the union of all reachable sets attained by using initial configuration  $\vec{S}_{i-1}$  plus any element of  $\text{TERM}_{i-1}$ , and let  $\text{TERM}_i$  denote the subset of these configurations that are terminal. The set of reachable configurations for a  $k$ -step CRN is the set  $\text{REACH}_k$ , and the set of terminal configurations is  $\text{TERM}_k$ . A classical CRN can be represented as a step CRN with  $k = 1$  steps and an initial configuration of  $\vec{A} = \vec{S}_0$ .

Note that our definitions assume only the terminal configurations of a given step are passed on to seed the subsequent step. This makes sense if we assume we are dealing with *bounded* systems, as this represents simply waiting long enough for all configurations to reach a terminal state before proceeding to the next step. In this paper, we only consider bounded void rule systems; we leave more general definitions to be discussed in future work.

## 2.4 Reachability

The computational problem studied in this paper is *reachability*. Informally, reachability asks if a given initial configuration  $\vec{A}$  can be turned into a target configuration  $\vec{B}$  by applying a sequence of rules from the given CRN  $\mathcal{C}$ . The precise problem statement is as follows.

► **Definition 5** (Reachability Problem). *Given an initial configuration  $\vec{A}$ , a destination (target) configuration  $\vec{B}$ , and a step CRN  $\mathcal{C}_S = ((\Lambda, \Gamma), (\vec{S}_0 = \vec{A}, \vec{S}_1, \vec{S}_2, \dots, \vec{S}_{k-1}))$ , determine if  $\vec{B} \in \text{REACH}_k$ , i.e., is configuration  $\vec{B}$  reachable for the given step CRN. In the case of basic CRNs, this simplifies to: given configurations  $\vec{A}$  and  $\vec{B}$ , and basic CRN  $\mathcal{C} = (\Lambda, \Gamma)$ , determine if  $\vec{B} \in \text{REACH}_{\vec{A}, \Lambda, \Gamma}$ .*

## 3 Membership in NP for void rule systems

We initiate our study of deletion-only systems by observing that reachability stays within the class NP with only void rules, even with step-CRNs. This is straightforward to see in the case of polynomial bounded volume (unary encoded species counts) as each rule reduces the system volume by at least 1. But in the case of binary encoded species counts, the argument is more subtle as such deletion sequences could be exponentially long. To deal with this issue, we use the following *rearrangement* lemma that states that any order of void rule applications can be rearranged such that all applications of a given rule occur in a contiguous sequence. Given this lemma, any sequence of void rule applications can be rearranged into a sequence that can be encoded and verified in polynomial time.

► **Lemma 6** (Rearrangement Lemma). *For any sequence of applicable void rules  $A$ , there exists a sequence  $B$  that is a permutation of  $A$  such that all applications of a given rule type occur contiguously.*

**Proof.** Consider a sequence of applicable void rules  $A$  that is not contiguous. Suppose rule  $x$  occurs at positions  $i$  and  $j$  in  $A$ ,  $i < j - 1$ , and there is at least one non- $x$  rule in between them. Construct a new sequence  $A'$  by shifting the  $x$  rule at position  $i$  up to position  $j - 1$ , and shifting all rules in between down one position. This new sequence must be applicable

as the only rule that moved to a higher index in the sequence is of type  $x$ , and we know that  $x$  is still applicable at position  $j - 1$  since  $x$  is known to be applicable at position  $j$ . As this swapping preserves the applicability of the sequence while reducing the number of non-contiguous blocks of one rule type in the sequence, we can repeat this process of swapping rule positions until the sequence is contiguous. ◀

This lemma implies the existence of a polynomial-time verifiable certificate for “yes” instances of the reachability problem for step-CRNs, giving us membership in NP.

► **Theorem 7.** *The reachability problem for step-CRNs with void rules is in NP.*

**Proof.** As a certificate, we utilize a contiguous sequence of applicable rules for each step of the CRN, which must exist by Lemma 6. This sequence, while potentially exponential in length, can be encoded with a sequence of rule types accompanied by a count on the number of applications of each rule type. The result of such a sequence can be computed in polynomial time and therefore can serve as a certificate for the reachability problem. ◀

## 4 Bimolecular Rules of Uniform-type: With or Without Catalysts

In this section, we focus on bimolecular systems with either all size- $(2, 0)$  rules (non-catalytic) or  $(2, 1)$  rules (catalytic). Recently, size- $(2, 0)$  rule reachability in a single step was proven to be polynomial [1]. For size- $(2, 1)$  1-step systems, we present polynomial-time algorithms for reachability. In contrast, we show that in either scenario the problem becomes NP-complete with the addition of a second step. Later in Section 5, we consider the scenario of CRNs that use both  $(2, 0)$  and  $(2, 1)$  rules together.

### 4.1 Bimolecular Void Rules Without a Catalyst: $(2, 0)$

In [1], the authors proved that reachability in a CRN system with only size- $(2, 0)$  rules is in P by reducing from the perfect  $b$ -matching problem, which is a generalization of matching. This takes the form of a traditional matching when all  $b$ -values are 1, and an uncapacitated  $b$ -matching occurs when all edge capacities are assigned  $u(e) = \infty$ .

► **Theorem 8 ([1]).** *Reachability for basic CRNs with binary encoded species with only rules of size  $(2, 0)$  is solvable in  $O(|\Lambda|^2 \log(|\Lambda|)(|\Gamma| + |\Lambda| \log(|\Lambda|)))$  time.*

We now look at the problem with only one additional step, and show that it becomes NP-complete by reducing from the graph 3-colorability (3-COL) problem. Given an instance  $\langle G \rangle$ , where  $G = (V, E)$  is an undirected graph, 3-COL asks if each vertex of  $G$  can be assigned one of three colors such that no adjacent vertices share the same color. An instance of 3-COL  $\langle G \rangle$  can be converted into a  $(2, 0)$  2-step CRN  $\mathcal{C}_S$  as follows.

**Species.** For each vertex  $v \in V$ , we create the species  $v$ ,  $v_R$ ,  $v_G$ , and  $v_B$ .  $v_C$  represents an assignment of color  $C \in \{R, G, B\}$  to vertex  $v$ ; the  $v$  species will be used to represent assigning only one color to vertex  $v$  through specific reactions. We also create the species  $X$  to verify that a corresponding color assignment of  $G$  in  $\mathcal{C}_S$  has no adjacent vertices that share a color.

**Steps and Rules.** In step one (or  $\vec{S}_0$ ), for each  $v \in V$ , we add two copies of  $v$  and one copy of  $v_R$ ,  $v_G$ , and  $v_B$ . We also create the *assignment* rule  $v + v_C \rightarrow \emptyset$  for each  $C \in \{R, G, B\}$ . Two of the three assignment rules created for  $v$  are applied to its respective species copies, consuming all  $v$  copies and two of the three copies of  $v_C$ . The remaining  $v_C$  copy then corresponds to assigning vertex  $v$  the color  $C$ . Additionally, for each edge  $(i, j) \in E$  and  $C \in \{R, G, B\}$ , we create the *edge* rule  $i_C + j_C \rightarrow \emptyset$ . If the remaining copy of both  $i_C$  and  $j_C$  share a color  $C$ , they will be deleted by one of the edge rules.

In the second step ( $\vec{S}_1$ ), we introduce  $|V|$  copies of the species  $X$ . We also construct the *verification* rule  $v_C + X \rightarrow \emptyset$  for each  $v \in V$  and  $C \in \{R, G, B\}$ . All existing copies of  $v_C$  will be consumed by a  $X$  species. Thus, any remaining copies of  $X$  in the terminal configuration indicates that some  $v_C$  copies were deleted by an edge rule.

► **Theorem 9.** *Reachability for 2-step CRNs with only rules of size-(2, 0) is NP-complete, even for unary encoded species counts.*

**Proof.** We reduce from the graph 3-colorability problem. Given an instance of 3-COL  $\langle G \rangle$ , we convert  $G$  into a 2-step  $(2, 0)$  CRN  $\mathcal{C}_S$ , following the construction outlined above, and set  $\vec{A} = \vec{S}_0$  and  $\vec{B}$  to the *empty configuration*  $\vec{0}$ .

**Forward Direction.** Assume there exists a color assignment in  $G$  where no adjacent vertices share a color. By the construction of  $\mathcal{C}_S$ , a sequence of assignment rules can be applied in  $\vec{A}$  that results in a configuration of one copy of  $v_C$  for each vertex that matches the color assignment in  $G$ . Denote this new configuration  $\vec{S}'_0$ . Since no adjacent vertices share a color in  $G$ , no edge rule will be applied in  $\vec{S}'_0$ , keeping the count of the  $v_C$  copies to  $|V|$ .  $\mathcal{C}_S$  then transitions to  $\vec{S}_1$ , introducing the  $|V|$   $X$  copies. Since  $|V|$   $v_C$  copies were preserved, all  $v_C$  and  $X$  copies are deleted by verification reactions to reach the final configuration  $\vec{0} = \vec{B}$ .

**Reverse Direction.** Assume there exists a sequence of applicable rules in  $\mathcal{C}_S$  that reaches  $\vec{B}$  from  $\vec{A}$ . First, removing all  $v$  species can only be accomplished by applying a sequence of assignment rules. The resulting configuration  $\vec{S}'_0$  is  $|V|$   $v_C$  copies. Assume no edge rule can be applied in  $\vec{S}'_0$ . By the construction of  $\mathcal{C}_S$ , this implies that the matching color assignment in  $G$  also does not have adjacent vertices sharing colors. We then add  $|V|$   $X$  copies at the second step, resulting in the deletion of all  $v_C$  and  $X$  copies in the system by the verification rules, resulting in a final terminal configuration of  $\vec{0}$ . If an edge rule was applied in  $\vec{S}'_0$ , at least two  $v_C$  copies were removed, causing the final count of  $X$  to be greater than 0. Note that applying an edge rule *before* an assignment rule also guarantees  $\vec{0}$  cannot be reached, as either 1) the assignment rules for the affected  $v_C$  copies are then applied, removing those copies and consequentially preventing some  $X$  copies from being deleted, or 2) more edge rules are applied on the affected  $v_C$  copies, which prevents all  $v$  copies from being deleted. Therefore, the only way for  $\mathcal{C}_S$  to reach  $\vec{0} = \vec{B}$  is for a color assignment to exist on all vertices in  $G$  where no adjacent vertices share a color.

Theorem 7 shows reachability with void step CRN systems to be in NP. ◀

## 4.2 Bimolecular Void Rules with Catalyst: (2, 1)

In this section, we first show that reachability for size-(2, 1) void rules resides in the class  $NL$ .

► **Lemma 10.** *Let the implication graph  $G$  of a CRN  $(\Lambda, \Gamma)$  with size-(2, 1) void rules be the graph where each node is a species  $\lambda_i$  and each reaction  $\lambda_i + \lambda_j \rightarrow \lambda_k$  implies a directed edge from  $\lambda_i$  to  $\lambda_k$ . A configuration  $\vec{B}$  is reachable from  $\vec{A}$  if and only if for each species  $\lambda_i$  there exists a path to a node  $\lambda_r$  that holds one of the following properties:*



1.  $\vec{A}[r] = \vec{B}[r] > 0$ ,
2.  $\lambda_r + \lambda_j \rightarrow \lambda_j \in \Gamma$  where  $\vec{B}[j] \geq 1$ , or
3.  $\vec{B}[r] \geq 1$  and  $\lambda_r + \lambda_r \rightarrow \lambda_r \in \Gamma$

**Proof.** We will refer to a node which satisfies one of these conditions as a *root node*. A path from species  $\lambda_i$  to a root node  $\lambda_r$  means that we can delete enough copies of  $\lambda_i$  to reach the target configuration. We will prove this recursively, inducing over the length of the path from  $\lambda_i$  to  $\lambda_r$ , to create a reaction sequence through this process with our base case being the end of the sequence.

For our base case, any node  $\lambda_i = \lambda_r$  can reach the target amount if it satisfies a condition in the Lemma statement. In Case 1, the number of species in the starting configuration is already the target amount so the claim is trivially true. In Case 2, we may use copies of the species  $\lambda_j$  to delete  $\lambda_i$  using the leftover species in the target configuration. In Case 3, the species may delete itself to reach the target amount.

For our inductive case, assume that there exists a reachable configuration such that any species  $\lambda_k$  with a shortest path of length  $\leq l$  to a node  $\lambda_r$  can be reduced to the target amount  $\vec{B}[k]$ . For a species  $\lambda_i$  with a shortest path of length  $l + 1$ , there exists an edge to a species  $\lambda_k$  with length  $l$ . We can use the reaction  $\lambda_i + \lambda_k \rightarrow \lambda_k$  to decrease  $\lambda_i$  to  $\vec{B}[i]$  copies at the start of the current sequence. It remains to prove that removing these copies does not affect anything later in the sequence. If the closest node  $\lambda_r$  falls under case 1 or 3 then removing  $\lambda_i$  does not affect it. If the closest node is case 2 and  $\lambda_i = \lambda_j$ , the species  $\lambda_i$  is the one used to remove  $\lambda_r$ ; thus, the condition  $\vec{B}[j] \geq 1$  means that we leave at least a single copy in the configuration that can be used to delete  $\lambda_r$ .

If a node does not have a path to a root node, then it either does not have any outgoing edges, or all of its outgoing edges are part of some cycle where all nodes along each cycle have a target count of 0. As a result, if the node does not fall into case 1, there is no way of reducing the respective species to its target count without leaving some other species unsatisfied. ◀

► **Theorem 11.** *Reachability for basic CRNs with size  $(2, 1)$  void rules is in NL.*

**Proof.** We will show that we can decide whether a node has no path to a root in log space; thus reachability is in  $\text{coNL} = \text{NL}$ . We non-deterministically check a species  $\lambda_i$ , then for every node  $\lambda_j$  reachable from  $\lambda_i$  in the implication graph, we check if  $\lambda_j$  satisfies any of the conditions in Lemma 10. If we do not find such a node  $\lambda_j$ , then we reject.

If any node does not have a path to a root node, then some branch of this algorithm will reject. Checking each path can be done in NL as this is a directed graph. Checking if a node is a root node can be done in log space as it only involves edge queries and queries to the target configuration. ◀

We now show that adding an extra step turns the problem NP-complete, as with  $(2, 0)$  CRN systems. Here, we reduce from the classic 3SAT problem. Given an instance of 3SAT  $\langle \Phi \rangle$ , we construct a  $(2, 1)$  2-step CRN  $\mathcal{C}_S$  as follows.

**Species.** For each variable  $x_i$ , we create a pair of species  $T_i$  and  $F_i$ , which represents assigning  $x_i$  a value of true or false, respectively. Additionally, for each clause  $c_j$ , we create the species  $C_j$ . The presence of a copy of  $C_j$  in a configuration of  $\mathcal{C}_S$  indicates  $c_j$  has yet to be satisfied by one of its assigned variables. Finally, we create the species  $X$  for “clean-up” procedures.

**Steps and Rules.** In the first step ( $\vec{S}_0$ ), for each variable  $x_i$ , we introduce one copy of  $T_i$  and  $F_i$ . We also create a pair of *assignment* rules, one to represent assigning true to  $x_i$  ( $T_i + F_i \rightarrow T_i$ ) and one for assigning false ( $T_i + F_i \rightarrow F_i$ ). One of two assignment reactions will be applied to the copies of  $T_i$  and  $F_i$ ; the non-deleted copy represents assigning  $x_i$  the corresponding boolean value.

In the second step ( $\vec{S}_1$ ), we add a single copy of a clause species  $C_j$  for each clause  $c_j$  and a single copy of  $X$ . Additionally, given a clause  $c_j$  and a variable of the clause  $x_k \in (x_a, x_b, x_c)$ , 3 separate *verification* rules created of the form  $C_j + T_k \rightarrow T_k$  (for non-negated variables) or  $C_j + F_k \rightarrow F_k$  (for negated variables). If  $T_k/F_k$  is still present in  $\mathcal{C}_S$ , then  $C_j$  will be consumed by that species. Finally, for each variable  $x_i$ , we create the *cleaning* rules  $X + T_i \rightarrow X$  and  $X + F_i \rightarrow X$  to consume all present copies of  $T_i$  and  $F_i$ . Any  $C_j$  species remaining in  $\mathcal{C}_S$  after the application of the cleaning rules indicates that it could not be deleted by a verification rule.

► **Theorem 12.** *Reachability for 2-step CRNs with only rules of size (2, 1) is NP-complete, even for unary encoded species counts.*

**Proof.** We reduce from 3SAT. Given an instance of 3SAT  $\langle \Phi \rangle$ , we convert  $\Phi$  into a 2-step (2,1) CRN  $\mathcal{C}_S$  via the reduction from above. Let  $\vec{A} = \vec{S}_0$  and  $\vec{B}$  be a single copy of  $X$  ( $\vec{X}$ ).

**Forward Direction.** Assume there exists an assignment of variables that satisfies  $\Phi$  to true. A sequence of assignment reactions can be then performed in  $\vec{S}_0$  that results in a configuration with only one  $T_i/F_i$  copy for each variable that matches the variable assignment. In the second step, by the construction of  $\mathcal{C}_S$ , since the assignment satisfies  $\Phi$ , each introduced copy of  $C_j$  can be deleted with a verification reaction. Finally, the added  $X$  copy deletes all remaining literal species. The final configuration of  $\mathcal{C}_S$  is then  $\vec{X}$ .

**Reverse Direction.** Assume there exists a sequence of applicable rules in  $\mathcal{C}_S$  that reaches  $\vec{B}$  from  $\vec{A}$ . First, a sequence of assignment reactions can be performed in  $\vec{S}_0$  that consumes one of the  $T_i$  and  $F_i$  copy for each variable. We then add one copy of each  $C_j$  species and one copy of  $X$  in the second step. Assume all  $C_j$  copies can be consumed by a verification reaction. By the construction of  $\mathcal{C}_S$ , this implies that there exists an assignment of variables in  $\Phi$  that evaluates the formula to true. The  $X$  copy can delete the remaining  $X_i/F_i$  copies with cleanup rules to reach the final configuration  $\vec{X}$ . If a  $C_j$  copy could not be removed, this implies that the variable assignment couldn't satisfy the corresponding clause. Therefore, the only way for  $\mathcal{C}_S$  to reach  $\vec{X} = \vec{B}$  is for an assignment of variables in  $\Phi$  to exist that satisfies all clauses of the formula.

Theorem 7 shows reachability with void step CRN systems to be in NP. ◀

## 5 Bimolecular Rules of Mixed Type: (2,0) and (2,1)

In this section, we show that the reachability problem for general single-step bimolecular void rules systems that include a mix of non-catalytic (2,0) rules and catalytic (2,1) rules is in P. We show this via a reduction to the perfect  $b$ -matching problem in an undirected graph.

► **Definition 13** (Perfect  $b$ -matching Problem). *Given a graph  $G = (V, E)$ ,  $u : e \in E \rightarrow \mathbb{N} \cup \{\infty\}$  to be edge capacities, and  $b : v \rightarrow \mathbb{N}$  to be the number of matchings a vertex can take, does there exists an assignment to the edges  $f : e \rightarrow \mathbb{N}$  such that  $f(e) \leq u(e)$  and  $\sum_{e \in \delta(v)} f(e) = b(v)$  for all  $v \in V$ ?*

Unlike reducing from reachability to perfect  $b$ -matching with only (2,0) rules in [1], the inclusion of catalyst rules requires a substantially more involved reduction. We therefore begin with a brief overview of our reduction and then describe each step in greater detail. Finally, we follow with a proof of correctness and a runtime analysis for the entire result.



## Overview

Our reduction transforms an instance of CRN reachability  $\langle \mathcal{C}, \vec{A}, \vec{B} \rangle$  into an instance of the perfect  $b$ -matching problem. The key idea is to identify which species involved in catalytic  $(2, 1)$  rules can be fully deleted using just catalytic rules and which must be further deleted using non-catalytic  $(2, 0)$  rules. If this distinction were known in advance, it would be straightforward to construct a corresponding graph and solve reachability via a matching instance.

To infer this structure, we first construct a directed graph  $T$ , which we refer to as the *catalytic deletion graph*, where each vertex corresponds to a species  $\lambda_i$  and each directed edge  $(\lambda_i, \lambda_j)$  represents a  $(2, 1)$  rule  $\lambda_i + \lambda_j \rightarrow \lambda_j$  that catalytically deletes  $\lambda_i$  using  $\lambda_j$ . We then compute the strongly connected components (SCC's) of  $T$  and build a condensation graph  $H$  whose nodes each represent a component of  $T$ . The structure of  $H$  allows us to identify which catalytic species *must* be completely deleted via a non-catalytic  $(2, 0)$  rule.

Using this information, we construct an undirected graph  $G$  whose nodes effectively represent species that must be deleted via  $(2, 0)$  or  $(2, 1)$  rules and whose edges represent those corresponding rules. We then formulate a perfect  $b$ -matching instance on  $G$  where a perfect matching exists exactly when there exists a valid sequence of  $(2, 0)$  and  $(2, 1)$  rules that complements the catalytic deletions to reach the target configuration  $\vec{B}$ .

## Creating Catalytic Deletion Graph $T$

Given a CRN  $\mathcal{C} = (\Lambda, \Gamma)$ , we construct the directed graph  $T = (V, E)$  as follows. For each catalyst void rule  $\lambda_i + \lambda_j \rightarrow \lambda_j \in \Gamma$ , if  $\vec{A}[i], \vec{A}[j] > 0$ , we create vertices  $\lambda_i$  and  $\lambda_j$  and the directed edge  $(\lambda_i, \lambda_j)$ .

The intuition of  $T$  is that an edge from  $\lambda_i$  to  $\lambda_j$  represents the species  $\lambda_i$  being deleted by the catalyst species  $\lambda_j$ . It then follows that a species whose respective vertex in  $T$  has an out-degree of 0 can only be deleted by a  $(2, 0)$  rule. We label these vertices as *mandatory vertices*. We also consider cycles in  $T$  in which each vertex (species) only has an out-going edge to another vertex in the cycle. If the count of all represented species in the cycle in  $\vec{B}$  is zero, then regardless of the application of rules corresponding to the edges of the cycle, there is guaranteed to be at least one remaining species left that can only be completely removed by a  $(2, 0)$  rule. We label these cycles as *mandatory cycles*.

► **Definition 14** (Mandatory Vertices). *A vertex in  $T$  with an out-degree of 0.*

► **Definition 15** (Mandatory Cycles). *A cycle in  $T$  in which each vertex 1) only has an out-going edge to another vertex in the cycle, and 2) has a corresponding species count of 0 in  $\vec{B}$ .*

## Creating SCC Condensation Graph $H$

Given a directed graph  $T = (V, E)$ , we construct the directed graph  $H = (V', E')$  as follows. First, we run Tarjan's Strongly Connected Component Algorithm on  $T$ , which returns a partition of  $T$ 's vertices of strongly connected components  $C = \{c_1, c_2, \dots, c_n\}$  [30]. For each component  $c_i \in C$ , we create the vertex  $c_i$ . For each directed edge from  $c_i$  to another component  $c_j$ , we create the directed edge  $(c_i, c_j)$ .

► **Observation 16.** *A vertex in  $H$  represents a mandatory vertex if it is not a condensed component of  $T$  and it has an out-degree of 0.*

► **Observation 17.** *A vertex in  $H$  represents a mandatory cycle if it is a condensed component of  $T$  in which all corresponding species have final counts of 0, and it has an out-degree of 0.*

### Creating $b$ -matching Instance Graph $G$

Given a CRN  $\mathcal{C} = (\Lambda, \Gamma)$ , configurations  $\vec{A}$  and  $\vec{B}$ , and directed graphs  $T = (V, E)$  and  $H = (V', E')$ , we create an instance of the perfect  $b$ -matching problem with the graph  $G = (V'', E'')$  as follows. Let the *difference configuration*  $\vec{D} = \vec{A} - \vec{B}$ .

**Creating  $V''$  and  $b(\cdot)$ .** For each species  $\lambda_i \in \Lambda$ , if  $\vec{D}[i] > 0$ , we create the vertices  $\lambda_{i1}$  and  $\lambda_{i2}$  and set both  $b(\lambda_{i1})$  and  $b(\lambda_{i2})$  to  $\vec{D}[i]$ . These vertices represent the number of copies of  $\lambda_i$  that must be removed from  $\vec{A}$  by the void rules. Additionally, if  $\vec{B}[i] > 0$ , we create the vertices  $\bar{s}_{i1}$  and  $\bar{s}_{i2}$  and set both  $b(\bar{s}_{i1})$  and  $b(\bar{s}_{i2})$  to  $\vec{B}[i]$ . These vertices exist just to “set aside” the final configuration for matchings, hence the bar labels.

We now consider species that can be deleted by catalyst void rules. For each catalytic rule  $\lambda_i + \lambda_j \rightarrow \lambda_j \in \Gamma$ , if  $\vec{A}[i], \vec{A}[j] > 0$  and its corresponding edge in  $T$  is not part of a cycle, we create the vertices  $\lambda'_{i1}$  and  $\lambda'_{i2}$ , if not already created, and set both  $b(\lambda'_{i1})$  and  $b(\lambda'_{i2})$  to  $\vec{D}[i]$ . Additionally, given a vertex of  $H$   $c_i \in V'$ , if  $c_i$  represents a condensed cycle  $\{\lambda_1, \dots, \lambda_n\}$ , we create the vertices  $c'_{i1}$  and  $c'_{i2}$ . If the cycle is mandatory, we assign  $b(c'_{i1})$  and  $b(c'_{i2})$  the value  $(\sum_{\lambda_i \in c_i} \vec{D}[i]) - 1$ ; else they are assigned  $(\sum_{\lambda'_i \in c_i} \vec{D}[i])$ , where  $\lambda'_i$  is a *non-mandatory* vertex of  $c_i$ . These vertices represent a choice to delete a species  $\lambda_i$  using a catalytic species.

Let the vertices with  $b$ -values from  $\vec{D}$  be the sub-graph  $G_D$ , and the vertices with  $b$ -values from  $\vec{B}$  be the sub-graph  $G_B$ .

**Creating  $E''$  and  $u(\cdot)$ .** For each  $(2, 0)$  rule  $\lambda_i + \lambda_j \rightarrow \emptyset \in \Gamma$ , if the vertices for both species were created in  $G$ , we create the edges  $(\lambda_{i1}, \lambda_{j1})$  and  $(\lambda_{i2}, \lambda_{j2})$ . Performing a matching on these edges corresponds to deleting  $\lambda_i$  and  $\lambda_j$  by a  $(2, 0)$  rule.

For each  $(2, 1)$  rule  $\lambda_i + \lambda_j \rightarrow \lambda_j \in \Gamma$ , if  $\lambda'_{i1}$  and  $\lambda'_{i2}$  were created in  $G$  and the rule is not part of a cycle in  $T$ , we create the edges  $(\lambda_{i1}, \lambda'_{i1})$  and  $(\lambda_{i2}, \lambda'_{i2})$ . For each vertex of  $T$  that represents a condensed cycle  $c_i = \{\lambda_1, \dots, \lambda_n\}$ , if the cycle is mandatory, we create the edges  $(\lambda_{k1}, c'_{i1})$  and  $(\lambda_{k2}, c'_{i2})$  for all  $s_k \in c_i$ . Otherwise, we only create  $(\lambda_{k1}, c'_{i1})$  and  $(\lambda_{k2}, c'_{i2})$  for the *non-mandatory* vertices of  $c_i$ . Matching the edges represents deleting a species  $\lambda_i$  with a catalyst rule.

We finally create the following edges: for all  $\bar{s}_{i1}$  and  $\bar{s}_{i2}$  vertices, create the edge  $(\bar{s}_{i1}, \bar{s}_{i2})$ , for all  $\lambda'_{i1}$  and  $\lambda'_{i2}$  vertices, create the edge  $(\lambda'_{i1}, \lambda'_{i2})$ , and for all  $c'_{i1}$  and  $c'_{i2}$  vertices, create the edge  $(c'_{i1}, c'_{i2})$ . Matching on these edges does not represent a rule application, but rather ensures a perfect  $b$ -matching can be performed on these vertices even if they were not perfectly matched by other  $(2, 1)$  edges. For all edges  $e \in E''$ , assign  $u(e) = \infty$ .

### Result

The overall effect is that  $G$  has a perfect  $b$ -matching exactly when configuration  $\vec{B}$  is reachable from configuration  $\vec{A}$  which yields our main theorem from this section.

► **Theorem 18.** *Reachability in void rule systems with  $(2, 0)$  and  $(2, 1)$  rules is solvable in  $O(|\Lambda|^2 \log(|\Lambda|)(|\Gamma| + |\Lambda| \log(|\Lambda|)))$  time.*

Due to space, the full proof can be found in Appendix A.

## 6 Larger Void Rules

Our next results involve CRNs with reactions that require more than two reactants. If a system's rules have all but one reactant serving as catalysts (i.e.,  $(k, k-1)$  void rules), then reachability remains polynomial-time solvable. In contrast, reachability for systems with any other form of void rule (with 3 or more reactants) becomes NP-complete.

### 6.1 Mostly-Catalytic Large Void Rules of Mixed-type: $(k, k-1)^+$

We provide a polynomial-time dynamic programming algorithm to decide reachability for CRNs that use mostly-catalytic void rules of the form  $(k, k-1)$ . We further argue that reachability remains in P, even for CRNs that use a combination of various size  $(k, k-1)$ . For simplicity, we refer to void rules of sizes  $(k_1, k_1-1), \dots, (k_b, k_b-1)$ , where all  $k_i \in \mathbb{N}$ , as  $(k, k-1)^+$ , meaning there is one or more rule of this type.

► **Lemma 19.** *Reachability for basic CRNs with only void rules of size  $(k, k-1)$  requires at most  $|\Lambda|$  distinct rules.*

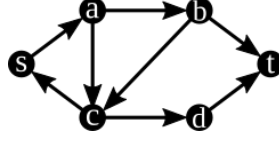
**Proof.** For simplicity, let  $n = |\Lambda|$ . Assume there exists a sequence of reactions  $a_1 r_1, \dots, a_{n+1} r_{n+1}$  for a CRN  $\mathcal{C}$  with set of species  $\Lambda$  and set of rules  $\Gamma$  that takes some initial configuration  $\vec{A}$  to configuration  $\vec{B}$ , where  $a_1, \dots, a_{n+1}$  are positive integers (denoting how many times to apply each rule) and  $r_1, \dots, r_{n+1}$  are rules in  $\Gamma$ . There must then exist some species  $s$  that gets consumed by 2 rules  $r_i$  and  $r_j$ , where  $i < j$ . Let  $s^i, s^x$  and  $s^f$  denote the initial, intermediate and final counts of species  $s$ , where  $r_i$  reduces  $s$  from  $s^i$  to  $s^x$  and  $r_j$  reduces  $s$  from  $s^x$  to  $s^f$ . Since  $s^i > s^x$ , any rule  $r_l$ , where  $l > i$ , that uses  $s$  with count  $s^x$  can also use  $s$  with count  $s^i$ . Thus, rule  $r_i$  is not needed. ◀

► **Theorem 20.** *Reachability for basic CRNs with only void rules of size  $(k, k-1)$  is solvable in  $O(|\Lambda|^2 |\Gamma|)$ .*

**Proof.** Let  $\Gamma$  denote the set of rules. We can use a dynamic programming approach to solve the problem. Construct an  $|\Lambda| \times (|\Lambda| + 1)$  table  $D(s, j)$  of boolean entries, where each row represents a different species. Reduce the count of each species to  $\max(k, s^f)$ , where  $s^f$  represents the final count of species  $s$ , if there exists a rule that can do so. Starting from the first column  $j = 0$ , place a 1 if the respective species is already in its final count. Then, for each entry  $D(s, j)$ , place a 1 if  $D(s, j-1)$  is a 1 or if there exists a reaction  $\gamma \in \Gamma$  that reduces  $s$  to its final count, where all the reactants of  $\gamma$  have either reached their final counts or will not prevent the reaction from occurring once they do. If column  $|\Lambda| + 1$  contains all 1's, then reachability is possible. Otherwise, it is not.

By Lemma 19, a solution to the problem requires at most  $|\Lambda|$  unique reactions, where each reaction directly reduces each species from its initial counts to its final counts. Thus, finding a solution to the problem takes at most  $|\Lambda|$  steps since we are implicitly selecting at least one reaction per column. This results in  $|\Lambda| + 1$  columns, with the first column representing the initial configuration.

Since every rule is a catalytic void rule, there must exist an ordering of reactions such that some reactions can occur first without impeding other species from getting reduced to their final counts. A bottom-up approach can be used to find this ordering, starting with the reactions that can be put off until later and working up to the reactions that must occur first. In table  $D$ , this ordering is implicitly represented between columns, with the reactions between the rightmost columns being the ones we do first. Any species with final count



■ **Figure 3** Directed graph used in the example  $(3, 1)$  reduction in Section 6.2.

greater than  $k$  is reduced before the algorithm is run if there exists an applicable rule, as reducing this species count does not prevent any other rule from occurring. Filling  $D$  takes at most  $O(|\Lambda|^2|\Gamma|)$  steps. Hence, reachability is solvable in polynomial time. ◀

► **Theorem 21.** *Reachability for basic CRNs with only void rules of size  $(k, k-1)^+$  is solvable in  $O(|\Lambda|^2|\Gamma|)$ .*

**Proof.** This follows from Theorem 20. Since we are considering a rule set  $\Gamma$  where the size of each rule  $\gamma \in \Gamma$  is  $\leq k$ , reducing each initial species count to  $\max(k, s^f)$  guarantees that any  $r$  that needs  $s$  will be able to occur. The algorithm remains the same. ◀

► **Corollary 22.** *Reachability for 2-step CRNs with only void rules of size  $(k, k-1)^+$  is NP-complete, even for unary encoded species counts.*

**Proof.** Follows from Theorem 12. ◀

## 6.2 Large Void Rules of Uniform-type: $(k \geq 3, g \leq k-2)$

Here we show that reachability for CRNs using any void rules with at least three reactants, and which remove at least two species, becomes NP-complete. To achieve this result, we show that reachability is NP-complete for CRNs using only  $(3, 1)$  void rules via a reduction from the Hamiltonian path problem for directed graphs. Since it was previously shown that reachability is NP-complete for CRNs using only  $(3, 0)$  rules [1], this implies Corollary 24.

We reduce from the Hamiltonian path problem in directed graphs. For simplicity, we reduce from the variation where each vertex has max in-degree and out-degree of two, which is still NP-complete [24]. Given an instance of the problem  $\langle G, s, t \rangle$ , we outline the species and the reactions for the reduction, and show correctness in the proof.

**Species.** We create a species for every path through a vertex (unless it starts at  $t$  or goes back to  $s$ ). For example, looking at Figure 3, for vertex  $b$ , we create two species  $S_{a,b,c}$  and  $S_{a,b,t}$ , where the notation denotes the vertex it came from, the vertex itself, and the vertex it goes to. With a maximum in/out degree of two, for each vertex, there are at most 4 paths through the vertex. In general, for each vertex  $j \in V$ , we create the species  $S_{i,j,k}$  where  $(i, j), (j, k) \in E$  and  $k \neq s, i \neq t$ . For  $s, t$ , we just have  $S_s$  and  $S_t$ . We also create a species  $P_i$  for each  $i \in V$  where  $i \neq s$ . Thus, we have  $n-1$  of these species.

**Reactions.** There are two main types of reactions: those that pick which path through a vertex and those that walk the Hamiltonian path.

- Since there are at most 4 species per vertex, we will create rules that create a tournament to choose one of the species for each vertex. Assuming 3 species for ease of explanation, we create the rules  $S_1 + S_2 + S_3 \rightarrow S_1$ ,  $S_1 + S_2 + S_3 \rightarrow S_2$ , and  $S_1 + S_2 + S_3 \rightarrow S_3$ . With only two species, we can create a dummy species that is not used as a catalyst. With 4 species, we augment the case of 3 species with another reaction that must occur with a dummy species ( $S_d$ ) and the previous choice. For each choice  $S_i$  with  $i \in \{1, 2, 3\}$ , we create the rules  $S_i + S_4 + S_d \rightarrow S_i$  and  $S_i + S_4 + S_d \rightarrow S_4$ .

- We create rules that walk the Hamiltonian path with valid connecting species and a “fuel” species ( $P_i$ ) so that we can only visit a vertex one time. We create each of the rules  $S_{i,j,k} + S_{j,k,l} + P_k \rightarrow S_{j,k,l}$  where  $i, j, k, l \in V$  and  $i \neq t, l \neq s$ . This rule indicates a walk from  $j$  to  $k$  along a valid edge and removes the  $P_k$  token to mark the vertex as visited.

**Example.** For Figure 3, we give the full reduction as follows. The final configuration  $\vec{S}_t$  (just a single copy of  $S_t$ ) is only reachable if there is a Hamiltonian path.

- The main species are  $S_s, S_{s,a,b}, S_{s,a,c}, S_{a,b,c}, S_{a,b,t}, S_{a,c,d}, S_{b,c,d}, S_{c,d,t}, S_t, P_a, P_b, P_c, P_d$ , and  $P_t$ . We also create a single copy each of dummy species  $D_a, D_b, D_c$ .
- The tournament reactions for each vertex are (d has only one path)
  - a)  $S_{s,a,b} + S_{s,a,c} + D_a \rightarrow S_{s,a,b}$  and  $S_{s,a,b} + S_{s,a,c} + D_a \rightarrow S_{s,a,c}$ ,
  - b)  $S_{a,b,c} + S_{a,b,t} + D_b \rightarrow S_{a,b,c}$  and  $S_{a,b,c} + S_{a,b,t} + D_b \rightarrow S_{a,b,t}$ ,
  - c)  $S_{a,c,d} + S_{b,c,d} + D_c \rightarrow S_{a,c,d}$  and  $S_{a,c,d} + S_{b,c,d} + D_c \rightarrow S_{b,c,d}$ .
- The walking reactions for each vertex are
  - a)  $S_s + S_{s,a,b} + P_a \rightarrow S_{s,a,b}$  and  $S_s + S_{s,a,c} + P_a \rightarrow S_{s,a,c}$ ,
  - b)  $S_{s,a,b} + S_{a,b,c} + P_b \rightarrow S_{a,b,c}$  and  $S_{s,a,b} + S_{a,b,t} + P_b \rightarrow S_{a,b,t}$ ,
  - c)  $S_{s,a,c} + S_{a,c,d} + P_c \rightarrow S_{a,c,d}$  and  $S_{a,b,c} + S_{b,c,d} + P_c \rightarrow S_{b,c,d}$ ,
  - d)  $S_{a,c,d} + S_{c,d,t} + P_d \rightarrow S_{c,d,t}$  and  $S_{b,c,d} + S_{c,d,t} + P_d \rightarrow S_{c,d,t}$ ,
  - t)  $S_{a,b,t} + S_t + P_t \rightarrow S_t$  and  $S_{c,d,t} + S_t + P_t \rightarrow S_t$ .

► **Theorem 23.** *Reachability for CRNs with only void rules of size  $(3, 1)$  is NP-complete, even for unary encoded species counts.*

**Proof.** We reduce from the directed Hamiltonian path problem. Given an instance  $\langle G, s, t \rangle$ , we convert this to an instance of the reachability problem, as outlined above, for CRN  $\mathcal{C}$  and target configuration  $\vec{S}_t$ . We show that  $H$  is true iff the configuration  $\vec{S}_t$  is reachable in  $\mathcal{C}$ .

**Forward Direction.** Assume there exists a Hamiltonian path in  $G$  from  $s$  to  $t$ . Then it is possible that the tournament for every vertex correctly produces the species that represents the Hamiltonian path through the vertex. If this does occur, all species have been removed from the system except  $|V|$   $S$  species for the path and  $|V| - 1$   $P$  species. Then, the walking reactions can occur successively by destroying the previous path vertex and the “fuel” species, which will only leave one copy of  $S_t$ .

**Reverse Direction.** Assume there exists a sequence of applicable rules in  $\mathcal{C}_S$  that reaches  $\vec{B}$  from  $\vec{A}$ . The only way to remove an  $S$  species is through the tournament and the walking reactions. The tournament will always leave at least one  $S$  for each vertex, meaning the walking reactions must be used to delete the other  $|V| - 1$ . Along with this, the  $P$  vertices ensure that each vertex can only be visited once. Thus, the walk can not occur before the tournament and take multiple paths to reach a vertex. Thus, reaching a configuration with only  $S_t$  ensures that a walk through the graph occurred starting at  $S_s$ , ending at  $S_t$ , and that every vertex was visited.

All void CRN systems are in NP with the certificate being the sequence of rules to apply, and the number of times to apply them [1]. ◀

► **Corollary 24.** *Reachability for CRNs with only void rules of size  $(k \geq 3, g \leq k - 2)$  ( $k, g \in \mathbb{N}$ ) is NP-complete, even for unary encoded species counts.*

**Proof.** For  $g \geq 1$ , this follows from Theorem 23. For  $g = 0$ , this follows from the fact that reachability for  $(3, 0)$  rules is NP-complete [1]. ◀

## 7 Primary Results

We restate the primary results formally with the corresponding proofs. Although not discussed, for completeness, we also include the following lemma.

► **Lemma 25.** *Reachability for step CRNs (including basic CRNs) uniformly using size  $(1, 0)$  void rules is in  $P$ .*

**Proof.** Simply decrease each species to the desired count. Since each step must become terminal, all species in rules will be removed before the subsequent step. Thus, there must exist a step that adds counts greater than or equal to the target counts. Then treat that step as a basic CRN. If no such step exists, the target configuration is not reachable. ◀

The collection of results presented, as a whole, yields the following main theorems of this work that characterize void rules within CRNs and step CRNs.

► **Theorem 26.** *Reachability for basic CRNs uniformly using size  $(k \geq 3, g \leq k - 2)$  void rules is NP-complete, and is in  $P$  when uniformly using any other size void rule.*

**Proof.** This follows from [1], Theorems 11, 20, Corollary 24, and Lemma 25. ◀

► **Theorem 27.** *Reachability for 2-step CRNs with only void rules that use exactly one reactant is in  $P$ , and is NP-complete otherwise.*

**Proof.** This follows from [1], Theorems 9, 12, Corollaries 22, 24, and Lemma 25. ◀

► **Theorem 28.** *Reachability for basic CRNs that use a combination of void rule types is in  $P$  for combinations of  $(2, 0) + (2, 1)$  rules as well as  $(k, k - 1)^+$  rules, and is NP-complete for any combination that uses  $(k \geq 3, g \leq k - 2)$  rules.*

**Proof.** This follows from [1], Theorems 18, 21, and 24. ◀

## 8 Conclusion and Future Work

This paper presents a nearly complete classification of the computational complexity of reachability for CRNs and step CRNs that consist of deletion-only rules. We provide polynomial-time algorithms for most combinations of void rules in basic CRNs and show NP-completeness for rules of size greater than  $(k, g \leq k - 2)$  for  $k \geq 3$ . Additionally, we prove that with the addition of a single step, these problems become NP-complete. We include some natural open directions to explore:

- **Mixed-size Void Rule Systems.** What is the complexity of reachability when you consider void rules of size  $(2, 0)$  and  $(k, k - 1)$  together? This combination of void rule types is the missing piece that would complete the picture for the entire complexity landscape of void rule reachability.
- **Staged CRNs.** The step CRN model augments the basic CRN model with steps that add species once reactions are completed. A generalization of this model could have multiple “stages”, where CRNs are left to react and the results of these stages are combined. How do stages affect reachability?
- **Model Variants.** What is the complexity of reachability in deletion-only extensions of CRNs, petri-nets, and vector addition systems?



## References

- 1 Robert M. Alaniz, Bin Fu, Timothy Gomez, Elise Grizzell, Andrew Rodriguez, Robert Schweller, and Tim Wylie. Reachability in restricted chemical reaction networks, 2022. doi:10.48550/arXiv.2211.12603.
- 2 Rachel Anderson, Alberto Avila, Bin Fu, Timothy Gomez, Elise Grizzell, Aiden Massie, Gourab Mukhopadhyay, Adrian Salinas, Robert Schweller, Evan Tomai, and Tim Wylie. Computing threshold circuits with void reactions in step chemical reaction networks. In *10th conference on Machines, Computations and Universality (MCU 2024)*, 2024.
- 3 Rachel Anderson, Bin Fu, Aiden Massie, Gourab Mukhopadhyay, Adrian Salinas, Robert Schweller, Evan Tomai, and Tim Wylie. Computing threshold circuits with bimolecular void reactions in step chemical reaction networks. In *International Conference on Unconventional Computation and Natural Computation*, pages 253–268. Springer, 2024. doi:10.1007/978-3-031-63742-1\_18.
- 4 Dana Angluin, James Aspnes, Zoë Diamadi, Michael J. Fischer, and René Peralta. Computation in networks of passively mobile finite-state sensors. *Distributed Computing*, 18(4):235–253, March 2006. doi:10.1007/s00446-005-0138-3.
- 5 Rutherford Aris. Prolegomena to the rational analysis of systems of chemical reactions. *Archive for Rational Mechanics and Analysis*, 19(2):81–99, January 1965. doi:10.1007/BF00282276.
- 6 Rutherford Aris. Prolegomena to the rational analysis of systems of chemical reactions ii. some addenda. *Archive for Rational Mechanics and Analysis*, 27(5):356–364, January 1968. doi:10.1007/BF00251438.
- 7 Michael Blondin, Matthias Englert, Alain Finkel, Stefan Göller, Christoph Haase, Ranko Lazić, Pierre McKenzie, and Patrick Totzke. The reachability problem for two-dimensional vector addition systems with states. *Journal of the ACM (JACM)*, 68(5):1–43, 2021. doi:10.1145/3464794.
- 8 Adam Case, Jack H Lutz, and Donald M Stull. Reachability problems for continuous chemical reaction networks. *Natural Computing*, 17(2):223–230, 2018. doi:10.1007/S11047-017-9641-2.
- 9 Matthew Cook, David Soloveichik, Erik Winfree, and Jehoshua Bruck. *Programmability of Chemical Reaction Networks*, pages 543–584. Springer Berlin Heidelberg, Berlin, Heidelberg, 2009. doi:10.1007/978-3-540-88869-7\_27.
- 10 Wojciech Czerwiński, Sławomir Lasota, Ranko Lazić, Jérôme Leroux, and Filip Mazowiecki. The reachability problem for Petri nets is not elementary. *Journal of the ACM (JACM)*, 68(1):1–28, 2020. doi:10.1145/3422822.
- 11 Wojciech Czerwiński, Sławomir Lasota, and Łukasz Orlikowski. Improved Lower Bounds for Reachability in Vector Addition Systems. In *48th International Colloquium on Automata, Languages, and Programming (ICALP 2021)*, volume 198 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 128:1–128:15, Dagstuhl, Germany, 2021. Schloss Dagstuhl – Leibniz-Zentrum für Informatik. doi:10.4230/LIPIcs.ICALP.2021.128.
- 12 Wojciech Czerwiński and Łukasz Orlikowski. Reachability in vector addition systems is Ackermann-complete. In *62nd Annual Symposium on Foundations of Computer Science, FOCS’21*, pages 1229–1240. IEEE, 2021.
- 13 Daniel T Gillespie. Stochastic simulation of chemical kinetics. *Annu. Rev. Phys. Chem.*, 58(1):35–55, 2007.
- 14 Michel Henri Théodore Hack. *Decidability questions for Petri Nets*. PhD thesis, Massachusetts Institute of Technology, 1976.
- 15 John Hopcroft and Jean-Jacques Pansiot. On the reachability problem for 5-dimensional vector addition systems. *Theoretical Computer Science*, 8(2):135–159, 1979. doi:10.1016/0304-3975(79)90041-0.
- 16 Richard M. Karp and Raymond E. Miller. Parallel program schemata. *Journal of Computer and System Sciences*, 3(2):147–195, 1969. doi:10.1016/S0022-0000(69)80011-5.

- 17 Ulla Koppenhagen and Ernst W Mayr. Optimal algorithms for the coverability, the subword, the containment, and the equivalence problems for commutative semigroups. *Information and Computation*, 158(2):98–124, 2000. doi:10.1006/INCO.1999.2812.
- 18 Jérôme Leroux. The reachability problem for Petri nets is not primitive recursive. In *62nd Annual Symposium on Foundations of Computer Science, FOCS'21*. IEEE, 2021.
- 19 Jérôme Leroux and Sylvain Schmitz. Reachability in vector addition systems is primitive-recursive in fixed dimension. In *2019 34th Annual ACM/IEEE Symposium on Logic in Computer Science (LICS)*, pages 1–13. IEEE, 2019. doi:10.1109/LICS.2019.8785796.
- 20 Richard J. Lipton. The reachability problem requires exponential space. Technical Report 62, Yale University, 1976.
- 21 Ernst W. Mayr. An algorithm for the general Petri net reachability problem. In *Proceedings of the Thirteenth Annual ACM Symposium on Theory of Computing, STOC '81*, pages 238–246, New York, NY, USA, 1981. Association for Computing Machinery. doi:10.1145/800076.802477.
- 22 Ernst W Mayr and Albert R Meyer. The complexity of the word problems for commutative semigroups and polynomial ideals. *Advances in mathematics*, 46(3):305–329, 1982.
- 23 Carl Adam Petri. *Kommunikation mit Automaten*. PhD thesis, Rheinisch-Westfälischen Instituten für Instrumentelle Mathematik an der Universität Bonn, 1962.
- 24 Ján Plesník. The np-completeness of the hamiltonian cycle problem in planar diagraphs with degree bound two. *Information Processing Letters*, 8(4):199–201, 1979. doi:10.1016/0020-0190(79)90023-1.
- 25 George S Sacerdote and Richard L Tenney. The decidability of the reachability problem for vector addition systems (preliminary version). In *Proceedings of the ninth annual ACM symposium on Theory of computing*, pages 61–76, 1977. doi:10.1145/800105.803396.
- 26 Sylvain Schmitz. The complexity of reachability in vector addition systems. *ACM SigLog News*, 2016. doi:10.1145/2893582.2893585.
- 27 David Soloveichik, Matthew Cook, Erik Winfree, and Jehoshua Bruck. Computation with finite stochastic chemical reaction networks. *natural computing*, 7(4):615–633, 2008. doi:10.1007/S11047-008-9067-Y.
- 28 Gergely Szlobodnyik and Gábor Szederkényi. Polynomial time reachability analysis in discrete state chemical reaction networks obeying conservation laws. *MATCH-Communications in Mathematical and in Computer Chemistry*, 89(1):175–196, 2023.
- 29 Gergely Szlobodnyik, Gábor Szederkényi, and Matthew D Johnston. Reachability analysis of subconservative discrete chemical reaction networks. *MATCH-Communications in Mathematical and in Computer Chemistry*, 81(3):705–736, 2019.
- 30 Robert Tarjan. Depth-first search and linear graph algorithms. *SIAM journal on computing*, 1(2):146–160, 1972. doi:10.1137/0201010.
- 31 Chris Thachuk and Anne Condon. Space and energy efficient computation with dna strand displacement systems. In *International Workshop on DNA-Based Computers*, 2012.
- 32 Boya Wang, Chris Thachuk, Andrew D Ellington, Erik Winfree, and David Soloveichik. Effective design principles for leakless strand displacement systems. *Proceedings of the National Academy of Sciences*, 115(52):E12182–E12191, 2018.

## **A** Proof Details for (2,0) and (2,1) Mixed rules

Here we show the full details for the proof of correctness and runtime for the Lemmas and Theorems from Section 5.

### **A.1** Proof of Correctness

► **Lemma 29.** *A species represented by a mandatory vertex must use (2,0) void rules to be completely deleted, and mandatory cycles must have at least one species represented by a mandatory vertex to completely delete the cycle.*



**Proof.** We first consider the case of trying to delete a non-zero amount of a species  $\lambda_i$  represented by a mandatory vertex. We thus know that  $\lambda_i$  cannot be deleted by any catalytic rules. Thus, it can only be deleted with  $(2, 0)$  void rules, which means we must have as many  $(2, 0)$  matchings with  $\lambda_i$  as the number of deletions required to remove the species.

To show that completely deleting a mandatory cycle requires at least one species to be represented by a mandatory vertex, recall that the definition of a mandatory cycle is a cycle of  $(2, 1)$  void rules where the only catalytic rules that can delete each vertex is within the cycle. This means to completely delete the species in the cycle only using the rules of the cycle will always leave at least one species in the cycle with a non-zero amount of copies. Since we can always perform catalytic rules in the cycle such that every species in the cycle has a count of 1, then no matter what order the catalytic rules are applied in the cycle, there is always at least one species  $\lambda_i$  with a count of one. Thus, to completely delete all species of the cycle, a matching from a  $(2, 0)$  rule must be performed on  $\lambda_i$ . ◀

► **Lemma 30.** *A perfect  $b$ -matching in  $G$  implies we have a set of rules that, if applied, can delete  $\vec{D}$  and only  $\vec{D}$ .*

**Proof.** We represent configurations  $\vec{D}$  and  $\vec{B}$  in disjoint subgraphs in  $G$  as  $G_D$  and  $G_B$ , respectively. If a perfect matching exists, then  $\sum_{e \in \delta(v)} f(e) = b(v)$  for all  $v \in V''$ , which holds for all vertices in  $G_D$ . Since every edge represents a deletion either from a  $(2, 0)$  void rule or a  $(2, 1)$  catalytic void rule, this implies that if all rules are applied that are represented in  $\delta(v)$ , this would delete at least  $v$  because  $\sum_{e \in \delta(v)} f(e) = b(v)$ , and  $b(v)$  is equal to the count of  $v$  in  $G_D$ . Since this is true for all  $v \in V''$ , we can delete the entirety of  $G_D$  and completely remove only  $D$  if we have a perfect matching. ◀

► **Lemma 31.** *A perfect  $b$ -matching in  $G$  contains  $(2, 0)$  void rule matchings for all species represented by mandatory vertices in  $G_D$ .*

**Proof.** If a species  $\lambda_i$  is represented in  $G_D$ , there exists a non-zero count of  $\lambda_i$  that must be deleted to reach  $\vec{B}$ . By definition, a species represented by a mandatory vertex can only be completely deleted by a  $(2, 0)$  rule. Thus, if we have a perfect matching, we have matched all of the species represented by mandatory vertices through these  $(2, 0)$  void rule edges. ◀

► **Lemma 32.** *If there exists a perfect  $b$ -matching in  $G$ , then there exists a vertex in each mandatory cycle that is matched to a  $(2, 0)$  rule.*

**Proof.** A perfect matching contains the property that for all vertices,  $\sum_{e \in \delta(v)} f(e) = b(v)$ . Let  $c_j$  be a collection of vertices that compose a mandatory cycle. By the construction of  $G$ , for all  $\lambda_i \in c_j$ , there exists an edge from  $\lambda_{i1}$  to  $c_{j1}'$  or  $\lambda_{i2}$  to  $c_{j2}'$ , where  $c_{jk}'$  is the node representing catalytic rules among the vertices in the cycle. The vertex  $c_{jk}'$  has  $b$ -value  $b(c_{jk}') = (\sum_{\lambda_{ik} \in c_j} b(v)) - 1$ , which means there is at most  $(\sum_{\lambda_{ik} \in c_j} b(v)) - 1$  matchings with the vertices in the cycle. This implies at least one matching must occur without the catalytic rules represented by  $c_{jk}'$ , and since the matching is perfect, this matching must exist. Furthermore, since this is a mandatory cycle, no species in this cycle can be deleted through a catalytic rule not represented in the cycle. Thus, there is at least one  $(2, 0)$  rule matching with some vertex in this cycle when we have a perfect matching. ◀

► **Theorem 33.**  *$G$  has a perfect  $b$ -matching if and only if  $\vec{B}$  is reachable from  $\vec{A}$ .*

**Proof.**

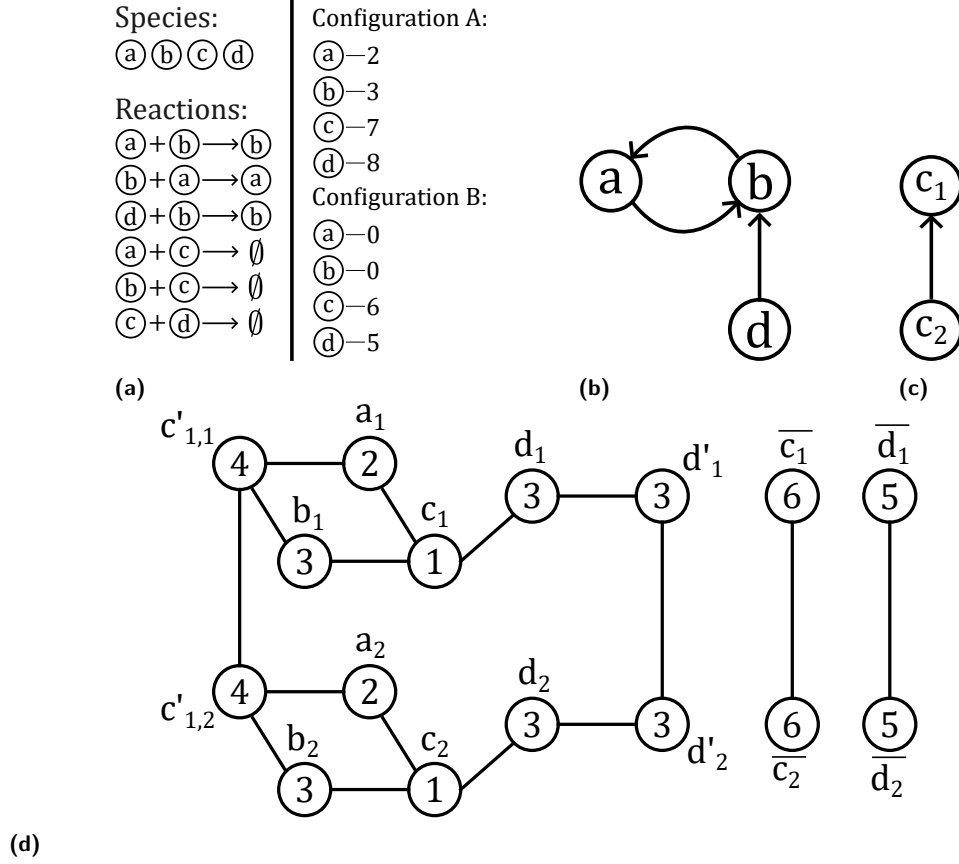
**Forward Direction.** Assume there exists a sequence of applicable rules in  $\mathcal{C}$  that reaches  $\vec{B}$  from  $\vec{A}$ . It thus follows that all species with non-zero counts in  $\vec{D}$  can be completely deleted by following this sequence. Then, by the construction of  $G$ , a perfect  $b$ -matching can be performed in the graph. Recall that our graph  $G$  has two disjoint subgraphs  $G_B$  and  $G_D$ . In our construction,  $G_B$  will always form a perfect matching since we have two vertices with equal  $b$ -values attached by one edge of infinite capacity. Thus, these vertices can always match with each other, and we only need to show how to create a perfect matching in  $G_D$ . Every vertex in  $G_D$  must be perfectly matched to create a perfect  $b$ -matching, and we can think of matching as a deletion of the deleted species for  $(2, 1)$  void rules and deletion of both species in  $(2, 0)$  void rules. We then make the assignment  $f((\lambda_{i1}, \lambda_{j1})) = f((\lambda_{i2}, \lambda_{j2}))$  on each edge the number of times we used that respective  $(2, 0)$  or  $(2, 1)$  rule, which perfectly matches all  $v_i \in V''$ . However, if our catalytic vertices  $\lambda_{i1}'$  are not perfectly matched through its respective  $(\lambda_{i1}, \lambda_{i1}')$  edge, this means that we deleted  $\lambda_{i1}$  using more than the catalytic rule corresponding with  $\lambda_{i1}'$ , which means we have excess  $\lambda_{i1}'$ . We thus use the edge  $(\lambda_{i1}', \lambda_{i2}')$  so that any excess matchings can be assigned along that edge, and thus create a perfect  $b$ -matching in  $G$ .

**Reverse Direction.** Assume there exists a perfect  $b$ -matching in  $G$ . Then  $\vec{B}$  is reachable from  $\vec{A}$  if there exists a valid sequence of  $(2, 0)$  and  $(2, 1)$  void rules to get from  $\vec{A}$  to  $\vec{B}$ . Recall that a perfect  $b$ -matching means that we have an assignment of edges such that  $\sum_{e \in \delta(v)} f(e) = b(v)$  for every  $v \in V''$ . Through Lemma 30, this perfect matching implies we can delete  $G_D$  and only  $G_B$ . Now we must show that there exists a valid sequence of rules that can be applied to delete  $G_D$ . This is done through Lemmas 31 and 32, where we show that all mandatory vertices and mandatory cycles have  $(2, 0)$  void rules to match with. This means we can apply all  $(2, 1)$  void rules the correct number of times ( $f(e)$  times) without worrying about deleting a catalytic species needed for a later catalytic rule. We can then execute all  $(2, 0)$  void rules their respective  $f(e)$  times to successfully delete  $\vec{D}$ , and thus reach  $\vec{B}$  from  $\vec{A}$ . ◀

## A.2 Runtime Analysis

We first construct  $T$ , which creates  $O(|\Lambda|)$  vertices and  $O(|\Gamma|)$  edges since it creates a vertex for each unique species involved in a  $(2, 1)$  rule and an edge for each unique  $(2, 1)$  rule. Thus, this step takes  $O(|V| + |E|)$  time, where  $V = O(|\Lambda|)$  and  $E = O(|\Gamma|)$ . We then run Tarjan's Strongly Connected Component Algorithm on  $T$  to produce graph  $H$ , which has a runtime of  $O(|V| + |E|)$  [30]. Finally, we create the graph  $G$  and functions  $u(\cdot)$  and  $b(\cdot)$ . It takes  $O(|\Lambda|)$  time to create  $G_D$ , and since we create at most 6 vertices per species when creating the vertex set for  $G$ , this takes  $O(|\Lambda|)$  time. Additionally, since we create one edge for every valid void rule in our CRN and an edge for every species in our final configuration, we create  $O(|\Lambda| + |\Gamma|)$  edges. We then assign a  $b$ -value  $b(\cdot)$  for each vertex in  $G$ , and since assigning a  $b$ -value to a vertex requires a look-up in  $\vec{D}$  that takes  $O(1)$  time, we assign all the  $b$ -values in  $O(|\Lambda|)$  time. Finally, we assign all edge capacities  $u(\cdot)$  to infinity, which takes  $O(1)$  time per edge with a runtime of  $O(|\Lambda| + |\Gamma|)$ . This results in an overall runtime of  $O(|\Lambda| + |\Gamma|)$  for the construction of  $G$ . Finally, the runtime of the maximum  $b$ -matching algorithm is proven to be strictly polynomial with a runtime of  $O(|V|^2 \log(|V|)(|E| + |V| \log(|V|)))$ . This results in a total polynomial runtime for the algorithm of  $O(|V|^2 \log(|V|)(|E| + |V| \log(|V|)))$ .

► **Theorem 34.** *The reachability problem in CRNs with only  $(2, 0)$  and  $(2, 1)$  rules is solvable in  $O(|\Lambda|^2 \log(|\Lambda|)(|\Gamma| + |\Lambda| \log(|\Lambda|)))$  time.*



**Figure 4** Transforming an instance of the  $(2,0)$  and  $(2,1)$  CRN reachability problem into an instance of the perfect  $b$ -matching problem, as described in Section 5. (a) An instance of  $(2,0)$ ,  $(2,1)$  CRN reachability. (b) The corresponding catalytic deletion graph  $T$ . Note that species  $a$  and  $b$  form a mandatory cycle. (c) The corresponding SCC condensation graph  $H$ .  $c_1$  represents the sole mandatory cycle in  $T$ , and  $c_2$  only the vertex  $d$ . (d) The corresponding  $b$ -matching instance graph  $G$ .

**Proof.** Subsection 5 shows how to convert any instance of the  $(2,0)$  and  $(2,1)$  reachability problem into an instance of the perfect  $b$ -matching problem in  $O(|\Lambda|^2 \log(|\Lambda|)(|\Gamma| + |\Lambda| \log(|\Lambda|)))$  time (Subsection A.2), with Subsection (A.1) proving its correctness. ◀



# Differentiable Programming of Indexed Chemical Reaction Networks and Reaction-Diffusion Systems

Inhoo Lee 

California Institute of Technology, Pasadena, CA, USA

Salvador Buse 

California Institute of Technology, Pasadena, CA, USA

Erik Winfree 

California Institute of Technology, Pasadena, CA, USA

---

## Abstract

Many molecular systems are best understood in terms of prototypical species and reactions. The central dogma and related biochemistry are rife with examples: gene  $i$  is transcribed into RNA  $i$ , which is translated into protein  $i$ ; kinase  $n$  phosphorylates substrate  $m$ ; protein  $p$  dimerizes with protein  $q$ . Engineered nucleic acid systems also often have this form: oligonucleotide  $i$  hybridizes to complementary oligonucleotide  $j$ ; signal strand  $n$  displaces the output of seesaw gate  $m$ ; hairpin  $p$  triggers the opening of target  $q$ . When there are many variants of a small number of prototypes, it can be conceptually cleaner and computationally more efficient to represent the full system in terms of indexed species (e.g. for dimerization,  $M_p$ ,  $D_{pq}$ ) and indexed reactions ( $M_p + M_q \rightarrow D_{pq}$ ). Here, we formalize the Indexed Chemical Reaction Network (ICRN) model and describe a Python software package designed to simulate such systems in the well-mixed and reaction-diffusion settings, using a differentiable programming framework originally developed for large-scale neural network models, taking advantage of GPU acceleration when available. Notably, this framework makes it straightforward to train the models' initial conditions and rate constants to optimize a target behavior, such as matching experimental data, performing a computation, or exhibiting spatial pattern formation. The natural map of indexed chemical reaction networks onto neural network formalisms provides a tangible yet general perspective for translating concepts and techniques from the theory and practice of neural computation into the design of biomolecular systems.

**2012 ACM Subject Classification** Computer systems organization → Molecular computing

**Keywords and phrases** Differentiable Programming, Chemical Reaction Networks, Reaction-Diffusion Systems

**Digital Object Identifier** 10.4230/LIPIcs.DNA.31.4

**Supplementary Material** *Software (Library)*: <https://github.com/SwissChardLeaf/icrn>

**Funding** *Inhoo Lee*: Schmidt Academy for Software Engineering, NSF CCF/FET Award 2212546. *Salvador Buse*: Open Philanthropy Good Ventures Foundation, NSF CCF/FET Award 2212546. *Erik Winfree*: NSF CCF/FET Award 2212546.

**Acknowledgements** Many thanks to inspirational and technical discussions with Alex Mordvintsev, Andrew Stuart, Lulu Qian, Kevin Cherry and members of the Winfree, Qian, and Rothemund groups. Inhoo Lee and Salvador Buse are co-first authors.

## 1 Introduction

As an embodiment of self-organized chemical complexity, living organisms challenge us to conceptualize how molecules can interact to create functional systems. Mathematical and computational models of molecular systems help us understand existing biology, the design space explored by evolution, and the engineering of sophisticated molecular technologies. Depending on purpose, different abstractions are available, from quantum chemistry to



© Inhoo Lee, Salvador Buse, and Erik Winfree;

licensed under Creative Commons License CC-BY 4.0

31st International Conference on DNA Computing and Molecular Programming (DNA 31).

Editors: Josie Schaeffer and Fei Zhang; Article No. 4; pp. 4:1–4:23

Leibniz International Proceedings in Informatics



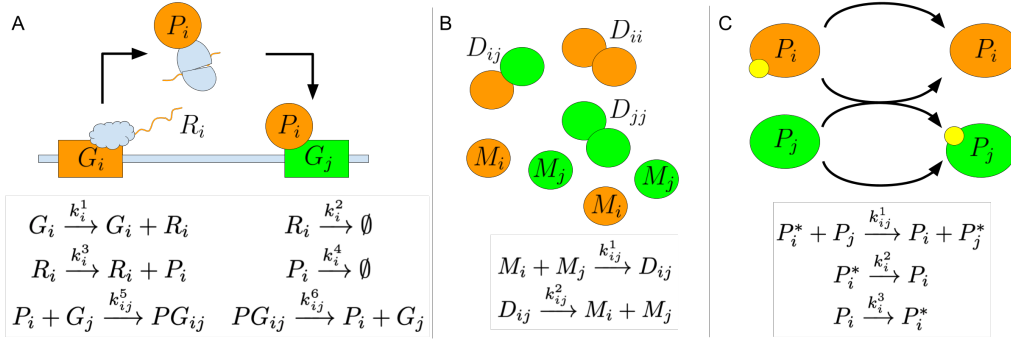
LIPICs Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

molecular dynamics to chemical kinetics and beyond. Formal chemical reaction networks with bulk mass-action kinetics occupy a particularly interesting place in the space of models: they have been used both as descriptive models for existing chemical systems [21], and as prescriptive models for specifying the target behavior for systems being engineered [59].

The basic chemical reaction network (CRN) model considers the temporal dynamics of the concentrations of a finite number of species undergoing a finite set of specified reactions. Software using the CRN representation for systematic modeling of complex biochemical pathways, such as COPASI [28] or Tellurium [12], has been an important ingredient for the development of systems biology. For these uses, generalizations of the basic CRN model have been important (non-mass-action rate formulas to allow for models involving Michaelis-Menten kinetics and other approximations; compartments to represent different parts of a cell; discrete events such as cell division) and have been incorporated into the systems biology markup language (SBML) that serves as a standard for thousands of models in the literature [32, 40]. Nonetheless, explicit listing of every relevant species and reaction can be cumbersome, leading to rule-based modeling languages such as Kappa [4] that allow concise descriptions of combinatorial modifications and even reaction mechanisms that potentially allow for an infinite number of distinct species, such as polymerization reactions. The bottom line is that the choice of representation impacts how we think about complex molecular systems and how we can efficiently simulate, analyze, and design them.

The starting point for our current work is to look for representations that reflect parallels between complex biochemical systems and neural networks. Canonical examples in biology include systems of metabolic enzymes [27], genetic regulatory networks [42, 7], signal transduction cascades [6, 14], and protein dimerization networks [48]. Inspired by these examples, synthetic biochemical systems have been explicitly designed to perform neural computation and experimentally demonstrated in cell-free enzyme systems [34, 35, 36, 24, 47], in DNA strand displacement cascades [51, 11, 64, 63, 67], and in living cells [52, 10]. The “neural” character of these systems stems from several factors: they have complex, multidimensional, nonlinear, analog dynamics that depend on a plethora of tunable parameters (such as reaction rate constants) and can perform information-processing tasks such as pattern recognition and attractor-based computation. Moreover, each type of network (e.g. metabolic, genetic regulation, signal transduction, dimerization) may be understood, in a simplified fashion, in terms of many instances of a prototypical mechanism, each differing only by its quantitative parameters – similar to how neural network models are often described in terms of a standard neural unit parameterized by weights and a threshold. Informally, we call such systems “index-amenable” because each species or reaction rate (or neuron or weight) can be identified by an index or tuple of indices; see Fig 1 for three examples based on genetic regulatory networks [7], dimerization networks [48], and phosphorelay networks [14, 9].

A particularly intriguing question is how biochemical “neural” computation can help make the decisions necessary to guide organismal development [42]. Reaction-diffusion systems, as pioneered by Alan Turing [62], provide a simplified model system – with local CRN dynamics spatially coupled by passive diffusion – for examining possible mechanisms for pattern formation. Even very simple CRNs can, in the reaction-diffusion context, give rise to complex textures and biological-looking patterns [49, 38]. Inspired by the neural cellular automaton model [45] that illustrated how a spatial array of neural networks can be trained to grow and maintain a precise “organismal” pattern, such as the image of a lizard, we have been exploring how a specific neural reaction-diffusion system can be trained for similar pattern formation tasks [8]. Both of these works employ differentiable programming



■ **Figure 1** Index-amenable chemical systems. **A.** An inhibitory gene regulatory network has genes  $G$  that are transcribed to RNA  $R$  that are translated to proteins  $P$ . The protein produced by one gene can bind on another gene and thereby block transcription. **B.** In a dimerization network, monomers can combine to form dimers. Dimers also dissociate into their constituent monomers. **C.** In phosphorelay networks, a phosphorylated protein can transfer its phosphate group to another protein. The proteins can also individually be phosphorylated or de-phosphorylated.

techniques [16], where physical simulation models are optimized by automatic differentiation and gradient descent. Relevant to the work here, differentiable programming has been applied to small CRNs [44] as well as to complex cellular morphogenesis models [19].

The work reported here extends our prior work [8] by abstracting from a specific “neural” CRN to a general model for what we call indexed chemical reaction networks (ICRNs). ICRNs use index symbols to succinctly describe a set of species and reactions that can be naturally scaled to be arbitrarily large. Not only does indexing provide notational convenience, but the representation allows computations to be efficiently performed with tensors. Furthermore – and making a very pragmatic connection to neural computation – software utilizing tensors can leverage machine learning technologies to train parameters of the chemical system to optimize its behavior. A natural consequence of the ICRN representation is that species and reactions may involve more than one index, representing potential all-to-all interactions that some molecular biologists call “promiscuous” [1, 60] but that also correspond to what neuroscientists call “synaptic weights”. In neuroscience it is well understood that many weak interactions can dominate over select strong interactions in processes such as visual perception [56], highlighting the potential importance of even weak “promiscuous” interactions in molecular biology. We hope that the ICRN model can provide a natural framework for describing chemical systems that embody neural network principles of collective computation, thus helping facilitate the transfer of concepts, insights, mathematics, and software between fields.

## 2 Chemical Reaction Networks

The chemical reaction network (CRN) model is a general mathematical formalism for (typically) mass-action chemical kinetics described as ordinary differential equations (ODEs) [30]. Chemical reaction network theory, as presented by Feinberg [22], provides a canonical representation that models chemical systems with sets of objects:  $\mathcal{S}$ ,  $\mathcal{C}$ ,  $\mathcal{R}$ ,  $\mathcal{K}$ . The set  $\mathcal{S}$  contains the species in consideration. For example,  $\mathcal{S}$  might be  $\{A, B, C\}$ . The set of complexes  $\mathcal{C}$  are multisets of species, i.e., each complex assigns a count to each of the species. For example,  $A + 2B$  is a complex. The set of reactions  $\mathcal{R}$  contains ordered pairs of complexes specifying the reactants and the products. For example,  $(A + 2B, C)$  is a reaction.

Lastly, the rate constants are specified by a function  $\mathcal{K} : \mathcal{R} \rightarrow \mathbb{R}_+$  that assigns a positive real number to a reaction. For example, if  $\mathcal{K}((A+2B, C)) = k$  then we may write  $A+2B \xrightarrow{k} C$ . A CRN is specified by the tuple  $(\mathcal{S}, \mathcal{C}, \mathcal{R}, \mathcal{K})$ .

In practice, Feinberg’s canonical representation is inconvenient for simulation software, where it may be desirable to allow reaction rate constants  $k = 0$  and to permit the same reaction to be listed multiple times, perhaps with different rate constants. The usual convention, which we take here, is to sum rate constants in that case, e.g., if a CRN contains  $A + B \xrightarrow{k_1} C$  and  $A + B \xrightarrow{k_2} C$  and  $A + B \xrightarrow{k_3} C$ , where  $k_1, k_2, k_3 \in \mathbb{R}_{\geq 0}$ , then this would correspond in Feinberg’s canonical representation to a single reaction  $A + B \xrightarrow{k} C$  with  $k = k_1 + k_2 + k_3$  unless  $k = 0$ , in which case the reaction would not be included in  $\mathcal{R}$ .

A chemical state is given as  $x \in \mathbb{R}_{\geq 0}^{\mathcal{S}}$ , representing the concentrations of each species. The notation  $\mathbb{R}_{\geq 0}^{\mathcal{S}}$  refers to  $|\mathcal{S}|$ -length tuples where each component corresponds to an element of  $\mathcal{S}$  and specifies a non-negative real number for that species. For complex  $c \in \mathcal{C}$ , define  $x^c = \prod_{s \in \mathcal{S}} x_s^{c_s}$ , the product over species in a complex of each concentration raised to the power of the respective stoichiometric coefficient. (Here, we define  $0^0 \equiv 1$  to completely ignore species whose stoichiometry is zero.) Then the mass action chemical kinetics for canonical CRN  $(\mathcal{S}, \mathcal{C}, \mathcal{R}, \mathcal{K})$  or equivalent non-canonical CRN list  $(a_r \xrightarrow{k_r} b_r : r \in \mathbb{Z}_M)$ , with  $M$  possibly repeated reactions listed, can be described by the ODE

$$\frac{dx}{dt} = \sum_{(a,b) \in \mathcal{R}} (b - a) \mathcal{K}((a,b)) x^a = \sum_{r \in \mathbb{Z}_M} (b_r - a_r) k_r x^{a_r}$$

where the difference of complex multisets  $b$  and  $a$  is interpreted as a vector in  $\mathbb{R}_{\geq 0}^{\mathcal{S}}$ , and  $\mathbb{Z}_M \equiv \{1, 2, 3, \dots, M\}$ .

### 3 Indexed Chemical Reaction Networks

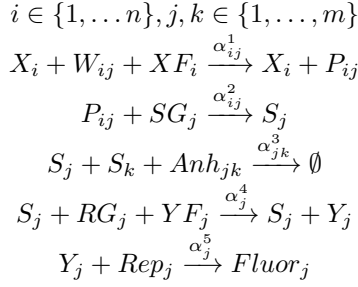
The indexed chemical reaction network (ICRN) model provides an abstraction and representation suitable for the specification and simulation of index-amenable chemical systems. To specify an ICRN for simulation, one provides symbolic information akin to the descriptions in Fig 1 together with the ranges for indices, complemented with numerical information for rate constant tensors and initial concentration tensors. Thus, the symbolic information can be very compact. Of course, since an ICRN may use any number of indices, including none, in a trivial sense every classical CRN can be considered to be an ICRN, and in this case no efficiency improvement is obtained for the specification of the system. In the other direction, given an ICRN we can obtain a CRN by enumeration: replacing each indexed reaction by a finite set of concrete CRN reactions in which index symbols take on particular values. The dynamics for ICRNs are defined below so as to ensure that an ICRN and its enumerated CRN always give identical chemical kinetics.

ICRNs and CRNs are structurally similar, and so share terminology, such as species, reactions, and rate constants. Yet the two representations differ in that ICRN objects are the multidimensional analogs of CRN objects. When necessary, to avoid confusion, the term indexed will be used to describe objects of the ICRN and the term concrete will be used to describe objects of the underlying CRN.

#### 3.1 Winner-Take-All ICRN

The winner-take-all DNA neural network from Cherry and Qian [11] is a prime example of a CRN with neural-network-like structure, and thus will be used both for illustrating the ICRN formalism here and for the first simulation and training example in Section 5. It can be described by the following indexed reactions.

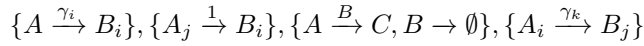




In an indexed reaction, base species, such as  $X, W, XF, \dots$ , and base rate constants, such as  $\alpha^1, \alpha^2, \dots$  are associated with index symbols,  $i, j, k$ . Associating specific values to all the index symbols in an indexed reaction describes a concrete reaction. For example, assigning values  $i = 1$  and  $j = 2$  in the first indexed reaction gives the concrete reaction  $X_1 + W_{12} + XF_1 \xrightarrow{\alpha_{12}^1} X_1 + P_{12}$ . The enumeration of the first indexed reaction is the collection of concrete reactions formed by substituting every combination of  $i, j \in \{1, \dots, n\} \times \{1, \dots, m\}$ . The enumeration of the ICRN is the collection of enumerations for each of its indexed reactions. This enumeration is a multiset: the same reaction may appear multiple times, possibly with different rate constants, and in such cases the effective rates constants are summed.

### 3.2 ICRN Definitions

By discussing ICRNs as representations of chemical systems, we have considered meaning, or semantics. Turning our attention to syntax, what are the rules for constructing ICRNs? The reader should pause and consider whether the examples below are valid ICRNs ( $\gamma_i, \gamma_k \in \mathbb{R}_{\geq 0}$ ).



We will define the ICRN, covering its syntax and its relationship to its semantics. Just as the various CRN conventions can be understood in terms of Feinberg's formalism, our formalism serves as a common way to precisely describe ICRNs.

An ICRN is specified by five collections:  $(\mathcal{U}, \mathcal{S}, \mathcal{K}, \mathcal{I}, \mathcal{R})$ . The base species  $\mathcal{S}$  and base rate constants  $\mathcal{K}$  are multidimensional analogs of species and rate constants from concrete CRNs. Indexed reactions  $\mathcal{R}$  use index symbols  $\mathcal{U}$  to describe interactions between the multidimensional objects. The index sets  $\mathcal{I}$  describe the index values that are valid to associate with base species, base rate constants, or indexed reactions.

Index symbols are essential for the abstraction provided by ICRNs. In the winner-take-all system, the set of index symbols  $\mathcal{U}$  contains index symbols  $i, j, k$ . Each index symbol  $u \in \mathcal{U}$  has an associated index set, denoted  $\mathcal{I}(u)$ , which is the set of values the index symbol can take on. For example,  $\mathcal{I}(i) = \{1, \dots, n\}$ . A tuple of index symbols  $\mathbf{u} \in \mathcal{U}^{<\mathbb{N}}$  has an index set that is the product of the index sets of its constituent index symbols  $\mathcal{I}(\mathbf{u}) = \mathcal{I}(u_1) \times \dots \times \mathcal{I}(u_w)$ .

Defining index symbols as objects that exist globally throughout the system, as opposed to each indexed reaction having its own notion of index symbols, has two advantages. Firstly, global index symbols avoid the confusion that arises when the same index symbol appears in multiple reactions but takes on different values across the reactions. Secondly, global index symbols directly represent entities of the system. In the winner-take-all system, there are  $n$  inputs and  $m$  outputs to a winner-take-all computation. Thus, the index symbol  $i$  represents a single input while  $j$  and  $k$  are used to represent a single output.

Base species and base rate constants are multidimensional objects representing groups of concrete species and concrete rate constants, respectively. Both base species and base rate constants have index sets, denoted  $\mathcal{I}(s)$  for base species  $s \in \mathcal{S}$  and  $\mathcal{I}(k)$  for base rate constants  $k \in \mathcal{K}$ . A base species maps from its index set to a concrete species, while a base rate constant maps from its index set to a concrete rate constant. For example, the base species  $W$  has an index set  $\{1, \dots, n\} \times \{1, \dots, m\}$  and represents a group of concrete species,  $W_{11}, \dots, W_{mn}$ . Base species can be thought of as dictating the overall structure of a group of biomolecules that vary at designated regions, or domains. Each concrete species is a specific variation of the overall structure and specifies the chemical identity of the domains.

Fixing  $\mathcal{U}$ ,  $\mathcal{S}$ , and  $\mathcal{K}$ , and their index sets, there is a notion of a set of valid indexed species  $V_{\mathcal{S}}$  and a set of valid indexed rate constants  $V_{\mathcal{K}}$ . An indexed object is valid if the index set of the base object and index set of the index symbols are the same.

$$V_{\mathcal{S}} = \{(s, \mathbf{u}) | \mathcal{I}(s) = \mathcal{I}(\mathbf{u})\}, \quad V_{\mathcal{K}} = \{(k, \mathbf{u}) | \mathcal{I}(k) = \mathcal{I}(\mathbf{u})\}$$

An indexed reaction  $r \in \mathcal{R}$  is a triple in  $\mathbb{N}^{V_{\mathcal{S}}} \times \mathbb{N}^{V_{\mathcal{S}}} \times V_{\mathcal{K}}$ . Reactants and products are multisets of  $V_{\mathcal{S}}$  and are accompanied by an indexed rate constant from  $V_{\mathcal{K}}$ . The set of index symbols present in the indexed reaction implies the index set of the indexed reaction, denoted  $\mathcal{I}(r)$ . Specifying values for each index symbol specifies a concrete reaction, so indexed reactions are functions from the index set to concrete reactions. The enumeration of an indexed reaction is the image of the index set under  $r$ ,  $\text{Im}(r)$ . The enumeration of an ICRN is the multiset union of the enumeration of each indexed reaction:  $\cup_{r \in \mathcal{R}} \text{Im}(r)$ .

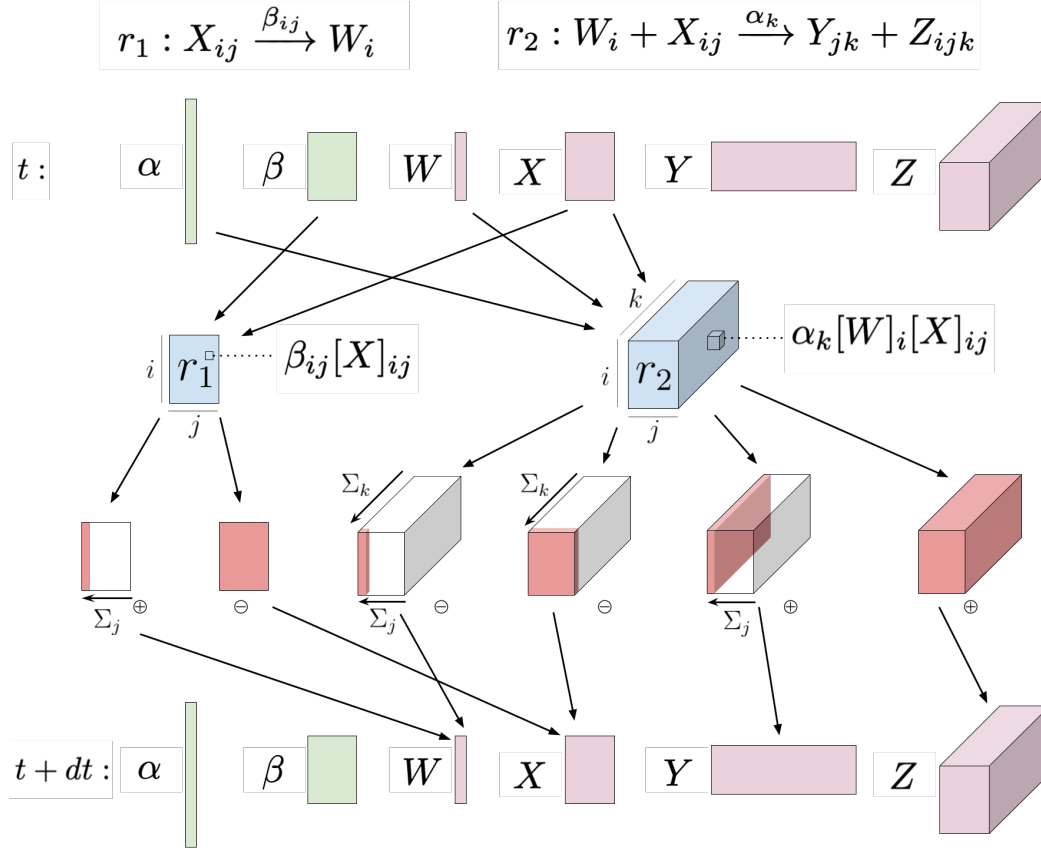
At this point, the reader should be able to identify  $\{A \xrightarrow{B} C, B \rightarrow \emptyset\}$  as the ill-defined ICRN. The other three ICRNs are valid given that the index sets of their objects are in agreement. Additionally, the reader may have noticed indexed species written as  $s_{\mathbf{u}}$  rather than  $(s, \mathbf{u})$ . For instance, we wrote  $W_{ij}$  instead of  $(W, (i, j))$ . The former notation emphasizes the concrete object that results from enumeration. However, indexed bases and indexed rate constants are objects in their own right and are, strictly speaking, the building blocks of indexed reactions. When there is room for confusion, the latter notation will be used to refer to indexed species and indexed rate constants.

ICRNs can be enumerated to capture a wide variety of concrete CRNs. There are systems of interest that are well suited to being described by indexed reactions but with restricted index sets. In many of these cases, the system of interest can be captured by creative use of the base species and base rate constants. For example, in the winner-take-all system, the implementation of the annihilation reaction in the DNA substrate produces a non-functioning annihilator molecule for the case where  $j = k$ . This restriction in indexing can be addressed by setting  $\alpha_{jk}^3 = 0$  when  $j = k$ , so the concrete reaction does not occur for  $j = k$ . Symmetric systems in particular are discussed further in appendix A.

### 3.3 Dynamics

The dynamics of an ICRN follows the dynamics of the enumerated CRN. Concentrations of base species and base rate constants are naturally represented as tensors. Accordingly, the dynamics of an ICRN can be expressed with tensor operations, which can be derived through symbolic manipulation of the ICRN.

The tensor representing the concentrations for base species  $s$  is written as  $[s] \in \mathbb{R}^s$ , where  $\mathbb{R}^s = \mathbb{R}_{\geq 0}^{|J_1| \times \dots \times |J_{N_s}|}$  with  $\mathcal{I}(s) = J_1 \times \dots \times J_{N_s}$ . To distinguish between the concentrations associated with the ICRN and the concrete CRN that it abstracts, we will use  $[s]_{\mathbf{u}}$  to refer to concentrations in the ICRN representation and  $[s_{\mathbf{u}}]$  to refer to the concentration



■ **Figure 2** Block schematic for ICRN dynamics. At time  $t$ , the values of the base rate constants, in green, and base species concentrations, in pink, are involved in the indexed reaction fluxes, in blue. The indexed reaction fluxes may be summed across their dimensions to produce contributions, in red, to each base species. These contributions positively or negatively contribute to the base species concentrations at time  $t + dt$ .

of the concrete species,  $s_{\mathbf{u}}$ , where  $\mathcal{I}(\mathbf{u}) = \mathcal{I}(s)$ . The time-dependent state of an ICRN is  $x = ([s_1], \dots, [s_{|\mathcal{S}|}]) \in \mathbb{R}^{\mathcal{S}}$  where  $\mathbb{R}^{\mathcal{S}} = \mathbb{R}^{s_1} \times \dots \times \mathbb{R}^{s_{|\mathcal{S}|}}$ . Thus,  $x$  specifies  $[s]$  for all  $s \in \mathcal{S}$ . The dynamics of an ICRN describes the time derivative of  $x$ , which is the tuple of time derivatives for each  $[s]$ :  $\frac{dx}{dt} = \left( \frac{d[s_1]}{dt}, \dots, \frac{d[s_{|\mathcal{S}|}]}{dt} \right)$

The dynamics for the ICRN follows the dynamics of the enumerated concrete CRN, so the time derivative for  $[s]$  is the collection of time derivatives for each concrete species represented by  $s$ , where the time derivative of  $[s]_{\mathbf{u}}$  is defined by the time derivative of  $[s_{\mathbf{u}}]$  from the enumerated concrete CRN:  $\frac{d[s]_{\mathbf{u}}}{dt} := \frac{d[s_{\mathbf{u}}]}{dt}$

As seen in Fig 2, each indexed reaction  $r$  has an associated indexed reaction flux  $\Phi(r)$ , a real-valued tensor. For each appearance of an indexed species  $(s, \mathbf{u})$ , sums are taken along the dimensions of  $\Phi(r)$  that correspond to index symbols that are present in the indexed reaction but not present in the indexed species. The result is the flux contribution to  $s$ , denoted  $\delta_r(s, \mathbf{u})$ , which is a real-valued tensor with the same shape as  $[s]$ . A base species  $s$  receives contributions from every appearance of any indexed species that derives from  $s$  among all the indexed reactions.

The indexed reaction flux,  $\Phi(r)$  is a real-valued tensor with the same shape as the index set of  $r$ . Let  $\mathbf{p}$  be index symbols such that  $\mathcal{I}(\mathbf{p}) = \mathcal{I}(r)$  and  $a_r, b_r \in \mathbb{N}^{V_S}$  be the multisets of reactants and products, respectively. Then  $a_r(y, \mathbf{u}), b_r(y, \mathbf{u}) \in \mathbb{N}$ , where  $(y, \mathbf{u})$  is an indexed species. Additionally, let  $(k_r, \mathbf{q}) \in V_K$  be the indexed rate constant used in reaction  $r$ . Here,  $k_{r, \mathbf{q}} \in \mathbb{R}_{\geq 0}$  is the real-valued rate constant. The following is an element-wise specification of  $\Phi(r)$ , indexing with  $\mathbf{p}$ , since all index symbols in  $\mathbf{p}$  are present in the right hand expression.

$$\Phi(r)_{\mathbf{p}} := k_{r, \mathbf{q}} \prod_{(y, \mathbf{u}) \in V_S} [y]_{\mathbf{u}}^{a_r(y, \mathbf{u})}$$

The flux contribution to  $s$  from  $r$  depends on which indexed species  $(s, \mathbf{u})$  is participating in the reaction. The following equation is an element-wise specification of the tensor  $\delta_r(s, \mathbf{u})$  using the index symbols  $\mathbf{u}$ .

$$\delta_r(s, \mathbf{u})_{\mathbf{u}} := \sum_{\mathbf{p} - \mathbf{u}} \Phi(r)_{\mathbf{p}} = \text{sum of } \Phi(r) \text{ over index symbols in } r \text{ and not in } \mathbf{u}$$

On the right hand side,  $\mathbf{p}$  is the tuple of index symbols in  $r$  and  $\mathbf{p} - \mathbf{u}$  refers to the index symbols that are in  $\mathbf{p}$  but not  $\mathbf{u}$ . Thus, the right hand side is an expression that is parameterized by  $\mathbf{u}$ . Moreover, the summation is an Einstein summation, a kind of indexed summation which can be efficiently computed [58].

The totality of the contributions to  $s$  from each appearance of a derived indexed species can be formulated as the following element-wise specification.

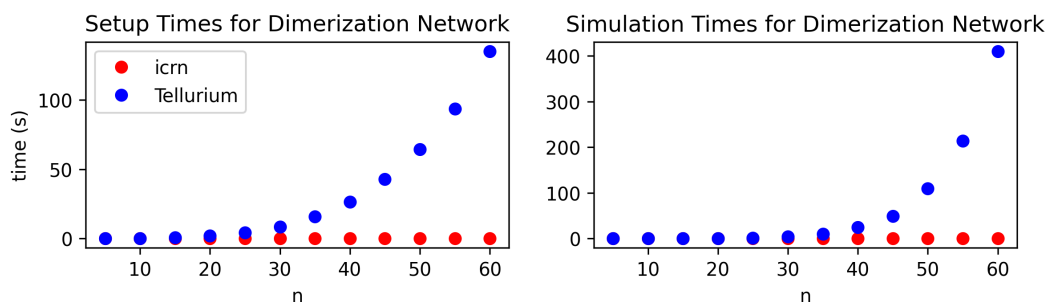
$$\frac{d[s]_{\mathbf{v}}}{dt} = \sum_{r \in \mathcal{R}} \sum_{(y, \mathbf{u}) \in V_S | y=s} (b_r(y, \mathbf{u}) - a_r(y, \mathbf{u})) \delta_r(y, \mathbf{u})_{\mathbf{v}}$$

The left hand side describes the rate of change of an element of  $[s]$  corresponding to index symbols  $\mathbf{v}$ . On the right,  $\delta_r(s, \mathbf{u})_{\mathbf{v}}$  refers to an element of the tensor  $\delta_r(s, \mathbf{u})$  indexed by  $\mathbf{v}$ . Since  $(b_r(y, \mathbf{u}) - a_r(y, \mathbf{u})) \in \mathbb{Z}$ , the right hand side is an expression parameterized by  $\mathbf{v}$ . An example of the dynamics derivation appears in appendix B.

ICRNs can be extended naturally from the well-mixed to a spatial setting, becoming reaction-diffusion systems. For classical CRNs, reaction-diffusion dynamics are described mathematically by a PDE:  $\partial_t \mathbf{X} = \mathbf{D} \nabla^2 \mathbf{X} + R(\mathbf{X})$ . For reaction-diffusion with species  $\mathcal{S}$  in  $d$  spatial dimensions, the concentration field  $\mathbf{X}$  is a function  $\mathbf{X} : \Omega \subset \mathbb{R}^d \rightarrow \mathbb{R}^{\mathcal{S}}$ , with  $\Omega$  a bounded subset of the spatial dimension.  $\mathbf{D}$  is a diagonal matrix of diffusion constants,  $\nabla^2$  is the Laplacian operator encoding diffusion, and  $R$  represents the derivatives due to the CRN's reactions. ICRNs require only a tensor generalization of these semantics.

## 4 Software

The `icrn` Python library allows for efficient simulation and training of ICRNs by compiling a general ICRN into its dynamics in a tensor-based and differentiable framework. The SymPy symbolic programming library [41] is used for transforming an ICRN representation into a function that computes the dynamics and influenced the nomenclature of the ICRN definitions. While the ICRN representation is symbolic, the dynamics function is numerical and takes as input real-valued tensors for concentrations and rate constant information to compute a change of concentrations. In the `icrn` library, tensors and the operations between them are backed by JAX [5], so the dynamics computation is not only efficient, making use of hardware acceleration to speed up tensor computations, but also is automatically differentiable, allowing for gradient based optimization methods.



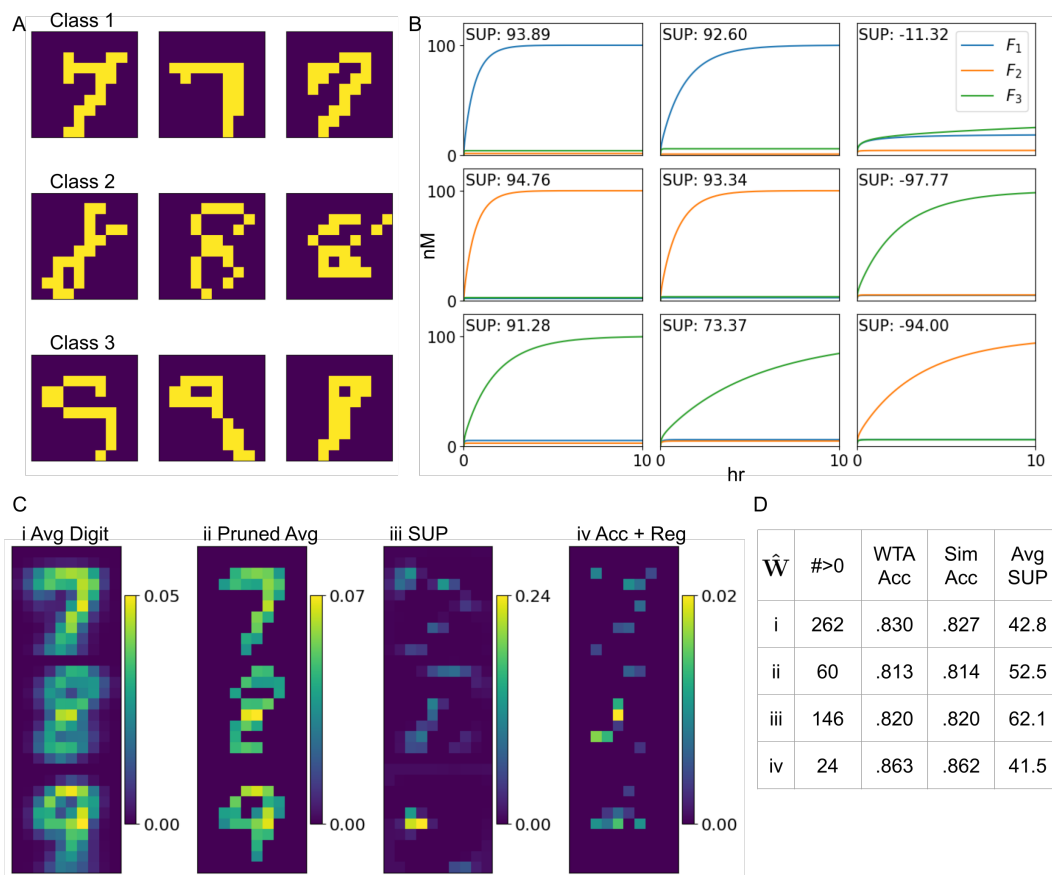
**Figure 3** Benchmarking of icrn. Setup is considered to be any action, such as reaction specification, necessary before simulations are run. We tested random dimerization networks with initial concentrations and rate constants in the range  $(0, 1)$  without requiring symmetry for  $k_{ij}^1$  and  $k_{ij}^2$ . For each  $n$ , a reference simulation was run using icrn and very small  $dt$ .  $\epsilon_{icrn}^n(s)$  and  $\epsilon_{te}^n(s)$  are the relative errors in the concentration of  $s$  with respect to the reference simulation. The two simulations achieved similar accuracy, as  $\max_n \{ |\max_s \{ \epsilon_{icrn}^n(s) \} - \max_s \{ \epsilon_{te}^n(s) \}| \} < 2 \times 10^{-4}$ . At  $n = 60$ , icrn was  $3.6 \times 10^3$ -fold and  $3.1 \times 10^4$ -fold faster than Tellurium for setup and simulation, respectively. All times were measured on the same Google Colab instance with an A100 GPU.

Existing reaction system simulators, such as Tellurium [12], do not take advantage of mapping of dynamics onto tensor operations. Fig 3 compares the use of the icrn library and Tellurium for setting up and simulating dimerization networks from Fig 1 with varying  $n$ , the number of monomer species.

Well-mixed systems are simulated explicitly using forwards Euler or Runge-Kutta methods adapted to tensor operations. As of writing, icrn has only limited support for time-scale separation: “fast” reactions that go to completion (not equilibrium or steady-state) within each time step, subject to the restriction that no species may appear as a reactant in more than one fast reaction. To simulate reaction diffusion, we use a finite-differences numerical scheme, dividing both the spatial and temporal dimensions into discrete points, and deriving operators for the reaction and diffusion update steps. Thus, spatial extent expands the dimensions of concentration tensors, with discretized positions playing a role analogous to indices. Our ICRN simulator software uses an implicit-explicit (IMEX) integration scheme: diffusion derivatives are solved implicitly in the Fourier domain [54], and the reaction derivatives are handled explicitly. The icrn package and notebooks with code for the examples below are available at <https://github.com/SwissChardLeaf/icrn>.

## 5 Examples

We provide three examples, demonstrating different aspects of the ICRN formalism and our icrn software. First, we simulate the winner-take-all example described earlier, showing that the starting concentrations may be optimized by gradient descent to perform the classification task, rather than set manually as in [11]. Second, we explore the Gray-Scott reaction-diffusion model [25]. While this is a simple CRN and does not stand to gain from indexing, it serves to illustrate how tensor-based simulation facilitates reaction-diffusion modeling and how differential programming can be used to find parameters that produce target textures. Finally, we replicate the developmental pattern formation task from [8], showing how icrn can be used to efficiently optimize a large (256 species) ICRN in the reaction-diffusion setting.



**Figure 4** Training a WTA ICRN. **A.** Three sample images are shown from each of the three classes. The 28 by 28 MNIST images are resampled into 10 by 10 images and then made binary by setting the 20 pixels with the highest values to 1 and setting the remaining 80 pixels to 0. We used 5000 training samples and 900 test samples per digit class. **B.** Plots show the time trajectory of the 3 fluorescence signals with the SUP score from 9 ICRN simulations, each initialized with an image from **A** by setting the initial concentration of base species  $X$  based on the pixel values of the image. The concentration of the base species  $W$  is initialized based on the pruned average-digit weight matrices. Each simulation runs for  $10^5$  steps with a  $dt = 10^{-4}$ . Sample images were chosen such that the left and center columns show correct classification while the right column is incorrect. **C.** 100 by 3 weight matrices are rearranged into 30 by 10 arrays. The (i) Avg Digit matrix was created by averaging images for each class, and the (ii) Pruned Avg matrix is produced by taking the top 20 pixel values for each class from the Avg Digit matrix. The (iii) SUP matrix is produced by training a uniform weight matrix with a loss function that optimizes the SUP score. The (iv) Acc + Reg matrix is produced by training with a cross entropy loss function with an additional  $L_{1/2}$  regularization term. **D.** The performance of each matrix from **C** is tabulated. The WTA Acc is the result of using a weight matrix as  $\hat{W}$  in the abstract winner-take-all computation while Sim Acc uses the ICRN as a classifier, where accuracy refers to the percent of test samples where the correct output is the largest output.

## 5.1 Indexed Well Mixed – Winner-Take-All

The abstract winner-take-all computation can classify hand written digits. Following Cherry and Qian [11], our dataset is a reduced-resolution subset of the MNIST [18] dataset and consists of input-output pairs  $(\mathbf{x}_d, \mathbf{y}_d)$  where  $d$  indicates a specific pair,  $\mathbf{x}_d \in \{0, 1\}^{100}$  specifies

a 10 by 10 image of a handwritten digit, and  $\mathbf{y}_d \in \{0, 1\}^3$  specifies the digit that the image represents (seven, eight, or nine). A digit classifier is a function from images to classes. The abstract winner-take-all computation acts as a classifier by computing weighted sums for each class and taking the class with the highest sum to be the output class. The weights are specified by a matrix  $\hat{\mathbf{W}} \in \mathbb{R}_{\geq 0}^{100 \times 3}$ , where a column contains the weights used in the weighted sum per class.

$$\text{WTA}_{\hat{\mathbf{W}}}(\mathbf{x}) = \operatorname{argmax}_{c \in \{1, 2, 3\}} (\mathbf{x} \hat{\mathbf{W}})_c$$

Cherry and Qian set the weights of the classifier by pruning an average-digit matrix. Average-digit weights are the averages of examples in the training data. A column of the weight matrix is populated by the average of  $\mathbf{x}$  from the digit class corresponding to the column. Due to experimental limitations from having many nonzero elements, the pruned matrix takes the 20 highest elements of each class.

The winner-take-all ICRN can be trained to optimize different qualities. One view is that the fluorescence signal for the correct class should completely saturate (at 100 nM) while for other classes it is 0. A large difference in the output fluorescence is important experimentally for being able to resolve the output signal from the classifier. The superiority score  $\text{SUP} = [F] \cdot (2y - 1)$  captures this idea, for a given target input-output pair  $(x, y)$  that results in endpoint fluorescence vector  $[F]$ . Ideal behavior gives a superiority metric of 100, and lower values reflect deviation from the ideal (Fig 4B). The loss function can be set to produce a weight matrix with a high superiority metric. Another possibility is to optimize the ICRN's ability to produce the correct output without a concern for how low the fluorescence is for the other output classes. The endpoint fluorescence can be interpreted as a probability by normalizing with the softmax function,  $p_j = \exp([F]_j) / \sum_k \exp([F]_k)$ . The loss function computes a cross entropy loss  $-\sum_j y_j \log(p_j)$  with an  $L_{1/2}$  regularization term,  $\sum_{i,j} \hat{\mathbf{W}}_{ij}^{1/2}$ , that penalizes high weights, which encourages the training process to produce fewer nonzero weights, increasing experimental feasibility. The results of applying these training methods are shown in Fig 4D.

## 5.2 Reaction Diffusion – Gray-Scott Textures

We will begin with the Gray-Scott model [25] to illustrate the simulation and training of reaction-diffusion systems within our software framework. The Gray-Scott model is a reaction-diffusion system with just two species,  $U$  and  $V$ , and four reactions:  $U + 2V \xrightarrow{1} 3V$ ,  $V \xrightarrow{F+k} 0$ ,  $U \xrightarrow{F} 0$ , and  $0 \xrightarrow{F} U$ . As such, it does not stand to gain simplicity from indexing, but nonetheless it is a (trivial) instance of an ICRN. The PDE can be written as:

$$\begin{aligned} \partial_t U &= D_U \nabla^2 U - UV^2 + F(1 - U) \\ \partial_t V &= D_V \nabla^2 V + UV^2 - (F + k)V \end{aligned}$$

This reaction-diffusion system has just four parameters: rate constants  $F$  and  $k$ , and diffusion constants  $D_U$  and  $D_V$ . However, tuning these parameters, the system generates a wide variety of patterns, both stable and time-varying, explored and categorized by Pearson [49].

Our task is to use differentiable programming [16] to find Gray-Scott parameters that yield a particular type of pattern – e.g. mazes, spots, waves. As in Pearson's paper, we will fix the diffusion constants (with  $U$  diffusing twice as fast as  $V$ ), and vary only  $F$  and  $k$ . We set up the task as follows: first, we choose  $F$  and  $k$  to yield a certain type of pattern when simulated forwards from initial conditions randomly sampled from a distribution (see

Fig 5). The Gray-Scott PDE is fairly stiff, necessitating many steps with small  $dt$  to let its characteristic patterns develop from unstructured initial conditions. We thus take a single long simulation output using these parameters, and call this our target pattern. During training, we will run shorter simulations that are more amenable to gradient descent, but which will introduce other challenges.

Next, we need to define a loss function measuring how closely simulations during training match the target. We would like the loss function to be relatively stable to different initial conditions, so we cannot use a simple pixel-by-pixel comparison of simulation output and target: rather, we want something that captures and compares the qualitative features of the patterns. Taking inspiration from work on neural transfer of artistic style [23] and matching textures with self-organizing neural cellular automata [43], we evaluate pattern matches using the low-level (early layer) features from VGG-16, a convolutional neural network trained for image classification [57]. We feed our target pattern to VGG-16 as an input, and extract the activations from the first through third max-pooling layers. We then compute Gram matrices from these features, capturing some sense of how different elements of the pattern vary with each other over space. The Gram matrix  $G^l$  element  $G_{i,j}^l = \sum_{x,y} f_{i,x,y}^l f_{j,x,y}^l$ , where  $x$  and  $y$  are the spatial dimensions of the layer and  $f_{i,x,y}^l$  is feature  $i$  at position  $(x,y)$  in layer  $l$ .

Finally, we can compute a stylistic loss function, defined as the mean squared distance between the target Gram matrices and those extracted from the simulated output by the same process:  $\mathcal{L}_{\text{texture}} = \frac{1}{|l \times i \times j|} \sum_{l,i,j} (G_{i,j}^{l, \text{target}} - G_{i,j}^{l, \text{output}})^2$ . For stability, we augment the loss by adding a “moments-matching” term, comparing the means and log-variances of the simulation outputs with reference values, drawn by simulating the Gray-Scott model in different parts of the pattern-forming parameter space:

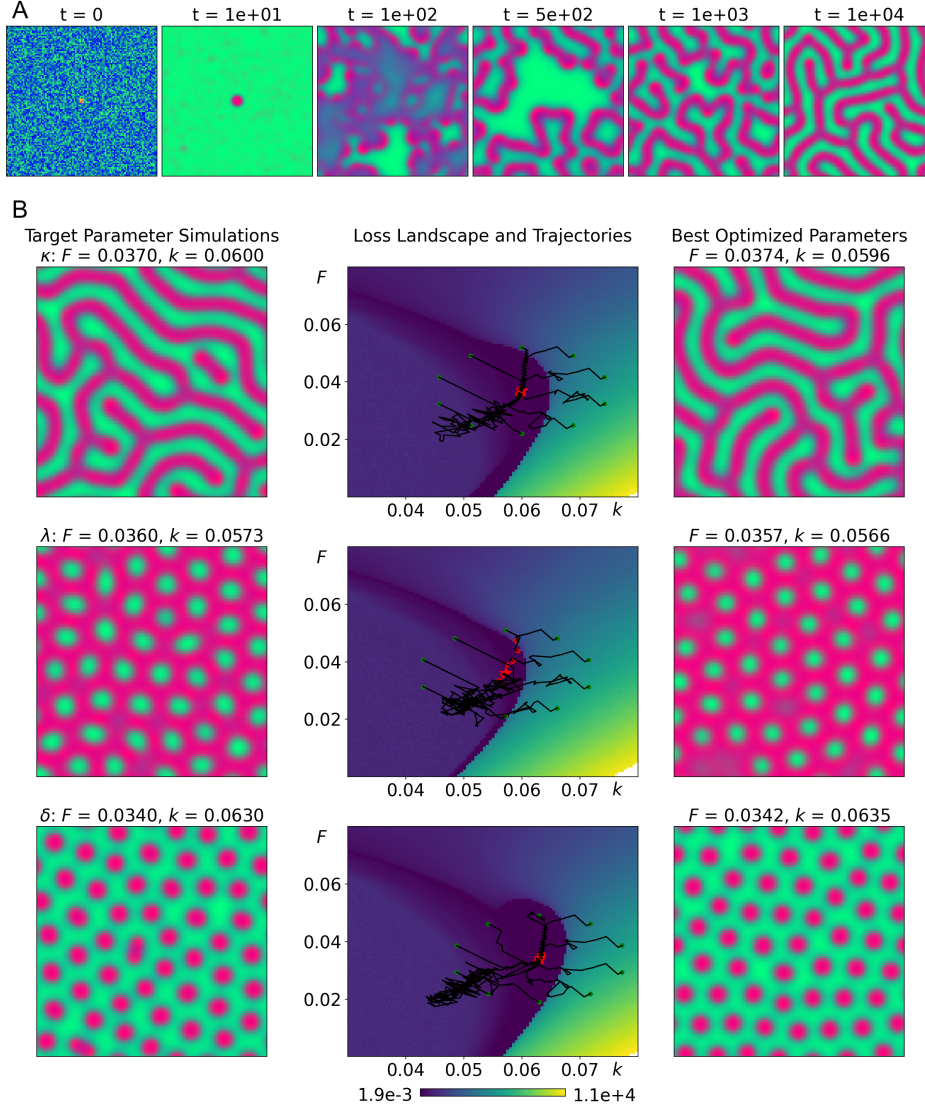
$$\mathcal{L}_{\text{moments}} = (\langle U \rangle - \langle \hat{U} \rangle)^2 + (\langle V \rangle - \langle \hat{V} \rangle)^2 + \left( \log \left( \frac{\sigma^2(U)}{\sigma^2(\hat{U})} \right) - \log \left( \frac{\sigma^2(V)}{\sigma^2(\hat{V})} \right) \right)^2.$$

Here,  $U$  is the simulated output, and  $\hat{U}$  is an average over ten reference simulation outputs, and  $\sigma^2(X)$  is the variance of  $X$ . Finally, we add a “potential well” regularization term, penalizing parameter values outside of the pattern-forming region of  $F \in [0, 0.08], k \in [0.03, 0.08]$ . We then have loss  $\mathcal{L} = \mathcal{L}_{\text{texture}} + \alpha_1 \mathcal{L}_{\text{moments}} + \alpha_2 \mathcal{L}_{\text{regularization}}$ .

We train the parameters using gradient descent on this loss function applied to the output of short simulations that start with the target pattern rather than a random noise state. Thus we are accommodating the stiffness of the Gray-Scott PDE by training for stability rather than for *de novo* generation of the texture. This has an additional benefit in that for some parameters, pattern formation in the Gray-Scott system can fail to “kick start” properly for random initial conditions. As in the winner-take-all example, both the simulation (using icrn) and loss function computation are written using differentiable JAX primitives, meaning that the derivatives may be calculated automatically using backpropagation-through-time (BPTT) [2, 53]. We integrate the gradients using mini-batch gradient descent, adding a small amount of randomness to the parameters when the learning converges to a local minimum.

The results shown in Fig 5, including a final long simulation from the lowest loss endpoint among the training runs, are perhaps deceptively well-behaved: training of this system was finicky and challenging. The vast bulk of parameter combinations give rise to spatially homogeneous outputs, and only in a sliver of parameter space are interesting patterns to be found – and there, the sensitivity to initial state was often confounding. Making this problem even harder, learning can turn just two knobs,  $F$  and  $k$ . The success of modern machine learning in many contexts is thanks to, among other things, the high dimensionality of deep neural networks. With so many parameters, there is almost always some direction in





**Figure 5** Texture matching with the Gray-Scott reaction-diffusion system. **A.** Following Pearson, we initialize with  $U = 1.0$ ,  $V = 0.0$ , but at the center,  $U = 0.25$ ,  $V = 0.5$ . We apply weak additive noise everywhere. We simulate forwards using an implicit-explicit Runge-Kutta integration scheme, with periodic boundaries, and parameters  $F$  and  $k$  corresponding to Pearson’s  $\kappa$  mazes. RGB visualization is achieved by  $(R, G, B) = (V, U, 1 - (U + V)/2)$ . **B.** While training the Gray-Scott system, we fix  $D_U = 0.2$ ,  $D_V = 0.1$ . Each row is a different target pattern (Pearson’s  $\kappa$ ,  $\lambda$ , and  $\delta$ ) obtained by simulating forward to time  $t = 5000$  with hand-chosen values  $F, k$  (left, marked by red stars in the middle plots). In the middle, we plot the learning trajectories through parameter space, superimposed on the loss landscape. On the right are the outputs produced using the lowest-loss parameters among the endpoints of the training trajectories. Each is simulated for  $t = 5000$  from a new target image created from a new random initial state, demonstrating long-term texture stability.

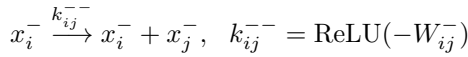
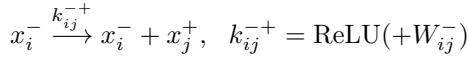
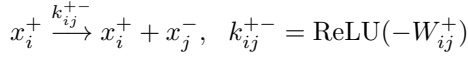
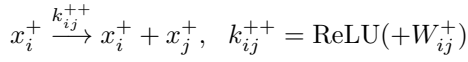
parameter space we can move along to reduce the loss, meaning that even simple, first-order optimization algorithms like gradient descent perform well. In the next section we will consider an ICRN with tens of thousands of parameters in the reaction-diffusion setting. Though small by modern standards, it is big enough to enjoy the blessings of dimensionality.

### 5.3 Indexed Reaction Diffusion – Hopfield CRN Images

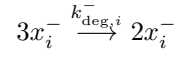
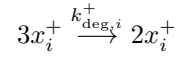
Reaction-diffusion systems can be trained to grow into significantly more complex patterns. Here, we take the Hopfield network-inspired CRN and training approach used in [8], and train an ICRN to grow from simple initial conditions into a rendering of a photograph.

The Hopfield CRN consists of a set of neurons  $\mathbf{x}$ , with variables  $x_i \in \mathbb{R}$ , and  $i, j \in \{1, \dots, N\}$  for an  $N$ -neuron system. In the well-mixed setting, the Hopfield CRN can be trained to store a set of “memories”, similar to the classical Hopfield network [29]. The neurons are connected all-to-all. Catalytic reactions between pairs of neurons correspond to weighted sums, and a trimolecular degradation reaction for each neuron corresponds to an activation function. To allow negative values for the variables, and negative weights between neurons, we use a dual-rail representation: the variable  $x_i$  is the difference in the concentrations of two species,  $x_i = x_i^+ - x_i^-$ , with  $x_i^+, x_i^- \in \mathbb{R}_{\geq 0}$ . A fast annihilation reaction is added between the positive and negative species for each variable. The ICRN is defined as

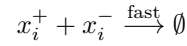
Weighted sums:



Degradation:



Annihilation:

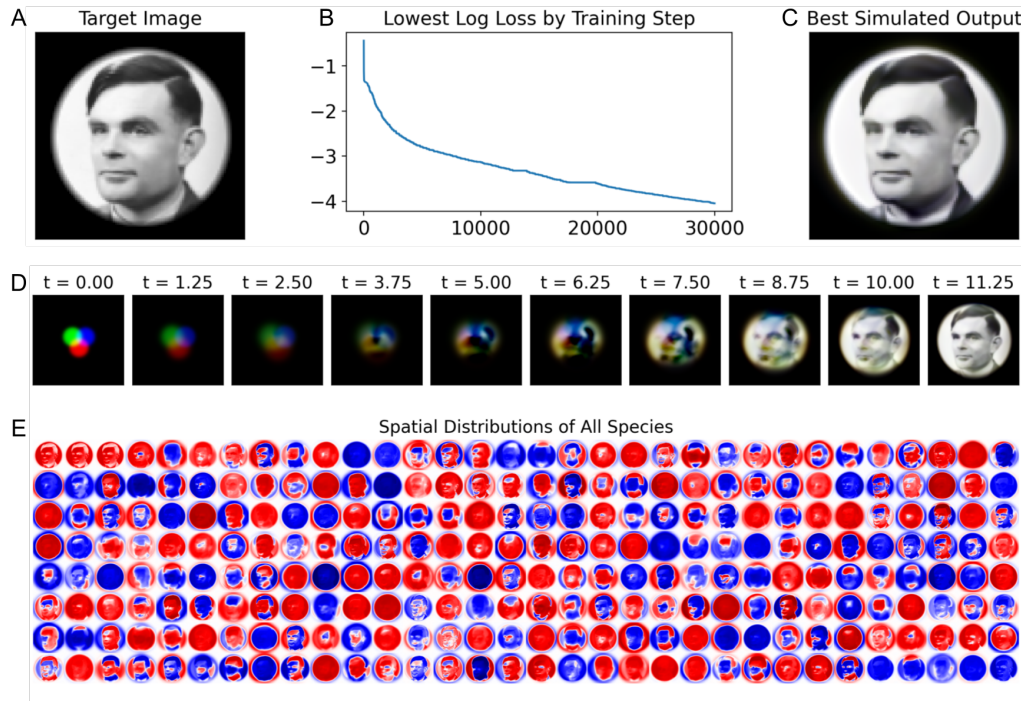


where  $\text{ReLU}(x) = \max(0, x)$ , ensuring that all rate constants are non-negative. The parameters for the Hopfield ICRN consist of two  $N \times N$  weight matrices  $W^+$  and  $W^-$  (with  $W_{ij}^\pm \in \mathbb{R}$ ), two  $N$ -vectors of degradation rate constants  $k_{\text{deg}}^+$  and  $k_{\text{deg}}^-$  (with  $k_{\text{deg},i}^\pm \in \mathbb{R}_{\geq 0}$ ), and two  $N$ -vectors of diffusion constants  $D^+$  and  $D^-$  (with  $D_i^\pm \in \mathbb{R}_{\geq 0}$ ). When the weight matrices are symmetric and degradation rate constants are equal, the Hopfield ICRN has equivalent attractor-basin structure as the classical Hopfield associative memory, but we find that allowing asymmetric and unequal parameters facilitates improved pattern formation [8]. The ICRN software admits an efficient representation of dual-rail ICRNs.

The training approach and loss function for the pattern forming task was inspired by Mordvintsev’s work on neural cellular automata [45]. We define the first three  $x$  variables ( $i \in \{0, 1, 2\}$ ) to be the visible units, corresponding to red, green, and blue, and the remaining  $N - 3$  are hidden. We pick an RGB target image  $\mathbf{T}$ , and train the parameters of the ICRN such that the visible state of the system matches the target as closely as possible, when it is grown from specified initial conditions for a specific duration of time,  $t_{\text{grow}}$ . The target in Fig 6 is grayscale, but we treat it in RGB for generality. We can write the pixel-wise loss as

$$\mathcal{L} = \sum_{h,w} \sum_{i \in \{0,1,2\}} \left( \mathbf{X}_{h,w,i}^{t_{\text{grow}}} - \mathbf{T}_{h,w,i} \right)^2$$

where  $h \in \{1, \dots, H\}$  and  $w \in \{1, \dots, W\}$  are the discretized points in space. The loss function considers only the visible species. This loss is simpler than that used in the Gray-Scott example, but is also less flexible – a pattern which to the eye appears similar in style may have a high loss, if the pixel values are different. As in the Gray-Scott training task, we train the rate constants, but for this task we also train the diffusion constants. Unlike in the winner-take-all example, the initial conditions are fixed, consisting of a simple asymmetric pattern in the visible units. We use the Optax implementation of the Adam optimizer [37, 17].



**Figure 6** Training complex reaction-diffusion pattern formation. **A.** A circular mask was placed on top of a photograph of Alan Turing was down-sampled to 100 by 100 pixels. **B.** From a random initialization, the parameters of the Hopfield ICRN were trained for 30,000 steps to minimize the pixel-wise squared distance between the target and the simulation output at time  $t_{\text{grow}}$ . **C.** Snapshots from a reaction-diffusion simulation. The initial conditions consist of three overlapping splotches of the visible species,  $x_0$ ,  $x_1$ , and  $x_2$ , plotted as R, G, B colors. The hidden channels all begin at zero. **D.** At  $t = t_{\text{grow}} = 12.5$ , this pattern, generated with the best parameters found during training, is “fully grown”. **E.** Though the loss function is applied only to the first three (visible) species, we can plot the spatial distributions of all 256 species in the ICRN. Red corresponds to a positive value and blue to a negative value.

Fig 6 shows the training of a Hopfield ICRN with  $N = 256$  neurons. The corresponding CRN has 512 species (two per dual-rail variable), 131072 ( $= 2N^2$ ) catalytic rate constants, 512 ( $= 2N$ ) degradation rate constants, and 512 ( $= 2N$ ) diffusion constants, all of which may be varied during training. This amounts to considerably more real values than the  $10^4$  grayscale pixels of the target image, helping explain the accuracy of pattern formation. In general, we observe that increasing  $N$  allows the training procedure to find a set of parameters with a lower loss.

The Hopfield CRN has several other notable differences from the Gray-Scott model. First, it is much less sensitive to how the parameters are initialized, and the choice of target pattern. Most training runs are at least somewhat successful, matching the target to some level of fidelity. Second, the Hopfield reaction-diffusion system is less stiff, and so can grow into the pattern in fewer steps without encountering numerical errors. With so many species, the state of the system occupies significant memory, and so we use gradient checkpointing to reduce the memory overheads from backpropagation-through-time [26]. Finally, note that the target image is a correct snapshot at the specified time, but is not stable – more sophisticated training procedures are discussed in [8] that improve the stability and robustness of pattern formation in Hopfield CRNs.

## 6 Discussion

We have demonstrated that the dynamics of ICRNs can be expressed and computed efficiently for systems with mass action kinetics. At this point the reader may be asking themselves whether there is a general principle for when a CRN is index-amenable and would benefit from the ICRN representation. We know of no concise characterization, and indeed the choice depends upon how simple the rate constant tensors can be. For example, any CRN involving only count-preserving bimolecular reactions can be represented as an ICRN with just one indexed reaction,  $X_i + X_j \xrightarrow{k_{ijkl}} X_k + X_l$ , together with a perhaps very sparse 4-dimensional tensor for  $k$ . A framework like this was the starting point for the prior differentiable programming of small CRNs [44]. The optimal tradeoff for the number of indexed reactions and the complexity of the rate constant tensors may be quite subtle.

Toward the goal of concise representations, observe that rate laws other than mass action as well as a variety of timescale separation conventions are used widely because they allow for simplifying assumptions, such as general pseudo-steady-state approximations, that result in more compact CRNs. For example, in systems with an enzyme concentration sufficiently lower than the substrate concentration, utilizing Michaelis-Menten kinetics allows the simulation to disregard the enzyme concentration. Appropriate formalisms and compilation methods that will produce efficient tensor operations for these cases have yet to be developed.

Once ICRNs are compiled into dynamics expressed through tensor operations, there are a variety of numerical methods for forward simulation of ICRNs, and choosing among these options is particularly important for the training of ICRNs. Simple integration methods often perform poorly on stiff problems, requiring small  $dt$  and multiple time steps, which has two significant consequences for training the ICRN. Firstly, an increase in simulation steps increases the memory requirements during backpropagation, limiting the size of ICRN that can be trained. Secondly, the learning algorithm may enter a region of the state space with low loss but with numerical artifacts. In these cases, the learned weights are invalid as the simulation no longer reflects the time evolution of the physical system. Expanding support for more sophisticated numerical methods will open doors for training stiff ICRNs.

All the ICRNs in the examples were trained by evaluating loss against a target state at the end of simulation. However, the target could be a trajectory through time and the loss could be evaluated over multiple time points. Others have trained parameters of chemical systems to fit trajectories but consider a restricted class of CRNs [44, 15]. On the other hand, our work considers the training of ICRNs with no restriction. One advantage of this flexibility is the increased potential to train experimental systems. If an experimental system can be written as an ICRN, and data regarding the target behavior can be gathered, the ICRN can be trained to perform the target behavior. Of course, there is no guarantee that the ICRN will respond well to training. The value of the ICRN formalism and software is precisely to uncover the classes of ICRNs that respond well.

DNA is especially well-suited as a molecular substrate to physically realize large ICRNs designed in this way. The biophysics of DNA-DNA interactions are relatively well-understood and modeled, and are simple enough that they can sometimes be abstracted without a critical loss of detail. Tools like NUPACK [68] enable the design of large libraries of sequences which are more or less orthogonal, enabling the creation of standardized components that can be scaled to large systems without too much cross-talk. The Hopfield CRN relies on non-orthogonal interactions (“promiscuous”, quadratic with the number of neurons), and progress has been made on tools to design DNA molecules with a range of non-orthogonal interactions [46, 3]. Prior work has also shown a theoretical basis for the morphogenesis of

arbitrary patterns in reaction-diffusion systems [55], although using a digital circuits-inspired approach which may be hard to scale to more complex patterns. On the experimental side, nucleic acids have been used to generate reaction-diffusion patterns in a wide range of approaches, as reviewed in [65]. Though the pattern-forming Hopfield CRN has not been implemented experimentally, we believe it is at least on the horizon of feasibility in DNA.

The abstraction of DNA design is not yet at the level of silicon and transistors, but is much more straightforward than the design of proteins or organic molecules. These molecules have much more complex dynamics, and in general systems built with them face the problem that parameters cannot be changed independently of each other: changing a protein's fold will affect how it interacts with all other proteins. Impressive work has been done in synthetic biology, including the synthetic morphogenesis of bacterial colonies [20] and mammalian cells [61]. Still, these patterns are closer in their complexity to those formed by simple reaction-diffusion systems like Gray-Scott. As tools for biomolecular design continue to improve – think of protein design models like AlphaFold [33] and RFDiffusion [66] – it is interesting to consider what sorts of biological design tasks will become possible.

Some biological and computational systems can also learn autonomously. This is clearly the case for humans, but even relatively simple CRNs (e.g. for linear regression) can be designed to do gradient descent on a chemically-specified loss function, learning a representation via their dynamics, rather than externally [39, 50]. *Homo sapiens* slowly changes from one generation to the next under evolution, but individual humans also learn from their experiences. This relationship can be thought of in the frame of inner-vs-outer optimization [13], or meta-learning [31]. It is thought-provoking to consider how CRNs could be designed to autonomously learn to execute more complex tasks, like pattern formation.

---

## References

- 1 Amir Aharoni, Leonid Gaidukov, Olga Khersonsky, Stephen McQ Gould, Cintia Roodveldt, and Dan S. Tawfik. The “evolvability” of promiscuous protein functions. *Nature Genetics*, 37:73–76, 2005.
- 2 Atılım Günes Baydin, Barak A. Pearlmutter, Alexey Andreyevich Radul, and Jeffrey Mark Siskind. Automatic differentiation in machine learning: a survey. *J. Mach. Learn. Res.*, 18:5595–5637, 2017.
- 3 Joseph Don Berleant. Rational design of DNA sequences with non-orthogonal binding interactions. In *International Conference on DNA Computing and Molecular Programming*, pages 4:1–4:22, 2023. doi:10.4230/LIPICS.DNA.29.4.
- 4 Pierre Boutillier, Mutaamba Maasha, Xing Li, Héctor F. Medina-Abarca, Jean Krivine, Jérôme Feret, Ioana Cristescu, Angus G. Forbes, and Walter Fontana. The Kappa platform for rule-based modeling. *Bioinformatics*, 34:i583–i592, 2018. doi:10.1093/BIOINFORMATICS/BTY272.
- 5 James Bradbury, Roy Frostig, Peter Hawkins, Matthew James Johnson, Chris Leary, Dougal Maclaurin, George Nécule, Adam Paszke, Jake VanderPlas, Skye Wanderman-Milne, and Qiao Zhang. JAX: composable transformations of Python+NumPy programs, 2018. URL: <http://github.com/jax-ml/jax>.
- 6 Dennis Bray. Protein molecules as computational elements in living cells. *Nature*, 376:307–312, 1995.
- 7 Nicolas E. Buchler, Ulrich Gerland, and Terence Hwa. On schemes of combinatorial transcription logic. *Proceedings of the National Academy of Sciences*, 100:5136–5141, 2003.
- 8 Salvador Buse and Erik Winfree. Developmental pattern formation in neural reaction-diffusion systems. Manuscript in preparation, 2025.
- 9 Chun-Hsiang Chan, Cheng-Yu Shih, and Ho-Lin Chen. On the computational power of phosphate transfer reaction networks. *New Generation Computing*, 40:603–621, 2022. doi:10.1007/S00354-022-00154-6.



- 10 Zibo Chen, James M. Linton, Shiyu Xia, Xinwen Fan, Dingchen Yu, Jinglin Wang, Ronghui Zhu, and Michael B. Elowitz. A synthetic protein-level neural network in mammalian cells. *Science*, 386:1243–1250, 2024.
- 11 Kevin M. Cherry and Lulu Qian. Scaling up molecular pattern recognition with DNA-based winner-take-all neural networks. *Nature*, 559:370–376, 2018.
- 12 Kiri Choi, J. Kyle Medley, Matthias König, Kaylene Stocking, Lucian Smith, Stanley Gu, and Herbert M. Sauro. Tellurium: an extensible Python-based modeling environment for systems and synthetic biology. *Biosystems*, 171:74–79, 2018. doi:10.1016/J.BIOSYSTEMS.2018.07.006.
- 13 Benoît Colson, Patrice Marcotte, and Gilles Savard. Bilevel programming: A survey. *4OR*, 3:87–107, 2005. doi:10.1007/S10288-005-0071-0.
- 14 Attila Csikász-Nagy, Luca Cardelli, and Orkun S. Soyer. Response dynamics of phosphorelays suggest their potential utility in cell signalling. *Journal of The Royal Society Interface*, 8:480–488, 2011.
- 15 Alexander Dack, Benjamin Qureshi, Thomas E. Ouldrige, and Tomislav Plesa. Recurrent neural chemical reaction networks that approximate arbitrary dynamics, 2024. arXiv:2406.03456.
- 16 Filipe de Avila Belbute-Peres, Kevin Smith, Kelsey Allen, Josh Tenenbaum, and J. Zico Kolter. End-to-end differentiable physics for learning and control. In *Advances in Neural Information Processing Systems*, volume 31, pages 7178–7189, 2018. URL: <https://proceedings.neurips.cc/paper/2018/hash/842424a1d0595b76ec4fa03c46e8d755-Abstract.html>.
- 17 DeepMind, Igor Babuschkin, Kate Baumli, Alison Bell, Surya Bhupatiraju, Jake Bruce, Peter Buchlovsky, David Budden, Trevor Cai, Aidan Clark, Ivo Danihelka, Antoine Dedieu, Claudio Fantacci, Jonathan Godwin, Chris Jones, Ross Hemsley, Tom Hennigan, Matteo Hessel, Shaobo Hou, Steven Kapturowski, Thomas Keck, Iurii Kemaev, Michael King, Markus Kunesch, Lena Martens, Hamza Merzic, Vladimir Mikulik, Tamara Norman, George Papamakarios, John Quan, Roman Ring, Francisco Ruiz, Alvaro Sanchez, Laurent Sartran, Rosalia Schneider, Eren Sezener, Stephen Spencer, Srivatsan Srinivasan, Miloš Stanojević, Wojciech Stokowiec, Luyu Wang, Guangyao Zhou, and Fabio Viola. The DeepMind JAX Ecosystem, 2020. URL: <http://github.com/google-deepmind>.
- 18 Li Deng. The MNIST database of handwritten digit images for machine learning research. *IEEE Signal Processing Magazine*, 29:141–142, 2012. doi:10.1109/MSP.2012.2211477.
- 19 Ramya Deshpande, Francesco Mottes, Ariana-Dalia Vlad, Michael P. Brenner, and Alma Dal Co. Engineering morphogenesis of cell clusters with differentiable programming, 2024. doi:10.48550/arXiv.2407.06295.
- 20 Salva Duran-Nebreda, Jordi Pla, Blai Vidiella, Jordi Piñero, Nuria Conde-Pueyo, and Ricard Solé. Synthetic lateral inhibition in periodic pattern forming microbial colonies. *ACS Synthetic Biology*, 10:277–285, 2021.
- 21 Irving R. Epstein and John A. Pojman. *An Introduction to Nonlinear Chemical Dynamics: Oscillations, Waves, Patterns, and Chaos*. Oxford University Press, 1998.
- 22 Martin Feinberg. *Foundations of Chemical Reaction Network Theory*. Springer International Publishing, 2019.
- 23 Leon A. Gatys, Alexander S. Ecker, and Matthias Bethge. Image style transfer using convolutional neural networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2414–2423, 2016. doi:10.1109/CVPR.2016.265.
- 24 Anthony J. Genot, Teruo Fujii, and Yannick Rondelez. Scaling down DNA circuits with competitive neural networks. *Journal of The Royal Society Interface*, 10:20130212, 2013.
- 25 Peter Gray and Stephen K. Scott. Autocatalytic reactions in the isothermal, continuous stirred tank reactor: Oscillations and instabilities in the system  $A + 2B \rightarrow 3B$ ;  $B \rightarrow C$ . *Chemical Engineering Science*, 39:1087–1097, 1984.

- 26 Andreas Griewank and Andrea Walther. Revolve: An implementation of checkpointing for the reverse or adjoint mode of computational differentiation. *ACM Transactions on Mathematical Software*, 26:19–45, 2000.
- 27 Allen Hjelmfelt, Edward D. Weinberger, and John Ross. Chemical implementation of neural networks and Turing machines. *Proceedings of the National Academy of Sciences*, 88:10983–10987, 1991.
- 28 Stefan Hoops, Sven Sahle, Ralph Gauges, Christine Lee, Jürgen Pahle, Natalia Simus, Mudita Singhal, Liang Xu, Pedro Mendes, and Ursula Kummer. COPASI—a complex pathway simulator. *Bioinformatics*, 22:3067–3074, 2006. doi:10.1093/BIOINFORMATICS/BTL485.
- 29 John J. Hopfield. Neural networks and physical systems with emergent collective computational abilities. *Proceedings of the National Academy of Sciences*, 79:2554–2558, 1982.
- 30 Fritz Horn and Roy Jackson. General mass action kinetics. *Archive for Rational Mechanics and Analysis*, 47:81–116, 1972.
- 31 Timothy Hospedales, Andreas Antoniou, Paul Micaelli, and Amos Storkey. Meta-learning in neural networks: A survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 44:5149–5169, 2021. doi:10.1109/TPAMI.2021.3079209.
- 32 Michael Hucka, Andrew Finney, Herbert M. Sauro, Hamid Bolouri, John C. Doyle, Hiroaki Kitano, Adam P. Arkin, Benjamin J. Bornstein, Dennis Bray, Athel Cornish-Bowden, et al. The systems biology markup language (SBML): a medium for representation and exchange of biochemical network models. *Bioinformatics*, 19:524–531, 2003. doi:10.1093/BIOINFORMATICS/BTG015.
- 33 John Jumper, Richard Evans, Alexander Pritzel, Tim Green, Michael Figurnov, Olaf Ronneberger, Kathryn Tunyasuvunakool, Russ Bates, Augustin Žídek, Anna Potapenko, Alex Bridgland, Clemens Meyer, Simon A. A. Kohl, Andrew J. Ballard, Andrew Cowie, Bernardino Romera-Paredes, Stanislav Nikolov, Rishub Jain, Jonas Adler, Trevor Back, Stig Petersen, David Reiman, Ellen Clancy, Michal Zielinski, Martin Steinegger, Michalina Pacholska, Tamas Berghammer, Sebastian Bodenstein, David Silver, Oriol Vinyals, Andrew W. Senior, Koray Kavukcuoglu, Pushmeet Kohli, and Demis Hassabis. Highly accurate protein structure prediction with AlphaFold. *Nature*, 596:583–589, 2021. doi:10.1038/s41586-021-03819-2.
- 34 Jongmin Kim, John Hopfield, and Erik Winfree. Neural network computation by in vitro transcriptional circuits. *Advances in Neural Information Processing Systems*, 17:681–688, 2004. URL: <https://proceedings.neurips.cc/paper/2004/hash/65f2a94c8c2d56d5b43a1a3d9d811102-Abstract.html>.
- 35 Jongmin Kim, Kristin S. White, and Erik Winfree. Construction of an in vitro bistable circuit from synthetic transcriptional switches. *Molecular Systems Biology*, 2:68, 2006.
- 36 Jongmin Kim and Erik Winfree. Synthetic in vitro transcriptional oscillators. *Molecular Systems Biology*, 7:465, 2011.
- 37 Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *International Conference on Learning Representations*, 2015. arXiv:1412.6980.
- 38 Shigeru Kondo and Takashi Miura. Reaction-diffusion model as a framework for understanding biological pattern formation. *Science*, 329:1616–1620, 2010.
- 39 Matthew R. Lakin. Design and simulation of a multilayer chemical neural network that learns via backpropagation. *Artificial Life*, 29:308–335, 2023. doi:10.1162/artl\_a\_00405.
- 40 Rahuman S. Malik-Sheriff, Mihai Glont, Tung VN Nguyen, Krishna Tiwari, Matthew G Roberts, Ashley Xavier, Manh T. Vu, Jinghao Men, Matthieu Maire, Sarubini Kananathan, et al. Biomodels—15 years of sharing computational models in life science. *Nucleic acids research*, 48:D407–D415, 2020. doi:10.1093/NAR/GKZ1055.
- 41 Aaron Meurer, Christopher P. Smith, Mateusz Paprocki, Ondřej Čertík, Sergey B. Kirpichev, Matthew Rocklin, Amit Kumar, Sergiu Ivanov, Jason K. Moore, Sartaj Singh, Thilina Rathnayake, Sean Vig, Brian E. Granger, Richard P. Muller, Francesco Bonazzi, Harsh Gupta, Shivam Vats, Fredrik Johansson, Fabian Pedregosa, Matthew J. Curry, Andy R. Terrel, Štěpán Roučka, Ashutosh Saboo, Isuru Fernando, Sumith Kulal, Robert Cimrman, and Anthony Scopatz. SymPy: symbolic computing in Python. *PeerJ Computer Science*, 3:e103, 2017. doi:10.7717/PEERJ-CS.103.

- 42 Eric Mjolsness, David H. Sharp, and John Reinitz. A connectionist model of development. *Journal of Theoretical Biology*, 152:429–453, 1991.
- 43 Alexander Mordvintsev, Eyvind Niklasson, and Ettore Randazzo. Texture generation with neural cellular automata, 2021. AI for Content Creation Workshop (CVPR). [arXiv:2105.07299](#).
- 44 Alexander Mordvintsev, Ettore Randazzo, and Eyvind Niklasson. Differentiable programming of chemical reaction networks, 2023. [doi:10.48550/arXiv.2302.02714](#).
- 45 Alexander Mordvintsev, Ettore Randazzo, Eyvind Niklasson, and Michael Levin. Growing neural cellular automata. *Distill*, 2020. <https://distill.pub/2020/growing-ca>. [doi:10.23915/distill.00023](#).
- 46 Maxim P. Nikitin. Non-complementary strand commutation as a fundamental alternative for information processing by DNA and gene regulation. *Nature Chemistry*, 15:70–82, 2023. [doi:10.1038/s41557-022-01111-y](#).
- 47 Shu Okumura, Guillaume Gines, Nicolas Lobato-Dauzier, Alexandre Baccouche, Robin Deteix, Teruo Fujii, Yannick Rondelez, and Anthony J. Genot. Nonlinear decision-making with enzymatic neural networks. *Nature*, 610:496–501, 2022.
- 48 Jacob Parres-Gold, Matthew Levine, Benjamin Emert, Andrew Stuart, and Michael B. Elowitz. Contextual computation by competitive protein dimerization networks. *Cell*, 188:1984–2002, 2025.
- 49 John E. Pearson. Complex patterns in a simple system. *Science*, 261:189–192, 1993.
- 50 William Poole, Thomas E. Ouldridge, and Manoj Gopalkrishnan. Autonomous learning of generative models with chemical reaction network ensembles. *Journal of The Royal Society Interface*, 22, 2025. [doi:10.1098/rsif.2024.0373](#).
- 51 Lulu Qian, Erik Winfree, and Jehoshua Bruck. Neural network computation with DNA strand displacement cascades. *Nature*, 475:368–372, 2011. [doi:10.1038/NATURE10262](#).
- 52 Luna Rizik, Loai Danial, Mouna Habib, Ron Weiss, and Ramez Daniel. Synthetic neuromorphic computing in living cells. *Nature Communications*, 13:5602, 2022.
- 53 David E. Rumelhart, Geoffrey E. Hinton, and Ronald J. Williams. Learning representations by back-propagating errors. *Nature*, 323:533–536, 1986.
- 54 Vojtěch Rybář. Spectral methods for reaction–diffusion systems. In *Proceedings of the Annual Conference of Doctoral Students – WDS*, volume 22, pages 97–101, 2013.
- 55 Dominic Scalise and Rebecca Schulman. Designing modular reaction-diffusion programs for complex pattern formation. *Technology*, 02:55–66, 2014. [doi:10.1142/s2339547814500071](#).
- 56 Elad Schneidman, Michael J. Berry, Ronen Segev, and William Bialek. Weak pairwise correlations imply strongly correlated network states in a neural population. *Nature*, 440:1007–1012, 2006.
- 57 Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. In *International Conference on Learning Representations*, 2015. [arXiv:1409.1556](#).
- 58 Daniel G. A. Smith and Johnnie Gray. `opt_einsum` - A Python package for optimizing contraction order for einsum-like expressions. *Journal of Open Source Software*, 3:753, 2018. [doi:10.21105/JOSS.00753](#).
- 59 David Soloveichik, Georg Seelig, and Erik Winfree. DNA as a universal substrate for chemical kinetics. *Proceedings of the National Academy of Sciences*, 107:5393–5398, 2010.
- 60 Christina J. Su, Arvind Murugan, James M. Linton, Akshay Yeluri, Justin Bois, Heidi Klumpe, Matthew A. Langley, Yaron E. Antebi, and Michael B. Elowitz. Ligand-receptor promiscuity enables cellular addressing. *Cell Systems*, 13:408–425, 2022.
- 61 Satoshi Toda, Lucas R. Blanch, Sindy K. Y. Tang, Leonardo Morsut, and Wendell A. Lim. Programming self-organizing multicellular structures with synthetic cell-cell signaling. *Science*, 361:156–162, 2018.
- 62 Alan Mathison Turing. The chemical basis of morphogenesis. *Philosophical Transactions of the Royal Society*, 237:37–72, 1953.



- 63 Marko Vasić, Cameron Chalk, Austin Luchsinger, Sarfraz Khurshid, and David Soloveichik. Programming and training rate-independent chemical reaction networks. *Proceedings of the National Academy of Sciences*, 119:e2111552119, 2022.
- 64 Marko Vasić, Cameron Chalk, Sarfraz Khurshid, and David Soloveichik. Deep molecular programming: a natural implementation of binary-weight ReLU neural networks. In *International Conference on Machine Learning*, pages 9701–9711. PMLR, 2020. URL: <http://proceedings.mlr.press/v119/vasic20a.html>.
- 65 Siyuan S. Wang and Andrew D. Ellington. Pattern generation with nucleic acid chemical reaction networks. *Chemical Reviews*, 119:6370–6383, 2019. doi:10.1021/acs.chemrev.8b00625.
- 66 Joseph L. Watson, David Juergens, Nathaniel R. Bennett, Brian L. Trippe, Jason Yim, Helen E. Eisenach, Woody Ahern, Andrew J. Borst, Robert J. Ragotte, Lukas F. Milles, Basile I. M. Wicky, Nikita Hanikel, Samuel J. Pellock, Alexis Courbet, William Sheffler, Jue Wang, Preetham Venkatesh, Isaac Sappington, Susana Vázquez Torres, Anna Lauko, Valentin De Bortoli, Emile Mathieu, Sergey Ovchinnikov, Regina Barzilay, Tommi S. Jaakkola, Frank DiMaio, Minkyung Baek, and David Baker. De novo design of protein structure and function with RFdiffusion. *Nature*, 620:1089–1100, 2023.
- 67 Xiewei Xiong, Tong Zhu, Yun Zhu, Mengyao Cao, Jin Xiao, Li Li, Fei Wang, Chunhai Fan, and Hao Pei. Molecular convolutional neural networks with DNA regulatory circuits. *Nature Machine Intelligence*, 4:625–635, 2022. doi:10.1038/S42256-022-00502-7.
- 68 Joseph N. Zadeh, Conrad D. Steenberg, Justin S. Bois, Brian R. Wolfe, Marshall B. Pierce, Asif R. Khan, Robert M. Dirks, and Niles A. Pierce. NUPACK: Analysis and design of nucleic acid systems. *Journal of Computational Chemistry*, 32:170–173, 2011. doi:10.1002/JCC.21596.

## A Symmetry

Symmetry arises when an index-amenable chemical system is naturally described by an ICRN but there are objects that can be considered equivalent with a permutation of index symbols. Specifically, we say an object of the ICRN is symmetric if the dimensions of the index set can be permuted to produce the same concrete object. There are two types of symmetry: species symmetry and reaction symmetry. Although our formalism does not explicitly model symmetries, the ICRN can capture many symmetries in the system of interest.

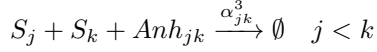
Symmetry at the level of species often reflects symmetry in the physical substrate. The annihilator of the winner-take-all network is an example of species level symmetry. The DNA substrate that implements the annihilator is a double stranded complex composed of two single strands with the same general structure, a toehold domain followed by two domains that are dependent on the value of the index symbol. For an annihilator molecule indexed  $Anh_{jk}$ , one strand has the complement of  $j$  domain followed by the  $k$  domain, and the other strand has the complement of  $k$  domain followed by the  $j$  domain. The toehold domains of the two strands are identical. Therefore, both  $Anh_{jk}$  and  $Anh_{kj}$  are composed of the same two identical single strands that hybridize identically to form the same double stranded complex. Thus, the annihilator is symmetric:  $Anh_{jk} = Anh_{kj}$ .

It is possible for the enumeration of an indexed reaction to produce multiple concrete reactions that could be considered equivalent. Reaction level symmetry can arise with or without the participation of base species that are symmetric. The annihilation reaction of the winner-take-all system exemplifies both kinds of reaction symmetry. The presence of the symmetric annihilator, with  $Anh_{jk} = Anh_{kj}$  symmetry, leads to symmetry in the annihilation reaction. The annihilation reaction  $r_{jk}$  has the same reactants and products as the reaction  $r_{kj}$ . The annihilator species exists to allow for the physical implementation of the competitive

annihilation between the  $S$  species. Thus, we can consider the idealized annihilation reaction without the annihilator. This idealized reaction  $r'_{jk}$  also displays reaction level symmetry as  $r'_{kj}$  has identical reactants and products.



There are generally two strategies that can be used so that an ICRN's dynamics matches the dynamics of symmetric index-amenable system. The tensors representing concentrations can be manipulated or the base rate constants can be manipulated. Suppose that the following index-amenable concrete CRN reactions exist.



Setting the initial concentrations and the base rate constants of the ICRN in the following way will produce the same dynamics.

$$[S]_j = [S_j], [Anh]_{jk} = \begin{cases} [Anh_{jk}] & j < k \\ 0 & \text{otherwise.} \end{cases}$$

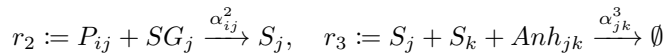
$$\begin{aligned} \frac{d[Anh]_{jk}}{dt} &= -\alpha_{jk}^3 [S]_j [S]_k [Anh]_{jk} = -\alpha_{jk}^3 [S_j] [S_k] [Anh_{jk}] = \frac{d[Anh_{jk}]}{dt} \\ \frac{d[S]_l}{dt} &= -\sum_k \alpha_{lk}^3 [S]_l [S]_k [Anh]_{lk} - \sum_j \alpha_{jl}^3 [S]_j [S]_l [Anh]_{jl} \\ &= -\sum_{l < k} \alpha_{lk}^3 [S]_l [S]_k [Anh]_{lk} - \sum_{l \geq k} \alpha_{lk}^3 [S]_l [S]_k [Anh]_{lk} \\ &\quad - \sum_{j < l} \alpha_{jl}^3 [S]_j [S]_l [Anh]_{jl} - \sum_{j \geq l} \alpha_{jl}^3 [S]_j [S]_l [Anh]_{jl} \\ &= -\sum_{l < k} \alpha_{lk}^3 [S]_l [S]_k [Anh]_{lk} - \sum_{l \geq k} \alpha_{lk}^3 [S]_l [S]_k (0) - \sum_{j < l} \alpha_{jl}^3 [S]_j [S]_l [Anh]_{jl} - \sum_{j \geq l} \alpha_{jl}^3 [S]_j [S]_l (0) \\ &= -\sum_{l < k} \alpha_{lk}^3 [S]_l [S]_k [Anh]_{lk} - \sum_{j < l} \alpha_{jl}^3 [S]_j [S]_l [Anh]_{jl} = \frac{d[S]_l}{dt} \end{aligned}$$

From the above equations, it is clear that the symmetry could also have been addressed by manipulating the base rate constant. The ICRN base rate constant  $\hat{\alpha}_{jk}^3$  can be set in the following way.

$$\hat{\alpha}_{jk}^3 = \begin{cases} \alpha_{jk}^3 & j < k \\ 0 & \text{otherwise} \end{cases}$$

## B Dynamics Example

Take for example, the process of deriving the time derivative of  $[S]$  in the winner-take-all ICRN.  $S$  is produced in the second reaction and degraded in the third reaction, where  $S$  appears twice as a reactant but with different associated indices. Let  $n = m = 3$  so that  $i \in \{1, 2, 3\}$  and  $j, k \in \{1, 2, 3\}$ . The following enumerated concrete reactions will determine the dynamics for  $S$ .



$$\text{Im}(r_2) = \begin{Bmatrix} P_{11} + SG_1 \xrightarrow{\alpha_{11}^2} S_1 & P_{12} + SG_2 \xrightarrow{\alpha_{12}^2} S_2 & P_{13} + SG_3 \xrightarrow{\alpha_{13}^2} S_3 \\ P_{21} + SG_1 \xrightarrow{\alpha_{21}^2} S_1 & P_{22} + SG_2 \xrightarrow{\alpha_{22}^2} S_2 & P_{23} + SG_3 \xrightarrow{\alpha_{23}^2} S_3 \\ P_{31} + SG_1 \xrightarrow{\alpha_{31}^2} S_1 & P_{32} + SG_2 \xrightarrow{\alpha_{32}^2} S_2 & P_{33} + SG_3 \xrightarrow{\alpha_{33}^2} S_3 \end{Bmatrix}$$

$$\text{Im}(r_3) = \begin{Bmatrix} S_1 + S_1 + Anh_{11} \xrightarrow{\alpha_{11}^3} \emptyset & S_1 + S_2 + Anh_{12} \xrightarrow{\alpha_{12}^3} \emptyset & S_1 + S_3 + Anh_{13} \xrightarrow{\alpha_{13}^3} \emptyset \\ S_2 + S_1 + Anh_{21} \xrightarrow{\alpha_{21}^3} \emptyset & S_2 + S_2 + Anh_{22} \xrightarrow{\alpha_{22}^3} \emptyset & S_2 + S_3 + Anh_{23} \xrightarrow{\alpha_{23}^3} \emptyset \\ S_3 + S_1 + Anh_{31} \xrightarrow{\alpha_{31}^3} \emptyset & S_3 + S_2 + Anh_{32} \xrightarrow{\alpha_{32}^3} \emptyset & S_3 + S_3 + Anh_{33} \xrightarrow{\alpha_{33}^3} \emptyset \end{Bmatrix}$$

Each indexed reaction flux is a real-valued tensor.

$$\Phi(r_2) = \begin{pmatrix} \alpha_{11}[P]_{11}[SG]_1 & \alpha_{12}[P]_{12}[SG]_2 & \alpha_{13}[P]_{13} + [SG]_3 \\ \alpha_{21}[P]_{21}[SG]_1 & \alpha_{22}[P]_{22}[SG]_2 & \alpha_{23}[P]_{23} + [SG]_3 \\ \alpha_{31}[P]_{31}[SG]_1 & \alpha_{32}[P]_{32}[SG]_2 & \alpha_{33}[P]_{33} + [SG]_3 \end{pmatrix}$$

$$\Phi(r_3) = \begin{pmatrix} \alpha_{11}^3[S]_1[S]_1[Anh]_{11} & \alpha_{12}^3[S]_1[S]_2[Anh]_{12} & \alpha_{13}^3[S]_1[S]_3[Anh]_{13} \\ \alpha_{21}^3[S]_2[S]_1[Anh]_{21} & \alpha_{22}^3[S]_2[S]_2[Anh]_{22} & \alpha_{23}^3[S]_2[S]_3[Anh]_{23} \\ \alpha_{31}^3[S]_3[S]_1[Anh]_{31} & \alpha_{32}^3[S]_3[S]_2[Anh]_{32} & \alpha_{33}^3[S]_3[S]_3[Anh]_{33} \end{pmatrix}$$

Every appearance of the indexed species of  $S$  contributes a summation to the dynamics expression.  $(S, j)$  appears in the second indexed reaction, and both  $(S, j)$  and  $(S, k)$  appear in the third reaction. Therefore, we have three contributions the dynamics of  $[S]$ :  $\delta_{r_2}(S, j), \delta_{r_3}(S, j), \delta_{r_3}(S, k)$ . To find each  $\delta$ , the sum of the indexed reaction flux is taken  $\Phi$  is taken over the dimensions corresponding to index symbols that are in the indexed reaction and not in the indexed species. For  $\delta_{r_2}(S, j)$ , the indexed symbol  $i$  participates in the reaction  $r_2$  but is not present in the indexed species  $(S, j)$ , so the sum proceeds over  $i$ . This summation naturally describes the element-wise specification for  $\delta_{r_2}(S, j)$ , indexed by  $j$ .

$$\delta_{r_2}(S, j)_j = \sum_i \alpha_{ij}^2[P]_{ij}[SG]_j, \quad \delta_{r_3}(S, j)_j = \sum_k \alpha_{jk}^3[S]_j[S]_k[Anh]_{jk},$$

$$\delta_{r_3}(S, k)_k = \sum_j \alpha_{jk}^3[S]_j[S]_k[Anh]_{jk}$$

Each  $\delta$  is a real-valued tensor.

$$\delta_{r_2}(S, j) = \begin{pmatrix} \sum_i \alpha_{i1}^2[P]_{i1}[SG]_1 \\ \sum_i \alpha_{i2}^2[P]_{i2}[SG]_2 \\ \sum_i \alpha_{i3}^2[P]_{i3}[SG]_3 \end{pmatrix}, \quad \delta_{r_3}(S, j) = \begin{pmatrix} \sum_k \alpha_{1k}^3[S]_1[S]_k[Anh]_{1k} \\ \sum_k \alpha_{2k}^3[S]_2[S]_k[Anh]_{2k} \\ \sum_k \alpha_{3k}^3[S]_3[S]_k[Anh]_{3k} \end{pmatrix},$$

$$\delta_{r_3}(S, k) = \begin{pmatrix} \sum_j \alpha_{j1}^3[S]_j[S]_1[Anh]_{j1} \\ \sum_j \alpha_{j2}^3[S]_j[S]_2[Anh]_{j2} \\ \sum_j \alpha_{j3}^3[S]_j[S]_3[Anh]_{j3} \end{pmatrix}$$

These summations can then be combined, accounting for the difference in product and reactant stoichiometric coefficients for each appearance of  $S$ .

$$\frac{d[S]}{dt} = (1-0)\delta_{r_2}(S, j) + (0-1)\delta_{r_3}(S, j) + (0-1)\delta_{r_3}(S, k) = \delta_{r_2}(S, j) - \delta_{r_3}(S, j) - \delta_{r_3}(S, k)$$

The reader can confirm that the mass action dynamics for  $[S]$ , derived from symbolic manipulation matches the dynamics from the enumerated concrete CRN.


$$\frac{d[S]_j}{dt} = \frac{\alpha_{1j}^2[P]_{1j}[SG]_j + \alpha_{2j}^2[P]_{2j}[SG]_j + \alpha_{3j}^2[P]_{3j}[SG]_j}{-\alpha_{j1}^3[S]_j[S]_1[Anh]_{j1} - \alpha_{j2}^3[S]_j[S]_2[Anh]_{j2} - \alpha_{j3}^3[S]_j[S]_3[Anh]_{j3}} = \frac{d[S]_j}{dt}$$



# Programmable Co-Transcriptional Splicing: Realizing Regular Languages via Hairpin Deletion

Da-Jung Cho ✉ 

Department of Software and Computer Engineering, Ajou University, Suwon, Republic of Korea

Szilárd Zsolt Fazekas ✉ 

Graduate School of Engineering Science, Akita University, Japan

Shinnosuke Seki ✉ 

University of Electro-Communications, Tokyo, Japan

Max Wiedenhöft ✉ 

Department of Computer Science, Kiel University, Germany

---

## Abstract

RNA co-transcriptionality, where RNA is spliced or folded during transcription from DNA templates, offers promising potential for molecular programming. It enables programmable folding of nanoscale RNA structures and has recently been shown to be Turing universal. While post-transcriptional splicing is well studied, co-transcriptional splicing is gaining attention for its efficiency, though its unpredictability still remains a challenge. In this paper, we focus on engineering co-transcriptional splicing, not only as a natural phenomenon but as a programmable mechanism for generating specific RNA target sequences from DNA templates. The problem we address is whether we can encode a set of RNA sequences for a given system onto a DNA template word, ensuring that all the sequences are generated through co-transcriptional splicing. Given that finding the optimal encoding has been shown to be NP-complete under the various energy models considered [4], we propose a practical alternative approach under the logarithmic energy model. More specifically, we provide a construction that encodes an arbitrary nondeterministic finite automaton (NFA) into a circular DNA template from which co-transcriptional splicing produces all sequences accepted by the NFA. As all finite languages can be efficiently encoded as NFA, this framework solves the problem of finding small DNA templates for arbitrary target sets of RNA sequences. The quest to obtain the *smallest* possible such templates naturally leads us to consider the problem of minimizing NFAs and certain practically motivated variants of it, but as we show, those minimization problems are computationally intractable.

**2012 ACM Subject Classification** Applied computing → Bioinformatics

**Keywords and phrases** RNA Transcription, Co-Transcriptional Splicing, Finite Automata Simulation, NFA Minimization

**Digital Object Identifier** 10.4230/LIPIcs.DNA.31.5

**Related Version** *Full Version:* <https://arxiv.org/abs/2506.23384>

**Funding** *Da-Jung Cho* was supported by Institute of Information & communications Technology Planning & Evaluation (IITP) under the Artificial Intelligence Convergence Innovation Human Resources Development (IITP-2025-RS-2023-00255968) grant funded by the Korea government (MSIT). *Szilárd Zsolt Fazekas* was supported by JSPS Kakenhi Grant No. 23K10976. *Max Wiedenhöft's* work was partially supported by the DFG project number 437493335.

## 1 Introduction

RNA splicing is a fundamental process in eukaryotic gene expression, where intronic sequences are removed from precursor messenger RNA (pre-mRNA) transcripts, and the remaining exonic sequences are ligated together to form a mature messenger RNA (mRNA). This essential process is mediated by the spliceosome, a dynamic and large macromolecular complex consisting of small nuclear RNAs (snRNAs) and associated proteins. The spliceosome



© Da-Jung Cho, Szilárd Zsolt Fazekas, Shinnosuke Seki, and Max Wiedenhöft; licensed under Creative Commons License CC-BY 4.0

31st International Conference on DNA Computing and Molecular Programming (DNA 31).

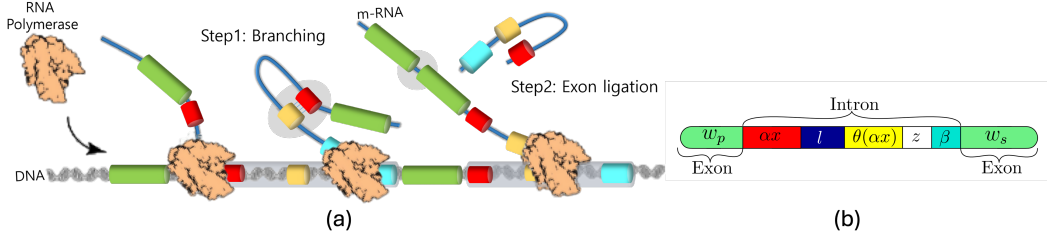
Editors: Josie Schaeffer and Fei Zhang; Article No. 5; pp. 5:1–5:22

Leibniz International Proceedings in Informatics



LIPICs Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

assembles on the pre-mRNA in a highly conserved and sequential manner. The assembly involves the sequential recruitment of specific snRNPs (small nuclear ribonucleoproteins) and various factors that play critical roles in splice site recognition and catalysis. The process begins when the 5'-splice site (5'-SS) is identified by the polymerase-spliceosome complex. This site is then kept in close proximity to the transcribed region of the RNA, awaiting the hybridization event with the complementary sequence at the 3'-splice site (3'-SS) after subsequent transcription of the RNA. Once the 3'-SS is transcribed, the intronic sequence is excised, and the two exons are ligated together to form the mature RNA. Splicing takes place while transcription is still ongoing, meaning that the spliceosome assembles and acts on the RNA as it is being made. This close coordination between transcription and splicing is known as *co-transcriptional splicing*. Increasing biochemical and genomic evidence has established that spliceosome assembly and even catalytic steps of splicing frequently occur co-transcriptionally [16]. As shown in Figure 1, co-transcriptional splicing involves the spliceosome's early recognition and action on splice sites while the RNA is still being transcribed. This coupling imposes temporal and structural constraints on splice site selection and nascent RNA folding, since spliceosome assembly and splicing catalysis occur during ongoing transcription.



**Figure 1** (a) An illustration of co-transcriptional splicing: The process begins when the 5'-splice site (5'-SS) is identified by the polymerase-spliceosome complex. This site is then kept in close proximity to the transcribed region of the RNA, awaiting the hybridization event with the complementary sequence at the 3'-splice site (3'-SS) after subsequent transcription of the RNA. The transcript forms a hairpin where the 5'-SS binds the branch point. After the 3'-SS is transcribed the spliceosome removes the intron and joins the exons. (b) A string representation of the DNA sequence in (a), factorized according to the scheme of co-transcriptional splicing, as used in Definition 1.

While co-transcriptional splicing has been studied as a natural phenomenon, it is increasingly being recognized as a programmable and engineerable process. Indeed, Geary, Rothmund, and Andersen have proved that one of such processes called co-transcriptional folding is programmable to assemble nanoscale single-stranded RNA structures in vitro [9] and then Seki, one of the authors, proved in collaboration with Geary and others that it is Turing universal by using an oritatami model [8]. Recently, we became curious about the potential of using RNA sequences as programmable components in molecular systems. Motivated by advances in co-transcriptional RNA folding and splicing, this raises the question: Can we encode a set of RNA sequences for a given system onto a single DNA template such that all the sequences are generated through co-transcriptional splicing, while avoiding sequences that may disrupt the system? As transcription proceeds and the 3'-SS is transcribed, RNA polymerase often pauses just downstream, allowing time for accurate splice site recognition and splicing to take place. The folding of the emerging RNA strand can influence this process. Motivated by this perspective, we introduced a formal model of co-transcriptional splicing [4], defined as *contextual lariat deletion* under various biologically plausible conditions

(energy models). For the sake of simplicity and to avoid confusion with the rather different but similarly named *contextual deletion* [14], we will refer to the operation here as *hairpin deletion*. We showed that the template construction problem is NP-complete when  $|\Sigma| \geq 4$ . The hardness of this problem led us to consider an alternative approach for the template construction problem. A natural observation is that the DNA template word can be obtained by finding the shortest common supersequence of the RNA target sequences. However, this problem is also known to be NP-complete [7, 15]. The contributions of this paper are as follows:

- **(Parallel) Logarithmically-bounded hairpin deletion operation:** In natural RNA folding, the destabilizing energy contributed by loop regions in hairpin structures increases logarithmically with loop size. While small loops incur significant energy penalties, this effect diminishes as loop length increases [6]. This behavior supports the logarithmic-bounded hairpin model, where the destabilization caused by a loop of length  $\ell$  is proportional to  $\log(\ell)$ . Our model exploits this property by favoring hairpin structures with long loops and short stems, which remain energetically viable under the logarithmic cost function. Furthermore, we introduce a parallel deletion model to represent this consecutive splicing behavior since co-transcriptional splicing tends to proceed consecutively rather than recursively due to kinetic and structural constraints during transcription.
- **Template word construction by encoding RNA target sequences to a finite automata on a circular DNA template word:** In Section 3, our main result shows, if we consider a set of target RNA sequences as a regular language represented by some (non)deterministic finite automaton, we can construct a DNA template from which we can obtain exactly that language using the logarithmic hairpin deletion model. The construction operates on a circular DNA template, which is widely used in molecular biology, for instance in plasmids and viral genomes [5, 9], and supports continuous transcription. We first demonstrate how to construct such a circular DNA word that encodes an arbitrary NFA according with the behavior of logarithmic hairpin deletion. Then, a detailed construction for the RNA alphabet  $\Sigma_{\text{RNA}} = \{\text{A}, \text{U}, \text{C}, \text{G}\}$  is also provided, illustrating how the states and transitions of the automaton are encoded. It is shown that this construction actually works for an appropriately designed DNA template word under the logarithmic hairpin deletion model.
- **Minimizing NFA-like models:** As the size of the constructed template depends on the number of states and transitions in the automaton, minimizing the NFA becomes an important consideration. As the minimization problem for NFAs is NP-hard in general [12], in Section 4, we consider restricted NFA-like models that fit our setting. We show that minimization in these restricted cases is still computationally hard. This highlights a fundamental complexity barrier in optimizing DNA template designs for generating arbitrary regular languages of RNA sequences via co-transcriptional splicing. Therefore, it may be more effective to focus on designing specific classes of target sequences and their corresponding automata representations.

## 2 Preliminaries

Let  $\mathbb{N}$  denote the set of positive integers and let  $\mathbb{N}_0 = \mathbb{N} \cup \{0\}$ . Let  $\mathbb{Z}$  denote the set of integers. For some  $m \in \mathbb{N}$ , denote by  $[m]$  the set  $\{1, 2, \dots, m\}$  and let  $[m]_0 = [m] \cup \{0\}$ . With  $\Sigma$ , we denote a finite set of symbols, called an *alphabet*. The elements of  $\Sigma$  are called *letters*. A *word* over  $\Sigma$  is a finite sequence of letters from  $\Sigma$ . With  $\varepsilon$ , we denote the *empty word*. The set of all words over  $\Sigma$  is denoted by  $\Sigma^*$ . Additionally, set  $\Sigma^+ = \Sigma^* \setminus \{\varepsilon\}$ .



Given some word  $w \in \Sigma^*$ , the word  $w^\omega$  represents an infinite repetition of  $w$ , called *circular word*. The *length* of a word  $w \in \Sigma^*$ , i.e., the number letters in  $w$ , is denoted by  $|w|$ ; hence,  $|\varepsilon| = 0$ . For some  $w \in \Sigma^*$ , if we have  $w = xyz$  for some  $x, y, z \in \Sigma^*$ , then we say that  $x$  is a *prefix*,  $y$  is a *factor*, and  $z$  is a *suffix* from  $w$ . We denote the set of all factors, prefixes, and suffixes of  $w$  by  $\text{Fact}(w)$ ,  $\text{Pref}(w)$ , and  $\text{Suff}(w)$  respectively. If  $x \neq w$ ,  $y \neq w$ , or  $z \neq w$  respectively, we call the respective prefix, factor, or suffix *proper*. For some word  $w = xy$ , for  $x, y \in \Sigma^*$ , we define  $w \cdot y^{-1} = x$  as well as  $x^{-1} \cdot w = y$ . A function  $\theta : \Sigma^* \rightarrow \Sigma^*$  is called an *antimorphic involution* if  $\theta(\theta(a)) = a$  for all  $a \in \Sigma$  and  $\theta(wb) = \theta(b)\theta(w)$  for all  $w \in \Sigma^*$  and  $b \in \Sigma$ . Watson-Crick complementarity, which primarily governs hybridization among DNA and RNA sequences, has been modeled as an antimorphic involution that maps **A** to **T** (**U**), **C** to **G**, **G** to **C**, and **T** (**U**) to **A**. A nondeterministic finite automaton (NFA) is a tuple  $A = (\Sigma, Q, q_0, \delta, F)$  where  $\Sigma$  is the input alphabet,  $Q$  is the finite set of states,  $\delta : Q \times \Sigma \rightarrow 2^Q$  is the multivalued transition function,  $q_0 \in Q$  is the initial state, and  $F \subseteq Q$  is the set of final states. In the usual way,  $\delta$  is extended as a function  $Q \times \Sigma^* \rightarrow 2^Q$  and the language accepted by  $A$  is  $L(A) = \{w \in \Sigma^* \mid \delta(q_0, w) \cap F \neq \emptyset\}$ . The automaton  $A$  is a deterministic finite automaton (DFA) if  $\delta$  is a single valued partial function. It is well known that deterministic and nondeterministic finite automata recognize the class of *regular languages* [11]. Finally, for a regular expression  $r$ , writing it down, immediately refers to its language, e.g., writing  $a(a|b)^*b$  refers to the set  $\{w \in \{a, b\}^* \mid w[1] = a, w[|w|] = b\}$ .

## 2.1 Hairpin deletion

Hairpin deletion is a formal operation introduced in [4] with the purpose of modeling co-transcriptional splicing in RNA transcription. The computational power of the operation in various energy models (length restrictions between the stem and the loop) has been thoroughly investigated in [4]. Here we introduce only the elements relevant to our results (in the *logarithmic energy model*). A pair of words,  $(u, v) \in \Sigma^* \times \Sigma^*$ , serves as a context;  $u$  and  $v$  are particularly called *left* and *right* contexts. A set  $C$  of such pairs is called a *context set*. Hairpins are an atom of DNA and RNA structures. They can be modeled as  $x\ell\theta(x)$  for words  $x, \ell \in \Sigma^*$ , where  $x = b_1b_2 \cdots b_n$  and its Watson-Crick complement  $\theta(x) = \theta(b_n) \cdots \theta(b_2)\theta(b_1)$  hybridize with each other via hydrogen bonds between bases  $b_i$  and  $\theta(b_i)$  into a stem, and  $\ell$  serves as a loop (in reality  $|\ell| \geq 3$  is required) [13]. Assume  $\Sigma$  to be some alphabet,  $w \in \Sigma^*$  to be some word over that alphabet, and  $\theta$  to be some antimorphic involution on  $\Sigma$ . We call  $w$  a *hairpin* if  $w = x\ell\theta(x)$ , for some  $x, \ell \in \Sigma^*$ . Furthermore, let  $c \in \mathbb{N}$  be some positive number. We call  $w$  a *logarithmically-bounded hairpin* (*log-hairpin*) if  $|x| \geq c \cdot \log(\ell)$ . We denote the set of all hairpins regarding  $\Sigma$  and  $\theta$  by  $H(\Sigma, \theta)$  and the sets of all log-hairpins regarding  $\Sigma$ ,  $\theta$ , and  $c$  by  $H_{\log}(\Sigma, \theta, c)$ . For readability purposes, we often write just  $H$  or  $H_{\log}$  and infer  $\Sigma$ ,  $\theta$  and  $c$  by the context. Notice that the logarithmic penalty of  $\ell$  in log-hairpins allows for an exponential size of the loop with respect to the length of the stem.

Using the notion of hairpins and log-hairpins, we can continue with the definition of bounded hairpin deletion. Splicing relies on the occurrence of left and right contexts, the formation of a stable hairpin containing the left context, and a potential margin between the hairpin and the occurrence of the right context. As only the logarithmic energy model is relevant in this paper, we focus only on log-hairpins. Figure 1(b) shows a string factorization where the logarithmic bounded hairpin deletion operation can be applied, reflecting the co-transcriptional splicing process.

► **Definition 1.** Let  $S = (\Sigma, \theta, c, C, n)$  be a tuple of parameters, with  $\Sigma$  an alphabet,  $w \in \Sigma$  a word,  $C$  a context-set,  $\theta$  an antimorphic involution,  $n \in \mathbb{N}$  a constant called margin, and  $c \in \mathbb{N}$  a multiplicative factor used in the logarithmic energy model definition. If

$$w = w_p \alpha x \ell \theta(x) \theta(\alpha) z \beta w_s$$



for some  $(\alpha, \beta) \in C$  and  $w_p, w_s, x, \ell, z \in \Sigma^*$ , then we say that  $w_p w_s$  is obtainable by logarithmic bounded hairpin deletion (or just log-hairpin deletion) from  $w$  over  $S$  (denoted  $w \xrightarrow{\log}_S w_p w_s$ ) if and only if  $\alpha x \ell \theta(x \alpha) \in H_{log}$  and  $|z| \leq n$ .

For example, if  $\theta$  is defined to represent the Watson-Crick complement, the context-set is set to  $C = \{(\text{AUA}, \text{CCC})\}$ , the margin is set to  $n = 1$ , the log-factor is set to  $c = 1$ , and  $w = \text{AAUAACCUUAUGCCCCGA}$ , then we have

$$\text{AAUAACCUUAUGCCCCGA} \xrightarrow{\log} \text{AGA}$$

by  $w_p = \text{A}$ ,  $\alpha = \text{AUA}$ ,  $x = \text{A}$ ,  $\ell = \text{CC}$ ,  $\theta(x)\theta(\alpha) = \text{UUAU}$ ,  $z = \text{G}$ ,  $\beta = \text{CCC}$ , and  $w_s = \text{GA}$ .

Given some input word or some input language, we can consider the language of all words obtained if hairpin deletion is allowed to be applied multiple times. In practice, it is observed that co-transcriptional splicing occurs in a consecutive manner on the processed DNA sequence, meaning that newly created splicing sites resulting from earlier splicing operations do not affect the outcome. Hence, we introduce a parallel deletion model in which only non-overlapping hairpins present in the sequence at the beginning can be deleted, representing consecutive co-transcriptional splicing.

► **Definition 2.** Let  $w \in \Sigma^*$  be a word and let  $S$  be a tuple of parameters for log-hairpin deletion. For some  $w' \in \Sigma^*$ , we call it obtainable by parallel log-hairpin deletion of  $w$  over  $S$ , denoted  $w \xrightarrow{p'log}_S w'$ , if there exist  $m \in \mathbb{N}$  and  $u_1, \alpha_1, \dots, u_m, \alpha_m, u_{m+1} \in \Sigma^*$  such that

$$w = u_1 \alpha_1 u_2 \alpha_2 \dots u_m \alpha_m u_{m+1},$$

$$w' = u_1 u_2 \dots u_m u_{m+1},$$

and, for all  $i \in [m]$ , we have  $\alpha_i \xrightarrow{\log}_S \varepsilon$ , i.e., each  $\alpha_i$  is removed by log-hairpin deletion. For readability purposes, we may just write  $\xrightarrow{\log}$  and infer  $S$  or  $\log$  by context.

For example, given the context set  $C = \{(\text{AA}, \text{CC}), (\text{CC}, \text{CC})\}$ ,  $\theta$  defined by  $\theta(\text{A}) = \text{U}$  and  $\theta(\text{C}) = \text{G}$ , and the word  $w = \text{CAGAGUCUGCCAAGGG}$ , we could delete two factors at once:

$$\text{CA AAGUCC UG CCAAGGCC G} \xrightarrow{p} \text{CAUGG}$$

Applying parallel log-hairpin deletion to a word or a given language can produce another language. Hence, given some  $w \in \Sigma^*$  or  $L \subseteq \Sigma^*$ , the parallel log-hairpin deletion sets over  $S$  of  $w$  (or  $L$ ) are defined by

$$[w]_{\xrightarrow{p'log}_S} = \{ w' \in \Sigma^* \mid w \xrightarrow{\log}_S w' \} \text{ (or } [L]_{\xrightarrow{p'log}_S} = \bigcup_{w \in L} [w]_{\xrightarrow{p'log}_S} \text{)}.$$

In practice and for the purpose of the following constructions in the main body of the paper, we define a variant of the parallel hairpin deletion that uses certain necessary assumptions about the words contained in the language. First of all, if there exist isolated 5'SS (left contexts) in a word that is read from left to right, there is a high chance that it is used in the formation of the hairpin if a corresponding 3'SS exists [2, 17]. In that sense, we assume some greediness when it comes to selecting left contexts. Similarly, we could assume the same about the choice of the right context. For the construction that follows, only the first assumption is necessary. If there exist overlapping 5'SS in the word, thermodynamics might result in some nondeterministic choice of the actual 5'SS that is used [18, 19]. To obtain a model that follows these constraints, we introduce a variant of parallel deletion, called maximally parallel deletion, in which it is assumed that no left contexts survive in each part  $u_i$ .

► **Definition 3.** Let  $w \in \Sigma^*$  be a word and  $S$  be some tuple of parameters for log-hairpin deletion. For some  $w' \in \Sigma^*$ , we call it obtainable by maximally parallel log-hairpin deletion over  $S$  of  $w$ , denoted  $w \xrightarrow[\text{p}\uparrow\text{-log}]{\text{p}\uparrow\text{-log}}_S w'$ , if there exists some  $m \in \mathbb{N}$  and  $u_i, \alpha_i, u_{m+1} \in \Sigma^*$ ,  $i \in [m]$ , such that  $w = u_1 \alpha_1 \dots u_m \alpha_m u_{m+1}$ ,  $w' = u_1 \dots u_{m+1}$ , bounded hairpin deletion cannot be applied to  $u_{m+1}$ , and, for all  $i \in [m]$ , we have  $\alpha_i \xrightarrow[\text{log}]{\text{log}}_S \varepsilon$  as well as, for all  $(x, y) \in C$ , we have  $x \notin \text{Fact}(u_i)$ .

Again, we denote the set of all words obtainable by maximally parallel log-hairpin deletion by  $[w]_{\text{p}\uparrow\text{-log}}^{\text{p}\uparrow\text{-log}}_S$  (analogously for input languages as before). Notice that  $[w]_{\text{p}\uparrow\text{-log}}^{\text{p}\uparrow\text{-log}}_S \subseteq [w]_{\text{p}\uparrow\text{-log}}^{\text{p}\uparrow\text{-log}}_S$  as it is a more restricted variant of the parallel log-hairpin deletion set. This concludes all necessary introductory terminology regarding the formal model of hairpin deletion.

### 3 Simulating Finite Automata with Maximal Parallel Log-Hairpin Deletion

This section investigates the possibility to obtain arbitrary regular languages of RNA sequences from a circular template DNA sequence using bounded hairpin deletion. We provide a construction that allows for the simulation of arbitrary (non)deterministic finite automata (DFA/NFA) using maximally parallel bounded hairpin deletion in the logarithmic energy model on circular DNA. In particular it is shown, given some NFA  $A$ , that we can construct a word  $w$  for which we have  $[w^\omega]_{\text{p}\uparrow\text{-log}}^{\text{p}\uparrow\text{-log}} = L(A)$ .

As an initial idea, each transition defined in some NFA  $A$  was encoded on a circular word in a consecutive matter, i.e., if for example  $A$  had the transitions  $(q_i, a, q_j)$  and  $(q_j, b, q_\ell)$ , then  $w$  would contain them directly as factors, resulting in a word  $w = \dots (q_i, a, q_j) \dots (q_j, b, q_\ell) \dots$ . A context-set  $C$  can be defined to contain some context  $(\alpha, \beta) \in C$  with  $\alpha = ,q_j)$  and  $\beta = (q_j,$ , that would allow for the potential deletion of the factor  $,q_j) \dots (q_j,$  and resulting in a word  $w' = \dots (q_i, ab, q_\ell)$ , allowing for further parallel deletions. This model works quite intuitively, but resulted in various problems. For example, the controlled formation of hairpins with a stem and a loop posed a serious challenge. Also a controlled notion of termination was missing. Due to the above, we decided on a different approach. Now, instead of encoding all transitions in a parallel manner, we use a single factor  $s_i$  per state  $q_i$  that allows a non-deterministic transition to jump to some other factor  $s_j$ , encoding  $q_j$ , using hairpin deletion. The general spirit, however, stays the same, as we are still moving around a circular DNA template and select transitions to be taken non-deterministically by the application of hairpin deletion, leaving only the letters labeling those transitions to remain in the resulting words.

First, we need some section-specific definitions, as we are now considering and utilizing infinite repetitions of some circular DNA template. As mentioned in Section 2, a circular word  $w^\omega$  is an infinite repetition of some word  $w \in \Sigma^*$ . In order to extend the notion of hairpin deletion to infinite words, we need some notion of intentionally stopping the transcription process on infinite words.

In practice there are, among others, two different ways to stop the transcription process and detach the produced RNA from the polymerase. First, there is rho-dependent transcription termination which uses other molecules that bind to certain factors on the produced RNA [20]. Second, there is rho-independent transcription termination that is based on the formation of a hairpin of certain length followed by a specific factor on the template DNA [3]. The rho-dependent model would be easier to embed in our definition, but it relies on external factors to work. Hence, to obtain a process that works as independent from external factors as possible, we will use the latter, rho-independent, model.

Most of the time, after the final hairpin, a factor containing only a certain number of U's/T's is enough to result in the decoupling of the produced RNA sequence from the polymerase. To obtain a general model, we allow for arbitrary words to be used at this point, elements of some  $T \subset \Sigma^*$ , called the *set of terminating-sequences*. We extend the notion of log-hairpin deletion by adding  $T$  and a *terminating stem-length*  $m \in \mathbb{N}$  to the properties  $S$ , i.e., writing  $S = (\Sigma, \theta, c, C, n, T, m)$  instead of  $S = (\Sigma, \theta, c, C, n)$ , if the processed word is a circular word  $w^\omega$  over  $w \in \Sigma^*$ . The number  $m$  represents the minimal length of the hairpin or stem that must be formed as a suffix in the RNA produced, before reading a terminating-sequence  $t \in T$  in  $w^\omega$ . Using this, we can adapt the notion of log-hairpin deletion for circular words by adding the notion of termination.

► **Definition 4.** Let  $w^\omega$  be a circular word over  $w \in \Sigma^*$ . Let  $S = (\Sigma, \theta, c, C, n, T, m)$  be a tuple of properties for log-hairpin deletion as defined before. Assume there exists a finite prefix  $w't$  of  $w^\omega$  with  $w' \in \Sigma^*$  and  $t \in T$  such that  $w' \xrightarrow{S} u$ , for some word  $u \in \Sigma^*$ . If  $u$  has a suffix  $s$  of length  $|s| = 2m$  such that  $s$  forms a stem<sup>1</sup>, i.e.,  $s = x\theta(x)$ , for some  $x \in \Sigma^*$ , then we say that  $u \cdot s^{-1}$  is obtainable from  $w^\omega$  by terminating log-hairpin deletion.

The remaining part of this section provides a general framework for constructing working DNA templates. Properties needed for the construction to work are provided. The following result is the main result of this section and contains the general construction of a template word  $w$  that allows for the simulation of arbitrary regular languages represented by finite automata using terminating maximally parallel bounded log-hairpin deletion. We provide a basic construction for NFAs which naturally follows for DFAs and potentially other finite automata models such as GNFA's or GDFAs (which allow sequences as transition labels instead of just single letters).

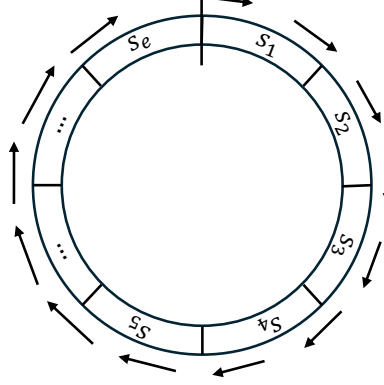
► **Theorem 5.** Let  $A = (Q, \Sigma, q_1, \delta, F)$  be some NFA. There exists a word  $w \in \Sigma^*$  and a properties tuple for log-hairpin deletion  $S = (\Sigma, \theta, c, C, n, T, m)$  as defined before such that  $L(A) = [w^\omega]_{\xrightarrow{S}^{\uparrow \log}}$ .

**Proof.** Let  $A = (Q, \Sigma, q_1, \delta, F)$  be some NFA with  $Q = \{q_1, \dots, q_o\}$ , for some  $o \in \mathbb{N}$ , and  $\Sigma = \{a_1, \dots, a_\sigma\}$ , for some  $\sigma \in \mathbb{N}$ . We begin by an explanation of the basic idea. Then, we continue with a formal construction which is then given an intuitive explanation that utilizes images afterwards. Due to space constraints, the full proof of the correctness of the construction is given in Appendix A. Additionally, an example of an actual implementation using the RNA alphabet  $\Sigma_{\text{RNA}} = \{A, U, C, G\}$  for a given NFA is provided in Appendix C.2.

**Basic idea.** For each state  $q_i \in Q$ , we construct a word  $s_i \in \Sigma^*$  that represents this state and the outgoing transitions from it, as defined by  $\delta$ . Each period  $w \in \Sigma^*$  of the circular word  $w^\omega$ , over which transcription is done, will be essentially made up of a concatenation of all  $s_i$ 's with the addition of one final word  $s_e$ , i.e., we obtain  $w = s_1 s_2 \dots s_o s_e$  (see Figure 2). The suffix  $s_e$  handles transcription termination. Hairpin deletion will be used to simulate a transition between two connected states  $q_i$  and  $q_j$ , while reading a letter  $a \in \Sigma$ . This is done by jumping from  $s_i$  to  $s_j$  by removing everything but the letter  $a$  between  $s_i$  and  $s_j$ . This is implemented using a two-step process for which context-sets will be deliberately designed

<sup>1</sup> The requirement that  $s$  is only a stem of length  $2m$  and not a hairpin with a stem of such length, i.e., no loop  $\ell$  is formed, is a technical one chosen for simplicity reasons. A long enough stem always allows for a loop to be formed. Hence, adapting the encoded suffix responsible for transcription termination in the following construction in the proof of Theorem 5 should also be possible by setting  $m$  short enough and adding enough letters to that suffix that do not interfere with the constructed context-set

such that these hairpin deletion steps actually simulate the process of reading letters in the automaton  $A$ . To terminate transcription, in final-states, a technical transition to the end of the template can be used that jumps to a factor  $t \in T$  at the end of  $w$  while obtaining a suffix which consists of a stem of size  $2m$  at the end of the transcribed word. By that, we successively build prefixes of words in  $L(A)$  and may finalize the transcription process anytime a final state is reached, effectively obtaining only words in  $L(A)$ .



■ **Figure 2** Representation of circular transcription (maximally parallel hairpin deletion over  $w^\omega$ ).

**Construction.** We continue with the formal construction of  $w$ . After that, we give an intuitive sketch of the functionality.<sup>2</sup> We need some general assumptions.  $\theta$  is not defined specifically in the general case, but rather implied by the constraints we give in this proof. The logarithmic factor  $c$  is set to 1 in this construction, but larger numbers also work. The margin number  $n$  is set to 0 in this construction. Such a constant may be added but is not necessary for this construction to work.  $\Sigma$  is copied from the definition of  $A$ , but a distinct alphabet may simplify the construction. For practical implementation purposes, we use this assumption to keep the possible alphabet size as low as possible. Consider the construction in the Appendix to see an actual working example of an encoding for an alphabet size of 4.  $T$  and  $m$  are not explicitly specified, but construction constraints are given.

In the final construction, we obtain a word  $w = s_1 \cdots s_o s_e$  for words  $s_i \in \Sigma^*$ ,  $i \in [o]$ , representing the states, and a technical final word  $s_e \in \Sigma^*$  that is also responsible for transcription termination. From now on, for each newly introduced word, assume that its specification or characterization via constraints is given later on in the proof (if it is not given immediately or before). Also, assume that any word that occurs as one side of a context in the context-set  $C$  (which is to be constructed later) does not appear anywhere else by other overlapping factors in  $w$ .

We start with the construction of  $s_e$ . Let  $f_s, f_e \in \Sigma^*$  be two words such that  $|f_s f_e| = 2m$  and  $f_s = \theta(f_e)$ , hence,  $f_s f_e$  forms a stem of length  $2m$ , i.e.,  $f_e = \theta(f_s)$ . Let  $T = \{t\}$  be a set of terminating sequences containing only one word  $t \in \Sigma^*$ . We set the suffix  $s_e$  by

$$s_e = \theta(S_{end})E_{end} f_e t \theta(S_{q_1})E_{q_1}$$

for words  $S_{end}, E_{end}, E_{q_1} \in \Sigma^*$ . Next, for each state  $q_i \in Q$ , we construct a word  $s_i \in \Sigma^*$  by setting

$$s_i = S_{i,1} \cdots S_{i,k} S_{i,k+1} e_{i,1} \cdots e_{i,k} e_{i,k+1} \theta(S_{q_{i+1}}) E_{q_{i+1}}$$

<sup>2</sup> Consider reading the intuition-part in parallel to the construction-part for a better understanding.

for words  $S_{q_{i+1}}, E_{q_{i+1}}, S_{i,1}, \dots, S_{i,k+1}, e_1, \dots, e_{k+1} \in \Sigma^*$ , with  $k$  being the number of outgoing transitions of  $q_i$ . If  $q_i = q_o$ , then  $S_{q_{i+1}} = E_{q_{i+1}} = \varepsilon$  is just an empty suffix. Assume some arbitrary ordering on the outgoing transitions of  $q_i$  and the transition labels to be given by  $\mathbf{a}_{i,1}$  to  $\mathbf{a}_{i,k}$ . Then, each  $e_{i,j}$ , for  $j \in [k]$ , is defined by

$$e_{i,j} = \theta(S_{i,1} \cdots S_{i,j}) E_{i,j} \mathbf{a}_{i,j} S_{q_{i,k}}$$

for the words  $S_{q_{i,j}} E_{i,j} \in \Sigma^*$ . Suppose that  $S_{i,j} = S_{q_{j'}}$  for some  $j' \in [o]$ , let  $q_{i,j}$  be the state reached by the  $j^{\text{th}}$  transition of  $q_i$ . If we have  $q_i \in Q \setminus F$ , i.e., it is not a final state, then we set  $S_{i,k+1} = e_{i,k+1} = \varepsilon$  to be empty. If we have  $q_i \in F$ , i.e., it is a final state, then  $S_{i,k+1} \in \Sigma^+$  is a nonempty word and  $e_{i,k+1}$  is given by

$$e_{i,k+1} = \theta(S_{i,1} \cdots S_{i,k+1}) E_{i,k+1} f_s S_{\text{end}}$$

for the word  $E_{i,k+1} \in \Sigma^*$  and the others as defined above. Notice that the only differences to the other  $e_j$ 's are, first, the fact that, instead of a letter  $\mathbf{a}$ , we write the word  $f_s$  and, second, instead of the suffix  $S_{q_{i,j}}$ , we add the word  $S_{\text{end}}$ . This concludes all structural elements.

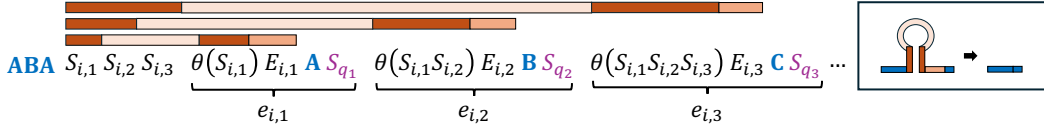
We continue with the definition of the context-set  $C$  over the words defined above. For each  $i \in [o]$ , we assume  $(S_{q_i}, E_{q_i}) \in C$ . These are contexts responsible for jumping between different  $s_i$  and  $s_j$  using hairpin deletion. In addition to  $i \in [o]$ , for each  $j \in [k]$  (or  $j \in [k+1]$  if  $q_i \in F$ ),  $k$  being the number of outgoing transitions of the state  $q_i$  as before, we assume  $(S_{i,1} \cdots S_{i,j}, E_{i,j}) \in C$ . Finally, we assume  $(S_{\text{end}}, E_{\text{end}}) \in C$ , concluding the definition of  $C$ .

Again, we mention that we assume that each occurrence of a left or right context in  $w$  is exactly given by the above construction. More occurrences resulting from overlapping factors are excluded by assumption. Also, for each context  $(\ell, r) \in C$ , we assume that each left context  $\ell$  is chosen long enough so that it forms a valid hairpin with  $\theta(\ell)$  regarding the logarithmic energy model for each occurrence of  $\ell$  with the closest occurrence of the right context  $r$  after it. Keep in mind that each left context has a unique corresponding right context, so no non-deterministic choice can happen here. We continue with an intuitive explanation of the functionality of the construction.

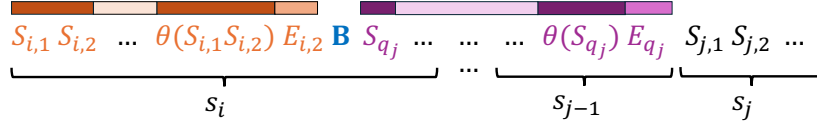
**Intuition.** Transcription starts in  $s_1$  and moves around  $w^\omega$  over and over again (see Figure 2). Every time the transcription is at the beginning of one of the factors  $s_i$ ,  $i \in [o]$ , a nondeterministic choice of one of the encoded transitions occurs. By the selection of one of the factors  $S_{i,1}, S_{i,1}S_{i,2}, \dots, S_{i,1}S_{i,2} \cdots S_{i,k}$  as a left context, we are forced to use the corresponding right context  $E_{i,j}$  (if the prefix  $S_{i,1} \cdots S_{i,j}$  is chosen,  $j \in [k]$ ). Everything in between gets removed by bounded hairpin deletion, using the stem  $S_{i,1} \cdots S_{i,j}$  with corresponding  $\theta(S_{i,1} \cdots S_{i,j})$  and a loop containing everything in between. This represents the choice of an outgoing transition of  $q_i$ . Immediately after the removed hairpin, the letter  $\mathbf{a}_{i,j}$  occurs in  $s_i$ . See Figure 3 for a visualization of this process.

Keep in mind, that the lengths of  $S_{i,j'}$ ,  $j' \in [j]$ , and  $E_{i,j}$  have to be adapted accordingly for this to work. Due to maximally parallel hairpin deletion, the next left context in  $w$ , in particular in  $s_i$ , has to be chosen for hairpin deletion. The immediate next left context is  $S_{\delta(q_i, \mathbf{a}_j)}$  from which we have to choose the unique right context  $E_{\delta(q_i, \mathbf{a})}$  to jump to  $s_{q_{i,k}}$ . This is the aforementioned jump between  $s_i$  and  $s_{q_{i,k}}$ . See Figure 4 for a visualization of this process.

After these two steps (transition selection and jump to the next state), we can repeat this process. A prefix of a word in the language  $L(A)$  is successively obtained by maximally parallel bounded hairpin deletion. To terminate transcription, in a final state, we can use the same mechanism to choose the left context  $S_{i,1}S_{i,2} \cdots S_{i,k}S_{i,k+1}$  and the right context  $E_{i,k+1}$  to obtain a suffix stem  $f_s f_e$  which is followed by  $t \in T$  in  $w^\omega$ , which results in transcription termination. See Figure 5 for a visualization.



■ **Figure 3** Nondeterministic selection of a transition out of the encoded  $q_i \in Q$  in  $s_i$ . This example assumes 3 outgoing transitions and the respective letters A, B, and C. Depending on the choice of the left context (dark color - left), a corresponding right context (medium color - right) can be chosen and a hairpin with the corresponding  $\theta$  part (dark color - right) is formed, having everything in between as part of the loop (bright color - middle). Finally, the whole marked region is then removed by hairpin deletion, leaving the blue marked letter as a new prefix.



■ **Figure 4** Representation of a jump between 2 state encodings  $s_i$  and  $s_j$ . Assuming some transition to be selected and the corresponding part to be removed by hairpin deletion (orange marked region), a deterministic left and right context selection results in everything between the letter B and the state encoding  $s_j$  to be removed (purple marked region).

Hence, as termination can only be initiated from final  $s_i$ 's that represent final states  $q_i \in F$ , all words obtainable from  $w^\omega$  by maximally parallel bounded hairpin deletion must be words in  $L(A)$ . As all transitions are encoded and available to be chosen in the beginning of each  $s_i$ , we can also obtain all words in  $L(A)$ , concluding this sketch. As mentioned before, due to space constraints, a full formal proof of correctness, i.e., that  $L(A) = [w^\omega]_{\text{p}\uparrow\text{-log}}^{\text{HOD}} S'$ , can be found in Appendix A. An engineered example that uses actual words over the RNA alphabet  $\Sigma_{\text{RNA}} = \{A, U, C, G\}$  can be found in the Appendix C.2. ◀

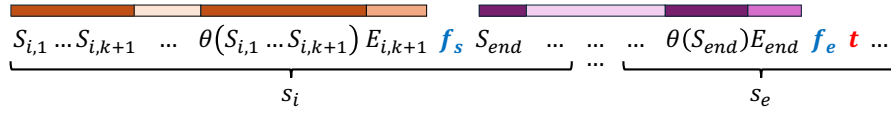
For this construction to work, we clearly needed some assumptions regarding the hairpin deletion model to simulate co-transcriptional splicing. These are especially the assumptions that we are able to do this in a parallel manner (modeling the independence of subsequent co-transcriptional deletions) and that we have some level of greediness in the system (maximally parallel bounded hairpin deletion). Without some greediness assumption regarding the left context, an infinite number of points where deletion could happen can be skipped, and we could start simulating a computation on  $A$  at the beginning of every iteration of  $w^\omega$ . We continue with a general framework that implements the construction from Theorem 5 using the RNA alphabet  $\Sigma_{\text{RNA}} = \{A, U, CG\}$ , sketching the existence of working encodings.

### 3.1 Applying Theorem 5 for the RNA-Alphabet $\Sigma_{\text{RNA}} = \{A, U, C, G\}$

For any working encoding, the following four questions need to be answered:

1. How to encode the internal transition selection in each  $s_i$ ?
2. How to realize the transitions between states, i.e., the jump between arbitrary  $s_i$  and  $s_j$ ?
3. How to realize termination of the parallel hairpin deletion process?
4. Do the words that encode contexts  $(x, y) \in C$  appear only in the intended positions?

In the main part, we discuss these questions in a more general manner. A specific example of encoding a specific NFA using this general framework can be found in Appendix C.



■ **Figure 5** Assuming  $s_i$  encodes a final state with  $k$  outgoing transitions, then a transition  $k + 1$  can be taken to obtain  $f_s$  as a suffix after some previously obtained word (orange marked region). From there, a deterministic choice of left and right contexts  $S_{end}$  and  $E_{end}$  allows for a jump to  $s_e$  (purple marked region), resulting in the suffix  $f_s f_e t$ . As  $t \in T$  and  $f_s f_e$  forms a stem with length  $2m$  by assumption, transcription can be terminated here.

Consider some NFA  $A = (Q, \Sigma_{\text{RNA}}, q_1, \delta, F)$  with  $Q = \{q_1, \dots, q_o\}$ , for some  $o \in \mathbb{N}$ . Let  $q_i \in Q$ , for  $i \in [o]$ , be some state in  $Q$ . Assume  $q_i$  has  $k$  transitions, for some  $k \in \mathbb{N}$ . For each  $j \in [k]$ , let  $\mathbf{a}_{i,j}$  be the letter on the  $j^{\text{th}}$  transition of  $q_i$  and let  $q_{i,j} \in \delta(q_i, \mathbf{a}_{i,j})$  be the resulting state from  $q_i$  by taking the  $j^{\text{th}}$  transition.

(1) First, we define  $\theta$  to represent the strongest base pair bonds, i.e.,  $\theta(\mathbf{A}) = \mathbf{U}$ ,  $\theta(\mathbf{U}) = \mathbf{A}$ ,  $\theta(\mathbf{C}) = \mathbf{G}$ , and  $\theta(\mathbf{G}) = \mathbf{C}$ . To encode  $q_i$  and its outgoing transitions in a word  $s_i \in \Sigma_{\text{RNA}}^*$ , generally,  $s_i$  will have the form as described in the construction given in Theorem 5, i.e., if w.l.o.g.  $q_i \notin F$ ,

$$s_i = S_{i,1} \cdots S_{i,k} e_{i,1} \cdots e_{i,k} \theta(S_{q_{i+1}}) E_{q_{i+1}}.$$

We can set

$$S_{i,1} \cdots S_{i,k} = \mathbf{AA}(\mathbf{C})^i \mathbf{AA}(\mathbf{GAA})^k.$$

Specifically, we set  $S_{i,1} = \mathbf{AA}(\mathbf{C})^i \mathbf{AAGAA}$  and, for each  $j \in [k]$  with  $j > 1$ , we set  $S_{i,k} = \mathbf{GAA}$ . For example, if  $q_i$  has 3 transitions, then

$$S_{i,1} \cdots S_{i,3} = \mathbf{AA}(\mathbf{C})^i \mathbf{AAGAAGAAGAA},$$

and we have  $S_{i,1} = \mathbf{AA}(\mathbf{C})^i \mathbf{AAGAA}$  and  $S_{i,2} = S_{i,3} = \mathbf{GAA}$ . Now, by the definition of  $\theta$ , we obtain, for each  $j \in [k]$ ,

$$\theta(S_{i,1} \cdots S_{i,j}) = (\mathbf{UUC})^j \mathbf{UU}(\mathbf{C})^i \mathbf{UU}.$$

From before, we know that each  $e_{i,j}$ ,  $j \in [k]$ , we have

$$e_{i,j} = \theta(S_{i,1} \cdots S_{i,j}) E_{i,j} \mathbf{a}_{i,j} S_{q_{i,j}}.$$

We obtain  $\theta(S_{i,1} \cdots S_{i,j})$  implicitly by the previous definition. For simplicity reasons, we can set  $E_{i,j} = \theta(S_{i,1} \cdots S_{i,j})$  (hence, equal to the  $\theta(S_{i,1} \cdots S_{i,j})$  part). This concludes already the implementation of the contexts  $(S_{i,1} \cdots S_{i,j}, E_{i,j}) \in C$ . The letter  $\mathbf{a}_{i,j}$  is just the letter of the transition. The final parts of  $s_i$  that need to be defined are  $S_{q_{i,j}}$  and  $E_{q_{i+1}}$ . More generally, we need to define the words for each context  $(S_{q_m}, E_{q_m}) \in C$ , for  $m \in [o]$ . Let  $o_m \in \mathbb{N}$  be a positive number for each state  $q_m \in Q$ . Then, we set  $S_{q_m} = \mathbf{UUU}(\mathbf{G})^{o_m} \mathbf{UUU}$  and  $E_{q_m} = \theta(S_{q_m})$ . We discuss the numbers  $o_m$  later when discussing the second question from above. With that, we have defined all parts of  $s_i$ .

To select any transition, pick a prefix  $S_{i,1} \cdots S_{i,j}$ ,  $j \in [k]$ , find the corresponding right context  $E_{i,j}$ , and then notice that before  $E_{i,j} = \theta(S_{i,1} \cdots S_{i,j})$ , there is another occurrence of  $\theta(S_{i,1} \cdots S_{i,j})$ . We obtain the following hairpin structure (red) with corresponding left and right contexts and the remaining letter  $\mathbf{a}_{i,j}$ :

$$\underbrace{\mathbf{AA}(\mathbf{C})^i \mathbf{AA}(\mathbf{GAA})^j}_{S_{i,1} \cdots S_{i,j}} \cdots \underbrace{(\mathbf{UUC})^j \mathbf{UU}(\mathbf{G})^i \mathbf{UU}}_{\theta(S_{i,1} \cdots S_{i,j})} \underbrace{(\mathbf{UUC})^j \mathbf{UU}(\mathbf{G})^i \mathbf{UU}}_{E_{i,j}} \mathbf{a}_{i,j} S_{q_{i,j}}$$



If, due to the logarithmic energy model, the length of the inner part between  $S_{i,1} \dots S_{i,j}$  and  $\theta(S_{i,1} \dots S_{i,j})$  gets too large, in the position of the last  $G$  in  $S_{i,1} \dots S_{i,j}$  (and in the position of the first  $C$  in  $\theta(S_{i,1} \dots S_{i,j})$ ), we can increase the number of  $G$ 's (and  $C$ 's respectively) until the stem supports the length of the loop. We might have to do this for multiple transitions in a single state encoding  $s_i$  until the values balance out, but due to the exponential increase in supported size per letter in the stem, we can always reach a length that works out. This concludes the question on how to encode the transitions selection using the alphabet  $\Sigma_{\text{RNA}}$ .

(2) Next, we briefly discuss, how to realize the jump between transition-encodings  $s_i$  and  $s_m$ . Notice, that we have already defined the encoding of the related contexts  $(S_{q_m}, E_{q_m}) \in C$ ,  $m \in [o]$ , by  $S_{q_m} = \text{AAA}(\text{C})^{o_m} \text{AAA}$  and  $E_{q_m} = \theta(\text{AAA}(\text{C})^{o_m} \text{AAA}) = \text{UUU}(\text{G})^{o_m} \text{UUU}$ . As before with the number of  $G$ 's and  $C$ 's in the transition encodings, we left the number of  $G$ 's and  $C$ 's in this case open as well. Depending on the number of letters between each left and right context  $(S_{q_m}, E_{q_m}) \in C$ , a different number might be needed to obtain a stem that supports the length of the loop. In the logarithmic energy model, we can always find such numbers as a linear increase in stem size results in an exponential increase of supported loop size. In the example case above, notice that after  $a_{i,j}$ , there is a unique occurrence of  $S_{q_{i,j}} = \text{AAA}(\text{C})^{o_m} \text{AAA}$ , assuming  $q_m = q_{i,j}$ . The unique occurrence of  $E_{q_m}$  is right in front of  $s_m$ . By the construction, we obtain the following unique hairpin formation. Here, we have no overlapping left contexts. So, no nondeterministic choice is possible, This hairpin deletion step must occur, if the letter before is used in an obtained word:

$$a_{i,j} \underbrace{\text{AAA}(\text{C})^{o_m} \text{AAA}}_{S_{q_m}} \dots \underbrace{\text{UUU}(\text{G})^{o_m} \text{UUU}}_{\theta(S_{q_m})} \underbrace{\text{UUU}(\text{G})^{o_m} \text{UUU}}_{E_{q_m}} s_m$$

This concludes the discussion on how to realize the jumps between state encodings  $s_i$  and  $s_m$ .

(3) Finally, we need to encode the part that results in termination of the hairpin deletion process. For that, we define  $s_e$ , i.e., the words  $f_s$ ,  $f_e$ ,  $t$ , and in particular the words in the context  $(S_{\text{end}}, E_{\text{end}}) \in C$ . In principle, we can consider  $s_e$  as its own state encoding but without any transitions. So, similar to the contexts  $(S_{q_m}, S_{q_m}) \in C$ , for  $m \in [o]$ , we can define, for some  $o_e \in \mathbb{N}$  that is not equal to any other  $o_m$ ,  $S_{\text{end}} = \text{AAA}(\text{C})^{o_e} \text{AAA}$  as well as  $E_{\text{end}} = \theta(S_{\text{end}}) = \text{UUU}(\text{G})^{o_e} \text{UUU}$ . Additionally, we set  $t = (\text{U})^{10}$ ,  $f_s = (\text{A})^4 \text{G}(\text{A})^4$ , and  $f_e = (\text{U})^4 \text{C}(\text{U})^4 = \theta(f_s)$ . In the encoding of each final state  $s_i$ , for some  $q_i \in F$ , we handle  $S_{i,k+1}$ , assuming  $q_i$  has  $k$  outgoing transitions, and  $E_{i,k+1}$  analogously to any other transition and add  $f_s$  in the place where the letter of a transition would have been. This results in the following 2 hairpin deletion steps in a final state. Assuming  $w \in \Sigma^*$  is a word in  $L(A)$  which has been obtained already as a prefix and  $q_i$  being the final state reached after reading  $w$ , the following becomes a general possibility:

$$\begin{aligned} w & \underbrace{\text{AA}(\text{C})^i \text{AA}(\text{GAA})^{k+1}}_{S_{i,1} \dots S_{i,k+1}} \dots \underbrace{(\text{UUC})^{k+1} \text{UUGUU}}_{\theta(S_{i,1} \dots S_{i,k+1})} \underbrace{(\text{UUC})^{k+1} \text{UUGUU}}_{E_{i,n+1}} \underbrace{(\text{A})^4 \text{G}(\text{A})^4}_{f_s} \underbrace{\text{AAA}(\text{G})^{o_e} \text{AAA}}_{S_{\text{end}}} \\ & \underbrace{(\text{A})^4 \text{G}(\text{A})^4}_{f_s} \underbrace{\text{AAA}(\text{G})^{o_e} \text{AAA}}_{S_{\text{end}}} \dots \underbrace{\text{UUU}(\text{C})^{o_e} \text{UUU}}_{\theta(S_{\text{end}})} \underbrace{\text{UUU}(\text{C})^{o_e} \text{UUU}}_{E_{\text{end}}} \underbrace{(\text{U})^4 \text{C}(\text{U})^4}_{f_e} \underbrace{\text{UUUUUUUUUU}}_t \end{aligned}$$

(4) We can see that, if only the intended hairpin deletion steps are possible then this encoding effectively works, assuming the words in the contexts are pumped to be big enough. However, it needs to be made sure that hairpin deletion and context recognition can only occur in the intended positions and that no other factor can be used for that purpose. By checking the encodings, one can make sure, that no left or right context appears in unintended positions as a factor. See Appendix C.1 for a detailed examination on why this is the case.



This concludes all elements needed to encode an arbitrary NFA  $A$  in some circular word  $w^\omega$  such that  $L(A) = [w]_{p^{\frac{1}{1-\log}}}$ . As mentioned before, in Appendix C.2, an exemplary encoding of a specific NFA using this framework is given.

This concludes this section. It has been shown that the mechanism of log-hairpin deletion, modeling co-transcriptional splicing, can be used to encode any (infinite) regular language by encoding its DFA or NFA representation on a finite circular DNA word. As the construction grows with the size of the DFA or NFA at hand, minimizing the input NFA is a second problem that can be looked at. Due to the practical nature of this problem, even restricted models of NFAs could be considered.

## 4 Minimizing Input NFA's

The construction from the previous section makes it possible to produce any given finite language from powers of a word  $w$  by parallel hairpin deletion. As our goal is to engineer DNA templates that efficiently encode a given set of sequences to be produced by co-transcriptional splicing, Theorem 5 shows that we can achieve our goal, and we can focus on optimizing the length of  $w$ , i.e., the period. To allow further efficiency gains and some leeway for possible lab implementations of the construction, we can relax the requirement that a given finite set is produced *exactly*, that is, without any other words obtained by the system. With implementation in mind, we require the “extra” sequences produced to not interfere with the words in the target set of sequences, i.e., that they do not share hybridization sites with our target.

We can formalize this as a set of forbidden patterns to be avoided as factors by the “extra” words. In some cases, we may also assume that sequences above given lengths can be efficiently filtered out from the resulting set. We propose the following problem(s).

► **Problem 1** (Small automaton for cover set avoiding forbidden patterns). *Given a set  $W = \{w_1, \dots, w_k\}$  of target words and a set  $F = \{f_1, \dots, f_\ell\}$  of forbidden patterns, such that each  $f_i \in F$  is a factor of some  $w_j \in W$ , find a smallest NFA  $M$ , such that:*

- $W \subseteq L$ , and
- $(L \setminus W) \cap \Sigma^* F \Sigma^* = \emptyset$ ,

where either

(exact)  $L = L(M)$ , in the general case, or

(cover)  $L = L(M) \cap \Sigma^{\leq n}$  for  $n = \max\{|w| \mid w \in W\}$ , in the length bounded setting of the problem.

Problem 1 can be either considered as a minimization problem for the number of states or a minimization problem for the number of transitions (notice that the size of the encoding in Section 3 primarily depends on the number of transitions). The following example shows that the problem is not trivial in the sense that there exist target and forbidden pattern sets  $W, F$  for which the smallest NFA  $M$  satisfying the conditions is smaller than both the minimal NFA for  $W$  and the minimal NFA for  $\overline{\Sigma^* F \Sigma^*} \cup W$ .

► **Example 6.** Let  $W = \{aab^k aa\}$  and  $F = \{ab^k a\}$ . It is easy to see that the minimal NFA for  $W$  must have at least  $k + 5$  states and  $k + 4$  transitions, otherwise it would have a loop which reads  $b^k$  and it would accept an infinite language. Similarly, the minimal NFA for the language containing  $W$  but avoiding all other words having forbidden patterns from  $F$ , that is,  $\overline{\Sigma^* F \Sigma^*} \cup W$ , has at least  $|Q| \in \Omega(k)$  states and  $\Omega(k)$  transitions, otherwise, for accepting inputs of the form  $ab^j aa$ , with  $|Q| \leq j < k$  the block of  $b$ 's would be read by

some cycle labeled  $b^\ell$ . Together with the fact that  $ab^{k-\ell}aa$  must be accepted, we get that the machine would also accept  $ab^k aa$ , a forbidden word. However, a simple automaton  $M$  accepting  $aab^*aa$  with 5 states and 5 transitions satisfies the conditions,  $W \subset L(M)$  and  $L(M)$  does not include any other word with a factor  $ab^k a$ .

As the number of states implicitly gives an upper bound on the possible number of transitions of an NFA, the hardness of Problem 1(cover, states), which is the version of the problem where we are looking for the smallest cover automaton in terms of the number of its states rather than transitions, suggests that all versions of Problem 1 are NP-hard, as similar straightforward reductions are possible with simple target and forbidden sets. Due to space constraints, all proofs in this section can be found in the full version on arXiv.

► **Proposition 7.** *Problem 1(cover, states) is NP-hard.*

#### 4.1 Practically Motivated Restricted NFA-models

As it is practically impossible to obtain infinite RNA sequences from circular DNA transcription, we propose the following NFA-like models that impose certain restrictions on valid computations. Encoding finite languages using those variants can reduce the size of the resulting machine w.r.t. to the minimal classical NFA encoding. Initially we also hoped that those restricted models would allow for more tractable minimization algorithms. In what follows, however, we show the NP-hardness for the minimization problem for all models we considered. The first restriction comes from the fact that the number of times the template word is “read” in circular transcription might be limited. There are various ways of expressing this limitation in terms of the computation of the automata. First, one can restrict the number of times that a certain state of the automaton can be reached in a computation.

► **Definition 8.** *A state-step-bounded nondeterministic finite automaton (SSB-NFA)  $A$  is a hexatuple  $A = (Q, c, \Sigma, \delta, q_0, F)$  where  $Q$  is a finite set of states,  $c \in \mathbb{N}$  denotes the state-step-bound,  $\Sigma$  denotes a finite alphabet,  $\delta : Q \times \Sigma \rightarrow Q$  denotes the transition function,  $q_0 \in Q$  the initial state, and  $F \subseteq Q$  the set of final states.*

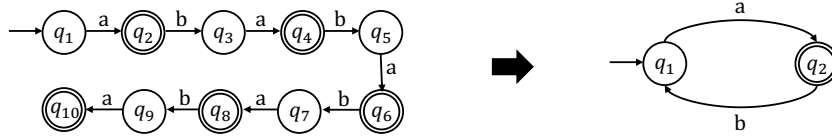
*The language  $L(A)$  of a SSB-NFA is the language of all words  $w \in \Sigma^*$  where  $\delta(q_0, w) \cap F \neq \emptyset$  and  $w$  has some accepting computation where each state  $q_i$  occurs at most  $c$  times.*

Next, in the encoding of NFAs onto templates for hairpin deletion proposed in Theorem 5, different states are encoded consecutively. So, simulating a transition from an earlier encoded state to a later encoded state still occurs on a single repetition of the template. Hence, we could also impose an order  $<_Q$  on the states and restrict the number of times that  $q_i$  follows  $q_j$  in a computation if  $q_i <_Q q_j$ , effectively resulting in having to use another repetition in  $w^\omega$ . This results in the notion of *return-bounded nondeterministic finite automata* (RB-NFAs). Their formal definition is given in Appendix B. Finally, another practical limitation might impose a bound on the length of the intron that is removed during hairpin deletion. This can be represented by imposing an order  $<_Q$  on  $Q$ , setting a distance for each 2 subsequent elements over  $<_Q$ , and setting a bound on the maximum forward distance between two subsequent states in a computation over the automaton. This results in the notion of *distance-bounded nondeterministic finite automata* (DB-NFAs). Their formal definition is given in Appendix B as well. The class of all SSB-NFAs, RB-NFAs, and DB-NFAs is denoted by  $C_{FA}$ . We will investigate SSB-NFAs as an exemplary case and see that we can obtain NP-hardness results for the decision variant of the minimization problem for all models in the class  $C_{FA}$  using very similar proofs.

## 4.2 Properties of State-Step-Bounded NFAs

SSB-NFAs restrict the number of times we are allowed to be in a specific state. Hence, for SSB-NFAs (and RB-NFAs)  $A$ , we know that  $L(A)$  is a finite language. One big advantage of SSB-NFAs is their potential to be significantly smaller than NFAs recognizing the same language. When encoding a finite language as NFA, each word in the language must occur as the label of a simple path in the state graph as any loop in the NFA with a path to a final state results in an infinite language. In SSB-NFAs, certain repetitions may be represented with a small loop. Consider the following example.

► **Example 9.** Let  $L = \{a, aba, ababa, abababa, ababababa\}$ . Let  $A = (Q, \Sigma, \delta, q_0, F)$  be a minimal NFA represented by the left state diagram in Figure 6 such that  $L = L(A)$ . We see that there exists a (minimal) SSB-NFA  $B = (Q', 5, \Sigma, \delta', q'_0, F')$  represented by the state diagram on the right in Figure 6 for which we also have  $L(B) = L = L(A)$ , but its size is significantly smaller than  $A$  regarding the number of transitions and states.



■ **Figure 6** One minimal NFA  $A = (Q, \Sigma, \delta, q_0, F)$  (left) and minimal SSB-NFA  $B = (Q', 5, \Sigma, \delta', q'_0, F')$  (right) recognizing the same language  $L$  found in example 9.

Indeed, we can find for any NFA  $A$  accepting a finite language a SSB-NFA  $B$  which is smaller or equal to the size of  $A$  in terms of the number of states or transitions but which accepts the same language. Trivially, as a NFA recognizing a finite language has no loops on paths that reach final states, we can essentially copy the whole automaton and set the step-bound  $c$  to any value to obtain a SSB-NFA which recognizes the same language. As we are interested in finding minimal templates for contextual splicing, we propose the following decision variant of the SSB-NFA minimisation problem.

► **Problem 2 (SSB-NFA-Min).** Let  $L$  be some regular language and let  $c, k \in \mathbb{N}$  be some positive integers. Does there exist some SSB-NFA  $A = (Q, c, \Sigma, \delta, q_0, F)$  such that  $L(A) = L$  with either  $|Q| \leq k$  (SSB-NFA-Min-States) or  $|\delta| \leq k$  (SSB-NFA-Min-Transitions).

We know that the answer to both problems is **false** if the input language  $L$  is an infinite language. This can be efficiently checked for any representation, i.e., DFAs, NFAs, or regular expressions. Due to the sizes of different representations of regular languages, we obtain the following results.

► **Proposition 10.** SSB-NFA-Min is in NP if the input language  $L$  is given as a finite list of words. If  $L$  is presented as a regular expression or NFA, then the problem is in PSPACE.

Similar to the decidability version of the problem of finding minimal NFAs for finite languages (see [10, 1]), we can show that SSB-NFA-Min is NP-hard by a reduction from the biclique covering problem.

► **Proposition 11.** SSB-NFA-Min (states/transitions) is NP-hard for all alphabets  $\Sigma$  with  $|\Sigma| \geq 2$ .

By that, we can conclude the results of this section in the following theorem.

► **Theorem 12.** *The SSB-NFA-Min problem is NP-complete for all alphabets  $|\Sigma| \geq 2$  if the input language is given as a list of words.*

For the minimization problems of RB-NFAs and DB-NFAs, almost the exact same reduction as for SSB-NFAs can also be applied to obtain NP-hardness in these models. A sketch of the differences is also given in the full version on arXiv. For the following result, assume that RB-NFA-Min and DB-NFA-Min are defined analogously to SSB-NFA-Min.

► **Proposition 13.** *RB-NFA-Min and DB-NFA-Min are NP-hard for all alphabets  $\Sigma$  with  $|\Sigma| \geq 2$ .*

This concludes the results regarding the automata models in  $C_{FA}$ . As even in these restricted cases we still have NP-hardness for minimization, this motivates identifying target languages with restricted structure for which efficient minimization algorithms may exist.

## 5 Conclusion

In this paper, we provided a framework which shows that any regular language can be obtained from circular words using maximally bounded parallel logarithmic hairpin deletion. This indicates that co-transcriptional splicing may be utilized to encode and obtain any set of RNA sequences representable by regular languages, even potentially infinite ones, from a circular DNA template of finite size. As the construction is based on the NFA representation of the encoded regular language, the problem of minimizing NFAs has been further investigated, in particular for well motivated restricted NFA-like models (SSB-NFAs, RB-NFAs, and DB-NFAs). As only hardness results could be obtained so far, future research could work on identifying practically motivated classes of languages for which we can efficiently find small NFAs, SSB-NFAs, RB-NFAs, or DB-NFAs. In addition to that, another future challenge lies in applying hairpin deletion to obtain language classes with higher expressibility, e.g. context-free or context-sensitive languages. But whether this is possible is left as an open question for which we have no specific conjecture so far.

► **Question 14.** *Let  $L_G$  be some context-free language. Does there exist a word  $w \in \Sigma^*$  and a properties tuple for log-hairpin deletion  $S$ , such that  $L(A) = [w^\omega]_{\text{pt-log } S}^{\text{pt-log } S}$ ?*

---

## References

- 1 Jérôme Amilhastre, Philippe Janssen, and Marie-Catherine Vilarem. Fa minimisation heuristics for a class of finite languages. In Oliver Boldt and Helmut Jürgensen, editors, *Automata Implementation*, pages 1–12, Berlin, Heidelberg, 2001. Springer Berlin Heidelberg.
- 2 Ann L Beyer and Yvonne N Osheim. Splice site selection, rate of splicing, and alternative splicing on nascent transcripts. *Genes & development*, 2(6):754–765, 1988.
- 3 Yves d’Aubenton Carafa, Edward Brody, and Claude Thermes. Prediction of rho-independent escherichia coli transcription terminators: a statistical analysis of their rna stem-loop structures. *Journal of molecular biology*, 216(4):835–858, 1990.
- 4 Da-Jung Cho, Szilárd Zsolt Fazekas, Shinnosuke Seki, and Max Wiedenhöft. A formalization of co-transcriptional splicing as an operation on formal languages, 2025. doi:10.48550/arXiv.2504.13354.
- 5 Gloria Del Solar, Rafael Giraldo, María Jesús Ruiz-Echevarría, Manuel Espinosa, and Ramón Díaz-Orejas. Replication and control of circular bacterial plasmids. *Microbiology and molecular biology reviews*, 62(2):434–464, 1998.
- 6 Thomas R Einert and Roland R Netz. Theory for RNA folding, stretching, and melting including loops and salt. *Biophysical journal*, 100(11):2745–2753, 2011.

- 7 Michael R Garey and David S Johnson. *Computers and intractability*, volume 29. W.H. Freeman New York, 2002.
- 8 Cody Geary, Pierre-Étienne Meunier, Nicolas Schabanel, and Shinnosuke Seki. Oritatami: a computational model for molecular co-transcriptional folding. *International Journal of Molecular Sciences*, 20(9):2259, 2019.
- 9 Cody Geary, Paul WK Rothmund, and Ebbe S Andersen. A single-stranded architecture for cotranscriptional folding of RNA nanostructures. *Science*, 345(6198):799–804, 2014.
- 10 Hermann Gruber and Markus Holzer. Computational complexity of NFA minimization for finite and unary languages. In Remco Loos, Szilárd Zsolt Fazekas, and Carlos Martín-Vide, editors, *LATA 2007. Proceedings of the 1st International Conference on Language and Automata Theory and Applications*, volume Report 35/07, pages 261–272. Research Group on Mathematical Linguistics, Universitat Rovira i Virgili, Tarragona, 2007.
- 11 John E. Hopcroft and Jeffrey D. Ullman. *Introduction to Automata Theory, Languages and Computation*. Addison-Wesley, 1979.
- 12 Tao Jiang and Bala Ravikumar. Minimal nfa problems are hard. *SIAM Journal on Computing*, 22(6):1117–1141, 1993. doi:10.1137/0222067.
- 13 Lila Kari, Elena Losseva, Stavros Konstantinidis, Petr Sosik, and Gabriel Thierrin. A formal language analysis of DNA hairpin structures. *Fundamenta Informaticae*, 71:453–475, 2006. URL: <http://content.iospress.com/articles/fundamenta-informaticae/fi71-4-05>.
- 14 Lila Kari and Gabriel Thierrin. Contextual insertions/deletions and computability. *Information and Computation*, 131(1):47–61, 1996. doi:10.1006/INCO.1996.0091.
- 15 David Maier. The complexity of some problems on subsequences and supersequences. *Journal of the ACM (JACM)*, 25(2):322–336, 1978. doi:10.1145/322063.322075.
- 16 Evan C. Merkhofer, Peter Hu, and Tracy L. Johnson. *Introduction to Cotranscriptional RNA Splicing*, pages 83–96. Humana Press, Totowa, NJ, 2014.
- 17 Evan C Merkhofer, Peter Hu, and Tracy L Johnson. *Introduction to cotranscriptional RNA splicing*. Springer, 2014.
- 18 Marina M O’reilly, Mark T McNally, and Karen L Beemon. Two strong 5’ splice sites and competing, suboptimal 3’ splice sites involved in alternative splicing of human immunodeficiency virus type 1 RNA. *Virology*, 213(2):373–385, 1995.
- 19 Xavier Roca, Ravi Sachidanandam, and Adrian R Krainer. Determinants of the inherent strength of human 5’ splice sites. *Rna*, 11(5):683–698, 2005.
- 20 V Stewart, R Landick, and C Yanofsky. Rho-dependent transcription termination in the tryptophanase operon leader region of escherichia coli K-12. *Journal of bacteriology*, 166(1):217–223, 1986.

## A Proof of Correctness for the Construction in Theorem 5

In this section, a formal correctness proof for the construction given regarding Theorem 5 is provided. First, we show that  $L(A) \subseteq [w^\omega]_{\text{p}\uparrow^{-\log}_{\text{S}'}}$  given a constructive argument. Then, we show that  $[w^\omega]_{\text{p}\uparrow^{-\log}_{\text{S}'}} \subseteq L(A)$  by an inductive argument of the hairpin deletion steps that have to be taken in order for parallel hairpin deletion to terminate.

**Formal proof of first direction ( $L(A) \subseteq [w^\omega]_{\text{p}\uparrow^{-\log}_{\text{S}'}}$ ).** Assume  $u \in L(A)$ . Assume  $q_{i_1}q_{i_2}\dots q_{i_{|u|+1}}$  to be a sequence of states to obtain  $u$  in  $L(A)$ , i.e., such that  $q_{i_1} = q_1$ ,  $q_{|u|+1} \in F$ , and for all  $i_j, i_{j+1}$ ,  $j \in [|u|]$ , we have  $q_{i_{j+1}} \in \delta(q_{i_j}, u[j])$ . Assume  $k_j$  to mark the number of the transition taken to obtain  $q_{i_{j+1}}$  from  $q_{i_j}$  with  $u[j]$ . By the construction above, we know that we can factorize  $w^\omega$  into

$$v_1 u[1] v_{2,\ell} v_{2,r} u[2] \dots u[|u| - 1] v_{|u|,\ell} v_{|u|,r} u[|u|] v_{|u|+1,\ell} v_{|u|+1,r} f_s v_{|u|+2} f_e t y w^\omega$$

such that  $t \in T$ ,  $y = \theta(S_{q_1})E_{q_1}$ , and

$$\begin{aligned} v_1 &= S_{1,1} \dots S_{1,\ell_1} \dots \theta(S_{1,1} \dots S_{1,k_1})E_{1,\ell_1}, \\ v_{j,\ell} &= S_{q_{i_{j+1}}} \dots \theta(S_{q_{i_{j+1}}})E_{q_{i_{j+1}}}, \text{ for } j \in [|u| + 1] \setminus \{1\}, \\ v_{j,r} &= S_{j+1,1} \dots S_{j+1,k_{j+1}} \dots \theta(S_{j+1,1} \dots S_{j+1,k_{j+1}})E_{k_{j+1}}, \text{ for } j \in [|u|] \setminus \{1\}, \\ v_{|u|+1,r} &= S_{i_{|u|+1},1} \dots S_{i_{|u|+1},k_{|u|+1}} \dots \theta(S_{i_{|u|+1},1} \dots S_{i_{|u|+1},k_{|u|+1}})E_{|u|+1}, \text{ and} \\ v_{|u|+2} &= S_{end} \dots \theta(S_{end})E_{end}. \end{aligned}$$

Each word  $v$  marks a single factor completely removed by a single hairpin-removal step. As each right context appears uniquely in  $w$ , we assume that the first following occurrence in  $w^\omega$  is taken. As we always pick a left context right after a removed factor and as each  $u[i]$  cannot be part of a left context by assumption in the construction, we know that this factorization is valid for maximal parallel bounded hairpin deletion. Intuitively,  $v_1$  is a factor removed by hairpin deletion that determines the first transition that is taken. For each  $j \in [|u| + 1] \setminus \{1\}$ , the factor  $v_{j,\ell}$  is a factor that can be completely removed by hairpin-deletion that brings us to the beginning of some factor  $s_i$ ,  $i \in [|Q|]$ , from which point on we can now select the next transition. The factor  $v_{j,r}$ , for  $j \in [|u|] \setminus \{1\}$ , similar to  $v_1$ , again is a factor removed by hairpin deletion that determines the next transition that is taken. Finally, we continue with two last removed hairpins that result in transcription termination. As  $q_{i_{|u|+1}}$  is a final state, the factor  $v_{|u|+1,r}$  actually exists and can be removed by hairpin deletion. It is followed by  $f_s$ . Then, in a last step, we can remove  $v_{|u|,r}$  to obtain a suffix  $f_s f_e$  in the transcribed word that is followed by  $tyw^\omega$ , resulting in termination, i.e., resulting in  $u \in [w^\omega]_{\xrightarrow[\text{S}']{\text{p}\uparrow\text{-log}}}$ .

**Formal proof second direction ( $[w^\omega]_{\xrightarrow[\text{S}']{\text{p}\uparrow\text{-log}}} \subseteq L(A)$ ).** Next, we need to show that we cannot obtain any word that is not in  $L(A)$ . Let  $u \in [w^\omega]_{\xrightarrow[\text{S}']{\text{p}\uparrow\text{-log}}}$ . By terminating hairpin deletion, we know that there exists a prefix  $w_p t$  of  $w^\omega$ , for  $t \in T$  and  $w_p \in \Sigma^+$ , such that  $w_p \xrightarrow{\text{S}} us$ , for some suffix  $s \in \Sigma^m$  of length  $m$  that forms a hairpin. For  $w_p$ , we know by the definition of parallel bounded hairpin deletion that there exists some  $n \in \mathbb{N}$  such that

$$w_p = u_1 v_1 \dots u_n v_n u_{n+1}$$

with  $u_i \in \Sigma^*$ , for all  $i \in [n + 1]$ , and  $v_j \in \Sigma^+$ , for all  $j \in [n]$ , for which we have  $v_j \xrightarrow{\text{S}} \varepsilon$ , and such that  $us = u_1 \dots u_{n+1}$ . By the definition of maximal parallel bounded hairpin deletion, we know that for all  $(x, y) \in C$  we have  $x \notin \text{Fact}(u_i)$ , for all  $i \in [n + 1]$ . For all  $j \in [n]$ , we know that  $v_j$  starts with some left context  $x$  for some  $(x, y) \in C$ .

First, we show inductively that we can only obtain letters that can be read during a computation of  $A$ . For that, we show that  $u_1$  must be empty, that  $v_1$  contains only a single deleted hairpin, that  $u_2$  is just a single letter that represents a label of some outgoing transition of  $q_1$  and that  $v_2$  consists of exactly two subsequently removed hairpins (for simplicity reasons, in this proof, we assume that a single  $v_i$  may contain multiple subsequently removed hairpins, also resulting in the empty word. Following the formal model, between each subsequently removed hairpin, a factor  $u$  would have to be added, which is set to the empty word. In this proof, we always assume that such a factorization is possible if we talk about subsequently removed hairpins in a single factor  $v$ .) By induction, and using the same arguments used for the basic step, we obtain that all  $u_i$ , for  $i \in [n - 1] \setminus 1$  must be single letters representing labels from transitions in  $A$ , that all  $v_i$ , with  $i \in [n - 1]$  being odd, contain just a single removed hairpin, and that all  $v_j$ , with  $j \in [n - 1]$  being even, contain exactly two subsequently

removed hairpins (hence, could be split up to a factor containing two distinct hairpins  $v_{j_1}$  and  $v_{j_2}$  that surround the empty word in the above factorization). After that, we proof the terminating condition, resulting in  $u$  being actually in  $L(A)$ .

Suppose  $u_1 \neq \varepsilon$ . Then  $v_1$  does not start at  $w[1]$ . But then, the next possible starting position for  $v_1$  is the next occurring left context  $x$  of some  $(x, y) \in C$  that does not start in  $w[1]$ . By construction, this is the left context  $S_{q_k}$  for some state  $q_k \in Q$  that is located after the first encoded transition. However, if  $v_1$  starts with that or a later occurring left context, then, e.g.,  $S_{1,1} \in \text{Fact}(u_1)$ , which is a contradiction as  $(s_{1,1}, E_{1,1}) \in C$ . So,  $u_1 = \varepsilon$  and  $v_1$  has a prefix  $S_{1,k}$  for some  $k^{\text{th}}$  transition of state  $q_1$ .

By construction, we know that there exists a unique right context  $E_{1,k}$  such that  $(S_{1,k}, E_{1,k}) \in C$ . Suppose  $v_1$  is factorized into  $v_1 = v_{1,1}v_{1,2}$  (or more factors) such that, for both  $i \in [2]$ , we have  $v_{1,i} \neq \varepsilon$ . Then,  $v_{1,1} = S_{1,k} \dots E_{1,k}$  and  $v_{1,2}$  would start immediately after  $E_{1,k}$  in  $w$ . As before,  $v_{1,2}$  would need to start with a left context  $x$  for some  $(x, y) \in C$ . However, by construction of  $w$  we know that after  $E_{1,k}$  follows just the letter of the  $k^{\text{th}}$  transition of  $q_1$ , but no left context. So, this is a contradiction and we have  $v = v_{1,1}$  as well as  $u_2[1] = \mathbf{a}_{1,k}$ , where  $\mathbf{a}_{1,k}$  represents the letter of the  $k^{\text{th}}$  transition of  $q_1$ .

Now, suppose that  $u_2$  has length  $|u_2| > 1$ . Then  $v_2$  cannot start with the left context  $S_{\delta(q_1, \mathbf{a}_{1,k})}$ . But then, there are only 3 possibilities for the next occurring left context in  $w^\omega$ . Either, it is the left context  $S_{\delta(q_1, \mathbf{a}_{1,k+1})}$ , i.e., the left context representing the jump to another state for the  $k+1^{\text{th}}$  transition of  $q_1$ , or, if  $q_1$  is also a final state, then the left context  $S_{\text{end}}$  which is used in the termination process, or, if  $q_1$  is not a final state and  $q_1$  only has  $k$  transitions, the collection of left contexts used to determine the choice of transitions in the state  $q_2$ , e.g.,  $S_{2,1}$ ,  $S_{2,2}$ , and so on. No matter the choice of the next left context, we observe that their occurrences are disjoint from the occurrence of  $S_{\delta(q_1, \mathbf{a}_{1,k})}$ . Hence,  $u_2$  has a prefix  $\mathbf{a}_{1,k} S_{\delta(q_1, \mathbf{a}_{1,k})}$ , which is a contradiction to the definition of maximal parallel bounded hairpin deletion. So, we can only have  $u_1 = \mathbf{a}_{1,k}$  and that  $v_2$  has the prefix  $S_{\delta(q_1, \mathbf{a}_{1,j})}$ .

For readability purposes, until defined otherwise, from now on assume that  $q_i = \delta(q_1, \mathbf{a}_{i,j})$ . Hence,  $S_{\delta(q_1, \mathbf{a}_{1,j})} = S_{q_i}$ . In contrast to  $v_1$ , for  $v_2$ , we have to show that it always contains exactly 2 subsequent hairpin deletion steps. First, as  $v_2$  has the prefix  $S_{q_i}$  for which only the unique right context  $E_{q_i}$  in  $(S_{q_i}, E_{q_i}) \in C$  exists, we know that  $v_2$  has a prefix  $S_{q_i} \dots E_{q_i}$ . By the definition of  $w^\omega$ , we know that immediately after that, we are in the beginning of the factor  $s_i$ . As  $s_i$  is analogously constructed to  $s_1$ , we know by analogous arguments to the ones made for  $u_1$  and  $v_1$ , that we have to start with some context  $(S_{i,k'}, E_{i,k'}) \in C$ ,  $k$  referring to the selected transition, to continue. This results in  $v_2$  containing at least 2 subsequently removed hairpins. We know by the arguments from before that after the occurrence of  $E_{i,k'}$  in  $w^\omega$ , there follows some letter  $\mathbf{a}_{i,k'}$  which represents the letter from the  $k'^{\text{th}}$  transition of  $q_i$ . Also, we know by the same arguments that this occurrence of  $\mathbf{a}_{i,k'}$  is not part of any left context in  $C$ . Hence,  $v_2$  consists of exactly two subsequent hairpin deletion steps.

In addition, we obtain that  $u_3$  has the letter  $\mathbf{a}_{i,k'}$  as a prefix. By analogous arguments from before, we get that  $u_2$ , and in particular every following  $u_i$ , at least for  $i \in [n-1] \setminus 1$  (shown in terminating condition), is a single letter representing the letter of some transition of  $A$ . By induction, we also know that the order of those always represents a valid computation on  $A$ .

So, by now we know that  $u$  must always have a prefix that represents the prefix of some word in  $L(A)$ . What's left to show is that termination can only occur, when  $u$  actually is in  $L(A)$ . This can be done by a combinatorial argument on the construction of  $w^\omega$ .

First, we know that we can only terminate before the position of the factor  $t \in T$  in  $w$ . Also, we know that we must need a hairpin stem of length  $2m$  obtained by hairpin deletion immediately before that occurrence of  $t$ . By the construction of  $s_e$  in  $w$ , we know that  $t$  is



preceded by the factor  $f_e$  of length  $m$ .  $f_e$  is not part of any context. Hence,  $u_{n+1}$  has a suffix  $f_e$ . We must obtain the suffix  $\theta(f_e)f_e$  by hairpin deletion from  $w^\omega$ . By construction, we know  $\theta(f_s) = f_e$ , so  $\theta(f_e) = f_s$ . By construction, we also know that  $f_e$  is preceded by  $E_{end}$  which is, by assumption, not a factor of  $f_s$ . Hence,  $E_{end}$  needs to be removed by hairpin deletion, resulting in  $u_{n+1} = f_e$ . This is only possible with the corresponding context  $(S_{end}, E_{end}) \in C$  which can only be found in the encoding words  $s_i$  of final states  $q_i \in F$ . By construction, these occurrences of  $S_{end}$  are preceded by  $f_s$ . Also by construction,  $f_s$  is preceded by some right context  $E_{i,k+1}$ , assuming the encoding  $s_i$  of some final state  $q_i$  with  $k$  transitions. Using the inductive arguments from before, we know that we can only obtain  $f_s$  as part of some factor  $u_n$  if and only if we select the context  $(S_{i,1} \dots S_{i,k+1}, E_{i,k+1})$  as the last removed hairpin in  $v_{n-1}$ . Hence,  $u_n = f_s$ .

Also by the inductive arguments from before, we know that  $v_{n-1}$  consists of exactly two subsequently removed hairpins, where the first one is enclosed by  $(S_{q_i}, E_{q_i})$ . For all other previous  $u_j$ , for  $j \in [n-1]$ , we know that they are all single letters representing a prefix of a valid computation on  $A$ . As we can obtain  $f_s$  only in final states, we know that  $u_1 \dots u_{n-1} \in L(A)$ . Also, we know that the suffix of length  $2m$  is actually  $s = f_s f_e = u_n u_{n+1}$ . So,  $u = u_1 \dots u_{n-1}$  and by that  $u \in L(A)$ . This concludes this direction and the proof of this construction.

## B Formal Definitions of RB-NFAs and DB-NFAs

► **Definition 15.** A return-bounded nondeterministic finite automaton (RB-NFA)  $A$  is a hexatuple  $A = (Q, c, \Sigma, \delta, q_1, F)$  where  $\Sigma, \delta, q_0$ , and  $F$  are defined as for usual NFAs,  $c \in \mathbb{N}$  denotes a return-bound and  $Q = \{q_1 < \dots < q_{|Q|}\}$  is an ordered finite set of states. A RB-NFA accepts a word  $w \in \Sigma^*$  if  $\delta(q_1, w) \in F$  and there exists an accepting path  $q_1 = p_1 \rightarrow \dots \rightarrow p_\ell = \delta(q_1, w)$  with no more than  $c$  pairs  $p_i > p_{i+1}$  for  $i \in [\ell-1]$ .

► **Definition 16.** A distance-bounded nondeterministic finite automaton (DB-NFA)  $A$  is a heptatuple  $A = (Q, f_d, c, \Sigma, \delta, q_1, F)$  where  $Q$  is an ordered set of states,  $\Sigma, \delta, q_0$ , and  $F$  are defined as for usual NFAs,  $c \in \mathbb{N}$  denotes a distance-bound, and  $f_d : Q \times Q \rightarrow \mathbb{N}$  is a distance function such that  $f_d(q_i, q_{i+1}) \in \mathbb{N}$  for  $i \in [|Q| - 1]$ ,  $f_d(q_{|Q|}, q_1) \in \mathbb{N}$ ,  $f_d(q_i, q_j) = \sum_{i' \in [i..j-1]} f_d(q_{i'}, q_{i'+1})$  if  $i < j - 1$ , and  $f_d(q_i, q_j) = f_d(q_i, q_{|Q|}) + f_d(q_{|Q|}, q_1) + f_d(q_1, q_j)$  if  $j < i$ . A DB-NFA accepts a word  $w \in \Sigma^*$  if  $\delta(q_0, w) \in F$  and there exists an accepting path  $q_0 = q_{i_1} \rightarrow \dots \rightarrow q_\ell = \delta(q_1, w)$  satisfying the condition that  $f_w(q_{i_j}, q_{i_{j+1}}) \leq c$  for  $j \in [\ell-1]$ .

## C Additional Content Regarding the Example for Theorem 5 using the RNA Alphabet $\Sigma_{RNA} = \{A, U, C, G\}$

### C.1 Detailed Examination of Factors in the General Construction

This detailed examination is related to point (4) in Section 3.1.

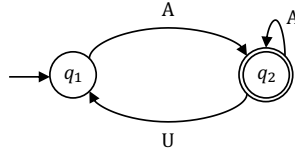
For all contexts  $(S_{i,1} \dots S_{i,j}, E_{i,j}) \in C$ , we see that they are bordered by either AA or UU, followed and preceded by either C or G. For all  $S_{i,1} \dots S_{i,j}$ , we know that after the first AA, there occur  $i$  many C's. We notice that there is only one position in the word where  $AA(C)^i$  occurs and that is in the beginning of  $s_i$ . In addition to that, now regarding  $E_{i,j}$ , we notice that there are two positions where the corresponding  $(G)^i UU$  occurs, and these are in either in the end of  $\theta(S_{i,1} \dots S_{i,j})$  or the end of  $E_{i,k}$ . As we need an occurrence of a right context that is preceded by a valid hairpin, we see that the first occurrence has to be used in the hairpin-formation between  $S_{i,1} \dots S_{i,j}$  and  $\theta(S_{i,1} \dots S_{i,j})$ , and we see that the second



occurrence can then be used as a right context. So, no other unintended placements are possible. The same holds for each context  $(S_{q_i}, E_{q_i}) \in C$  which are bordered by AAA (resp. UUU), encapsulating a unique number of  $o_i$  many C's or G's. As before, the word used in the right context also occurs two times, once for the hairpin formation and once for the right context recognition. The first occurrence has to be used to form a valid hairpin and the second one, as the margin is set to 0, has to be a consecutive factor, i.e., the intended occurrence of  $E_{q_i}$ . The same can be said analogously for  $(S_{end}, E_{end}) \in C$ . Hence, this construction generally results in no unintended factors, as long as the numbers of C's and G's between the respective borders are pumped up enough. One has to make sure that the letters from the transitions  $a_{i,k}$  are not part of some context. But this is prevented as these are always preceded by UU in the end of  $E_{i,k}$  and followed by AA in  $S_{q_{i,k}}$ . So, they cannot be part of any context. The words  $f_s$  and  $f_e$  might also interfere with some context, but as both are bordered by  $(A)^4$  (resp.  $(U)^4$ ), this cannot happen. The termination word  $t = (U)^{10}$  can also not be part of any context, as is preceded by  $(U)^4$  in  $f_e$  and followed by UUU in  $\theta(S_{q_1})$  (see construction in proof of Theorem 5 to verify this).

## C.2 Specific Example Construction

This section contains a specific example for the construction of a circular word  $w^\omega$  for some NFA  $A$  for which we have  $L(A) = [w^\omega]_{\text{p}\uparrow\text{-log}}^{\text{uuu}}$ , using the framework given in Section 3.1. Let  $A = (Q, \Sigma_{\text{RNA}}, q_1, \delta, F)$  be some NFA that is defined by the following graph representation: We use the construction in Theorem 5 and encoding given above to obtain a word  $w$  for



■ **Figure 7** NFA  $A$  with  $Q = \{q_1, q_2\}$ ,  $F = \{q_2\}$ , and  $\delta = \{((q_1, A), q_2), ((q_2, U), q_1), ((q_2, A), q_2)\}$ . So it accepts the regular language  $A^*(UA^*)^*$ .

which we have  $[w^\omega]_{\text{p}\uparrow\text{-log}}^{\text{uuu}} = L(A)$  for the properties tuple  $S = (\Sigma_{\text{RNA}}, \theta, 1, C, 0, \{t\}, 9)$ . Notice that the margin is of length 0, that the set of terminating sequences only contains  $t = (U)^{10}$ , that the log factor is 1, that the alphabet is  $\Sigma_{\text{RNA}}$ , and that  $\theta$  is given as above.

First, we set the state encodings  $s_1$  and  $s_2$ . Notice that  $q_1$  only has 1 outgoing transition while  $q_2$  has two outgoing transitions and is a final state. For  $s_1$ , we obtain

$$s_1 = \underbrace{\text{AACAAGAA}}_{S_{1,1}} \underbrace{\text{UUCUUGUU}}_{\theta(S_{1,1})} \underbrace{\text{UUCUUGUU}}_{E_{1,1}} \underbrace{\text{A}}_{a_{1,1}} \underbrace{\text{AAA(G)}^6 \text{AAA}}_{S_{q_2}} \underbrace{\text{UUU(G)}^6 \text{UUU}}_{\theta(S_{q_2})} \underbrace{\text{UUU(G)}^6 \text{UUU}}_{E_{q_2}}.$$

$e_{1,1}$

Notice that we set  $o_2 = 6$  in  $S_{q_2}$  and  $E_{q_2}$  due to the total size of the word. For  $s_2$ , due to space constraints, we first give the abstract structure and then the assignment to each word. Notice that  $s_2$  is the last state encoding before appending  $s_e$  to  $w$ . Hence, no  $\theta(S_{q_3})E_{q_3}$  has to be added in the end. We have

$$s_2 = S_{2,1}S_{2,2}S_{2,3} \underbrace{\theta(S_{2,1})E_{2,1}\text{US}_{q_1}}_{e_1} \underbrace{\theta(S_{2,1}S_{2,2})E_{2,2}\text{AS}_{q_2}}_{e_2} \underbrace{\theta(S_{2,1}S_{2,2}S_{2,3})E_{2,3}\text{fS}_{end}}_{e_3}$$

## 5:22 Realizing Regular Languages via Hairpin Deletion

for the following assignments:

$$\begin{aligned}
S_{2,1}S_{2,2}S_{2,3} &= \underbrace{\text{AACCAAGAA}}_{S_{1,1}} \underbrace{\text{GAA}}_{S_{2,2}} \underbrace{\text{GAA}}_{S_{2,3}} \\
\theta(S_{2,1})E_{2,1} \text{ U } S_{q_1} &= \underbrace{\text{UUCUUGUU}}_{\theta(S_{2,1})} \underbrace{\text{UUCUUGUU}}_{E_{2,1}} \text{ U } \underbrace{\text{AAA(C)}^5\text{AAA}}_{S_{q_1}} \\
\theta(S_{2,1}S_{2,2})E_{2,2} \text{ A } S_{q_2} &= \underbrace{\text{UUCUUCUUGUU}}_{\theta(S_{2,1}\theta(S_{2,2}))} \underbrace{\text{UUCUUCUUGUU}}_{E_{2,2}} \text{ A } \underbrace{\text{AAA(C)}^6\text{AAA}}_{S_{q_2}} \\
\theta(S_{2,1}S_{2,2}S_{2,3})E_{2,3}f_sS_{end} &= \underbrace{\text{UUCUUCUUCUUGUU}}_{\theta(S_{2,1}S_{2,2}S_{2,3})} \underbrace{\text{UUCUUCUUCUUGUU}}_{E_{2,3}} \underbrace{(\text{A})^4\text{G}(\text{A})^4}_{f_s} \underbrace{\text{AAA(C)}^7\text{AAA}}_{S_{end}}
\end{aligned}$$

Notice that we set, in addition to  $o_2 = 6$ , the numbers  $o_1 = 5$  and  $o_e = 7$  for the words  $S_{q_1}, E_{q_1}, S_{end}$  (thus also for  $E_{end}$  in  $s_e$ ). The numbers are selected such that hairpin deletion regarding the logarithmic model works. Finally,  $s_e$  is encoded in the following manner:

$$s_e = \underbrace{\text{UUU(G)}^7\text{UUU}}_{\theta(S_{end})} \underbrace{\text{UUU(G)}^7\text{UUU}}_{E_{end}} \underbrace{(\text{U})^4\text{C}(\text{U})^4}_{f_e} \underbrace{\text{UUUUUUUUU}}_t \underbrace{\text{UUU(G)}^5\text{UUU}}_{\theta(S_{q_1})} \underbrace{\text{UUU(G)}^5\text{UUU}}_{E_{q_1}}$$

Remember, that  $\theta(S_{q_1})E_{q_1}$  is needed to jump to  $s_1$ . It is encoded in the end of  $s_e$  such that each word starts with the left context  $S_{1,1}$  in  $s_1$ . In total, we set

$$w = s_1 s_2 s_e.$$

We have to check that each intended hairpin can actually be formed, regarding the length bound obtained by the logarithmic energy model. But, for that we notice that each left context for the transition selection has at least length  $|S_{1,1}| = 8$ , which supports a loop of length  $2^8 = 256$ , which clearly works out inside of each  $s_1$  and  $s_2$ . Regarding jumps between  $s_1, s_2$  and  $s_e$ , notice that the shortest context responsible for such a jump,  $S_{q_1}$ , has length  $|S_{q_1}| = 11$ , which supports a loop of length  $2^{11} = 2048$ . That is longer than  $w$  itself, hence such a log-hairpin can also always be formed.

We obtain  $[w]_{\text{p}\uparrow\text{-log}} = L(A)$  by the proof of Theorem–5 and the fact, that no factor occurs in an unintended position.

# Secondary Structure Design for Cotranscriptional 3D RNA Origami Wireframes

Pekka Orponen ✉ 

Department of Computer Science, Aalto University, Finland

Shinnosuke Seki ✉ 

Department of Computer and Network Engineering, University of Electro-Communications,  
Tokyo, Japan

Antti Elonen ✉ 

Department of Computer Science, Aalto University, Finland

---

## Abstract

We address the task of secondary structure design for *de novo* 3D RNA origami wireframe structures in a way that takes into account the specifics of a cotranscriptional folding setting. We consider two issues: firstly, avoiding the topological obstacle of “polymerase trapping”, where some helical domain cannot be hybridised due to a closed kissing-loop pair blocking the winding of the strand relative to the polymerase–DNA–template complex; and secondly, minimising the number of distinct kissing-loop designs needed, by reusing KL pairs that have already been hybridised in the folding process. For the first task, we present an efficient strand-routing method that guarantees the absence of polymerase traps for *any* 3D wireframe model, and for the second task, we provide a graph-theoretic formulation of the minimisation problem, show that it is NP-complete in the general case, and outline a branch-and-bound type enumerative approach to solving it. Key concepts in both cases are depth-first search in graphs and the ensuing DFS spanning trees. Both algorithms have been implemented in the *DNAforge* design tool (<https://dnaforge.org>) and we present some examples of the results.

**2012 ACM Subject Classification** Theory of computation → Theory and algorithms for application domains

**Keywords and phrases** RNA origami, wireframe nanostructures, cotranscriptional folding, secondary structure, kissing loops, algorithms, self-assembly

**Digital Object Identifier** 10.4230/LIPIcs.DNA.31.6

**Funding** This work was initiated during a visit by P.O. to the University of Electro-Communications, supported by grant “Design Tools for DNA Nanotechnology” from the Finnish Cultural Foundation, and significantly advanced during a visit by S.S. to Aalto University, supported by a Visiting Researcher grant from the Aalto Science Institute.

**Acknowledgements** We thank Mr. Robin Runne for essential contributions to the implementation of our DFS tree enumeration method.

## 1 Introduction

Concurrently to the advances in DNA nanotechnology, there has been increasing interest in using RNA as the fabrication material for self-assembling bionanostructures. In comparison to DNA, the appeal of RNA is that the strands can be produced by the natural process of polymerase transcription, and the structures can thus be created in room temperature *in vitro*, and possibly eventually *in vivo*, from genetically engineered DNA templates. The challenge, on the other hand, is that the folding process of RNA is kinetically more complex and hence less predictable than DNA helix formation, at least at the present stage of RNA engineering.

The first design technique applied in this area of *RNA nanotechnology* was modular “RNA tectonics”, in which naturally occurring RNA structures are connected together to create bigger target complexes using specific connector motifs such as sticky-end pairings and a



© Pekka Orponen, Shinnosuke Seki, and Antti Elonen;  
licensed under Creative Commons License CC-BY 4.0

31st International Conference on DNA Computing and Molecular Programming (DNA 31).

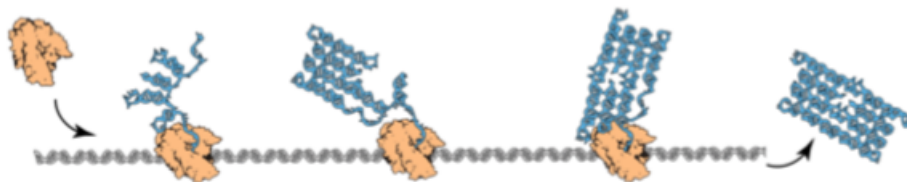
Editors: Josie Schaeffer and Fei Zhang; Article No. 6; pp. 6:1–6:18

Leibniz International Proceedings in Informatics



LIPICs Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

variety of pseudoknots [11, 12]. A complementary top-down method of “RNA origami”, in which a task-specific strand is rationally designed to fold into the desired target structure, was then introduced in a pioneering work by Geary et al. in 2014 [8]. Geary et al. demonstrated the feasibility of their method by synthesising 2D “RNA tiles” of different sizes, and this approach has since then been further developed with new design motifs, techniques, and tools [14, 6]. (For an overview, see [17].)



**Figure 1** Cotranscriptional folding of a 2D RNA origami tile structure from a DNA template, mediated by an RNA polymerase enzyme. (Reprinted with permission from [7].)

One emphasis in the work of Geary et al. [8, 6] has been the *cotranscriptional* nature of the polymerase transcription process, that is, the way the transcribed RNA strand starts to fold into secondary structures already while being spooled off the polymerase enzyme (Figure 1). This characteristic of natural RNA generation introduces new challenges and also opportunities for the rational design process, some of which we shall explore in the present work.

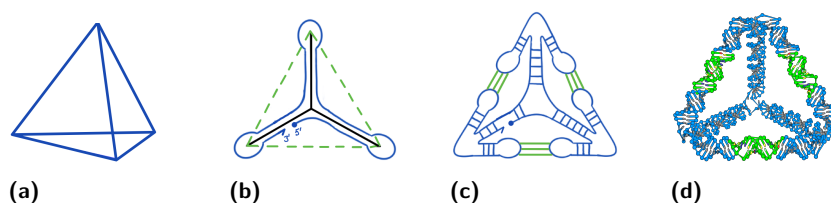
In the following, Section 2 presents a spanning-tree based framework for self-assembly of wireframe structures by co-transcriptional folding, and introduces the topological folding obstacle of polymerase trapping. Section 3 then demonstrates how this obstacle can always be avoided by using a depth-first-search (DFS) based design scheme. Next, Section 4 introduces the notion of the KLX number of a DFS tree, which corresponds to the maximum number of kissing loops that are simultaneously “open” in the folding process, and hence need different designs in order to avoid nonspecific pairings. Minimising this number provides the possibility of efficiently reusing KL designs, although, as proved in Section 5, the KLX minimisation problem is in the general case NP-hard. Since an efficient minimisation algorithm is thus unlikely, Section 6 provides a branch-and-bound type enumeration approach to the problem. Section 7 provides some examples of using the *DNAforge* tool to compute the DFS tree based designs and KLX minimisation. Section 8 summarises the results and suggests some directions for further work.

## 2 Wireframe RNA origami and the polymerase trapping obstacle

An extension of the RNA origami method to the design of 3D wireframe structures was presented by Elonen et al. in [3]. We conduct our discussion in this framework, but the basic ideas apply, *mutatis mutandis*, also to the task of designing 2D RNA origami tiles (cf. [15]). The general spanning-tree based 3D wireframe design scheme is outlined in Figure 2.

In this scheme, one starts from the targeted wireframe, which in the case of Figure 2(a) is a simple tetrahedron. (Or more precisely the wireframe skeleton of a tetrahedral mesh.) In the first design step (Figure 2(b)) one chooses some spanning tree  $T$  of the wireframe graph  $G$ ,<sup>1</sup> and designs the primary structure of the RNA strand so that it folds to create a twice-around-the-tree walk on  $T$ , covering each edge of  $T$  twice in antiparallel directions. In the second design step (Figure 2(c)) one then extends the walk halfway along each of the

<sup>1</sup> A spanning tree of a graph  $G$  is an acyclic subgraph that connects all the vertices of  $G$ .

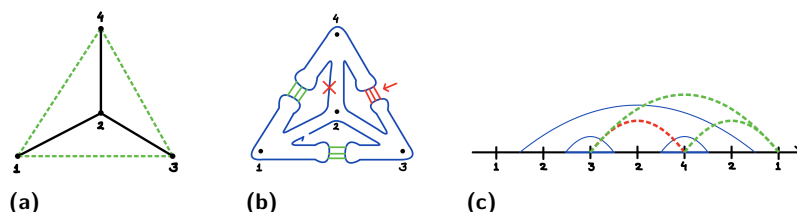


■ **Figure 2** A spanning-tree based design scheme for 3D RNA wireframe origami. (a) Targeted wireframe model. (b) A spanning tree and strand routing of the wireframe graph. (c) Routing-based stem and kissing-loop pairings. (d) Nucleotide-level model. (Adapted with permission from [3].)

co-tree (= non-spanning tree) edges of  $G$  into a hairpin loop, and designs the base sequences at the termini of the hairpins so that pairwise matching half-edges are connected by the  $180^\circ$  kissing-loop design motif introduced in [8], thus constituting the co-tree edges. Figure 2(d) presents a nucleotide-level model of the eventual nanostructure.

One potentially significant topological obstacle to cotranscriptional folding in this framework is the phenomenon of *polymerase trapping*, first identified by Geary and Andersen in [9] and also addressed in the recent 2D RNA origami design tool ROAD by Geary et al. [6, pp. 551–552, SI p. 102 ff]. Our mathematical model of this phenomenon is basically the same as introduced and analysed by Mohammed et al. in [15], but adapted to the present setting of 3D wireframe origami designs.

To explain this concern, let us review the previous tetrahedron design, presented in more detail in Figure 3. Figure 3(a) shows the tetrahedral wireframe as a Schlegel diagram, that is, as a planar projection from a point above one of the tetrahedron's faces. The edges of the chosen spanning tree, which in this case is a 3-pointed star, are indicated by solid black lines, and the co-tree edges by dashed green lines.



■ **Figure 3** A tetrahedron design based on a 3-star spanning tree. (a) Spanning tree and co-tree of the tetrahedral graph. (b) Strand routing and kissing-loop pairs for the design. (c) Domain-level arc diagram of the design.

Figure 3(b) depicts again the corresponding twice-around-the-tree strand routing (blue) and the complementary kissing-loop pairings (green and red). The helix junctions in the design, which constitute the vertices of the eventual 3D nanostructure, are now indexed according to their first-visit order in the strand routing.

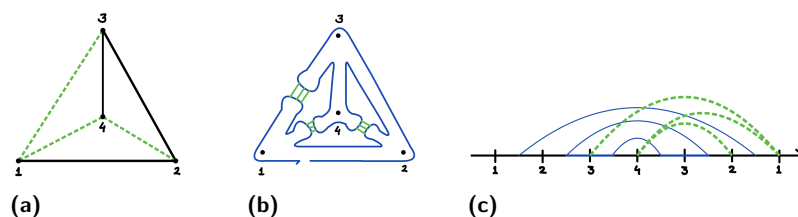
The schematic in Figure 3(c) presents the design as a domain-level arc diagram, where the strand is laid out along a line in the  $5'$  to  $3'$  direction, the vertex visits are marked by the corresponding indices, the domain-to-domain helical pairings are indicated by solid blue arcs, and the kissing-loop pairings by green and red dashed arcs. (For simplicity and clarity, the pairings of the half-edge stem domains flanking each kissing-loop hairpin are not shown.)

Consider now how a cotranscriptional folding process of this structure might proceed. Instead of thinking of the RNA strand being spooled out of the polymerase starting at the  $5'$  end and folding as the appropriate base pairings become available, it may be easier to visualise the large polymerase–DNA–template complex as traversing the  $5'$ – $3'$  strand route outlined in Figure 3(c) and transcribing the nucleotide domains as it goes.

First the domains 1-2 and 2-3 are transcribed, and the RNA strand stays linear until the transcription of domain 3-2 begins. (For simplicity, we are ignoring any transient nonspecific nucleotide pairings that arise during the folding process.) Between the completion of domain 2-3 and the initiation of domain 3-2, the two opening hairpins for the kissing loops 3-4 and 3-1 are transcribed. (The best relative ordering of these two transcriptions is a geometric and sequence-design issue, and we leave this choice open in this high-level view.) After (or while) the complementary domains 2-3 and 3-2 hybridise, domain 2-4 is transcribed, and after that the closing hairpin of the 3-4 kissing loop and the opening hairpin of the 4-1 kissing loop, in some order.

Consider now what happens when the polymerase reaches domain 4-2 (marked with a red cross in diagram 3(b)), where it should create a double-stranded helix with domain 2-4, by winding the strand around it in antiparallel direction. If the 3-4 kissing loop (drawn in red and marked with a red arrow in 3(b)) has already closed, the strand with the big polymerase–DNA complex coupled to it cannot achieve this, since the kissing-loop pairing is blocking the pathway. (This is of course also a time-scale issue, and depends among other things on the strand distance between the closing hairpin of the kissing loop and the closing domain of the helical pairing; but let us again ignore these lower-level details at this presentation.<sup>2</sup>)

Schematically, one can see that the risk of this kind of “polymerase trapping” obstacle emerges when a kissing-loop pair, initiated (opened) before a given helical pairing, closes between the opening and closing of the helical pairing; or in terms of our arc diagram when a “dashed arc” that has been initiated before a “solid arc” terminates inside that solid arc.



■ **Figure 4** A tetrahedron design based on a 4-vertex path spanning tree. (a) Spanning tree and co-tree of the tetrahedral graph. (b) Strand routing and kissing-loop pairs for the design. (c) Domain-level arc diagram of the design.

As another example, let us consider the tetrahedron design presented in Figure 4. As shown in Figure 4(a), in this case the spanning tree is a simple 4-vertex path. Figure 4(b) again outlines the corresponding strand route and kissing-loop pair arrangement, with the helix junctions numbered according to their first-visit order. As witnessed by Figure 4(c), this time there is no risk for the polymerase trapping obstacle. That is, every kissing loop closes only after the completion of all the helical pairings that have been initiated after the kissing loop was opened.

Such complete absence of polymerase traps seems like a very particular property, and one may wonder for which kinds of wireframe models this situation can be achieved. As we shall see in the next section, however, such an arrangement of the helical and kissing loop pairings can in fact be found for *any* connected wireframe graph, by an application of the fundamental algorithmic method of depth-first search [1, Sec. 20.3].

<sup>2</sup> Furthermore, moving from our strand-centric to a polymerase-centric view, the polymerase of course does not stop transcribing even if the folding is temporary blocked. Eventually the polymerase detaches from the fully transcribed strand and leaves it to fold the unfolded 3' tail the best it can. From this perspective, the polymerase trapping phenomenon is more of a kinetic than a topological obstacle.

### 3 Cotranscription-friendly secondary structure design

■ **Algorithm 1** Depth-first search of a graph  $G = (V, E)$  from root vertex  $r \in V$ .

---

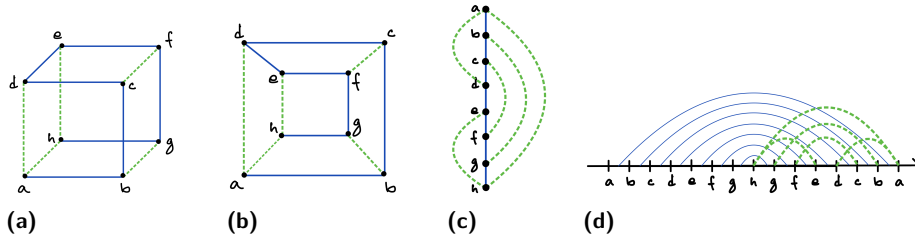
```

1: Initially all vertices  $v \in V$  and edges  $e \in E$  are set to be unmarked.
2:
3: function DFS( $G, r$ )
4:   mark vertex  $r$  as visited
5:   for each edge  $e = \{r, v\}$  incident to  $r$  do
6:     if vertex  $v$  is not marked as visited then
7:       mark  $e$  as a tree edge
8:       perform search DFS( $G, v$ )
9:     else
10:      mark  $e$  as a back edge, unless it is already marked (= edge to parent)
11:    end if
12:  end for
13: end function

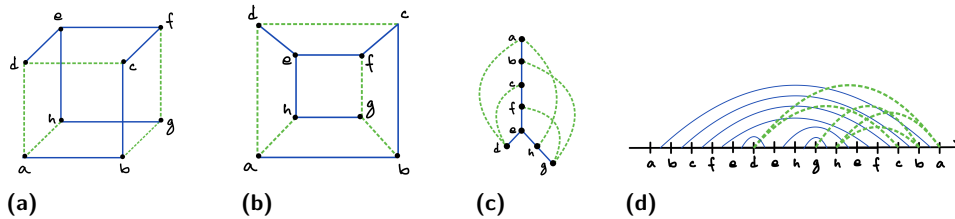
```

---

To streamline the presentation, we assume henceforth that any graph under consideration is connected and undirected, unless stated otherwise. The *depth-first search (DFS)* method for systematically traversing and labelling a (connected, undirected) graph is presented as Algorithm 1.



■ **Figure 5** A cube design based on a path-like DFS tree. (a) DFS tree and co-tree of the cube mesh. (b) Corresponding Schlegel diagram. (c) DFS tree with back edges. (d) Arc diagram of the resulting design.



■ **Figure 6** A cube design based on a branching DFS tree. (a) DFS tree and co-tree of the cube mesh. (b) Corresponding Schlegel diagram. (c) DFS tree with back edges. (d) Arc diagram of the resulting design.

Consider a graph  $G = (V, E)$  with  $|V| = n$  vertices and  $|E| = m$  edges. Then a DFS traversal of  $G$ , starting from any chosen *root* vertex  $r \in V$ , partitions the set of edges  $E$  in time  $O(m)$  in two disjoint classes:  $n - 1$  *tree edges* and  $m - n + 1$  *back edges*. The tree edges constitute a spanning tree  $T$  of  $G$ , which can be considered to be rooted at  $r$  and oriented



accordingly, whereas the back edges (which constitute the corresponding co-tree  $E \setminus T$ ) have the important property that they can always be oriented to point “upward” towards the root of the tree [13]; in other words, there are no “cross edges” connecting two different branches of the directed tree, as exemplified in Figures 5 and 6 (c), and also in Figure 7.<sup>3</sup>

To make this precise and to introduce another important notion, consider such a *DFS (spanning) tree* for a graph  $G = (V, E)$  to be a four-tuple  $T = (V, S, r, \delta)$ , where  $S \subseteq E$  is the set of tree edges,  $r \in V$  is the chosen root which determines the orientation of the tree, and  $\delta : V \rightarrow [1..n]$  is a *pre-ordering* that labels the vertices in order of their first visits in the traversal. As discussed above, any edge of  $G$  is either part of the stem  $S$  or connects a vertex and its ancestor along the unique path in  $T$  from the vertex to the root. We often consider  $T$  as embedded in the plane so that the children of each vertex are ordered from left to right in increasing order of their  $\delta$  values. Thus, for example, vertex  $d$  in Figure 6(c) is presumed to have been visited earlier than vertex  $h$  in the traversal that created the tree.

A plane embedded DFS tree  $T = (V, S, r, \delta)$  provides an initial blueprint for designing a cotranscriptionally folding RNA wireframe nanostructure. In the first stage of the design process, the contour of the tree  $T$  is traced by an RNA strand  $w$  so that each edge of  $T$  becomes assembled as an RNA helix, hybridised from two complementary antiparallel domains of  $w$ . In the second stage the co-tree edges of  $G$ , which are now back edges of the DFS tree  $T$ , are constituted as kissing loops made of two half-edge hairpins that extend to meet from the end-vertices of the respective edges.

Let us call the sequence of vertex labels encountered during a walk around the contour of any labelled, plane embedded tree  $T$  a *contour trace* or *linear arrangement* of  $T$ .<sup>4</sup> In the special case that  $T$  is a DFS tree we refer to this sequence as a *DFA arrangement* based on  $T$ . By construction, any contour trace of a tree  $T$  accommodates every edge  $e$  of  $T$  exactly twice: firstly, when the traversal crosses  $e$  in a forward direction, and secondly when the traversal crosses  $e$  in the opposite direction. Since any DFS tree (or more generally any spanning tree) of a graph  $G$  with  $n$  vertices has  $n - 1$  edges, any DFS arrangement in  $G$  contains  $2n - 1$  vertex labels. For example, every DFS tree of a tetrahedron is a simple 4-path in its Schlegel diagram, and the corresponding DFS arrangement has the pattern 1, 2, 3, 4, 3, 2, 1 (see Figures 4(a) and (c)).

To complete our blueprint for designing a cotranscriptionally folding RNA nanostructure, the DFS arrangement needs to be augmented into a (domain-level) *arc diagram* by adding information that indicates when, for each pair of hybridising domains  $(\alpha, \alpha^*)$ , the forward domain  $\alpha$  is transcribed and when its complementary reverse domain  $\alpha^*$ . For each such pair, these time points are connected by an edge (“arc”), with the first time point considered as the “opening” and the second the “closing” time for this pair. Since in our design scheme, DFS tree edges correspond to helical domains, we correspondingly say that a tree edge is *opened* when it is crossed by the contour traversal in the forward direction, and *closed* when it is crossed in the reverse direction. Note that the chosen DFS completely determines the scheduling of the opening and closing pattern of the helical domains.

Let us then consider scheduling the opening and closing times of the back edges, i.e. kissing loops in the eventual RNA nanostructure. Multiple occurrences of internal vertices (neither a root nor a leaf) provide us with freedom to choose, for each back edge  $(a, b)$ , at which visits of  $a$  and  $b$  each of the two constituent hairpins of it is to be formed. For each

<sup>3</sup> The simple reason for the absence of cross edges is that if for an edge  $e = \{u, v\}$ , vertex  $v$  is still unmarked when the traversal first considers  $e$  at vertex  $u$  (or vice versa), then  $e$  becomes a tree edge.

<sup>4</sup> Also known as a “twice-around-the-tree walk” and *full walk* in [1, p. 1112].



back edge  $(a, b)$ , an occurrence of  $a$  and that of  $b$  can be chosen and connected by an arc; then we say that this edge is *opened* at the time point which corresponds to the left end of the arc, and *closed* at the right end.

The freedom in timing opening and closing of back edges brings multiple arc diagrams based on the same DFS arrangement. Which should we choose then? Given an arc diagram of  $G$ , we say that a cycle in  $G$  is *closed at an edge  $e$*  if  $e$  is in the cycle, and in the arc diagram, all the other edges involved in the cycle have already been closed before  $e$  is closed. It is known to be kinetically unfavourable to close a cycle at a tree edge. This experimental obstacle motivates our “phloem principle” for ordering back edge connections.<sup>5</sup> Recall that the vertices  $a$  and  $b$  are labelled with distinct integers  $\delta(a)$  and  $\delta(b)$ , respectively, and whichever labelled smaller is an ancestor of the other. This principle says that, if  $\delta(a) < \delta(b)$ , then the assembly of this edge (by a kissing loop) should begin at  $b$  and end at  $a$ .

► **Lemma 1.** *The phloem principle prevents any cycle of  $G$  from being closed at a tree edge.*

**Proof.** Any cycle of  $G$  involves a back edge as  $T = (V, S, r, \delta)$  is acyclic. Let  $E'$  be the set of edges in this cycle; then  $E' \cap S$  and  $E' \setminus S$  are the set of tree edges in the cycle and the set of back edges in the cycle, respectively. The absence of cross edges implies that subtrees of a vertex are connected only via the vertex even in  $G$ . Hence, the vertex with the smallest  $\delta$ -value in a cycle must be incident to a back edge in the cycle; in other words, the following inequality must hold:

$$\min_{(a,b) \in E' \setminus S} \{\min(\delta(a), \delta(b))\} \leq \min_{(u,v) \in E' \cap S} \{\min(\delta(u), \delta(v))\}.$$

Thus, this cycle is closed at the back edge. ◀

This lemma ensures that arc diagrams serve as a blueprint of the polymerase-trap-free co-transcriptional folding pathway as long as they obey the phloem principle. Now we count out any arc diagram that violates the principle. Thus, from now on, an arc diagram of  $G = (V, E)$  is a pair of a DFS arrangement  $p_1, p_2, \dots, p_m$  based on a DFS tree  $T = (V, S, r, \delta)$  of  $G$  and a mapping  $\alpha : E \setminus S \rightarrow [1..m] \times [1..m]$  with  $m = 2|V| - 1$  such that for all back edge  $e = (a, b) \in E \setminus S$  with  $\delta(a) < \delta(b)$ , if  $\alpha(e) = (o, c)$ , then  $o < c$ ,  $p_o = b$ , and  $p_c = a$ . (Arcs for tree edges do not appear anywhere in this definition, but they are uniquely identified by the DFS arrangement.) As  $|S| = |V| - 1$ , all arc diagrams of  $G$  are provided with  $|E| - |V| + 1$  arcs for kissing loops. An arc diagram of the tetrahedron is shown in Figure 4 (d) and those of the cube without any branch and with a branch are shown respectively in Figures 5 (d) and 6 (d), where the arcs for kissing loops are coloured in green, while those for the tree stem are in blue. All of them follow the phloem principle.

The phloem principle does not fully eliminate the freedom in drawing an arc for  $(a, b)$  with  $\delta(a) < \delta(b)$  since  $b$  is visited more than once before the last visit at  $a$ , unless  $b$  is a leaf. For the sake of the kissing loop crossing (KLX) number, a measure of how good an arc diagram is that we shall discuss next, the arc should be drawn as short as possible, that is, from the last occurrence of  $b$  to the immediate occurrence of  $a$  (as  $a$  is an ancestor of  $b$ , the search returns to  $a$  after the last visit to  $b$ ). However, this criterion of KLX optimisation does not pay any attention to the possible adverse topological effect of focusing a lot of hairpin formations at one time point. Therefore, we have left this freedom in the above definition of arc diagram.

<sup>5</sup> Phloem is a pathway for transporting products of photosynthesis from leaves to the rest of a plant.

#### 4 Minimising kissing loop crosstalk

A set of kissing loop types should be as orthogonal as possible to minimise the risk of *crosstalk*, that is the risk of mismatching hairpins hybridising. However, sets of kissing loops that have proven orthogonal enough in the laboratory are limited in size (see, e.g., [10]). The size of largest orthogonal KL sets available in reality serves as a standard for deciding whether a specific (rooted, ordered) DFS tree should be chosen or not. If the design leaves more than this number of kissing loops open simultaneously at any point of folding, it increases the risk of crosstalk.

Recall that any DFS arrangement corresponds one-to-one with a rooted and ordered DFS tree. Given an arc diagram  $D = ((p_1, \dots, p_m), \alpha)$  of a DFS tree  $T = (V, S, r, \delta)$  of a graph  $G = (V, E)$ , the *KLX number* of a segment  $(p_i, p_{i+1})$  is the number of arcs that cross the vertical line drawn between  $p_i$  and  $p_{i+1}$ . It is defined formally as

$$\kappa(p_i, p_{i+1}) = |\{e \in E \setminus S \mid \alpha(e) = (o, c), o \leq i, i+1 \leq c\}|. \quad (1)$$

The maximum of these values across all segments is the *KLX number of this diagram*  $D$ , that is,  $\kappa(D) = \max_{1 \leq i < m} \{\kappa(p_i, p_{i+1})\}$ . Finally, the *KLX number of the graph*  $G$ , denoted by  $\kappa(G)$ , is the minimum among the KLX numbers of all the possible arc diagrams of  $G$ .

The 1-to-1 correspondence between DFS arrangements and pairs of a graph and its rooted and preordered DFS tree justifies the introduction of the notation  $\kappa(G, T)$  as an alias of  $\kappa(G)$ .

► **Lemma 2.** *Let  $G = (V, E)$  and  $T = (V, S, r, \delta)$  be its rooted DFS tree. Let  $T'$  be a (connected) subtree of  $T$ , and  $G'$  be the subgraph of  $G$  induced by the vertex set of  $T'$ . Then  $\kappa(G', T') \leq \kappa(G, T)$ .*

**Proof.** It is known that  $T'$  becomes a DFS tree of  $G'$  [13]. Indeed, it suffices to traverse  $T'$  according to the preorder  $\delta$ . Note that  $T'$  may preorder the vertices differently and more favorably for the KLX number. ◀

This lemma can be used to prune the search tree for DFS trees with small KLX number, as outlined in Section 6.

As an algorithmic tool, it is useful to exclude some back edges from computing of the KLX number. For a subset of back edges  $B \subseteq E \setminus S$ , the KLX number of the segment  $(p_i, p_{i+1})$  restricted to  $B$ , denoted by  $\kappa_B(p_i, p_{i+1})$ , can be computed by replacing the occurrence of  $E \setminus S$  in Eq. (1) with  $B$ . It is also convenient to define the KLX number of a tree edge  $e \in S$  as the number of kissing loops that are opened but yet to be closed during the backward traversal across the edge; the following inequality justifies this definition.

► **Lemma 3.** *In the setting above, let  $(p_i, p_{i+1})$  and  $(p_j, p_{j+1})$  be the segments that correspond to the forward and backward traversals through an edge  $(u, v)$  of  $T$ , that is,  $p_i = p_{j+1} = u$  and  $p_{i+1} = p_j = v$ . Then  $\kappa(p_i, p_{i+1}) \leq \kappa(p_j, p_{j+1})$ .*

**Proof.** In order for this inequality not to hold, there must be an arc  $(o, c)$  that crosses the segment  $(p_i, p_{i+1})$  but not  $(p_j, p_{j+1})$ , that is,  $o \leq i$  and  $i+1 \leq c \leq j$ . Then  $\delta(p_o) < \delta(p_c)$  would hold, but this contradicts the phloem principle. ◀

► **Example 4 (KLX number of the cube).** See an arc diagram of the cube in Figure 5 (d);  $\kappa(h, g) = \kappa(b, a) = 2$ ,  $\kappa(g, f) = \kappa(c, b) = 3$ ,  $\kappa(f, e) = \kappa(d, c) = 4$ , and  $\kappa(e, d) = 3$ , and therefore, the KLX number of this diagram is 4. Compare this with another diagram of the cube in Figure 6 (d), whose KLX number is 5. Consequently, the KLX number of the cube is at most 4.

Recall that, with one DFS tree fixed along with the preorder  $\delta$ , any back edge  $(a, b)$  with  $\delta(a) < \delta(b)$  should be opened at the last visit to  $b$  and then closed ASAP, that is, at the next visit to  $a$ . No other timing of opening/closing this edge that respects the phloem principle improves in terms of KLX minimisation.

Given a rooted DFS tree  $T = (V, S, r)$  without any preorder specified, the sibling order with the minimum KLX number can be computed bottom-up. Consider a branch  $v$  with its siblings  $v_1, \dots, v_d$ , and suppose that they are visited in this order:  $v_1$  first,  $v_2$  next, and so on. Then any back edge between the subtree rooted at  $v_i$  and a vertex strictly above  $v$  increments by 1 the KLX number of all the segments corresponding to the edges  $(v, v_{i+1}), (v, v_{i+2}), \dots, (v, v_d)$  or all the subtrees below them, although this contribution may not be very clear on the drawing of a DFS tree annotated with back edges, unless the tree is without any branch. Compare (c) with (d) in Figure 6; the back edges  $(d, a)$  and  $(d, c)$  even cross the segments that correspond to the whole traversal of the other subtree of  $e$ , which consists of the edges  $(e, h)$  and  $(h, g)$ ; this is as clear as day on the arc diagram. The subtrees below  $v_1, \dots, v_{i-1}$  are spared this increment because they have been fully explored before such an edge is opened. The back edges that cross over  $v$  increase the KLX numbers of the path from the root to  $v$  independently of the order in which the children of  $v$  are visited. These observations allow the KLX-minimum sibling orders for a given rooted but unordered tree to be computed in a *bottom-up* manner, starting from leaves, by comparing at each branch (the domainial number of) all permutations of its children.

Let  $v$  be a vertex with  $k$  children  $v_1, v_2, \dots, v_d$ ,  $T_v$  be the subtree of  $T$  below  $v$ , and  $T_i$  be the subtree of  $T$  below  $v_i$ . Let  $B$  be the set of back edges one of whose endpoints is in  $T_v$ , and let  $B_i$  be defined analogously with respect to  $T_i$ . Suppose that for all edges  $e$  in  $T_i$  with  $1 \leq i \leq k$ , the KLX numbers restricted to the back edges opened below  $v_i$ , that is,  $\kappa_{B_i}(e)$ , have already been calculated. Let us compute the KLX number restricted less severely to the back edges opened below  $v$ . The back edges that come from inside  $T_i$  and go outside, that is, towards the path of  $T$  from the root to  $v$  can be categorised into those ending at  $v$  and those that go beyond; let us count them and denote the counts, respectively, by  $\kappa(T_i, v)$  and by  $\kappa(T_i, > v)$ . Suppose that the children  $v_1, \dots, v_d$  of  $v$  are visited in this order. The KLX number of an edge  $e$  in  $T_i$  would be then incremented by  $\sum_{k < i} \kappa(T_k, > v)$ , that is,

$$\kappa_B(e) = \kappa_{B_i}(e) + \sum_{k < i} \kappa(T_k, > v). \quad (2)$$

That of an edge  $(v, v_i)$  would be set as

$$\kappa_B((v, v_i)) = \left( \sum_{k < i} \kappa(T_k, > v) \right) + \kappa(T_i, v) + \kappa(T_i, > v). \quad (3)$$

With the maximum among these numbers in Eqs. (2) and (3), this order competes with the others, and the children of  $v$  should be ordered according to the one that achieves the minimum; then the KLX number restricted to the back edges opened below  $v$  should be updated for all tree edges below  $v$  accordingly.

Ordering the children of even a single vertex in this way may require domainial time in  $|V|$ . For the class of 3-regular graphs, quadratic time suffices as a vertex can have at most two children.

## 5 The minimum kissing loop crossing and minimum tree depth problems

The *minimum KLX (kissing loop crossing) number problem* (MINKLX) asks, given an undirected, connected graph  $G$  and a positive integer  $k$ , if there exists an arc diagram of  $G$  whose KLX number is at most  $k$ . This problem is computationally hard as stated in the next theorem; its proof can be found in the technical appendix A.

► **Theorem 5.** *The MINKLX problem is NP-hard.*

Another relevant problem is that of determining the depth of the shallowest DFS tree for a given graph  $G$ , because finding a shallower DFS tree decreases the number of helical domains kept open in parallel, and may result in a sequence with fewer helical domain types. This quantity is closely related to the *tree-depth* of  $G$ , which is defined to be the depth of the shallowest DFS tree for a *supergraph* of  $G$ . It is NP-hard to compute the tree-depth even for the class of triangulated graphs [2].

Considering the NP-hardness of the MINKLX problem, the only approach to finding a KLX-minimal, cotranscription-friendly designs for a given graph  $G$  may be via full enumeration of all the DFS trees for  $G$ . In the next section, we present a branch-and-bound approach to this enumeration process.<sup>6</sup>

Before proceeding, let us note that Theorem 5 does not exclude the possibility that KLX-optimal DFS trees could still be found in polynomial time for some graph classes of practical importance such as 3-regular or polyhedral graphs. As any graph can be approximated by a 3-regular one by replacing each vertex by a network of vertices of degree 3, it would be quite interesting to know whether the KLX-minimisation task remains hard for this class. As regards the class of polyhedral graphs, it is known that the MINKLX-related cutwidth minimisation problem (described in the proof of Theorem 5 in Appendix A) is NP-hard for the class of planar graphs with maximum degree 3 [16].

## 6 Solving the MinKLX and MinTD problems by enumeration

Let us now propose a prototype of algorithmic enumeration of DFS trees for computing the KLX number of a given graph  $G = (V, E)$ . Algorithm 2 is its pseudocode. Starting from the empty forest, it enumerates the DFS trees by inserting edges of  $G$  one by one in a predetermined order as a *tree* edge, that is, as an edge of a DFS tree to be built up. Biconnectedness is utilized, the property of a graph being free from an articulation point, whose removal disconnects the graph. DFS trees of a graph cannot circumvent any biconnected components, or *blocks*, of  $G$ . We shall demonstrate how each local branching of a DFS tree orients edges of  $G$  globally once it is accommodated inside a block of  $G$ . Orientations thus insisted by more than one such local branching are highly likely to contradict each other, enabling the incremental enumeration of DFS trees to prune the edge-insertion-based search tree. As a shallower tree involves more branches, the tighter an optional upper bound on the depth of DFS trees to be obtained is set, the more effective this block-based pruning should get.

<sup>6</sup> As an aside, if one gives up the goal of cotranscription-friendly design, a strand routing that minimises the total number of kissing loops needed to complement it can be found efficiently for any graph, and this number is for many typical wireframe models just zero or one [4].

---

**Algorithm 2** Enumerative KLX minimisation for a graph  $G = (V, E)$ .
 

---

```

1: Let  $n = |V|$ ,  $m = |E|$ , and edges be indexed as  $E = \{e_1, e_2, \dots, e_m\}$ 
2:
3: function KLXT( $T$ ) ▷ Compute KLX number of tree  $T$ 
4:   compute the KLX number of tree  $T$  in the way described in Section 4
5: end function
6:
7: function KLX( $F, k$ )
8: ▷ Compute min KLX number over all completions of forest  $F$  with edges in  $\{e_k, \dots, e_m\}$ 
9:   if  $F$  comprises a single tree with  $n - 1$  edges then
10:      $klx_{\text{tree}} \leftarrow \text{KLXT}(F)$ 
11:      $klx_{\text{min}} \leftarrow \min\{klx_{\text{min}}, klx_{\text{tree}}\}$ 
12:     return  $klx_{\text{tree}}$ 
13:   end if
14:   if edge  $e_k$  does not create a cycle and is admissible in  $F$  then ▷ See Figure 9
15:      $F' \leftarrow F \cup \{e_k\}$  ▷  $e_k$  included as a tree edge
16:     let  $T$  be the tree that contains edge  $e_k$  in forest  $F'$ 
17:     if  $\text{KLXT}(T) \geq klx_{\text{min}}$  then ▷ Prune if cost of  $T \geq klx_{\text{min}}$ 
18:        $klx_1 \leftarrow m$ 
19:     else
20:        $klx_1 \leftarrow \text{KLX}(F', k + 1)$ 
21:     end if
22:   end if
23:    $klx_0 \leftarrow \text{KLX}(F, k + 1)$  ▷  $e_k$  not included as a tree edge
24:   return  $\min\{klx_1, klx_0\}$ 
25: end function
26:
27:  $klx_{\text{min}} \leftarrow m$ 
28: return  $\text{KLX}(\emptyset, 1)$  ▷ Start with an empty forest

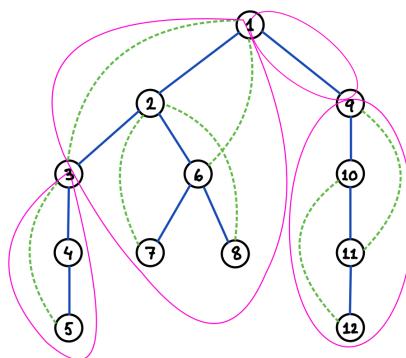
```

---

A graph  $G = (V, E)$  and its spanning tree  $T = (V, S)$  can be uniquely decomposed into a set of biconnected components, or *blocks*,  $E_i \subseteq E$ , along with a spanning tree  $T_i = (V_i, S_i)$  that is the intersection of  $T$  and  $G_i = (V_i, E_i)$ , the induced subgraph of  $G$  by  $E_i$ ; unless confusion arises, we may call even  $G_i$  a block. The blocks can then be uniquely organised into a so-called *block-cut tree* by connecting two blocks by an edge if these blocks induce the subgraphs of  $G$  that share (exactly one) vertex in common, which is an articulation point of  $G$  (see Figure 7 for an example); any articulation point of  $G$  thus serves as an *interface* among multiple blocks. We say that  $G$  is the underlying graph of this block-cut tree. Each block  $G_i$  is one of the following types:

- **Branching.** if  $T$  involves a vertex that is incident to at least three edges in  $E_i$  (*branch*);
- **Spinal.** otherwise.

Every block contains at least one edge of  $T$  and no blocks share an edge; hence, the block-cut tree is composed of at most  $|S| = |V| - 1$  blocks. If a node in a block  $E_i$  is incident to at least two tree edges of the block, say  $e_1, e_2 \in S \cap E_i$ , then we say the node is *internal*. Hence, a spinal block can contain at most two non-internal nodes. Note that an internal node in a spinal block can be a branch of  $T$ ; unlike a branch in a branching block, this branch is an articulation point of  $G$ . Therefore, a spinal block can have three or more interfaces. A spinal block consisting of a single edge is particularly called a *bridge* in [18], and we will borrow this term when it matters whether a spinal block involves an internal vertex or not.



■ **Figure 7** A DFS tree and the biconnected components of a 12-vertex graph.

## 6.1 KLX computation

The decomposition of a graph into blocks may facilitate the computation of the KLX number of  $G$ , but may not yet save it from brute-forcing sibling orders at every branch, which was discussed at the end of Section 4. Consider a block-cut tree that is free from a branching block; note that the underlying graph may admit a DFS tree with a branch because an internal node of a spinal block can serve as an interface as explained above, and three or more spinal blocks may be incident to an articulation point. The absence of branching block enables the KLX number of the underlying graph according to a DFS tree to be computed as the maximum of those of the blocks according to the respective restrictions of the DFS tree; indeed, it makes no sense for the depth-first-search to stay in a spinal block at any internal interface unless all the other blocks incident to the interface have been already traversed.

The computation of the KLX number of each block according to a DFS tree under construction does not require the DFS tree to be rooted but suffices for each block to know at or beyond which interfaces of them lies the global root. As observed shortly, the DFS tree cannot be rooted at any internal node. Hence, a spinal block can be locally rooted only at one of its two non-internal nodes. Its KLX number does not depend on which of these two vertices the global root is on or beyond. Branching blocks rather restrict where the global root can be due to intrinsic orientation of each of their branches, as we see from now on.

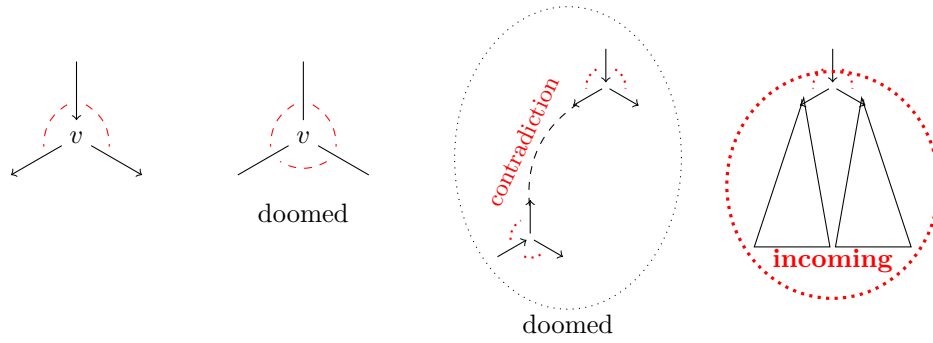
## 6.2 Pruning

The enumerative computation of the KLX number of  $G$  can adopt two kinds of pruning strategies.

The one based on the downward closedness of the KLX number along a specific spanning tree (Lemma 2) is simple; while growing a DFS tree as a block-cut forest, once the KLX number of any tree in the forest exceeds a predetermined target value or the current best, then this path of the enumerative search is not worth being pursued further, and hence should be pruned.

The other strategy is based on the following two structural properties of blocks and their DFS rooting from [13]: given a biconnected graph and its spanning tree like the block  $E_i$  and its spanning tree  $T_i$ ,

1. in order for the spanning tree to be a DFS tree of the graph, it must be rooted at a leaf (of the tree, not of the biconnected graph, all of whose vertices are of degree at least 2);
2. if the spanning tree has a branch, then there exists at most one vertex (indeed, a leaf as just recalled) starting from which the tree can be traversed in a DFS manner.



■ **Figure 8** Intrinsic orientation of a branch, an internally contradicting block, and global prohibition of root by a single branch. Dotted red lines indicate a bypass around the center of a branch, which is necessary for a graph to be biconnected.

Let us reproduce a proof for Property 2 (Observation 3.2 in [13]). Suppose there were two such vertices  $a$  and  $b$ . If rooted at  $a$ , the branch, say  $v$ , has at least two subtrees  $T_1$  and  $T_2$ , neither of which contains  $a$ ; without loss of generality, we assume that  $T_1$  does not contain  $b$ . Since the tree becomes a DFS tree by being rooted at  $a$ , and  $v$  is not an articulation point, there must be a back edge from  $T_1$  to some vertex above  $v$  along the unique tree path from  $v$  to the root  $a$ , but this would become a cross edge once the tree is rather rooted at  $b$ .

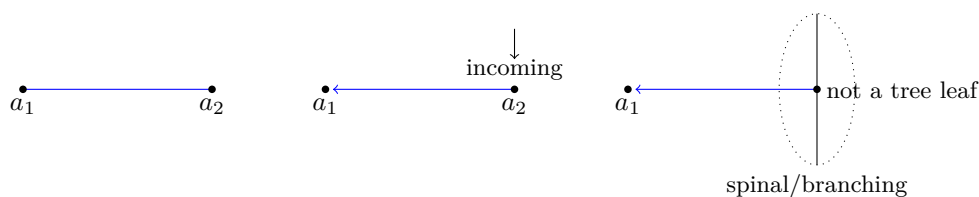
This proof can be applied to observe in our context that every branch inside a branching block is intrinsically oriented, independently of other branches, in such a way that among the edges incident to it, one is incoming while the others are outgoing, as illustrated in Figure 8 (one must be incoming as the branch cannot be a root due to Property 1). Each (local) branch thus globally prohibits a DFS tree of  $G$  from being rooted at any vertex beyond these outgoing edges. As indicated by red dashed lines in Figure 8, a branch cannot be part of a DFS tree of  $G$  if it can be bypassed to go back and forth between any two of its three subtrees (second from the left in the figure). A branch inside a block thus orients some edges across the tree that contains the block, and the in-degree of a vertex is defined naturally. Intrinsic orientations imposed by two branches may contradict each other even inside a block, for example, by yielding a vertex of in-degree 2 or higher (second from the right). As soon as the addition of an edge results in such branch(es), a forest under construction is doomed and should be pruned from the search tree.

### 6.3 Edge insertion and merging of blocks

An edge can be added to grow a forest only if all the following conditions hold: (1) it bridges two separate trees, say  $T_1$  and  $T_2$ , (connecting two vertices in the same tree would result in a contradictory cycle of tree edges), (2) at least one of the endpoints, say  $v_1$  and  $v_2$ , is an articulation point of  $G$ , and (3)  $v_1$  or  $v_2$  is of in-degree 0; it is then oriented from the endpoint of in-degree 1, if any, to the other (whose in-degree must be 0); if both endpoints are of in-degree 0, then the added edge remains yet-to-be-oriented (see Figure 9). Let us note, related to (2), that a bridge between an internal node and another node is intrinsically oriented away from the internal node in order to prevent a block from being rooted locally at its internal node. Hence, no tree edge can be added between two internal nodes.

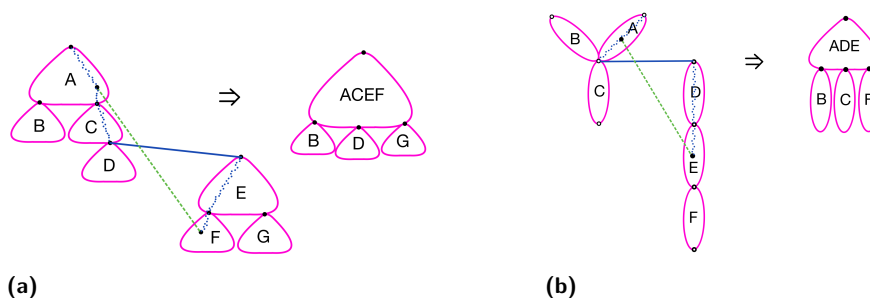
All the other edges between  $T_1$  and  $T_2$  are ruled out as a candidate of tree edges but introduced rather as an ancillary edge. They are however not guaranteed to be consistent; for example, if one of them is between the leaves  $u_1$  and  $u_2$ , another is between the leaves





■ **Figure 9** The three admissible edge additions between two trees: (left) between two articulation points that are not incoming; (middle) between two articulation points exactly one of which is incoming; (right) between a non-incoming articulation point and an internal vertex of a spinal or branching block that is not a tree leaf. In the latter two cases, the added edge is oriented towards the non-incoming articulation point,  $a_1$  here, and makes the whole tree where the point is, that is, the left tree here, incoming, that is, prohibited from being rooted.

$v_1$  and  $v_2$ , and all these four vertices are pairwise distinct, then the tree must be rooted globally at  $u_1$  or  $u_2$  in order for the edge not to become a cross edge, and  $v_1$  or  $v_2$  claims the ownership of the global root analogously, but these two claims are obviously not compatible. Bridging two trees by an edge may merge some of the blocks in  $T_1$  and in  $T_2$  into one (see Figure 10). The resulting block can be computed efficiently [18] but its KLX number should also be computable more efficiently from those of the merged blocks and from the cost due to the newly added ancillary edges than being computed from scratch.



■ **Figure 10** Two admissible ancillary edge types and their corresponding block-cut tree updates. (a) Edge connecting oriented blocks. (b) Edge connecting unoriented blocks. (Note that in this case the block ADE becomes branching and hence oriented.)

## 7 Examples

The cotranscription-friendly DFS-tree based design method presented in Section 3 is implemented and available for use in the online design tool *DNAforge* (<https://dnaforge.org>), together with an option for minimising the KLX cost of the design with a preliminary version of the enumeration method presented in Section 6.<sup>7</sup>

Figures 11 and 12 illustrate some outcomes from the tool. Figure 11 shows designs of wireframe dodecahedra based on a randomly chosen spanning tree (upper row) and a DFS spanning tree (lower row). The DFS-tree based design has also been KLX-optimised, resulting in a reduction from a KLX number of 9 in the initial DFS tree to 6 in the optimal

<sup>7</sup> Design method ST-RNA, additional parameters “co-transcriptional route” and “minimise the number of kissing loop sequences”.



■ **Table 1** Effect of KLX minimisation on some 3D mesh models.

Model	Vertices	Edges	Initial KLX	Min KLX
Tetrahedron	4	6	3	3
Cube	8	12	4	4
Octahedron	6	12	6	5
Dodecahedron	20	30	9	6
Icosahedron	12	30	12	10
Bunny	66	192	60	33

one. (The spanning tree diagrams in Figures 11(b) and (e)) have been manually reconstructed from the tool-generated diagrams in Figures 11(c) and (f).) Figure 12 displays random-tree and DFS-tree designs for a 66-vertex, 192-edge wireframe model of a bunny. Also here the DFS-tree based design has been KLX-optimised, resulting in a KLX number reduction from 60 in the initial DFS tree to 33 in the optimal one. Table 1 summarises the KLX number reductions for some basic mesh models.

## 8 Conclusions and further work

We have presented models and algorithms for addressing two tasks in secondary structure design for cotranscriptionally folding DNA origami wireframe nanostructures: avoiding the topological folding obstacle of polymerase trapping and minimising the number of distinct kissing loop designs (the KLX number). The key tools in this work have been the algorithmic method of depth-first search in graphs and the ensuing DFS spanning trees. Our branch-and-bound approach to the KLX minimisation problem can also be used for any other effectively computable objective function on DFS trees, such as the DFS tree depth of a given graph (the TD number).

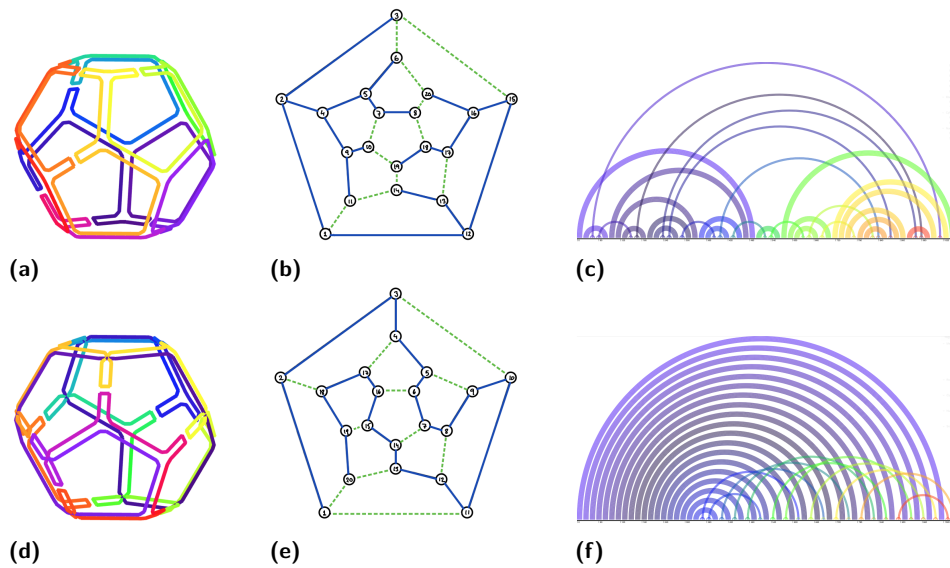
Relevant directions for further work include for instance the following:

1. Nucleotide-level sequence design for DNA origami wireframes in the cotranscriptional setting.
2. Efficient combinatorial algorithms for minimising the KLX and TD numbers in some interesting classes of graphs, such 3-regular or polyhedral graphs, or proving the problems NP-hard in these classes.
3. Efficient fixed-parameter or approximation algorithms for minimising the KLX and TD numbers in some relevant classes of graphs.

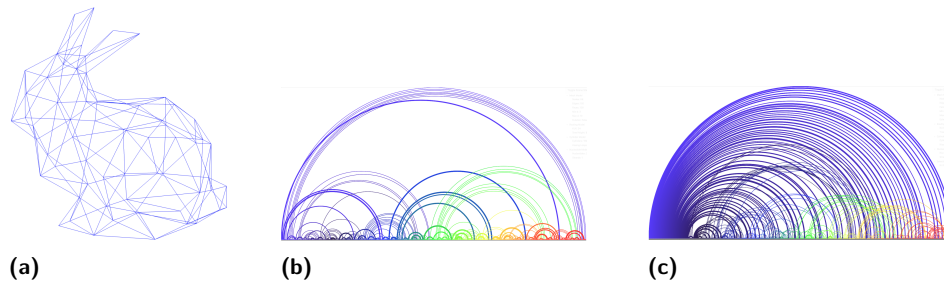
---

## References

- 1 Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein. *Introduction to Algorithms*. MIT Press, Cambridge MA, USA, 4th edition, 2022.
- 2 Dariusz Dereniowski and Adam Nadolski. Vertex rankings of chordal graphs and weighted trees. *Information Processing Letters*, 98(3):96–100, 2006. doi:10.1016/J.IPL.2005.12.006.
- 3 Antti Elonen, Ashwin Karthick Natarajan, Ibuki Kawamata, Lukas Oesinghaus, Abdulmelik Mohammed, Jani Seitsonen, Yuki Suzuki, Friedrich C. Simmel, Anton Kuzyk, and Pekka Orponen. Algorithmic design of 3D wireframe RNA polyhedra. *ACS Nano*, 16:16608–18816, 2022. doi:10.1021/acsnano.2c06035.
- 4 Antti Elonen and Pekka Orponen. Designing 3D RNA origami nanostructures with a minimum number of kissing loops. In *30th International Conference on DNA Computing and Molecular Programming*. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2024. doi:10.4230/LIPICS.DNA.30.4.



■ **Figure 11** Upper row: a dodecahedron design based on a randomly chosen spanning tree. (a) Strand routing on the 3D wireframe. (b) Spanning tree (solid blue) and back edges (dashed green) on the Schlegel diagram. (c) Domain-level arc diagram. (Long helical domain pairings thick, short kissing-loop domain pairings thin.) Lower row (d)-(f) A dodecahedron design based on a KLX-optimised DFS spanning tree (KLX = 6).



■ **Figure 12** Designs for a 3D mesh model of a bunny. (a) Wireframe model. (b) Arc diagram of random spanning tree routing. (c) Arc diagram of KLX-optimised DFS spanning tree routing (KLX = 33).

- 5 Fănică Gavril. Some NP-complete problems on graphs. In *Proceedings of the 1977 Conference on Information Sciences and Systems*, pages 91–95, 1977. URL: <https://scispace.com/papers/some-np-complete-problems-on-graphs-4102n07t1h>.
- 6 Cody Geary, Guido Grossi, Ewan K. S. McRae, Paul W. K. Rothmund, and Ebbe S. Andersen. RNA origami design tools enable cotranscriptional folding of kilobase-sized nanoscaffolds. *Nature Chemistry*, 13(6):549–558, 2021. doi:10.1038/s41557-021-00679-1.
- 7 Cody Geary, Pierre-Étienne Meunier, Nicolas Schabanel, and Shinnosuke Seki. Programming biomolecules that fold greedily during transcription. In *41st International Symposium on Mathematical Foundations of Computer Science*, pages 43:1–43:14. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2016. doi:10.4230/LIPICS.MFCS.2016.43.
- 8 Cody Geary, Paul W. K. Rothmund, and Ebbe S. Andersen. A single-stranded architecture for cotranscriptional folding of RNA nanostructures. *Science*, 345(6198):799, 2014. doi:10.1126/science.1253920.

- 9 Cody W. Geary and Ebbe Sloth Andersen. Design principles for single-stranded RNA origami structures. In *20th International Conference on DNA Computing and Molecular Programming*, pages 1–19. Springer, 2014. doi:10.1007/978-3-319-11295-4\_1.
- 10 Wade W. Grabow, Paul Zakrevsky, Kirill A. Afonin, Arkadiusz Chworos, Bruce A. Shapiro, and Luc Jaeger. Self-assembling RNA nanorings based on RNAI/II inverse kissing complexes. *Nano Letters*, 11(2):878–887, 2011. doi:10.1021/nl104271s.
- 11 Peixuan Guo. The emerging field of RNA nanotechnology. *Nature Nanotechnology*, 5(12):833–842, December 2010. doi:10.1038/nnano.2010.231.
- 12 Daniel Jasinski, Farzin Haque, Daniel W. Binzel, and Peixuan Guo. Advancement of the emerging field of RNA nanotechnology. *ACS Nano*, 11(2):1142–1164, 2017. doi:10.1021/acsnano.6b05737.
- 13 Ephraim Korach and Zvi Ostfeld. DFS tree construction: Algorithms and characterizations. In *Graph-Theoretic Concepts in Computer Science*, pages 87–106. Springer Berlin Heidelberg, 1989. doi:10.1007/3-540-50728-0\_37.
- 14 Di Liu, Cody W. Geary, Gang Chen, Yaming Shao, Mo Li, Chengde Mao, Ebbe S. Andersen, Joseph A. Piccirilli, Paul W. K. Rothmund, and Yossi Weizmann. Branched kissing loops for the construction of diverse RNA homooligomeric nanostructures. *Nature Chemistry*, 12(3):249–259, 2020. doi:10.1038/s41557-019-0406-7.
- 15 Abdulmelik Mohammed, Pekka Orponen, and Sachith Pai. Algorithmic design of co-transcriptionally folding 2D RNA origami structures. In *Unconventional Computation and Natural Computation: 17th International Conference*, pages 159–172. Springer, 2018. doi:10.1007/978-3-319-92435-9\_12.
- 16 Burkhard Monien and Ivan Hal Sudborough. Min cut is NP-complete for edge weighted trees. *Theoretical Computer Science*, 58(1):209–229, 1988. doi:10.1016/0304-3975(88)90028-X.
- 17 Erik Poppleton, Niklas Urbanek, Taniya Chakraborty, Alessandra Griffo, Luca Monari, and Kerstin Göpprich. RNA origami: design, simulation and application. *RNA Biology*, 20(1):510–524, 2023. doi:10.1080/15476286.2023.2237719.
- 18 Jeffery Westbrook and Robert E. Tarjan. Maintaining bridge-connected and biconnected components on-line. *Algorithmica*, 7(1):433–464, 1992. doi: 10.1007/BF01758773. doi: 10.1007/BF01758773.

## A NP-completeness of the MinKLX problem

► **Theorem 5.** *The MINKLX problem is NP-hard.*

**Proof.** The proof is based on the proof by Gavril [5] for the NP-hardness of computing the cutwidth of a graph  $G = (V, E)$ , which asks, given also an integer  $k$ , to arrange the vertices of  $G$  along a horizontal line in such a way that, for any vertical line drawn between adjacent vertices, dividing  $V$  into those to its left and those to its right,

The reduction is from MAX CUT, which asks to split the vertex set  $V$  of a given weighted graph  $G = (V, E)$  into two subsets  $V' \subseteq V$  and  $V \setminus V'$  so as to maximise the sum of the weights of edges that connect these subsets in  $G$ . Given a pair  $(G, w)$  of a  $n$ -vertices graph  $G = (V, E)$  and a positive integer  $w$ , let us convert this instance of max cut into an instance of KLX computation problem  $(\bar{G}, k)$  as follows. With a “large enough”  $r$ , let  $U = \{u_1, u_2, \dots, u_r\}$  be a set of auxiliary “universal” vertices. Let  $\bar{G} = (V \cup U, \bar{E})$  with  $\bar{E} = ((V \cup U) \times (V \cup U)) \setminus E$ . Note that, for any  $x \geq 1$ , all DFS trees of the complete graph  $K_x$  are equivalent, they are indeed a path (no branch), and their KLX number  $f(x) = \lceil x/2 \rceil \times \lfloor x/2 \rfloor$  can be computed in polynomial time. Let  $k = f(n + r) - w$ . Now we are ready to show that  $G$  has a cut  $(A, B)$  with at least  $w$  edges between  $A$  and  $B = V \setminus A$  if and only if the KLX number of  $\bar{G}$  is at most  $k$ .

Firstly, suppose  $\overline{G}$  has such a cut, and let  $A = \{a_1, a_2, \dots, a_m\}$  and  $B = \{b_1, b_2, \dots, b_{n-m}\}$ . The linear arrangement of  $V \cup U$

$$a_1, u_1, a_2, u_2, \dots, u_{m-1}, a_m, u_m, u_{m+1} \dots u_{r-(n-m-1)}, b_1, u_{r-(n-m-1)+1}, \dots, u_r, b_{n-m}$$

amounts to a DFS tree of  $\overline{G}$  thanks to a universal “glueing” vertex between  $a_i$  and  $a_{i+1}$  (or  $b_j$  and  $b_{j+1}$ ), which are not necessarily connected in  $\overline{G}$  (indeed, by definition, they are not connected in  $\overline{G}$  iff they are connected in  $G$ ). With large enough  $r$ , a tree edge that is crossed by the largest number of back edges is located in the interval  $u_m, \dots, u_{r-(n-m-1)}$ , and this number is at most  $k$ . Thus, the KLX number of  $\overline{G}$  is at most  $k$ .

For the opposite implication, suppose that  $\overline{G}$  has a DFS tree  $T$  whose KLX number is at most  $k$ . This tree must be “almost” a path in the sense that below a branch, if any, no universal vertex can appear in order not to introduce any cross edge between subtrees below the branch. Therefore, the path from the root of  $T$  to its first branch, if any, is of length at least  $r$ , and since  $r$  is large enough, an edge  $e$  that determines the KLX number of this DFS tree is somewhere along this path. This path can be extended into a DFS path  $P$  of the complete graph  $K_{n+r}$ . The edge which is crossed by  $f(n+r)$  back edges is on this path. Among these back edges, at most  $k$  of them belong to  $\overline{E}$ , and the others are edges of the original graph; let  $E_1$  be the subset of  $E$  that consist of these edges. Then,  $f(n+r) \leq k + |E_1|$ , which implies  $|E_1| \geq f(n+r) - k = w$ . Let  $S$  be the set of vertices in  $V$  that occur above this edge. Then  $(S, V \setminus S)$  is a cut which is crossed by at least  $w$  edges. ◀

# Tile Blockers as a Simple Motif to Control Self-Assembly: Kinetics and Thermodynamics

Constantine G. Evans   

Hamilton Institute and Department of Computer Science, Maynooth University, Ireland

Angel Cervera Roldan  

Hamilton Institute and Department of Computer Science, Maynooth University, Ireland

Trent Rogers  

Hamilton Institute and Department of Computer Science, Maynooth University, Ireland

Computer Science & Computer Engineering, University of Arkansas, Fayetteville, AR, USA

Damien Woods   

Hamilton Institute and Department of Computer Science, Maynooth University, Ireland

---

## Abstract

A fundamental problem in crystallisation, and in molecular tile-based self-assembly in particular, is how to simultaneously control its two main constituent processes: seeded growth and spontaneous nucleation. Often, we desire out-of-equilibrium growth without spontaneous nucleation, which can be achieved through careful calibration of temperature, concentration and experimental time-scale a laborious and overly-sensitive approach. Another technique is to find alternative nucleation-resistant tile designs [Mineev et al, 2001]. Rogers, Evans and Woods [In prep] propose *blockers*: short DNA strands designed to dynamically block DNA tile sides, altering self-assembly dynamics. Experiments showed independent and tunable control on nucleation and growth rates.

Here, we provide a theoretical explanation for these surprising results. We formally define the *kBlock* model where blockers bind to tiles at thermodynamic equilibrium in solution and stochastic kinetics allow self-assembly of a tiled structure. In an intentionally simplified mathematical setting we show that blockers permit reasonable seeded growth rates, akin to a non-blocked tile system at lower tile concentration, crucially giving nucleation rates that are exponentially suppressed. We then implement the kBlock model in a stochastic simulator, with results showing remarkable alignment with oversimplified theory. We provide evidence of blocker-induced tile buffering, where a large reservoir of blocked tiles slowly feeds a small unblocked tile subpopulation which acts like a regular, non-blocked, low tile concentration system, yet is capable of long-term buffered assembly. Finally, and perhaps most satisfyingly, theory and simulations align remarkably well with DNA self-assembly experiments over a wide range of concentrations and temperatures, matching the size of growth temperature windows to within 12%. Blockers are a straightforward solution to the challenging problem of simultaneously and independently controlling growth and nucleation, using a motif compatible with many DNA tile systems.

**2012 ACM Subject Classification** Applied computing → Chemistry; Applied computing → Physics; Theory of computation → Models of computation

**Keywords and phrases** Self-assembly, kinetic model, kinetic simulation, thermodynamic prediction

**Digital Object Identifier** 10.4230/LIPIcs.DNA.31.7

**Supplementary Material** Collection (Data, code, and source): <https://doi.org/10.5281/zenodo.15272400>

**Funding** Supported by Science Foundation Ireland (SFI) under grants numbers 20/FFP-P/8843 and 18/ERCS/5746, European Research Council (ERC) under the European Union's (EU) Horizon 2020 research and innovation programme (grant No 772766, Active-DNA project), and European Innovation Council and SMEs Executive Agency (EISMEA), No 101115422, DISCO project. Views and opinions expressed are however those of the author(s) only and do not necessarily reflect those of the EU, ERC, EISMEA or SFI. Neither the EU nor the granting authority can be held responsible for them.



© Constantine G. Evans, Angel Cervera Roldan, Trent Rogers, and Damien Woods;  
licensed under Creative Commons License CC-BY 4.0

31st International Conference on DNA Computing and Molecular Programming (DNA 31).

Editors: Josie Schaeffer and Fei Zhang; Article No. 7; pp. 7:1–7:19

Leibniz International Proceedings in Informatics



LIPICs Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

**Acknowledgements** We thank Erik Winfree for questions seeking a theoretical explanation of our experimental results at DNA28, as well as David Doty for suggesting, and Rebecca Schulman for discussions on, tile-buffering [20].

## 1 Introduction

Favourable self-assembly conditions typically catalyse two processes: continued growth of existing structures by monomer attachment, and spontaneous nucleation of new structures from free monomers in solution. Researchers usually prefer to have these two processes be at well-controlled and independent rates, a feat that can be incredibly difficult to achieve in practice. For example, perfect seeded growth [2, 21, 6, 26] requires no spontaneous nucleation but positive growth rate from a seed structure. Yet fabrication of seedless structures [23, 13, 10] requires a non-zero, ideally slow, rate of spontaneous nucleation, while having slow enough growth to avoid monomer depletion before structures complete. In almost all cases, we want to avoid low-temperature spurious growth. Thus, a fundamental problem for tile-based self-assembly is how to control both growth and nucleation rates for arbitrary tile-based systems.

A central principle of tile-based molecular self-assembly is that as temperature decreases, bonds strengthen, and so growth and nucleation should ordinarily become more favourable. This simple relationship allows systems to undergo controlled growth, for example, to create conditions where attachments by  $b$  bonds are favourable, while attachments by  $b - 1$  bonds are not. The control of the number of bonds required for favourable attachment allows for the computational power of algorithmic self-assembly, and is so closely linked to temperature that abstract models of tile assembly often use “temperature” to refer to the number of bonds required for attachment [24, 16]. Clean growth conditions with good bond specificity also seem important for good yield of uniquely addressed and periodic tile-based DNA nanostructures, exemplified by the use of slow anneals or long holding times at specific temperature, usually just below the lattice melting temperature [23, 10].

Growth from a pre-assembled seed structure of some form allows for some control of nucleation, and, sufficiently close to the melting temperature, “zig-zag” tile systems allow for an exponential reduction in spontaneous nucleation rates with increasing assembly width [22, 21]. However, these systems are inherently limited by nucleation pathways through increasing-size assemblies of tiles attaching purely by single-strength bonds, and must be close to the system’s melting temperature to avoid those pathways becoming significant. This practically creates a temperature window for seeded growth, above the temperature where spontaneous nucleation is significant, but sufficiently below the melting temperature so that growth is significant on experimental timescales. The window can be quite small and difficult to widen: Woods, Doty et al. [26] found a seeded growth window of  $<0.5^\circ\text{C}$ , increasing to  $2.5^\circ\text{C}$  only with a ten-fold reduction in concentration and corresponding decrease in growth rates.

A more dramatic approach to reducing spontaneous nucleation is to increase the difference in the number of bonds formed between a monomer and a growing assembly compared to between two monomers: from the 2 vs. 1 of usual threshold-2 systems, to  $k$  vs. 1. Mineev et al. [15], and later Wintersinger et al. [25], showed that criss-crossing multi-bond tile motifs binding to several other tiles in a growing assembly, rather than only two, strongly reduced spontaneous nucleation across a broad range of temperatures. However, the technique may require further characterisation [15] or use of large DNA origami rods [25, 5]. And theoretically, even if the temperature window is much wider, at sufficiently low temperatures, the system must still allow for spontaneous nucleation, even if to a disordered assembly: one-bond attachments and nucleation pathways of some sort will eventually become favourable.



Rogers, Evans and Woods [17, 18] recently demonstrated a third approach, adding additional “blocker” strands, each complementary to individual glue domains on tiles, as shown in Figure 1. While binding weakly at growth temperatures, the transitory binding of the higher-concentration blockers to tiles, both in solution and on assemblies, was conjectured to interfere with tile attachment, reducing both growth and spontaneous nucleation rates, but hopefully, reducing nucleation rates, reliant on pathways with many one-bond attachments, more than two-bond growth. Experimental results were promising, showing that for some choices of blocker concentration, it is possible to have a system with seeded growth across a range of temperatures and no significant spontaneous nucleation at *any* temperature. However, the mechanism by which these effects were achieved remained unclear, lacking a theoretical basis beyond conjecture.

### Contribution and paper structure

In Section 2, we define the kBlock kinetic model of tile assembly with blockers with the goal of explaining the results of Rogers et al. [17, 18]. In Section 3 we give two results. First, in Theorem 15, we show that behaviour of the kBlock model at a fixed temperature is largely equivalent to that of the kinetic Tile Assembly Model [24, 9] (kTAM) with a lowered, *effective tile concentration* and at the initial stages of growth before tile depletion dominates the kinetics. This effective tile concentration is dependent both on blocker concentration and temperature. Our second result, Theorem 19, is that with sufficient blocker concentration, the effective tile concentration decreases with decreasing temperature by enough to prevent spontaneous nucleation from ever becoming significant. Growth, meanwhile, can be confined to a temperature window of controllable width, with no significant growth, either seeded or by spontaneous nucleation, above or below the window (Remarks 16 and 17). And by increasing both tile and blocker concentrations, a “buffering” effect can be produced: for the same initial growth rate, the system with modified concentrations can incorporate far more tiles into assemblies before tile depletion prohibits further growth (Remark 18). Figures 2 and 4 illustrate these findings.

In Section 4, we show that this theoretical analysis fits well with kinetic model simulations for both growth and nucleation, and that simulations closely fit experimental results: for multiple experimentally-implemented systems, we show that temperature windows and relative growth rates with varying temperature can be closely matched by simulations using only tile and blocker concentrations, nearest-neighbour energies of glues, and a single lattice free energy parameter taken from the kTAM literature [10]. Finally, in Section 5 we draw several conclusions and suggest directions for future work.

## 2 kBlock model

### Intuition behind the kBlock model

We construct our kBlock model of tile assembly with blockers by starting with the same assumptions as the kTAM [24, 9] for tile-assembly interactions, and then adding blocker-assembly and blocker-tile interactions. Intuitively, we will assume that blockers behave like tiles, but rather than taking a place in the lattice (assembly), they simply block a particular edge of the free (in solution) tile or free edge of an assembly-bound tile where they bind, and that blocked tile-edges are unavailable for tile binding. However, unlike the kTAM, where there is an assumption of operating in the limit of low assembly concentrations and hence all reactions can be approximated as taking place with constant tile concentrations, in

the blockers model, even disregarding assemblies entirely, tiles and blockers will interact in solution, since the concentrations involved may make those interactions significant. Rather than considering the kinetics of those interactions, since tile and blocker concentrations will generally be much higher than those of assemblies, we will assume that the concentrations of *free blockers* (not bound to a tile or assembly), and free *tile states* – each specific tile-blocker complex configuration – remain at the equilibrium concentrations they would reach in the absence of any assemblies. Then, these concentrations will be the concentrations for tile states and free blockers when considering attachment to assemblies.

## 2.1 kBlock model definitions

We let  $\mathbb{R}, \mathbb{R}^+, \mathbb{R}^-$ , respectively denote the real numbers, non-negative real numbers and non-positive real numbers (i.e. all three include 0). We use the notation  $[x]$  to denote the number of instances of any object  $x$  in a fixed volume  $V$ , the *concentration*, where  $[x] \in \mathbb{R}^+$ .  $\Sigma$  denotes a finite alphabet, and  $\Sigma^*$  is the set of all strings over  $\Sigma$ . Throughout this paper,  $T$  is temperature, and we use  $\beta = 1/(RT)$  for brevity, where  $R$  is the molar gas constant. Free energy always refers to Gibbs free energy.

► **Definition 1** (Tile type, Blocker). A tile type  $t = (l, E)$ , or simply tile, is a square with a string label  $l \in \Sigma^*$  over alphabet  $\Sigma$ , and 4 unit length edges each of which is a pair  $e = (d, g) \in E$  where  $d \in \{N, E, S, W\}$  is a direction and  $g \in \Sigma^*$  is a glue name/type. Each glue  $g$  has a complement, denoted  $g^*$ . A blocker  $\mathbf{b}$  is a pair of the form  $\mathbf{b} = (d, g^*)$ . No two blockers have complementary glues.

► **Definition 2** (kBlock instance). An instance  $\mathcal{B}$  of the kBlock model consists of a set of tiles  $\mathcal{T}$  and blockers  $\mathcal{B}$ , each  $t \in \mathcal{T}$  and  $\mathbf{b} \in \mathcal{B}$  having an associated non-negative real-valued *total species concentration* respectively denoted  $c_t, c_{\mathbf{b}} \in \mathbb{R}^+$ . The system has a function  $\Delta G_{\text{bond}} : \Sigma^* \times \mathbb{R}^+ \rightarrow \mathbb{R}$  such that  $\Delta G_{\text{bond}}(g, T)$  is the free energy of glue  $g$  bound to its complement  $g^*$  at temperature  $T$ .

► **Definition 3** (Tile state). A tile state is an instance (copy) of a tile type  $t$ , with 0 to 4 blockers, each bound to an edge. A blocker  $\mathbf{b} = (d, g)$  may bind only to edge  $e = (d, g^*)$ , i.e. if directions match and glues are complementary. We write  $t\mathbf{b}$  to denote a tile-blocker bonded complex. The *available edges* of a tile state do not have a blocker bound to them.

### 2.1.1 Tile states and blockers in solution bind together at equilibrium concentrations

In the remainder of this section we define the kBlock model as having two parts: (1) free blockers and tile states *in solution* at equilibrium concentrations as a function of their initial concentration and temperature, remaining unchanged during assembly, and (2) an assembly that grows by attachments of tile states and blockers via stochastic interactions; the tile states and blockers on an assembly are not required to be at solution equilibrium concentrations.

Hence, the kBlock model is “powered” [22]: individual events do not change blocker or tile state concentrations. That is, we take the limit of large volume so that removal or addition of individual blockers or tile states does not change their concentrations. Depletion effects can be approximated by changing the concentration parameters in the model.

► **Definition 4** (Tile states in solution). The *solution* is a set of pairs of  $(s, c)$  where  $s$  is a blocker or tile state and  $c \in \mathbb{R}^+$  is its concentration. For an edge of a tile type  $e$  and a temperature  $T$ , we let  $p_a(T, e) \in [0, 1] \subseteq \mathbb{R}$ , the *available edge probability*, be the probability that the edge is available (unblocked) in solution at equilibrium. We simply write  $p_a$  when  $T$  and  $e$  are clear from the context.



Although we defined notation for probabilities and concentrations, we have not yet given them explicit values. In Remark 6 those come from chemical equilibrium, and a lemma:

► **Lemma 5.** *The concentration of unbound tile states of a tile type  $t$  with a particular edge available is  $p_a c_t$ , and the concentration of unbound blocker instances of a blocker  $\mathbf{b}$  is  $[\mathbf{b}] = c_b - p_a C_t$ , where  $C_t = \sum_{t'} n_{t', \mathbf{b}} c_{t'}$  is the sum of the concentrations of all tiles with  $n_{t', \mathbf{b}} > 0$  glues complementary to the blocker  $\mathbf{b}$ .*

**Proof.** By Definitions 2 and 4 (initial tile concentration  $c_t$  and available edge probability  $p_a$ ), we get the first conclusion ( $p_a c_t$ ). The second comes from conservation of mass. ◀

► **Remark 6 (Equilibrium concentrations of tiles and blockers).** Practically,  $p_a(T, e)$ , written here as a function of temperature  $T$  and edge identity  $e$ , can be determined by the equilibrium concentrations of tiles and blockers in isolation in solution, ignoring assemblies. While the system may have many tile types and blockers, reactions involving each blocker are independent and can be considered in isolation. By standard chemical reaction thermodynamics in solution [14, 3], we define an abstract reaction coordinate  $\xi_{i, \mathbf{b}}$  for each reaction  $t_i + \mathbf{b} \rightleftharpoons t_i \mathbf{b}$ . Then, the change in free energy  $G$  with each reaction is

$$\frac{dG}{d\xi_{i, \mathbf{b}}} = (RT \ln[t_i \mathbf{b}] - RT \ln[t_i] - RT \ln[\mathbf{b}] + \Delta G_{\text{bond}}(T)) \quad (1)$$

$G$  has a minimum where, for each tile-blocker pair,  $\frac{dG}{d\xi_{i, \mathbf{b}}} = 0$  [3], implying equilibrium concentrations  $[t_i \mathbf{b}] = [t_i][\mathbf{b}]e^{-\beta \Delta G_{\text{bond}}(T)}$ . By conservation of mass  $[t_i] + [t_i \mathbf{b}] = c_t$ , hence

$$[t_i] = \frac{c_t}{1 + [\mathbf{b}]e^{-\beta \Delta G_{\text{bond}}(T)}} \quad p_a = \frac{1}{1 + [\mathbf{b}]e^{-\beta \Delta G_{\text{bond}}(T)}} \quad (2)$$

We claim that  $p_a$  is a probability, in other words that  $p_a \in [0, 1] \subset \mathbb{R}$  as follows: without blockers  $p_a = 1$  for all temperatures  $T$  and  $\Delta G_{\text{bond}}$  since  $0 = [\mathbf{b}] \leq c_b$ , otherwise  $0 < p_a < 1$  for  $[\mathbf{b}] > 0$  and all  $T$  and  $\Delta G_{\text{bond}}$ . Furthermore, the expression for  $p_a$  is in terms of  $[\mathbf{b}]$ , the concentration of unbound blocker instances, rather than the total species concentration  $c_b$  of the blocker  $\mathbf{b}$ . In the limit where total blocker concentrations are large multiples of total tile concentrations, the concentration  $[\mathbf{b}]$  is close to  $c_b$ , and where noted we use that approximation:  $[\mathbf{b}] = c_b$ . But more precisely, for all concentrations, the combination of conservation of mass constraints and equilibrium conditions results in the analytic solution

$$[\mathbf{b}] = \frac{1}{2} \left( c_b - C_t - e^{\beta \Delta G_{\text{bond}}(T)} + \sqrt{(C_t - c_b + e^{\beta \Delta G_{\text{bond}}(T)})^2 + 4c_b e^{\beta \Delta G_{\text{bond}}(T)}} \right) \quad (3)$$

This has the effect, at low blocker concentrations relative to  $C_t$ , of increasing  $p_a$ . However, while our theoretical analyses and simulations use the precise value of  $[\mathbf{b}]$  in Equation (3) to calculate  $p_a$ , in this and related experimental work [17, 18] we focus on high concentration blockers, where depletion of unbound blockers is not significant, and the aforementioned  $[\mathbf{b}] = c_b$  approximation is sufficient. Figure 2(left) shows how close the approximation is.

## 2.1.2 Assembly reactions are stochastic

► **Definition 7 (Assembly).** An assembly of a kBlock system  $\mathcal{B}$  is a function  $\alpha : \mathbb{Z}^2 \rightarrow \mathcal{T}$  where the domain of  $\alpha$  is connected in  $\mathbb{Z}^2$  and where  $\mathcal{T}$  is the set of all tile states of  $\mathcal{B}$ . Intuitively, an assembly is a connected lattice of tile states.

► **Definition 8** (kBlock dynamics and rates). A single step  $\alpha \rightarrow_1 \alpha'$  in the kBlock model maps one assembly  $\alpha$  to another assembly  $\alpha'$  where both differ by the attachment or detachment of a single tile state or blocker governed by four rate equations, illustrated in Figure 1:

$$r_{t,\text{att}} = k_f[t] \quad (4)$$

$$r_{t,\text{det}} = k_f e^{b\beta\Delta G_{\text{bond}}(T) - (b-1)R\Delta S_{\text{lat}}} \quad (5)$$

$$r_{b,\text{att}} = k_f[b] = k_f(c_b - p_a C_t) \quad (6)$$

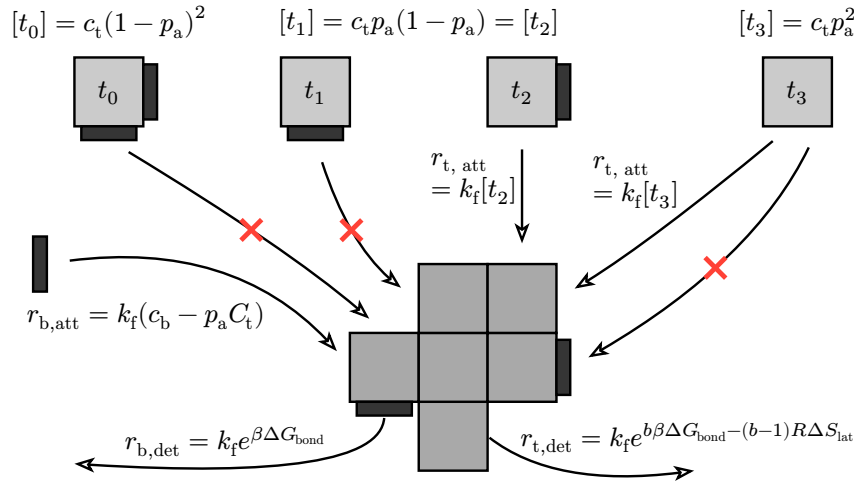
$$r_{b,\text{det}} = k_f e^{\beta\Delta G_{\text{bond}}(T)} \quad (7)$$

These are applied and explained as follows:

1. Tile state attachment ( $r_{t,\text{att}}$ ): at every empty coordinate in  $\mathbb{Z}^2$  adjoining at least one tile state in  $\alpha$ , each tile state in solution may attach at the coordinate if (1) the tile state has at least one glue complementary to the glue on the adjoining tile state, (2) the tile state would not be adjacent to any edge of any tile state in  $\alpha$  with a bound blocker, and (3) no edge of the tile state with a bound blocker would be adjacent to another tile state in  $\alpha$ . The rate of attachment per tile state  $t$  satisfying the criteria is  $r_{t,\text{att}} = k_f[t]$ , where  $[t]$  is the concentration of the tile state satisfying the criteria.
2. Tile state detachment ( $r_{t,\text{det}}$ ): any tile state in  $\alpha$  may detach, if detaching does not result in the assembly becoming disconnected, at a rate of  $r_{t,\text{det}} = k_f e^{b\beta\Delta G_{\text{bond}}(T) - (b-1)R\Delta S_{\text{lat}}}$ , where  $b$  is the number of bonds between the tile state and adjoining tiles in  $\alpha$ .  $\Delta S_{\text{lat}}$  is defined as an entropic penalty for the formation of multiple bonds.
3. Blocker attachment ( $r_{b,\text{att}}$ ): at every edge of every tile state in an assembly where no blocker is bound, there is no adjacent tile, and the system has a blocker complementary to the edge's glue, that blocker may attach to the edge with a rate of  $k_f[b] = k_f(c_b - p_a C_t)$ , where  $C_t$  is the sum of the total species concentrations of all tile types with a glue complementary to the blocker (Lemma 5).
4. Blocker detachment ( $r_{b,\text{det}}$ ): at every edge of every tile state in an assembly that has a bound blocker, the blocker may detach from the edge with a rate of  $k_f e^{\beta\Delta G_{\text{bond}}(T)}$ .

► **Definition 9** (Seed). The assembly  $\alpha$  of a kBlock system prior to any step being taken is defined as the seed of the system. This serves as a starting point from which the system may grow. Growth is said to be unseeded if the seed is empty (i.e.  $\text{dom}(\alpha) = \emptyset$ ).

► **Remark 10** (Why do we disallow attachment of partially blocked tiles?). These event definitions include the choice of disallowing tile state attachment if *any* blockers interfere. Including these attachments would seem more physically realistic. However, as the model treats tile and blocker attachments and detachments as elementary steps, rather than modeling individual bond formation and breakage, allowing partially blocked attachments would violate detailed balance. Much of the theoretical analysis of the kBlock model can be adapted to accommodate the extra events. However, we found that the temperature-dependent behaviour of the kBlock model fit experimental observations significantly better without partially blocked attachments. Although surprising, physically-plausible mechanisms could result in partially-blocked attachments being unlikely to result in stable tile attachment and thus make the model without partially blocked attachments more accurate. For example, there could be a small difference in blocker and single-bond tile free energies that would make blocker detachment less likely than tile detachment, and there could be a delay to forming an additional bond that would make a tile attached by a single bond more likely to detach before forming a second bond to a newly-available adjacent glue. We explore the two model choices further in Section B.



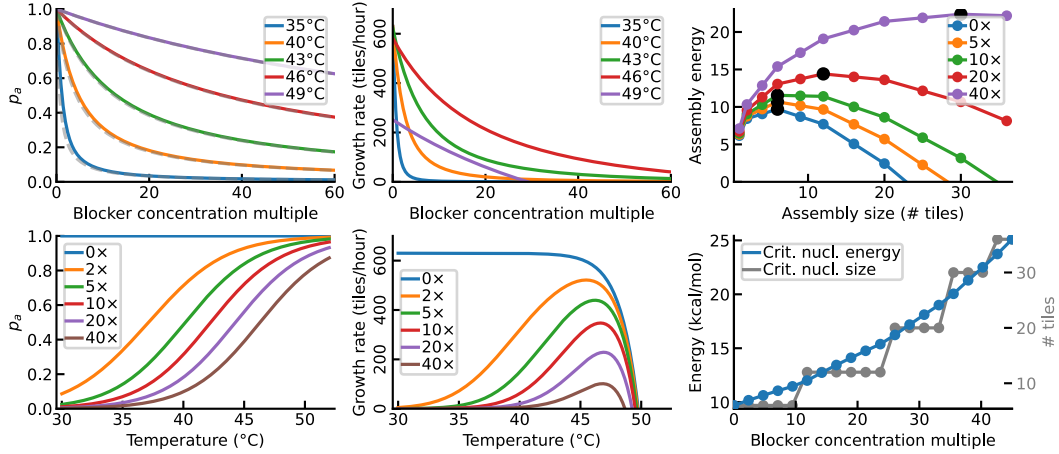
■ **Figure 1** Summary of the kBlock model. **Top:** Blockers and tile states bind in solution and are held at a temperature-dependent equilibrium. **Centre:** Blockers and tile states stochastically attach to, or detach from, a dynamically growing/shrinking assembly (centre). Blockers may attach to any available matching glue on a tile on the assembly, while tiles can attach to any empty position where the tile can make at least one bond, and where there is no blocker between tiles; both blockers and tiles attach with the same forward rate constant, simply dependent on concentration. **Bottom:** Tile states and blockers detach based on bond free energy.

We assume that the lattice penalty  $\Delta S_{\text{lat}}$ , used above, is entropic and proportional to the number of bonds made beyond one. This penalty was calculated in [10] as  $\alpha = -7.12$ , where  $\alpha$  is a parameter in the unitless formulation of the kTAM (Section A), based on [11, 12]. This parameter is equivalent in our formulation to  $-14.12 \text{ cal mol}^{-1} \text{ K}^{-1}$  for both SSTs and DX tiles; we do not expect that core tiles will significantly differ. The model as defined above excludes any detachment that would disconnect the assembly, as this would violate detailed balance; in practice, as with the kTAM, implementations may choose to include “fission” events, for example, by discarding one of the disconnected regions. While the model includes blocker attachment and detachment events on assemblies, for a given edge on an assembly that is not adjacent to another tile state, the blocker attachment and detachment rates result in the probability of that edge being available being  $r_{\text{b,det}}/(r_{\text{b,det}} + r_{\text{b,att}}) = 1/(1 + [\text{b}]e^{-\beta \Delta G_{\text{bond}}(T)})$ , which is the same as  $p_a$  in Equation (2). This probability is useful in the theoretical analysis of Section 3.

## 2.2 kBlock<sub>2</sub>: a simplified theoretical model for mathematical results

The kBlock and kTAM [24, 9] models are used for our computer-simulation-based results (Section 4). However, for mathematically derived results on growth rates in Section 3, we use two simpler models, called the kTAM<sub>2</sub> and kBlock<sub>2</sub>, which we now justify.

Most tile systems of experimental interest have intended growth pathways primarily using two-bond attachments, and are grown at temperatures where two-bond attachments are slightly favourable. In the limit of low concentrations and slow growth slightly below the two-bond melting temperature where attachment and detachment by two bonds are equally favourable, one-bond attachments detach much faster than they attach (and are mainly of relevance to erroneous growth pathways), while three- and four-bond detachments are slow enough to be insignificant. Thus, we view two-bond tile attachment and detachment events as being most significant [9]. For ease of analysis, we define models constrained to two-bond attachment and a few other reasonable simplifications:



■ **Figure 2** Plots derived from theoretical results in Sections 2.1.1 and 3. **Left:** The probability  $p_a$  of a tile side being unblocked, Equation (2), decreases with increasing blocker concentration and decreasing temperature. Solid curves: blocker concentration  $[b]$  from Equation (3). Dashed: high-blocker-concentration approximation  $[b] = c_b$ , Remark 6. **Middle:** Two-bond growth rates (Section 3.1) increasing blocker concentration suppresses growth at low temperatures, and increasing detachment rates prohibits growth at high temperatures, creating a growth temperature window (bottom). **Right:** Nucleation (Section 3.2): increasing blocker concentrations increases assembly energies. Nucleation pathways go through a sequence of assemblies of increasing size and decreasing favourability, until a critical nucleus size is surpassed wherein growth is mostly favourable. Plots: critical nucleus size and free energy (unfavourability) increases with blocker concentration, with black dots in the top plot being critical nuclei.

► **Definition 11** ( $kTAM_2$ ). The  $kTAM_2$  model is the  $kTAM$  [24, 9] restricted to only allow tile attachments by two bonds and detachments by two bonds, to have all bond strengths to be equal, and to have all tile concentrations to be equal ( $c_t$ ).

► **Definition 12** ( $kBlock_2$ ). The  $kBlock_2$  model is the  $kBlock$  restricted so that: (1) each tile type has at most two blockers, each for a distinct edge of the tile, and these blockers attach only to that tile; (2) every tile state attachment involves two bonds, each having either a blocker in the system for the glue on the attaching tile, or a blocker in the system for the glue on the adjoining tile; (3) all bond strengths are equal; and (4) all total tile concentrations are equal ( $c_t$ ) and all total blocker concentrations are equal ( $c_b$ ).

### 3 Theoretical analysis

In this section we prove some results for growth in the restricted  $kBlock_2$  model (Definition 12), and nucleation in the  $kBlock_2$  generalised to allow 1- and 2-bond attachments.

#### 3.1 Growth rate in the $kBlock_2$ model

Theoretically modelling growth rates is challenging due to assemblies changing frontier size over time. Indeed for algorithmic systems it can be provably algorithmically hard, or even undecidable, to predict whether a single tile is placed at some location [16, 4]. Hence, we consider an intentionally oversimplified growth rate in our simplified kinetics models:

► **Definition 13** (Two-bond growth rate). *In the  $kTAM_2$  and  $kBlock_2$ , the two-bond growth rate is the average rate an assembly increases in size, via both attachments and detachments.*

In the  $\text{kTAM}_2$ , the two-bond growth rate for a system with an assembly growth front, or frontier, constant in size over time is  $r_2 = \gamma \cdot (r_{\text{t,att}} - r_{\text{t,det}})$ , where  $\gamma$  is a positive constant factor dependent on the shape of growth front [24, 9]. For example,  $\gamma = 1$  for a “zig-zag” system [22] with frontier size 1. For frontier sizes that change over time,  $\gamma$  is a more complicated function, something we do not wish to explicitly handle here. In this subsection we seek a 2-bond growth rate result for the  $\text{kBlock}_2$  model, in terms of the  $\text{kTAM}_2$ , and leaving  $\gamma$  as an undefined system-dependent function.

► **Lemma 14.** *For any  $\text{kBlock}_2$  instance  $\mathcal{B}$ , at fixed temperature  $T$ , the available edge probability  $p_a$  (Equation (2)) is identical for every tile edge.*

**Proof.** By Equation (2),  $p_a$  is a function of (1) temperature – which is fixed, (2) glue free energy  $\Delta G_{\text{bond}}(T)$  – which is equal for all glues. and (3) blocker concentration  $[\text{b}]$  – which is equal for all blockers in the system by analysis of Equation (3). ◀

► **Theorem 15** (Each  $\text{kBlock}_2$  system has a corresponding  $\text{kTAM}_2$  system). *Let  $\mathcal{B}$  be a  $\text{kBlock}_2$  system with tile concentration  $c_{\mathcal{B}}$ , and let  $p_a$  be  $\mathcal{B}$ ’s available edge probability (a single value for all edges by Lemma 14). There is a  $\text{kTAM}_2$  system  $\mathcal{K}$  with the same tile types, seed and tile attachment/detachment rate constants as  $\mathcal{B}$ , with tile concentration per tile type  $c_{\mathcal{K}} = p_a^2 c_{\mathcal{B}}$  such that both systems have the same 2-bond growth rate. We call  $c_{\mathcal{K}}$  the “effective tile concentration”.*

**Proof.** By hypothesis, tile detachment rates for the  $\text{kBlock}_2$  and  $\text{kTAM}_2$  systems are the same. Tile attachments, however, will be influenced by blockers. In the  $\text{kTAM}_2$  system, a potential tile attachment has rate  $k_{\text{f}}c_{\mathcal{B}}$ , where we use notation  $c_{\mathcal{B}}$  or  $c_{\mathcal{K}}$ , depending on the system, instead of the usual  $c_{\text{t}}$ .

In the  $\text{kBlock}_2$  system, for the same assembly position, the same attachment will require that no blockers are bound to the edges between the tile and adjoining tiles in the assembly. In the  $\text{kBlock}_2$  model, there are exactly two blockers involved, and each may be complementary to either a glue on the attaching tile state, or a glue on a tile state in the assembly. If the blocker is complementary to a glue on the tile, then there is a  $p_a$  probability of a tile state that has that edge unbound, by the definition of  $p_a$  in Equation (2). If the blocker is complementary to a glue on the assembly, there is also probability  $p_a$  that the edge will be available at any given moment since the attachment and detachment rates satisfy the same equilibrium concentrations of blocker/no blocker for a single site.

As neither blocker can be present for attachment to be possible, there is a  $p_a^2$  probability that, for a randomly chosen tile state from solution at a random moment, the attachment will be possible; this results in an attachment rate of  $r_{\text{f},\mathcal{B}} = p_a^2 \cdot k_{\text{f}}c_{\mathcal{B}}$ . The expression for the two-bond growth rate for a  $\text{kTAM}_2$  or  $\text{kBlock}_2$  system is of the same form  $r_2 = \gamma(r_{\text{f}} - r_{\text{r},2})$ : Since, by hypothesis, both systems have identical detachment rates  $r_{\text{r},2}$ , the 2-bond growth rate  $r_2$  will be equal for both systems if the attachment rates  $r_{\text{f}}$  are equal for both systems. Setting the  $\text{kTAM}_2$  tile concentration to  $c_{\mathcal{K}} = p_a^2 c_{\mathcal{B}}$  gives exactly that since for  $\mathcal{K}$ , the attachment rate is  $r_{\text{f},\mathcal{K}} = k_{\text{f}}c_{\mathcal{K}} = k_{\text{f}}p_a^2 c_{\mathcal{B}} = r_{\text{f},\mathcal{B}}$ . ◀

► **Remark 16** (Blocked growth at low temperature). A corollary of the previous theorem is that blockers completely kill low-temperature growth, which can be seen as follows. From Equation (2), the probability  $p_a$  that a side is available is inversely exponentially dependent on temperature. Higher blocker concentration also drives this probability down, albeit linearly. Hence, in the systems studied here, at low temperatures and high blocker concentration,  $p_a$  will tend to zero, and thus the 2-bond growth rate will also tend to zero (Figure 2 (middle-top)). By the same reasoning, and since every bond or its complement has a blocker,

the attachment rate for 1-bond growth,  $k_f p_a c_B$ , will always be less than the detachment rate,  $k_f e^{\beta \Delta G_{\text{bond}}(T)}$ , so long as the unbound blocker concentration [6] is greater than the tile concentration. In contrast, in regular unblocked systems growth simply happens below the melting temperature, and at low temperatures suffer from unintended structures likely created from 1-bond growth and nucleation [26] (Suppl. Info. A, Sect. S5.1).

► **Remark 17 (Growth window).** There is a temperature growth window: above the melting temperature the growth rate is less than zero, and, by Remark 16, at low temperatures the 2-bond growth rate tends to zero. In between these extremes, there is positive growth, as shown in Figure 2 (middle-bottom). The width of the 2-bond growth window is blocker concentration dependent, with more blockers resulting in a narrower window.

► **Remark 18 (Buffering).** Blockers also result in a “tile-buffering” effect [20], causing changes to the growth rate to be slower, as tiles are incorporated into assemblies, than the same system without blockers. Or, equivalently, as tiles are used, the effective tile concentration  $c_{\text{eff}}$  (defined in the statement of Theorem 15) changes more slowly than the concentration of tiles used. While the kBlock model does not account for concentration changes from tile attachment and detachment events, the effect of a certain concentration of tiles being incorporated into assemblies, and thus unavailable for attachment, can be examined by changing the concentrations in the model. Defining  $c_{\text{used}}$  as the concentration of tiles incorporated into assemblies at some point during growth, so remaining tile concentration is  $c_t = c_t^{(\text{initial})} - c_{\text{used}}$ , and from Section 3.1 letting  $r_2$  be the 2-bond growth rate, it is easily seen that  $dr_2/dc_{\text{used}} = -\gamma k_f p_a$ , and  $dc_{\text{eff}}/dc_{\text{used}} = -p_a$ : if a particular concentration  $c_{\text{used}}$  of tiles is used, the effective concentration available for attachment is reduced only  $p_a c_{\text{used}}$ , rather than  $c_{\text{used}}$  as it would be without blockers. Practically, this means that, if a particular growth rate is desired, a system with blockers can use a larger total tile concentration, and incorporate more tiles with less effects from depletion. See Figure 4.

### 3.2 Nucleation rate in the kBlock model

Here, spontaneous nucleation from solution, or simply nucleation, refers to the self-assembly process whereby assemblies form in slightly supersaturated conditions, in other words just below the 2-bond crystal melting temperature. Nucleation rates are difficult to model due to multiple simultaneous pathways and geometric concerns. Schulman and Winfree [22] applied classical nucleation theory to derive a mathematical upper bound on the nucleation rate in DNA tile self-assembly. Close to the crystal melting temperature, nucleation pathways are unfavourable until some large enough assembly called a critical nucleus has assembled, from which growth is favourable. Their approach considers the assemblies visited along nucleation pathways to critical nuclei, observing that the most favourable of these, including any critical nucleus, can be used to derive an upperbound on the nucleation rate. The proof of the following theorem uses that upperbound [22]. The theorem is stated for a generalisation of the kBlock<sub>2</sub> that allows both 1- and 2-bond attachments (because nucleation is impossible if there are no 1-bond attachments), but more restricted than the kBlock to avoid complications of arbitrary combinations of blockers and concentrations. Also, using ideas from [22] we consider only the critical nucleus that is the most stable of all the critical nuclei, and that growth from it is favourable (by two-bond attachments).

► **Theorem 19.** *For a system in the kBlock with a critical nucleus of  $N$  tiles and  $B$  bonds, where every tile has exactly two blockers, attachments by one bond are not favourable, and every glue on the critical nucleus perimeter is required for continued growth, the classical nucleation theory upper bound [22] on the nucleation rate through that critical nucleus is reduced by  $(p_a)^{2(N+1)}$  compared to the same system without blockers.*



**Proof.** We first define the free energy of an assembly  $\alpha$ , in the kTAM [22]:

$$G(\alpha) = B\Delta G_{\text{bond}}(T) - T(B - N)\Delta S_{\text{lat}} - RT \sum_{t \in \alpha} \ln[t] \quad (8)$$

where  $N = |\alpha|$  is the total number of tiles in assembly  $\alpha$ ,  $B$  is the total number of bonds in  $\alpha$ , and each  $t$  is an instance (copy) of a tile in  $\alpha$ , i.e. the sum has  $N$  terms.

Next, in Equation (9), we define the assembly free energy of a critical nucleus in the kBlock model (the version assumed by the theorem hypotheses) which is justified as follows. In the kBlock and kTAM, energetic contributions of bonds in the assembly are equal. However, in the kBlock, the tile concentration of tiles available to bind to an assembly is  $p_a^2 c_t$ : by assumption, each tile (with, as usual, initial unblocked concentration  $c_t$  and available edge probability  $p_a$ ) has two blockers that bind to it and hence the completely unblocked tile state has concentration  $p_a^2 c_t$  through the entire assembly process, which we substitute for  $[t]$  in Equation (8), noting that the sum of logarithms in the assembly free energy  $G(\alpha)$  allows these multiplicative terms to be separated in Equation (9). Also, in Equation (9) we ignore free energy contribution from blockers bound to the perimeter of the assembly, as by hypothesis only tile states without blocked edges on the perimeter of the assembly will allow for continued growth.

$$G(\alpha) = B\Delta G_{\text{bond}}(T) - T(B - N)\Delta S_{\text{lat}} - RTN \ln c_t - 2RTN \ln p_a \quad (9)$$

Given a particular critical nucleus  $\alpha$ , as shown for the kTAM in similar conditions [22], we assume  $[\alpha]$  at any time is bounded by the equilibrium concentration  $[\alpha]_{\text{eq}} = \exp(-\beta G(\alpha))$ . Thus, the nucleation rate through a pathway containing that critical nucleus is bounded from above by  $r_{\text{cn}} = Fr_f \exp(-\beta G(\alpha))$ , where  $r_f$  is the tile attachment rate for two-bond attachments and  $F$  is the number of two-bond attachment sites: in other words, the flux through the critical nucleus, ignoring reverse reactions and one-bond attachments (which, being unfavourable, would mean the  $\alpha$  would not be the critical nucleus). Since two-bond attachments to the critical nucleus must involve two unblocked edges, the attachment rate is  $k_f p_a^2 c_t$ . The kBlock assembly free energy differs from the kTAM assembly free energy by  $-2RTN \ln p_a$ ; this results in a change to the nucleation rate, via the  $\exp(-\beta G(\alpha))$  term, of  $\exp(-2\beta RTN \ln p_a) = p_a^{-2N}$ . When combined with the additional  $p_a^2$  from  $r_f$ , this results in  $r_{\text{cn}}$  being multiplied by  $p_a^{2(N+1)}$  compared to the kTAM. ◀

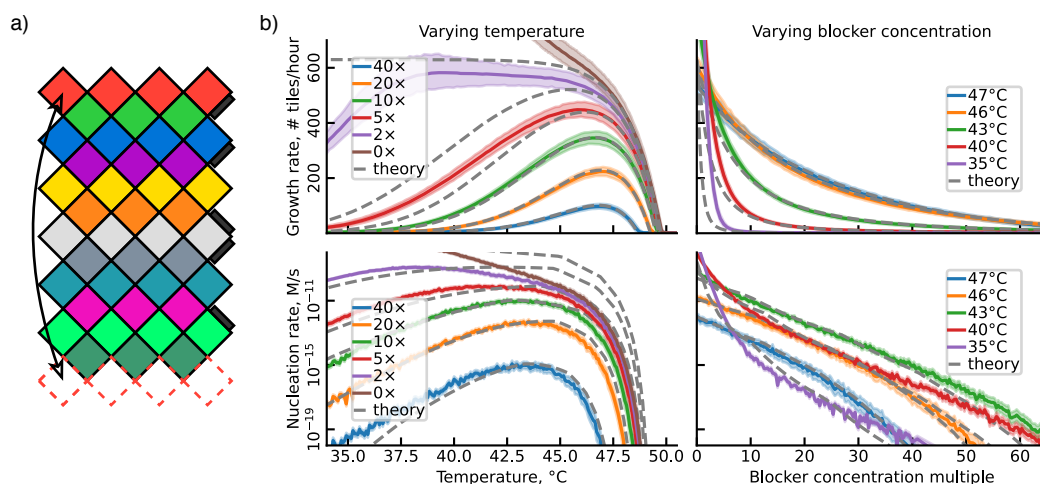
It follows from the previous theorem and the definition of  $p_a$  that the nucleation rate, like the growth rate, will tend towards zero as temperature decreases when the blocker concentration is high, resulting in a temperature window where unseeded nucleation occurs.

## 4 Simulations and comparisons with experimental results

As an extension of the kTAM, the kBlock model is amenable to similar simulation techniques. We implemented the model as a Gillespie algorithm simulation in the Rgrow tile assembly simulator [8]. When sequences for glues are provided, Rgrow calculates the free energy for the glues based on the nearest-neighbor model using parameters from [19]; these were used for comparisons with experimentally-implemented systems. We consistently used a value of  $\Delta S_{\text{lat}} = -14.12 \text{ cal mol}^{-1} \text{ K}^{-1}$  for all simulations [10].

Experiments with blockers in Rogers et al. focused on systems forming periodic nanotubes, uniquely-addressed around the tube to force a minimum circumference of 12 helices, and using a DNA origami seed for seeded growth [17, 18]. Seeded tubes are particularly simple





**Figure 3** (a) A simple 12-tile tube system with blockers. (b) Plots showing both simulation (solid curves) and theory (dashed) for growth and nucleation rates: on the left, systems with different blocker concentrations as a function of temperature, and on the right, systems at different fixed temperatures as a function of blocker concentration. Growth rate simulations used 1,024 simulated seeded assemblies, each grown until reaching 1,200 tiles or until 10 hours of simulated time; shaded regions show 90% range of growth rates. Nucleation rate simulations used Rgrow's FFS implementation for flat assemblies.

to analyse: growth can proceed entirely by pathways with two-bond attachments, and the growth front for two-bond attachments remains approximately of constant size. Rgrow can be set to flat or tube simulations; in simulations set for tube assemblies, rgrow treats the tube topology and circumference as a fixed property of the assembly, not allowing for tubes of different circumference, or a 2D lattice. Regardless of its topology, the assembly begins from an initial seed assembly which can be specified at the start of the simulation.

#### 4.1 Simulations, and comparison with theory

For comparisons between kBlock simulations and our theoretical analysis, we used a simple 12-tile system forming a periodic 12-helix tube (Figure 3(a)). Every glue was defined to have an identical strength, with  $\Delta G(37^\circ\text{C}) = -10.15 \text{ kcal mol}^{-1}$  and  $\Delta S = 193.9 \text{ cal mol}^{-1} \text{ K}^{-1}$  (from the representative sequence TGTCTGTGCA). Growth rates for tubes used no adjustable parameter other than the growth frontier size parameter  $\gamma = 3.5$  (see Section 3.1), set for a 12-helix tube with 0 to 6 two-bond attachment and detachment sites during kBlock<sub>2</sub> growth.

Nucleation rate estimates in simulations used the forward-flux-sampling (FFS) implementation in Rgrow [7, 1]. Application of the theoretical nucleation rate analysis, however, required an estimate of the critical nuclei. To simplify the analysis, nucleation simulations used a flat plane, rather than tube, effectively making the system a 2D periodic lattice. The critical nucleus size was then estimated by taking the maximum critical nucleus assembly free energy along a pathway of increasing  $k \times k$  and  $k \times (k + 1)$  rectangular assemblies. Additionally, classical nucleation theory as applied to tile assembly [22] primarily calculates an upper bound, often orders of magnitude higher than observed nucleation rates (either in simulation or experiments), and is primarily used to consider scaling of nucleation rates when varying parameters. To consider this scaling, for Figure 3, we scaled the theoretical rate to be equal to the simulation rate at  $46^\circ\text{C}$  when varying temperature, and to the simulation rate at  $20\times$  blocker concentration when varying blocker concentration.

**Simulations: growth**

Figure 3 shows simulation results with varying temperature and blocker concentration. In general, simulations align with theory near the two-bond melting point. Unlike the simulations, the theoretical growth rate (dashed curve) assumes that growth occurs only by two-bond attachments, leveraging the assumption that one-bond attachments are too unfavourable to be significant. As temperature decreases, and bond strength thus increases, one-bond attachments may become favourable, with different kinetics and the possibility of a changing growth front size; this would be expected to increase the observed growth rate compared to the theoretical two-bond rate, as seen in simulations at low blocker concentrations. However, at sufficiently high blocker concentrations, even at low temperatures, simulated growth rate remains near the two-bond rate, suggesting that blockers may reduce one-bond attachment pathways, and potentially disordered growth, at low temperatures. For blocker concentrations above  $5\times$ , where theory and simulation agree well, as blocker concentrations increase, the growth rate decreases, but slowly at optimal growth temperatures. Both the maximum temperature with a positive growth rate, and the temperature of maximum growth rate, decrease, but only very slightly with increasing blocker concentration, going from  $46.0^\circ\text{C}$  at  $5\times$  blocker concentration to  $46.8^\circ\text{C}$  at  $40\times$  blocker concentration. Growth occurs only in an increasingly narrow temperature range, going from a full-width-half-max of  $8.6^\circ\text{C}$  at  $5\times$  blocker concentration to  $4.1^\circ\text{C}$  at  $40\times$  blocker concentration.

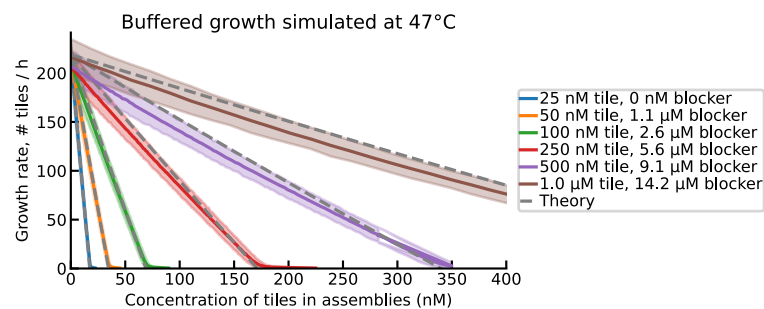
**Simulations: nucleation**

For nucleation, the scaling of nucleation rates at high temperatures, varying blocker concentration, largely fits the theoretical analysis, particularly when moving closer to the melting temperature. At lower temperatures and low blocker concentrations, however, the scaling of the FFS-calculated nucleation rate diverges significantly from the theoretical estimate. It is likely that in these conditions, our simple assumption of an increasing-size-rectangle pathway breaks down, as single-bond attachments may not be significantly unfavourable, or may be even be favourable (eg, below  $36.0^\circ\text{C}$  for  $0\times$  blocker concentration). It is also possible that in these conditions, the classical nucleation theory assumption of nucleation rates being constrained by a maximally-unfavourable critical nucleus may break down.

**Simulations: tile buffering**

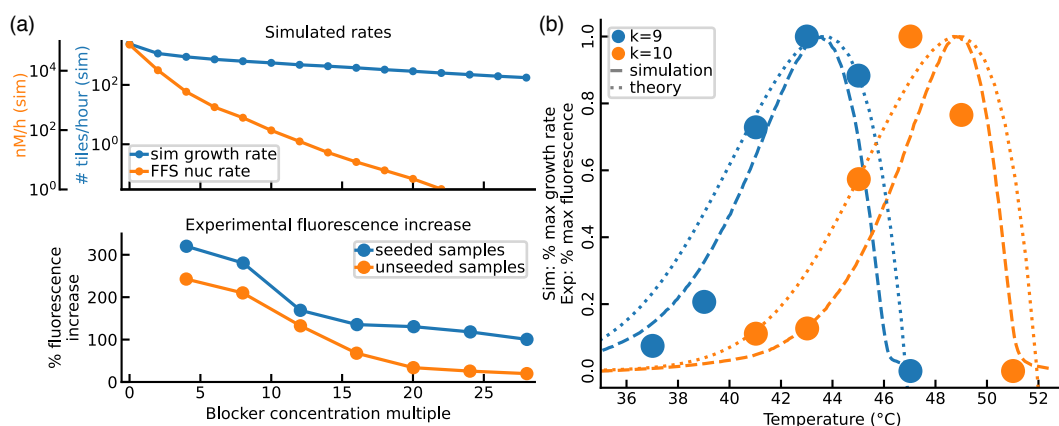
As predicted in Remark 18, simulations show a “buffering” effect, where for a particular target initial growth rate, systems with blockers decrease in growth rate more gradually than a system without blockers that starts with a similar growth rate: while a system in the kBlock model will have the growth rate of an equivalent kTAM system with effective tile concentration  $p_a c_t$ , as tiles are incorporated into assemblies, and  $c_t$  decreases, the reduction in growth rate will be slower if  $p_a < 1$ . Practically, for high blocker concentrations, where  $p_a$  does not depend on  $c_t$ , this gives a constant multiplicative reduction in the change in growth rate, depending on  $p_a$ , which can be influenced by temperature and blocker concentration.

Figure 4 gives an example of how this effect might be applied. Starting from a base 25 nM tile concentration, 12-tile-tube system without blockers, at a fixed temperature, systems with higher tile concentrations can be made to matching initial growth rate by adding a calculated concentration of blockers. Though they grow similarly at first, these systems will continue to grow even when significantly more tiles have been incorporated into assemblies.



**Figure 4** The growth rates of a simple twelve-tile tube system under varying tile and blocker concentrations, representing behaviour when free tiles have been depleted by growth. Solid lines represent simulated results from 1,024 simulated assemblies grown to a size of 1,200 tiles or until 10 hours of simulated time elapsed (shaded regions show 90% range of growth rates), while dashed curves indicate theoretical predictions. All simulations were seeded, with initial blocker concentrations adjusted such that all systems have the same initial growth rate.

## 4.2 Comparisons between simulation and experimental implementations



**Figure 5** (a) Simulated and experimental results from a twelve-helix tube system. Simulated results show growth rates for a single assembly and spontaneous nucleation rates, without considering depletion. Experimental results measure total growth of assemblies, both from seeds and through spontaneous nucleation, over 96 hours. (b) Simulated, theory and experimental seeded growth results from multiple systems with different glue strengths. Dots show increase in fluorescence (experiment). Dashed line shows mean simulated growth rate from 1,024 simulated assemblies. Dotted line shows theoretical growth rate, assuming all bonds have strength equal to the mean bond strength of the simulated system. Curves are scaled from minimum (or zero, for the theoretical growth rate) to maximum values within the plotted temperature range.

In the first series of experiments, Rogers et al. [17, 18] used a periodic 12-helix DNA tile nanotube system similar to the system used for comparisons between simulations and theoretical values, though for comparison, simulations use sequence-dependent nearest-neighbour energies to reflect the experimental system. Both seeded and spontaneously nucleated tubes were grown. Seeds were purpose-designed 12-helix DNA origami barrels. Tubes were grown in a series of temperature holds, using a qPCR machine repurposed to measure fluorescence under a programmable temperature protocol. The experimental implementations measured growth by a fluorescent reporter mechanism that periodically incorporated Forster resonance energy transfer (FRET) pairs into growing tubes, resulting

in an increase in bulk fluorescence corresponding to tube growth. Experimental data is in Figure 5. Each data point in Figure 5(a) is the maximum, over 6 temperatures (43–53°C in 2°C increments), of the measured fluorescence increase after a 96-hour temperature hold [17]. Figure 5(b) shows measured fluorescence increase after a 48-hour temperature hold from [18].

### Growth rate: Simulation versus experiment

For seeded experiments, with fixed seed count and no unseeded nucleation, fluorescence increase over time is a measure of the experimental growth rate. In Figure 5(a), we qualitatively compare maximum growth (via fluorescence increase) across temperatures for the experimental system, both when seeded and unseeded – in which case growth required spontaneous nucleation. Despite mixing system behaviours across a wide range of temperatures, and not accounting for tile depletion in simulations, simulations show a mild and decreasing decline in growth rates with increasing blocker concentration, as is also seen experimentally. A nucleation rate that is very high is also seen (and thus would be depletion-limited) for blocker concentrations where significant increases in fluorescence were observed in unseeded samples, exponentially decreasing across the range of blocker concentrations to be below ranges, per FFS predictions, likely matching minimal experimental nucleation.

### Multiple bond strengths: Simulation versus experiment

In a second series of experiments, Rogers et al. implemented multiple 12-helix tube systems, each with glues of different DNA domain lengths, and thus binding strengths, and measured the growth of these systems with fixed blocker concentrations. We consider primarily the growth rate of these systems when seeded. Complexities of fluorescent reporting, however, meant that these were not expected to be measures of absolute growth rate across different tile systems: rather, the increase in fluorescence, for each system separately, was used as a relative measure of growth rate when comparing growth at different temperatures. Our comparison, in Figure 5, thus uses the increase in fluorescence scaled from the minimum increase for each system (always at a temperature above the melting temperature for the system), to the maximum increase within the temperature range. Our simulated growth rate is scaled similarly, while our theoretical growth rate is scaled between the theoretical zero growth rate and the maximum within the range, as the theoretical rate includes negative rates for melting of existing structures at high temperatures, which was not possible in simulations or experiments starting from seeds. As the theoretical two-bond growth rate is for systems with equal bond strengths, the mean bond strength for each system was used.

Simulation closely match experimental temperatures in the multi-system case, especially considering the paucity of experimental temperature points. The temperature of maximum growth rate in simulation is between the highest two experimental growth rate temperatures for all systems, and the full-width-half-max temperature window is comparable, smaller in simulations by 9.8% for  $k = 9$  and 11.7% for  $k = 10$ . This close agreement of temperature window size motivates the omission of partially-blocked attachment events in the model: as discussed in Section B, the inclusion of partially-blocked attachment rates results in a significantly wider temperature window than seen experimentally, with a shallower fall-off in rates at low temperatures. Theoretical growth rate estimates, while matching the simulated temperatures of maximum growth rate, have a wider temperature window. As the theoretical estimate uses the mean bond strength, this result is expected: compared to the equal bond strength case, the weakest tile-assembly bonds will reduce growth at high temperatures, and the strongest cover-tile bonds will reduce growth at low temperatures.

## 5 Discussion

Despite the simplicity of our models, both theoretical analysis and simulations fit experiments well. That they do so despite disallowed reactions is perhaps unexpected. Remark 10 describes physically-plausible phenomena that may result in these reaction pathways being rare, but further analysis and experimental investigation would be fruitful. Nevertheless the simple model seems to provide a good approximation of experimental behaviours. The model, theoretical tools and simulations may be predictive of blocker systems beyond the tile systems studied here, and are likely applicable to a wider range of tile motifs.

Our model predicts that, with sufficient blocker concentration, both growth and spontaneous nucleation will be possible only within a fixed temperature window and both will have maximum rates at particular temperatures. If the maximum rate of spontaneous nucleation is low, then no matter how low the temperature is spontaneous nucleation will not be significant, unlike the no-blocker scenario where nucleation rates are very high at low temperatures. Similarly, below the temperature window for growth, growth will not occur: unlike tile assembly without blockers, where lower temperatures lead to disordered growth, with sufficient blockers, no growth will take place at all. While these results are in a model that does not consider non-lattice interactions that could be viable at low temperature, intuitively, they are based on simple tile-blocker dimerization that should continue to be relevant at low temperatures. Such temperature-dependent behaviour offers several possibilities. For example, could temperature be used as a programmable, time-dependent input to the growth of structures? Could ordered assemblies be grown without a typical high-to-low temperature annealing protocol, instead starting at room temperature, below the growth temperature window, and then simply heating the system to a viable growth temperature? We intentionally model only simple blockers in our model, to match those used in experiments. Numerous variations could have potentially interesting behaviours, especially if combined. Increasing and decreasing the strength of blocker glues compared to tile glues (for example, by making them longer or shorter than the binding regions) could be a stronger mechanism for tuning effective tile concentration than blocker concentration alone, potentially avoiding the need for large blocker concentrations. Multiple-edge blockers could have different, perhaps even error-reducing, properties, incorporating cooperative attachment into the blocker mechanism. We also have not yet explored the effect of blockers on algorithmic self-assembly, where it could change error rate properties.

More generally, our work provides a kinetic explanation for a tile assembly mechanism beyond the colder-is-faster dynamics of passive tile self-assembly, through addition of simple reactions interfering with/mediating the self-assembly process. What other forms of “modified kinetics” might exist, and what behaviours could they exhibit? Could modifications allow for new tradeoffs, e.g. speed-accuracy tradeoff in algorithmic kTAM self-assembly, or other error reduction mechanisms entirely? Could they offer resilience to temperature, energy or concentration variability? Do such simple mediating self-assembly reactions exist in nature?

---

## References

- 1 Rosalind J. Allen, Chantal Valeriani, and Pieter Rein ten Wolde. Forward flux sampling for rare event simulations. *Journal of Physics: Condensed Matter*, 21(46):463102, October 2009. doi:10.1088/0953-8984/21/46/463102.
- 2 Robert D. Barish, Paul W. K. Rothmund, and Erik Winfree. Two Computational Primitives for Algorithmic Self-Assembly: Copying and Counting. *Nano Letters*, 5(12):2586–2592, December 2005. doi:10.1021/nl0520381.

- 3 Robert M. Dirks, Justin S. Bois, Joseph M. Schaeffer, Erik Winfree, and Niles A. Pierce. Thermodynamic Analysis of Interacting Nucleic Acid Strands. *SIAM Review*, 49(1):65–88, January 2007. doi:10.1137/060651100.
- 4 David Doty. Theory of algorithmic self-assembly. *Communications of the ACM*, 55(12):78–88, December 2012. doi:10.1145/2380656.2380675.
- 5 Phillip Drake, Daniel Hader, and Matthew J. Patitz. Simulation of the Abstract Tile Assembly Model Using Crisscross Slats. In Shinnosuke Seki and Jaimie Marie Stewart, editors, *30th International Conference on DNA Computing and Molecular Programming (DNA 30)*, volume 314 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 3:1–3:25, Dagstuhl, Germany, 2024. Schloss Dagstuhl – Leibniz-Zentrum für Informatik. doi:10.4230/LIPIcs.DNA.30.3.
- 6 Constantine Evans. *Crystals That Count!* PhD thesis, California Institute of Technology, Pasadena, CA, USA, 2014.
- 7 Constantine Evans, Jackson O’Brien, Damien Woods, Erik Winfree, and Arvind Murugan. Controlling kinetic pathways in self-assembly through subtle concentration patterns. In preparation.
- 8 Constantine Evans and Angel Cervera Roldan. Rgrow tile assembly simulator, April 2025. doi:10.5281/zenodo.15232767.
- 9 Constantine Evans and Erik Winfree. Physical principles for DNA tile self-assembly. *Chemical Society Reviews*, 46(12):3808–3829, 2017. doi:10.1039/C6CS00745G.
- 10 Constantine Glen Evans, Jackson O’Brien, Erik Winfree, and Arvind Murugan. Pattern recognition in the nucleation kinetics of non-equilibrium self-assembly. *Nature*, 625(7995):500–507, January 2024. doi:10.1038/s41586-023-06890-z.
- 11 Pedro Fonseca, Flavio Romano, John S. Schreck, Thomas E. Ouldrige, Jonathan P. K. Doye, and Ard A. Louis. Multi-scale coarse-graining for the study of assembly pathways in DNA-brick self-assembly. *The Journal of Chemical Physics*, 148(13):134910, April 2018. doi:10.1063/1.5019344.
- 12 Shuoxing Jiang, Fan Hong, Huiyu Hu, Hao Yan, and Yan Liu. Understanding the Elementary Steps in DNA Tile-Based Self-Assembly. *ACS Nano*, 11(9):9370–9381, September 2017. doi:10.1021/acsnano.7b04845.
- 13 Yonggang Ke, Luvena L. Ong, William M. Shih, and Peng Yin. Three-Dimensional Structures Self-Assembled from DNA Bricks. *Science*, 338(6111):1177–1183, November 2012. doi:10.1126/science.1227268.
- 14 L D Landau and E M Lifshitz. *Statistical Physics*, volume 5. Pergamon Press, 2 edition, 1970.
- 15 Dionis Mineev, Christopher M. Wintersinger, Anastasia Ershova, and William M. Shih. Robust nucleation control via crisscross polymerization of highly coordinated DNA slats. *Nature Communications*, 12(1):1741, March 2021. doi:10.1038/s41467-021-21755-7.
- 16 Matthew J. Patitz. An introduction to tile-based self-assembly and a survey of recent results. *Natural Computing*, 13(2):195–224, June 2014. doi:10.1007/s11047-013-9379-4.
- 17 Trent Rogers, Constantine Evans, and Damien Woods. Covered DNA core tiles for robust tuning of spurious nucleation. *DNA28*, 2022. (conference talk, full paper in preparation).
- 18 Trent Rogers, Constantine Evans, and Damien Woods. A 1D nanoscale printer: assembling DNA nanotubes of arbitrary and precise length from a fixed DNA tile set. *DNA30*, 2024. (conference talk, full paper in preparation).
- 19 John SantaLucia. A unified view of polymer, dumbbell, and oligonucleotide DNA nearest-neighbor thermodynamics. *Proceedings of the National Academy of Sciences*, 95(4):1460–1465, February 1998. doi:10.1073/pnas.95.4.1460.
- 20 Samuel W. Schaffter, Dominic Scalise, Terence M. Murphy, Anusha Patel, and Rebecca Schulman. Feedback regulation of crystal growth by buffering monomer concentration. *Nature Communications*, 11(1):6057, November 2020. doi:10.1038/s41467-020-19882-8.



- 21 Rebecca Schulman and Erik Winfree. Synthesis of crystals with a programmable kinetic barrier to nucleation. *Proceedings of the National Academy of Sciences*, 104(39):15236–15241, September 2007. doi:10.1073/pnas.0701467104.
- 22 Rebecca Schulman and Erik Winfree. Programmable Control of Nucleation for Algorithmic Self-Assembly. *SIAM Journal on Computing*, 39(4):1581–1616, January 2010. doi:10.1137/070680266.
- 23 Bryan Wei, Mingjie Dai, and Peng Yin. Complex shapes self-assembled from single-stranded DNA tiles. *Nature*, 485(7400):623–626, May 2012. doi:10.1038/nature11075.
- 24 Erik Winfree. Simulations of Computing by Self-Assembly. Caltech Technical Report CS-TR:1998.22, Caltech, 1998.
- 25 Christopher M. Wintersinger, Dionis Mineev, Anastasia Ershova, Hiroshi M. Sasaki, Gokul Gowri, Jonathan F. Berengut, F. Eduardo Corea-Dilbert, Peng Yin, and William M. Shih. Multi-micron crisscross structures grown from DNA-origami slats. *Nature Nanotechnology*, 18(3):281–289, March 2023. doi:10.1038/s41565-022-01283-1.
- 26 Damien Woods, David Doty, Cameron Myhrvold, Joy Hui, Felix Zhou, Peng Yin, and Erik Winfree. Diverse and robust molecular algorithms using reprogrammable DNA self-assembly. *Nature*, 567(7748):366–372, March 2019. doi:10.1038/s41586-019-1014-9.

## A A unitful-energy reparametrization of the kTAM

For comparisons to the kTAM model [24, 9], we use a reparametrization of the kTAM in terms of unitful energies, rather than dimensionless, sign-reversed energies. In the reparametrized model, the governing rate equations are

$$r_{\text{att}} = k_{\text{f}} c_{\text{t}} \quad r_{\text{det}} = k_{\text{r}} e^{\beta b \Delta G_{\text{bond}}(T) - (b-1) \Delta S_{\text{lat}} / R} \quad (10)$$

These parameters are related to the dimensionless kTAM [24, 9] as:

$$c_{\text{t}} = e^{-G_{\text{mc}} + \alpha} \quad \Delta G_{\text{bond}}(T) = -RT(G_{\text{se}} - \alpha) \quad \Delta S_{\text{lat}} = R\alpha \quad (11)$$

$$G_{\text{mc}} = \alpha - \ln c_{\text{t}} \quad G_{\text{se}} = -\beta \Delta G_{\text{bond}}(T) + \alpha \quad \alpha = \Delta S_{\text{lat}} / R \quad (12)$$

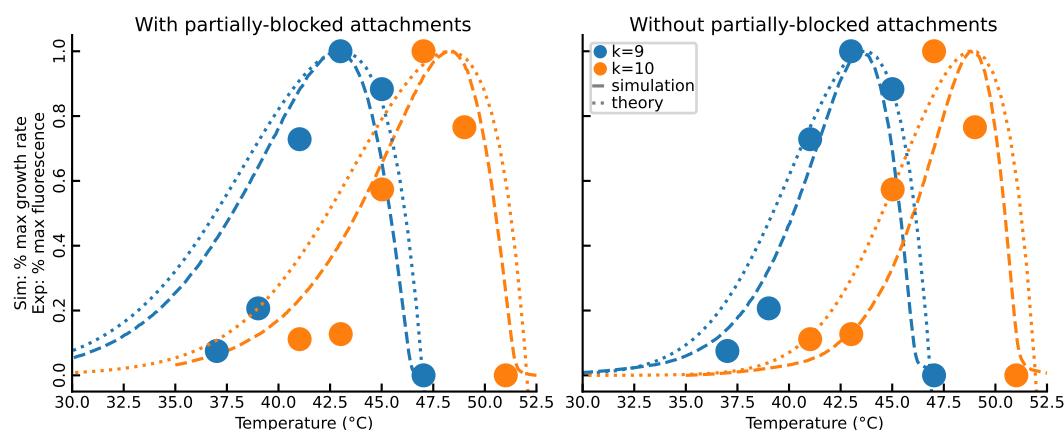
It is easily seen that the kBlock model, with  $c_{\text{b}} = 0$  and  $p_{\text{a}} = 1$ , is equivalent to this parametrization of the kTAM.

## B Partially-blocked attachment events

In the kBlock model, we make the assumption that a tile state may only attach if the tile would not be adjacent to any edge of any tile state in the assembly with a bound blocker. It is possible, and perhaps even more physically plausible, to consider the model where the opposite choice is made: where a tile can attach if there is at least one edge where it can make a bond, regardless of blockers on other edges. We will refer to this model as the kBlockP, where “P” stands for partial binding. Rates will remain the same: the only significant difference is that now, there may be blockers blocking bonds between adjacent tiles in an assembly. We will assume that these bonds are not able to form with the blocker present, but that those blockers will detach with the same rate as blockers elsewhere, and that when they do, the adjoining tiles will immediately make a bond if they have complementary glues on those edges, changing their detachment rates.

The kBlockP model does not satisfy detailed balance: there is no blocker attachment reaction corresponding to a blocker detaching from a position where there is a neighbouring tile. Like the kTAM, the blocker model treats reactions at a tile (and blocker) level as being single steps, rather than considering the attachment and detachment of individual bonds





**Figure 6** Comparison of growth rates. **Left:** With partially-blocked attachment events, i.e. using the  $k\text{BlockP}_2$  model discussed here in Section B. **Right:** Without partially-blocked attachment events, i.e. using the  $k\text{Block}_2$  model defined in Definition 12.

as the elementary reactions. This approach is considerably simpler and easier to consider physically-realistic parameters for: it is not clear, for example, what the kinetics of individual bond formation and breakage would be. However, for reaction pathways where individual bond formation and breakage must be considered, such models either violate detailed balance, if only one reaction direction can be included, as is the case here and with fission reactions in many  $k\text{TAM}$  implementations, or omit reaction pathways entirely, for example, for the fission of a 1D line of tiles, where the  $k\text{TAM}$  would calculate the line breaking in two based on tiles attached by two bonds, whereas a bond-level model would allow the line to break by the much faster reaction of only a single bond breaking. For the  $k\text{BlockP}$  model, we expect the omitted reaction pathways would be sufficiently rare, and the violation of detailed balance sufficiently small, so as to not significantly affect simulation results. However, a more general bond-based model, with or without blockers, is an open area for future work.

A similar  $k\text{BlockP}_2$  can be developed, but in this case, we will allow attachments by two bonds, or by tiles that could form two bonds, but have one bond blocked. Similar theoretical results can then be shown, primarily that  $k\text{BlockP}_2$  will have the same two-bond growth rate as a  $k\text{TAM}_2$  system with the effective tile concentration of  $c_{k\text{TAM}} = p_a c_{k\text{Block}}$ , rather than  $p_a^2 c_{k\text{Block}}$ , and that nucleation rates through a critical nucleus with  $N$  tiles and  $B$  bonds will be reduced by a factor of  $p_a^B$ . These results also fit simulation results well, when the simulation uses the  $k\text{BlockP}_2$  model.

However, in comparison to experimental results for multiple systems of varying bond strengths, shown in Figure 6, the temperature dependence of the  $k\text{BlockP}$  growth rate is significantly different from the experimental results, compared to the  $k\text{Block}$  model, which matches surprisingly well.



# An Axiomatic Study of Leveraging Blockers to Self-Assemble Arbitrary Shapes via Temperature Programming

Matthew J. Patitz ✉ 

University of Arkansas, Fayetteville, AR, USA

Trent A. Rogers ✉ 

University of Arkansas, Fayetteville, AR, USA

---

## Abstract

In this paper we present a theoretical design for tile-based self-assembling systems where individual tile sets that are universal for various tasks (e.g. building shapes or encoding arbitrary data for algorithmic systems) can be provided their input solely using sequences of variations in temperatures. Although there is prior theoretical work in so-called “temperature programming,” the results presented here are based upon recent experimental work with “blocked tiles” that provides plausible evidence for their practical implementation. We develop and present an abstract mathematical model, the BlockTAM, for such systems that is based upon the experimental work, and provide constructions within it for universal number encoding systems and a universal shape building construction. These results mirror previous results in temperature programming, covering the two ends of the spectrum that seek to balance the scale factors of assemblies with the number of temperature phases required, while leveraging the features of the BlockTAM that we hope provide a pathway for future experimental implementations.

**2012 ACM Subject Classification** Theory of computation → Models of computation

**Keywords and phrases** DNA tiles, self-assembly, temperature programming

**Digital Object Identifier** 10.4230/LIPIcs.DNA.31.8

**Supplementary Material** *Software*: [http://self-assembly.net/software/BlockTAM\\_examples/BlockTAM\\_examples-source-2025-04-23-14-42-34.zip](http://self-assembly.net/software/BlockTAM_examples/BlockTAM_examples-source-2025-04-23-14-42-34.zip)

*Software*: [http://self-assembly.net/software/WebTAS\\_beta/WebTAS\\_beta-latest/](http://self-assembly.net/software/WebTAS_beta/WebTAS_beta-latest/) [8]

**Funding** *Matthew J. Patitz*: This author’s work was supported in part by NSF grant 2329908.

## 1 Introduction

One of the central themes of technological advancement is mastering the manipulation of matter. In particular, humans have been fascinated with the construction of arbitrary structures since the beginning of human history. At times, this fascination has been due to the fact that survival has depended on a sufficient mastery of crafting structures while at other times this fascination has had more recreational roots. Over the millennia, humans have mastered building macroscale structures, but the idea of constructing nanoscale structures was unfathomable until relatively recently. Even though the manufacture of arbitrary nanoscale shapes may be a new endeavor for humanity, we are already seeing it begin to take a central role in the development of our future. Nanoscale structures are already proving to be useful both in terms of aiding in humanity’s survival and recreational pursuits as nanoscale structures have seen a wide range of uses including molecular computers [24], drug delivery [22], and lithographically-patterned electronic/photonic devices [6].

Observing and influencing matter at the nanoscale presents a seemingly insurmountable challenge since the structures at this scale are smaller than the wavelengths of visible light. How can we build a structure that we cannot see using components whose location we cannot



© Matthew J. Patitz and Trent A. Rogers;

licensed under Creative Commons License CC-BY 4.0

31st International Conference on DNA Computing and Molecular Programming (DNA 31).

Editors: Josie Schaeffer and Fei Zhang; Article No. 8; pp. 8:1–8:20

Leibniz International Proceedings in Informatics



LIPICs Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

control? As opposed to the traditional macroscopic approach, self-assembly uses a bottom up approach: instead of attempting to mechanically manipulate components to place them in certain relative locations, chemically manipulate components so that they are prone to stick together in specific locations relative to one another. Unfortunately, this general approach is still quite abstract and difficult to leverage, with successful physical realizations remaining elusive. In 1981, Ned Seeman proposed leveraging the programmability and specificity of deoxyribonucleic acid (DNA) to implement such self-assembling systems, and he constructed the first intentionally designed DNA nanostructure [3, 19].

Building on the work of Ned Seeman, Erik Winfree introduced and physically realized a framework to make the design of DNA nanostructures more tractable (and hence, less error prone) [23]. This framework uses designed DNA strands to assemble molecules that behave like square “tiles” with programmable “glues” that allow a tile to bind to other particular tiles. This approach is known as *tile-based DNA self-assembly*. Peng Yin et al. [9] leveraged this approach to develop a technology capable of assembling arbitrary nanoscale structures (where the resolution is limited by the size of the molecule that acts as a tile). William Shih et al. [12] further refined this approach (albeit with non-planar, non-square tiles) to eliminate technical hurdles which resulted in less than desirable error rates.

In 2006, Paul Rothemund introduced a new paradigm for the assembly of arbitrary nanostructures from DNA [17]. Like Winfree’s framework, Rothemund’s framework constrains the problem of designing DNA nanostructures by placing it into a logical framework to make the problem more tractable. This framework is known as DNA origami. The idea is to create the target structure by folding a very long DNA strand, called the scaffold strand, into the target shape through the use of small staple strands which bind to multiple positions along the scaffold. These staple strands pinch, or collocate, certain parts of the scaffold strand together and the net effect of this is to fold the scaffold strand into the target structure.

Regardless of which ubiquitous method listed above is used, each significantly different target structure requires a unique set of DNA strands to assemble it. Designing, synthesizing, ordering, and preparing these DNA strands for experiments is costly in terms of time, money and labor. This process also introduces the possibility of human errors and unintended sequence interactions. One way to reduce this burden of requiring new DNA strands is to reduce the number of new strands needed by as much as possible. Soleveichik and Winfree showed that in an axiomatic model of tile-based self-assembly, a target structure can be assembled using (nearly) information theoretic optimal number of strands [20]. While the result of Soleveichik and Winfree greatly reduces (theoretically) the number of strands that needed to assemble a new target structure, it does not completely eliminate it.

Is it possible to design a fixed set of DNA strands that can assemble any arbitrary shape without the need for pipetting or any other manual intervention steps? In the standard axiomatic model of tile-based self-assembly, it is not possible to eliminate the need for new DNA strands for assembling new target structures (since this is how input is passed into the system specifying what structure it should assemble). But, in [21], Summers showed (theoretically) that a fixed set of tiles could assemble any arbitrary 2D shape by controlling the growth of the assembly using controlled temperature fluctuations. This was shown in the multiple temperature model originally introduced in [1]. Unfortunately, this model makes a couple of assumptions which have yet to be demonstrated in the laboratory setting, namely that temperature fluctuations do not lead to tiles aggregating off of the seed assembly and that the binding of one tile to an assembly can prevent the binding of another tile to the assembly. Fortunately, recent experimental work has shown temperature programming may be feasible in the laboratory setting through the use of blockers. This is discussed in the next section.

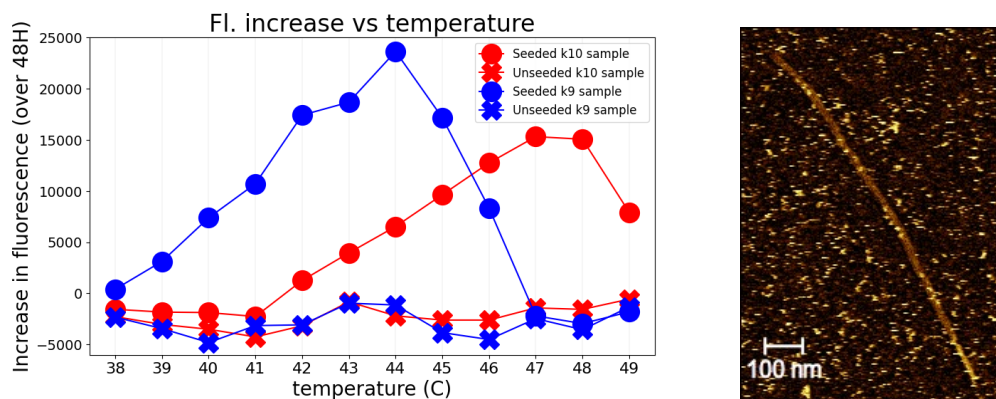
## Experimental Motivation

As mentioned above, previous theoretical results on temperature programming relied on two yet to be demonstrated mechanisms: 1) suppressing interactions of sticky glues off-seed at lower temperatures and 2) using the existence of a tile in the assembly to prevent the binding of another tile.

Recent experimental work [16] has shown that by “poisoning” systems with single glue domain length strands, called *blockers*, which are complementary to “input” glues, that one can not only “turn off” high strength (that is, very sticky) glues at lower temperatures, but also suppress spurious nucleation (that is, off-lattice interactions that lead to structures not grown from a seed) across all temperatures. This is thought to be due to the binding penalty induced by blockers being multiplicative for pathways that lead to spurious nucleation (which leads to an exponential impact on the spurious nucleation rate) while having a less significant impact on the growth rate of seeded assemblies since the penalty will result in linear slowdown [5]. Figure 1a, constructed from data in [15], shows this effect in practice. In the presence of a seed, growth may be slowed, but still proceeds at a reasonable rate (as indicated by the increase in fluorescence in samples with seeds), while spurious nucleation is suppressed to undetectable levels across all temperatures (as indicated by the relatively flat fluorescence trace of unseeded samples). Also, shown in this plot is the deactivation effect blockers have on stickier (that is, high-strength) glues at lower temperatures. It shows the growth rate of tiles consisting of length 10 glues becomes suppressed to undetectable levels at lower temperatures (evidenced by the insignificant increase in the seeded sample containing tiles with length 10 glues at temperatures 41C and below) where the growth rate of tiles consisting of length 9 glues is still significant (evidenced by the significant increase in the seeded samples at these temperatures).

The Blocked Tile Assembly Model is inspired by the glue deactivation and spurious nucleation suppression afforded by blockers. But, it still remains in the realm of theory since attempting to implement such system in the laboratory could face technical hurdles such as spurious nucleation due to the oscillation of temperature or strain at the interfaces of tiles with different glue lengths preventing further growth. In [15], temperature programming was physically realized. The authors showed that the key mechanisms required to physically realize temperature programming are attainable, and they used these features to program nanotube length using temperature oscillations. When interpreted through the lens of the Blocked Tile Assembly Model, the construction in [15] is simple. Two sets of tiles which assemble a cross-section of a nanotube were designed: one having length 10 glues (which are the high strength glues) and the other having length 9 glues (which are the low strength glues). These tile sets were designed to form alternating cross-sections of the nanotube. Blockers were added to the input side of the length 10 glues preventing those tiles from attaching at a lower temperature where the length 9 glues are active. On the other hand, the tiles with length 9 glues are unable to attach at warmer temperatures where the length 10 glues are active. The authors leverage the exclusivity of growth temperatures to design a system which requires the temperature to be oscillated in order to grow the next segment of the tube. In this manner, the number of oscillations dictates the length of the nanotube. (See Figure 2 for an example system which illustrates this idea.)

Figure 1b presents atomic force microscopy (AFM) data from [15]) which shows a nanotube grown using the temperature programming technique described above. The segments composed of tiles with length 9 glues were marked with a special molecule so that they appear with bright dots on the image. The temperature was oscillated 4 times (that is, it was held a high temperature and then a low temperature and this process was repeated 4



(a) Bulk fluorescence data taken from [15] providing evidence that blockers suppress spurious nucleation and deactivate stickier glues at lower temperatures. Each seeded sample is intended to grow a 12 helix nanotube and the unseeded samples consist of the tiles used in the seeded sample but do not contain the seed. A pair of molecules were added to two distinct tiles in each sample, so that the fluorescence signal increases when they are collocated indicating a nanotube is growing. Four samples were held at 12 temperatures for 48 hours and their fluorescence increase over that time is recorded on the plot. The samples labeled k10 have tiles with length 10 glues and the samples labeled k9 have tiles with length 9 glues. The relatively flat lines with x markers (which are the unseeded samples) indicate the fluorescence increase in unseeded samples is undetectable which we interpret to mean that the spurious nucleation rate is very low. Note that the negative fluorescence values are thought to be due to photo-bleaching. The seeded samples (the data points with circle markers) see significant fluorescence increase over the hold indicating that growth is occurring.

(b) A visualization of atomic force microscopy (AFM) data taken from [15]. The system used to grow this nanotube via a temperature programming technique was designed to grow the nanotube using alternating segments of tiles with length 10 glues and tiles with length 9 glues. The segment grown from tiles with length 9 glues was marked with a special molecule so that bright dots appear on the segment. This image shows a nanotube resulting from a system that underwent four temperature oscillations, so we would expect there to be four segments consisting of tiles with length 10 glues (unmarked) interleaved with four segments consisting of tiles with length 9 glues (marked).

■ **Figure 1** Evidence that the key mechanisms assumed by the Blocked Tile Assembly Model are experimentally feasible.

times) so that we would expect to see 8 cross-section segments alternating between being unmarked and marked. Note that this is indeed what we see in the image. We see the seed, the bright tube in the bottom right-hand corner of the image, followed by eight alternating unmarked and marked segments. This provides evidence that the system is working as intended.

## Our contributions

We first formally define the Blocked Tile Assembly Model that presents a formalization of the intuitive model used to design the system presented in [15], which is capable of using temperature to program tube length. A natural extension of the work presented in [15] is to examine whether we can leverage the mechanisms used in that paper to use temperature to program not just length, but the shape of the structure. This is akin to the idea of a 3D printer (although we work in 2 dimensions) which uses a universal ink to construct arbitrary structures. That is, can we assemble a target nanostructure purely through temperature fluctuations without the need to design, order, or add new strands?

We answer this question affirmatively through our two main results. The constructions we present rely on the same mechanisms already demonstrated in [15] with two exceptions. Our constructions rely on having three or more tile sets which have disjoint growth temperatures.

In other words, our constructions would require something like length 8 glues in addition to length 9 and length 10 glues. Another yet to be experimentally demonstrated mechanism is having mismatches in the glues between neighboring tiles. However, this mechanism has been demonstrated experimentally, albeit unintentionally, with other DNA tile motifs through the occurrence of errors in algorithmic self-assembly [24]. Neither of these technical challenges appear insurmountable which provides us hope that these constructions, or at least somewhat simplified versions of them, may be experimentally feasible.

Our first main result provides a way for a single BlockTAM tile and blocker set, always beginning from the same seed tile, to receive, and then encode, an arbitrary input value into the glues of a row of tiles solely via a sequence of temperature changes. We call this a *universal number encoder*, and such a system could be combined with an aTAM system designed to perform any algorithmic self-assembly task by first reading that information as its input and then carrying out its algorithmic growth. Dozens of powerful examples of algorithmic self-assembling systems have been theoretically demonstrated, e.g. [2, 10, 11, 14, 18, 20], and one in particular worth mentioning is that of [20] in which such an input sequence could be the information-theoretically optimal encoding of the definition of an arbitrary shape so that the assembly then proceeds to grow into a scaled version of that shape.

This leads naturally to our second main result, which we call a *universal shape builder*, and is a fixed tile and blocker set such that, given any arbitrary shape  $S$ , a system that always begins from the same seed tile can be controlled by a temperature sequence taken from the set of temperatures  $\{2, 3, 4\}$  that is specific to  $S$  so that the resulting terminal assembly is in the shape of  $S$  at a constant scale factor of 6.

As an input module to the construction of [20], our universal number encoder is also a version of a universal shape builder, and although it is optimal in terms of the number of temperature phases required to build an arbitrary shape, the scale factor of the shape is extremely large.<sup>1</sup> However, while our second construction requires a number of temperature phases that is linear in the number of points in the target shape, its scale factor is a mere 6. In combination, these results provide evidence of the broad powers and potential of the BlockTAM and inspiration for both further theoretical and experimental implementations.

Both results have been fully designed, implemented, and simulated. Python code that can be used to generate example systems for both results (along with pre-existing examples) [13] and a web-based simulator [8] can be accessed from: [http://self-assembly.net/wiki/index.php/Blocked\\_Tile\\_Assembly\\_Model\\_\(BlockTAM\)](http://self-assembly.net/wiki/index.php/Blocked_Tile_Assembly_Model_(BlockTAM)).

## 2 Preliminary Definitions and Models

### 2.1 The Blocked Tile Assembly Model Intuition

We begin by providing a high-level overview of the aTAM and BlockTAM so that readers familiar with the aTAM may skip the formal definitions section.

The abstract Tile Assembly Model is a tile-based model of self-assembly where growth proceeds from an initial assembly called the seed and proceeds asynchronously and non-deterministically via single-tile attachments. The systems which make up the abstract Tile Assembly Model (aTAM) are called tile assembly systems (TAS) and they are triples  $(T, \sigma, \tau)$  where  $T$  is the set of tile types,  $\sigma$  is the seed assembly, and  $\tau$  is the temperature, or minimum

<sup>1</sup> The scale factor would depend on the particular shape, but would likely be on the order of at least tens of thousands.



binding threshold, of the system. Growth of the system begins from the seed  $\sigma$  and grows via single tile attachments (where the tiles come from the tile set  $T$ ) provided that each tile binds to the assembly with at least a combined strength of  $\tau$ .

The Blocked Tile Assembly Model (BlockTAM) is an adaptation of the Multiple Temperature Model introduced in [4]. To formally define this model, we first define a 1 Phase Blocked Tile Assembly Model (1BlockTAM). The BlockTAM can be thought of as consisting of a sequence of 1BlockTAM systems where the resulting terminal assembly of the  $i^{\text{th}}$  system is fed into the  $(i + 1)^{\text{th}}$  system as a seed.

A system in the 1 Phase Blocked Tile Assembly Model (1BlockTAM) is a 4-tuple  $(T, C, \sigma, \tau)$  where  $T$ ,  $\sigma$  and  $\tau$  are defined as in the aTAM, and  $C$  is defined to be a set of “blockers”. A blocker is defined as a triple  $(t, d, s)$  where  $t$  is a tile,  $d$  is a direction on the tile, and  $s$  is a strength. Growth of the system proceeds as in the aTAM with the exception that any glue which has a blocker whose strength is  $\geq \tau$  is treated as a strength 0 glue (to reflect the fact that the glue is “blocked”). We formalize this by constructing an aTAM system from the 1BlockTAM system where any glues that are “blocked” in the system have their strength set to 0 and defining the behavior of the 1BlockTAM system to be the same as the derived aTAM system.

## 2.2 Abstract Tile Assembly Model

The definitions below are an adaptation of the abstract Tile Assembly Model (aTAM) definition presented in [7]. We note that [18] and [11] are good introductions to the model for unfamiliar readers.

Let  $\mathbb{N}$  be the set of nonnegative integers, and for  $n \in \mathbb{N}$ , let  $[n] = \{0, 1, \dots, n - 2, n - 1\}$ . Similarly, for  $l, u \in \mathbb{Z}$ , let  $\llbracket l, u \rrbracket = \{k \in \mathbb{Z} \mid l \leq k < u\}$ .

Fix  $\Sigma$  to be some alphabet with  $\Sigma^*$  its finite strings. A *glue*  $g \in \Sigma^* \times \mathbb{N}$  consists of a finite string *label* and non-negative integer *strength*. A *tile type* is a tuple  $t \in (\Sigma^* \times \mathbb{N})^4$ , thought of as a unit square with a glue on each side. A *tile set* is a finite set of tile types. We always assume a finite set of tile types, but allow an infinite number of copies of each tile type to occupy locations in the  $\mathbb{Z}^2$  lattice, each called a *tile*.

Given a tile set  $T$ , a *configuration* is an arrangement (possibly empty) of tiles in the lattice  $\mathbb{Z}^2$ , i.e. a partial function  $\alpha : \mathbb{Z}^2 \dashrightarrow T$ . Two adjacent tiles in a configuration *interact*, or are *bound* or *attached*, if the glues on their abutting sides are equal (in both label and strength) and have positive strength. Each configuration  $\alpha$  induces a *binding graph*  $B_\alpha$  whose vertices are those points occupied by tiles, with an edge of weight  $s$  between two vertices if the corresponding tiles interact with strength  $s$ . An *assembly* is a configuration whose domain (as a graph) is connected and non-empty. The *shape*  $S_\alpha \subseteq \mathbb{Z}^2$  of assembly  $\alpha$  is the domain of  $\alpha$ . For some  $\tau \in \mathbb{Z}^+$ , an assembly  $\alpha$  is  $\tau$ -*stable* if every cut of  $B_\alpha$  has weight at least  $\tau$ , i.e. a  $\tau$ -stable assembly cannot be split into two pieces without separating bound tiles whose shared glues have cumulative strength  $\tau$ . Given two assemblies  $\alpha, \beta$ , we say  $\alpha$  is a *subassembly* of  $\beta$  (denoted  $\alpha \sqsubseteq \beta$ ) if  $S_\alpha \subseteq S_\beta$  and for all  $p \in S_\alpha$ ,  $\alpha(p) = \beta(p)$  (i.e., they have tiles of the same types in all locations of  $\alpha$ ).

A *tile-assembly system* (TAS) is a triple  $\mathcal{T} = (T, \sigma, \tau)$ , where  $T$  is a tile set,  $\sigma$  is a finite  $\tau$ -stable assembly called the *seed assembly*, and  $\tau \in \mathbb{Z}^+$  is called the *binding threshold* (a.k.a. *temperature*). If the seed  $\sigma$  consists of a single tile, i.e.  $|\sigma| = 1$ , we say  $\mathcal{T}$  is *singly-seeded*. Given a TAS  $\mathcal{T} = (T, \sigma, \tau)$  and two  $\tau$ -stable assemblies  $\alpha$  and  $\beta$ , we say that  $\alpha$   $\mathcal{T}$ -*produces*  $\beta$  *in one step* (written  $\alpha \rightarrow_1^{\mathcal{T}} \beta$ ) if  $\alpha \sqsubseteq \beta$  and  $|S_\beta \setminus S_\alpha| = 1$ . That is,  $\alpha \rightarrow_1^{\mathcal{T}} \beta$  if  $\beta$  differs from  $\alpha$  by the addition of a single tile. The  $\mathcal{T}$ -*frontier* is the set  $\partial^{\mathcal{T}} \alpha = \bigcup_{\alpha \rightarrow_1^{\mathcal{T}} \beta} S_\beta \setminus S_\alpha$  of locations in which a tile could  $\tau$ -stably attach to  $\alpha$ . When  $\mathcal{T}$  is clear from context we simply refer to these as the *frontier* locations.

We use  $\mathcal{A}^T$  to denote the set of all assemblies of tiles in tile set  $T$ . Given a TAS  $\mathcal{T} = (T, \sigma, \tau)$ , a sequence of  $k \in \mathbb{Z}^+ \cup \{\infty\}$  assemblies  $\alpha_0, \alpha_1, \dots$  over  $\mathcal{A}^T$  is called a  $\mathcal{T}$ -assembly sequence if, for all  $1 \leq i < k$ ,  $\alpha_{i-1} \rightarrow_1^{\mathcal{T}} \alpha_i$ . The *result*  $\lim \alpha$  of an assembly sequence  $\alpha$  is the unique limiting assembly of the sequence. For finite assembly sequences, this is the final assembly; whereas for infinite assembly sequences, this is the assembly consisting of all tiles from any assembly in the sequence. We say that  $\alpha$   $\mathcal{T}$ -produces  $\beta$  (denoted  $\alpha \rightarrow^{\mathcal{T}} \beta$ ) if there is a  $\mathcal{T}$ -assembly sequence starting with  $\alpha$  whose result is  $\beta$ . We say  $\alpha$  is  $\mathcal{T}$ -producible if  $\sigma \rightarrow^{\mathcal{T}} \alpha$  and write  $\mathcal{A}[\mathcal{T}]$  to denote the set of  $\mathcal{T}$ -producible assemblies. We say  $\alpha$  is  $\mathcal{T}$ -terminal if  $\alpha$  is  $\tau$ -stable and there exists no assembly that is  $\mathcal{T}$ -producible from  $\alpha$ . We denote the set of  $\mathcal{T}$ -producible and  $\mathcal{T}$ -terminal assemblies by  $\mathcal{A}_{\square}[\mathcal{T}]$ . If  $|\mathcal{A}_{\square}[\mathcal{T}]| = 1$ , i.e., there is exactly one terminal assembly, we say that  $\mathcal{T}$  is *directed*. When  $\mathcal{T}$  is clear from context, we may omit  $\mathcal{T}$  from notation.

### 2.3 The 1-Phase Blocked Tile Assembly Model

Assembly in the BlockTAM proceeds through a series of temperature *phases*, each of which consists of a hold at a specified temperature during which growth proceeds until all assemblies are terminal. Once all assemblies are terminal, the temperature of the system is set to the next temperature in a specified series and remains there until all growth is again terminal. This process continues until all growth is terminal during the last temperature phase. Note that a temperature sequence must be finite and that infinite growth (in the limit) during a phase is only allowed during the final phase.

Although the BlockTAM shares similarities with other tile-assembly models that allow for temperature variations (e.g. [1, 21]), additional complexity arises from the fact that, at different temperatures, different subsets of the blockers in a system may be actively blocking glues. Therefore, to ease explanation of the BlockTAM, here we first define its dynamics during a single phase, then in Section 2.4 define the full model in terms of phases.

In order to easily translate between a tile and its blocked version, we assume all distinct glues have distinct labels. That is, there are not two glues with the same labels, but different strengths. This is a simple transformation that can be obtained by relabeling each glue so that the new label is a concatenation of the glue's original label along with its strength.

A blocker is defined as a triple  $(t, d, s)$  where  $t$  is a tile,  $d$  is a direction on the tile, and  $s$  is a strength. We say that a blocker  $c = (t, d, s)$  binds to tile  $t$  on its  $d$  edge with strength  $s$ . Let  $t$  be a tile,  $C$  be a set of blockers, and  $\tau$  be a temperature, then we define the tile  $t$  blocked at temperature  $\tau$  by blocker set  $C$ , denoted  $\text{blocked}(t, C, \tau)$  to be the tile with the same glues as  $t$  unless there is a blocker which binds at the location of the glue on the tile with strength greater than or equal to  $\tau$ . Note that when  $C$  and  $\tau$  are obvious from context, we simply write  $\text{blocked}(t)$ . Also, since we define an assembly to map points to tiles we can compose this function with an assembly to obtain a function which maps each point to a blocked version of the tile. We define the function  $\text{unblocked}$  to be the inverse of the blocked function. That is,  $\text{unblocked}(\text{blocked}(t, C, \tau)) = t$ . The function  $\text{blocked}$  is invertible due to our assumption distinct glues have distinct labels.

A 1-Phase BlockTAM system (1BlockTAS) is a 4-tuple  $\mathcal{B} = (T, C, \sigma, \tau)$  where  $T$  is a tile set,  $C$  is a blocker set,  $\sigma$  is the seed assembly, and  $\tau$  is the temperature. We define the equivalent TAS of  $\mathcal{B}$ , denoted  $\text{TEQ}(\mathcal{B})$ , to be the aTAM system  $\mathcal{T} = (T', \text{blocked}(\sigma), \tau)$  where  $T'$  consists of a modified version of  $T$  such that any glue whose blocker binds with strength  $\geq \tau$  is set to have strength 0. That is  $T' = \{\text{blocked}(t, C, \tau) | t \in T\}$ . Given a 1BlockTAS  $\mathcal{B} = (T, C, \sigma, \tau)$  and two  $\tau$ -stable assemblies  $\alpha$  and  $\beta$  formed from tiles in  $T$ , we say that  $\alpha$   $\mathcal{B}$ -produces  $\beta$  in one step, written  $\alpha \rightarrow_1^{\mathcal{B}} \beta$ , if  $\text{blocked}(\alpha) \rightarrow_1^{\text{TEQ}(\mathcal{B})} \text{blocked}(\beta)$ . The  $\mathcal{B}$ -frontier is the set  $\partial^{\text{TEQ}(\mathcal{B})}(\text{blocked}(\alpha))$ .

Given a 1BlockTAS  $\mathcal{B} = (T, C, \sigma, \tau)$ , a sequence of  $k \in \mathbb{Z}^+ \cup \{\infty\}$  assemblies  $\alpha_0, \alpha_1, \dots$  over  $\mathcal{A}^T$  is called a  $\mathcal{B}$  assembly sequence if  $\text{blocked}(\alpha_0), \text{blocked}(\alpha_1), \dots$  is a  $\text{TEQ}(\mathcal{B})$ -assembly sequence. We say that  $\alpha$   $\mathcal{B}$ -produces  $\beta$  denoted  $\alpha \rightarrow^{\mathcal{B}} \beta$  if  $\text{blocked}(\alpha) \rightarrow^{\text{TEQ}(\mathcal{B})} \text{blocked}(\beta)$ . We say that  $\alpha$  is  $\mathcal{B}$ -producible if  $\sigma \rightarrow^{\mathcal{B}} \alpha$  and write  $\mathcal{A}[\mathcal{B}]$  to denote the set of  $\mathcal{B}$ -producible assemblies. We say  $\alpha$  is  $\mathcal{B}$ -terminal if  $\text{blocked}(\alpha)$  is  $\text{TEQ}(\mathcal{B})$ -terminal. We denote the set of  $\mathcal{B}$ -producible and  $\mathcal{B}$ -terminal assemblies by  $\mathcal{A}_{\square}[\mathcal{B}]$ . We say that  $\mathcal{B}$  is directed provided that  $\text{TEQ}(\mathcal{B})$  is directed.

## 2.4 The Blocked Tile Assembly Model

The following formal definitions are an adaptation of those used to define the Multiple Temperature Model in [21].

A BlockTAM system (BlockTAS) is a 4-tuple  $\mathcal{B}^* = (T, C, \sigma, \langle \tau \rangle_{i=0}^{k-1})$  where  $T$  is a tile set,  $C$  is a blocker set,  $\sigma$  is the seed assembly, and  $\langle \tau \rangle_{i=0}^{k-1}$  is a sequence of non-negative binding thresholds. The number  $k$  is the temperature complexity.

We define an *assembly sequence for the  $i^{\text{th}}$  temperature phase*, for  $0 \leq i < k$ , of  $\mathcal{B}^*$  as follows. For  $i = 0$ , an assembly sequence is an assembly sequence  $\vec{\alpha}$  of the 1BlockTAS  $\mathcal{B}_0 = (T, C, \sigma, \tau_0)$ . For  $i > 0$ , an assembly sequence for temperature phase  $i$  of  $\mathcal{B}^*$  is a sequence of assemblies  $\vec{\alpha} = (\alpha_0, \alpha_1, \dots)$  satisfying the following conditions.

1. There exists  $l$  such that  $\alpha_l$  is  $\tau_{i-1}$ -stable  $\partial^{\mathcal{B}_{i-1}} \alpha_l = \emptyset$ , and  $(\alpha_0, \alpha_1, \dots, \alpha_l)$  is an assembly sequence for temperature phase  $i - 1$  of  $\mathcal{B}^*$ ; and
2. for all  $j > l$ , either  $\alpha_{j-1} \rightarrow_1^{\mathcal{B}_i} \alpha_j$  or  $\alpha_j$  is obtained from  $\alpha_{j-1}$  by deleting the portion of  $\alpha_{j-1}$  which does not contain the seed across a cut having strength less than  $\tau_i$ .

Intuitively, an assembly sequence for the  $i^{\text{th}}$  temperature phase is defined recursively. That is, we say that  $(\alpha_0, \alpha_1, \dots)$  is an assembly sequence for the  $i^{\text{th}}$  temperature phase provided that (1) if  $i = 0$ , then  $(\alpha_0, \alpha_1, \dots)$  is an assembly sequence of the 1BlockTAS  $(T, C, \sigma, \tau_0)$  and (2) if  $i > 0$ , then two conditions must be met. The first condition says that there exists  $l$  such that the sequence can be broken up at  $l$  so that the sequence up to index  $l$  is an assembly sequence for the  $(i - 1)^{\text{th}}$  temperature phase. Condition 2 says that the sequence after index  $l$  is a sequence of assemblies such that  $\alpha_j$  can be obtained from  $\alpha_{j-1}$  by either (1) a single tile addition (with combined binding strength of at least  $\tau_i$ ) or (2) by “melting off” a subassembly from  $\alpha_{j-1}$  with a cut of strength  $< \tau_i$ .

An *assembly sequence* in  $\mathcal{B}^*$  is an assembly sequence for the  $i^{\text{th}}$  temperature phase of  $\mathcal{B}^*$  for some  $i \in \mathbb{N}$ . We say that an assembly sequence  $\vec{\alpha}$  in  $\mathcal{B}^*$  *finishes every temperature phase of  $\mathcal{B}^*$*  if  $\vec{\alpha}$  is a finite assembly sequence for temperature stage  $k - 1$  of  $\mathcal{B}^*$  and its final assembly, denoted  $\alpha$ , is  $\tau_{k-1}$ -stable and  $\partial^{\mathcal{B}_{k-1}} \alpha = \emptyset$ . In this case, we call  $\alpha$  a *terminal assembly* and write  $\alpha \in \mathcal{A}_{\square}[\mathcal{B}^*]$ . If  $|\mathcal{A}_{\square}[\mathcal{B}^*]| = 1$ , then we say that  $\mathcal{B}^*$  is *directed*. Given system  $\mathcal{B}^*$ , let  $\tau_{\max} = \max(\tau_0, \dots, \tau_{k-1})$  (i.e. the maximum  $\tau$  value of any stage). We say that  $\mathcal{B}^*$  is  $\tau_{\max}$ -robust if every terminal assembly of every temperature phase is  $\tau_{\max}$ -stable. In a  $\tau_{\max}$ -robust system, no temperature change causes any portions of any assemblies to dissociate and thus growth is monotone. All constructions in this paper are  $\tau_{\max}$ -robust.

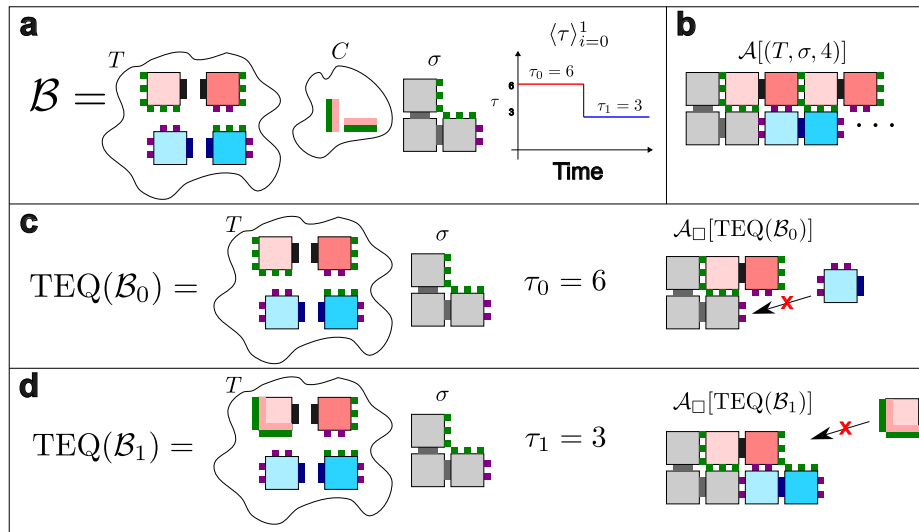
If for every  $\alpha \in \mathcal{A}_{\square}[\mathcal{B}^*]$ ,  $\text{dom } \alpha = S$ , then we say that  $\mathcal{B}^*$  self-assembles the shape  $S$ . Given  $S$ , a connected set of points in  $\mathbb{Z}^2$ , we define a version of  $S$  *scaled by factor  $c$* , and denote it by  $c \cdot S$ , as  $c \cdot S = \{(x, y) \in \mathbb{Z}^2 \mid (\lfloor \frac{x}{c} \rfloor, \lfloor \frac{y}{c} \rfloor) \in S\}$ . If for every  $\alpha \in \mathcal{A}_{\square}[\mathcal{B}^*]$ , there exists  $c \in \mathbb{N}$  such that  $\text{dom } \alpha = c \cdot S$ , then we say that  $\mathcal{B}^*$  self-assembles the shape  $S$  at scale factor  $c$ . Note that the Blocked Tile Assembly Model is defined so that a “blocked glue” on a tile which binds into the assembly immediately binds to any neighboring glue.

## 2.5 BlockTAM System Example

We now provide a simple example of a BlockTAM system to familiarize the reader with key ideas and notation regarding the model. This example is inspired by the system designed in [15]. Part (a) of Figure 2 shows a BlockTAS  $\mathcal{B}$ . In this example, we assume that all blockers bind with the same strength as the glue. If a temperature 4 TAS  $\mathcal{T}$  was defined to have the same tile set and seed as  $\mathcal{B}$ , the assembly shown in part (b) of Figure 2 would be an element of  $\mathcal{A}[\mathcal{T}]$  (that is, the assembly is producible by  $\mathcal{T}$ ). Parts (c) and (d) Figure 2 shows the equivalent blocked tile assembly systems for each temperature phase of  $\mathcal{B}$  and the terminal assembly for each. Note that in part (c), the shown producible assembly of  $\text{TEQ}(\mathcal{B}_0)$  is terminal since the blue tile lacks the strength required to attach. In part (d), the shown assembly is terminal in the context of  $\text{TEQ}(\mathcal{B}_1)$  since the red tile is blocked (meaning it has strength 0) and is consequently unable to attach.

### 3 Encoding Input Values via Temperature Sequences

In this section we present a construction that works in two parts. In the first, the input is a set of temperatures (i.e. positive integers), which we'll denote as  $\Gamma$ . Using this set of temperatures, the first part of the construction returns a tile set, set of blockers, and seed tile that compose a *universal number encoder* that is specific for that set of temperatures.



■ **Figure 2** An illustration of an example BlockTAM system. Part **a** shows its definition. Here, a glue label is represented as the color of the protruding squares on a tile and its strength is the number of protruding squares. For example, the red tile has a green strength 3 glue on its east side. We represent a strength 6 glue by a long rectangle protruding from the edge of the tile. We represent a blocker as two rectangular slices where one rectangle is the same color as the glue which it blocks and the other rectangle is the color of the tile to which the blocker binds. In this example, we assume that all blockers bind with the same strength as the glue. Part **b** shows the assembly formed by the tile set in a traditional TAS without blockers. The tile set grows a width 2 ribbon with alternating red and blue tile segments. Part **c** shows the equivalent blocked tile assembly system for temperature phase 0 of  $\mathcal{B}$  along with its terminal assembly. Note the blue tile is unable to bind to the terminal assembly because the two purple glues are strength 2, summing to 4 which is less than the binding threshold. Part **d** shows the equivalent blocked tile assembly system for temperature phase 1 of  $\mathcal{B}$  along with its terminal assembly. Note the red tile is unable to bind to the terminal assembly because it is blocked (making its strength 0 which is less than the binding threshold).

The input to the second part is an arbitrary value  $n \in \mathbb{N}$ , and the output is a sequence of temperatures over  $\Gamma$  such that a BlockTAM system made using that sequence and the universal number encoder previously produced results in a terminal assembly that is a rectangle with an encoding of the value  $n$  (in a base to be discussed) represented by its northern glues. The design is such that, as previously mentioned, a standard temperature-2 aTAM tile set could then attach to read that encoding as its input and carry out arbitrary algorithmic self-assembly.

A notable feature of our construction is that it is generalized so that it works for any set  $\Gamma$  consisting of at least 3 temperatures, creating a universal number encoder specific to that  $\Gamma$ . This allows for maximal design flexibility in terms of optimizing between tile complexity (i.e. the number of unique tile types required), glue diversity (in terms of numbers of unique sequences and unique strengths), and the temperature complexity (i.e. the number of distinct temperature phases required). In general, the greater the variety of temperatures allowed (i.e. larger  $\Gamma$ ) the greater the tile and glue complexity but the lower the temperature complexity.

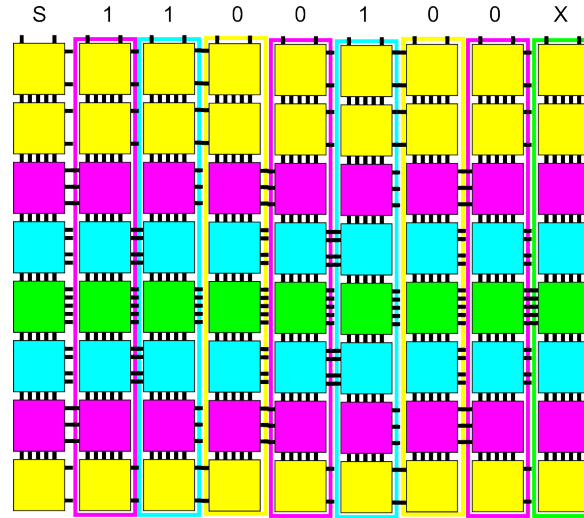
### 3.1 Generation of a universal number encoder

As mentioned, the input is a set of temperatures,  $\Gamma$ . The construction requires at least three distinct allowable temperature values since having only two possible temperatures in a BlockTAM system does not allow any way to create series of temperature sequences that are distinct from each other for different values to encode without using unary encoding, which is infeasible for all but the very smallest values.

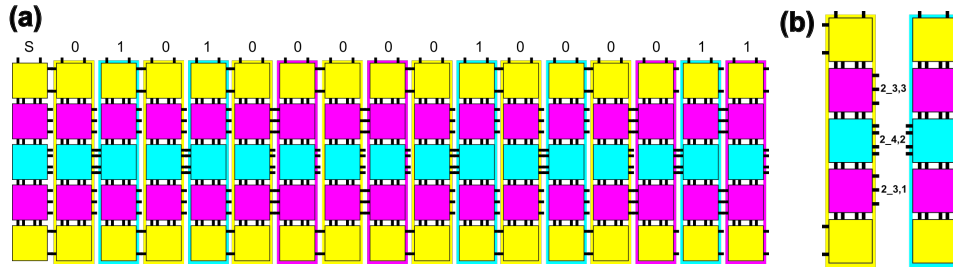
Let  $k = |\Gamma|$  be the number of temperatures, and let  $(\tau_1, \tau_2, \dots, \tau_k)$  be  $\Gamma$  sorted from lowest to highest. For clarity, we will often use  $\tau_{max}$  to refer to  $\tau_k$ . The tile set that we are creating will be specific to the values of  $\Gamma$ , but will be able to build assemblies representing arbitrary values of  $n \in \mathbb{N}$ . Each such  $n$  will be encoded in some base  $b$ , which is determined as follows: if  $k = 3$  then  $b = 2$ , otherwise  $b = k - 2$ . Let  $n \in \mathbb{N}$  be an arbitrary number to eventually be encoded. In general, encoding  $n$  in base  $b$  requires  $\lceil \log_b n \rceil$  values. However, if  $k = 3$  it will be necessary to represent each bit of the base 2 representation of  $n$  using 2 bits (for reasons to be explained later). Therefore, let  $v$  represent the number of values needed to represent the encoding of a given value  $n$  in base  $b$ .

The target assembly will consist of  $c = v + 1$  columns. The glue exposed on the north of the top tile of the first column will be a special “start” glue, and the northern glue of each of the following  $c - 1$  columns will encode subsequent values of the encoding of  $n$ . The north-south glues binding the tiles of a column will all be of strength equal to  $\tau_{max}$  so that each column is  $\tau$ -stable across all temperatures of the system. However, the glues that bind each column to another will consist of varying strengths from  $\langle \tau \rangle$ . Therefore, when glues of any strength  $< \tau_{max}$  are used, it will be necessary for multiple glues of that strength to bind the columns together to ensure stability if/when the system temperature is set to  $\tau_{max}$ .

Let  $h$  be the height of a column (with the determination of the value  $h$  to be explained later). We refer to the bottom location of a column as  $r_0$  and the one above that as  $r_1$ , continuing up to the topmost at  $r_{h-1}$ . Across all columns, the row at each location  $r_i$  for  $0 \leq i < h$  is dedicated to tiles whose east/west glues are of the same strength, which we’ll refer to as  $s_i$ . Furthermore, the binding of any pair of columns to each other will be via one or more glues all of the same strength. To compute  $h$  we first determine how many instances of each temperature in  $\Gamma$  are required to sum to the smallest value  $\geq \tau_{max}$ . We then take the sum of that sum to be  $h$ . For example, if the set of possible temperatures is  $\{2, 3, 4, 5\}$ ,  $h = 3 + 2 + 2 + 1 = 8$  since it takes 3 glues of strength 2 to sum to  $\geq \tau_{max} = 5$ , 2 each of strength 3 and 4, and of course only 1 of strength 5. Thus, every column in a system using those temperature values would be of height  $h = 8$ . (See Figures 3 and 4 for examples.)



■ **Figure 3** An example of the terminal assembly of the universal number encoder for temperatures  $\{2, 3, 4, 5\}$  encoding the number  $100_{10}$  in binary.



■ **Figure 4** (a) An example of the terminal assembly of the number encoding system with temperatures  $\{2, 3, 4\}$  encoding the number  $100_{10}$  in binary. Note that the fact that there are only 3 permissible temperatures requires that each bit of the binary representation of  $100_{10}$  (i.e.  $1100100_2$ ), as well as the end marker, is represented by 2 bits so that the final column can be accurately identified by any tiles that may later grow other the northern-facing encoding. Each 0 is replaced by 00, each 1 by 01, and the end marker is represented by 11. Thus, the full encoded value is “S0101000001000011”. (b) An example pair of columns for a universal number encoder for temperatures  $\{2, 3, 4\}$ . The glue strengths for each position, from bottom to top, are 2, 3, 4, 3, 2. Growth of the column on the left would be initiated by a temperature 2 phase, and the column on the right by a temperature 4 phase (which is clear since each column only has input, i.e. west-facing, glues in the locations specific to those strengths). Note that the left column does not have output (east-facing) glues in the strength 2 position since its own growth is initiated at temperature 2, and similarly the right column does not have a glue in the strength 4 position. Labels of east/west glues between the columns consist of  $t_i\_t_j, r_k$  where  $t_i$  the growth temperature of the left column,  $t_j$  is the growth temperature of the right column, and  $r_k$  is the row index. The label “2\_4, 2” is common to the east/west glues of the blue tiles, allowing the right tile to attach and the right column to then form, assuming the next temperature phase after completion of the left column was at temperature 4. If, for instance, it was temperature 3, tiles unique to the (2, 3) column would instead be unblocked and attach to the pink tiles of the left column, allowing the (2, 3) to grow instead.

With the value  $h$  as the necessary height of each column, we then compute the assignment of each  $s_i$ , that is, which east/west strengths are assigned to each row  $r_i$ . Since these are the glues that bind neighboring columns to each other, we try to assign them to maximize stability. Therefore, we assign the center location to the strongest glue,  $s_{max}$ . Then, since each weaker glue requires two or more locations (so that the multiple copies of each can sum to  $\geq \tau_{max}$ ), we assign them moving outward from the center, alternating up and down for each subsequently weaker glue until all positions are assigned. Examples can be seen in Figures 3 and 4. For instance, Figure 3 shows how the strengths 2, 3, 4, and 5 are assigned, with the ordering from bottom to top being 2, 3, 4, 5, 4, 3, 2, 2.

Since the north/south glues between the tiles of each column are of strength  $s_{max}$ , and now we know the strengths of the east/west glues for the tiles at each height, we can determine how to create the columns necessary to represent the values of the numbers to be encoded. Note that a seemingly straightforward method would be to create one unique column for each value that needs to be represented to the north. However, then it is not possible for the encoding to contain the same symbol in two consecutive positions (e.g. the binary representation of  $14_{10} = 1110_2$ , which has two pairs of consecutive 1s) since then the same column would have to be able to connect to a copy of itself, which would result in “pumping” into an infinite sequence of 1s. A relatively simple fix to that problem would be to have a column that doesn’t represent any value that could then be located between each value-encoding column. However, this would double the number of temperature phases needed, and that is the primary metric that we are trying to minimize. Therefore, we use the following scheme which compresses the temperature sequence at the expense of requiring slightly more unique tile types (approximately double).

Our method of creating columns for encoded values works by permuting over all pairs of (non-identical) temperatures from  $\Gamma$ . Let  $t = |\Gamma|$  be the number of unique temperatures. For each  $\tau_i \in \Gamma$ , there are  $t - 1$  temperatures  $\tau_j \in \Gamma$  such that  $t_i \neq t_j$ . For each such pair we create exactly one unique column and we assign a value for that column to encode from the set  $\{0, 1, \dots, b - 1\} \cup \{“X”\}$ . (Note that, in the case where  $t = 3$ , no “X” character is explicitly used but instead the ending condition is encoded using a special sequence of bits, to be discussed in Section 3.2.) This mapping from each pair of temperatures to an encoded value is referred to as the `column_transition_map` (and the code used to generate it can be seen in Listing 1 in the Appendix). The resulting map ensures that any encoded symbol can follow any other, except for the “X” symbol which is the final symbol to be encoded by the last temperature phase. (Again, when  $t == 3$ , there is no “X” symbol, just 0s and 1s, which will each be able to follow each symbol.) Note that the column that contains the seed tile is unique, composed of tiles that only appear in that column, and its northernmost glue encodes the start value “S”. The glue encoding “S” is of strength 2, and all other value encoding glues are of strength 1. Once the number encoding assembly is terminal, the system temperature can be set to 2, allowing for a standard aTAM tile set to attach and grow across from left to right, reading the encoded value then proceeding with arbitrary algorithmic behavior.

To generate the tile type for each location of each column, we use the previously computed glue strengths for each side of each location  $r_i$  and the information in the `column_transition_map`. For each temperature pair  $(t_i, t_j)$ , again where  $t_i \neq t_j$ , we generate a set of  $h$  distinct tile types to create a unique column. The purpose of the  $(t_i, t_j)$  column is to (1) encode the value `column_transition_map`( $t_i, t_j$ ) in the north glue of its northernmost tile, (2) to have its growth initiated by a temperature phase of temperature  $t_2$ , and (3) to grow immediately to the right of a column whose growth was initiated by a temperature



phase of temperature  $t_1$ . To effect this, each north/south glue label between two tiles of a column are specific to the column type, i.e.  $(t_i, t_j)$ , and location  $r_i$ , e.g.  $t_i\_t_j, r_k$  for the glue pair between the  $k$ th and  $k + 1$ th tile in the column for  $(t_i, t_j)$ . On the east, output side of the tile in location  $r_k$  is a glue of strength  $s_k$  (which is the strength value assigned to that row) for all locations except that for which  $s_k == t_j$ , which has no east-facing glue. As previously mentioned, this is to prevent the column from being able to initiate growth of another column to its right at the same temperature at which it grew. The label of each such glue will be  $t_j\_s_k, k$ , so that it can bind to a tile in the  $k$ th location of the column for temperature sequence  $(t_j, s_k)$  (assuming that the next phase is at temperature  $s_k$ ). Figure 4 shows an example of these glues.

In this way, the full tile set is constructed. Next, the blocker set is created. This simply consists of a blocker for every west-facing glue that is strength  $s - 1$  if the glue's strength is  $s$ . This ensures that those “input” glues, which initiate the growth of each new column, are only active during phases of the temperatures for which their columns were designed, and growth proceeds with a single column forming for each phase.

With the designated seed tile being a tile of the type designed for the middle of the leftmost column, the resulting tile set, blocker set, and seed is a universal number encoder for  $\Gamma$ . We next describe how to, given a specific value of  $n$  to encode, compute the temperature sequence necessary for a BlockTAS to assemble an assembly encoding  $n$ .

### 3.2 Temperature sequence generation

The input to this portion of the construction is a number  $n \in \mathbb{N}$  and a universal number encoder from the previous portion. From that encoder, we can determine the base  $b$  in which to encode  $n$ . Additionally, we can know which of two paths to take to compute the encoding  $e$  of  $n$ : If the number of temperatures,  $t$ , is 3, then  $e$  will consist of a “doubled” base 2 encoding, otherwise  $e$  will simply be the encoding of  $n$  in base  $b$  (which is set to  $t - 2$ ), with the character “X” appended to the end. Note that in either case, the encoding will begin with the “S” symbol which is encoded by the column with the seed tile.

To compute the doubled encoding for the case of  $t == 3$ , we note that with only three temperatures to select from, once a given phase of assembly completes, which is necessarily at some  $t_i \in \Gamma$ , then there are only two choices for some other  $t_j \in \Gamma$  such that  $t_i \neq t_j$ . Because of this, we can't directly encode a symbol to mark the end of sequence, which makes it difficult or impossible for an algorithmic (aTAM-style) tile set to then read the encoded value and effectively use it as input that controls its algorithmic behavior. (It would either have to pause at the end of the assembly “waiting” for a stop symbol, or instead be able to grow on its own past the last symbol without cooperating with the assembly which means that at other points of growth it could “ignore” the assembly and grow without cooperation, yielding an invalid input. This is a well-known problem in the domain of tile-based self-assembly.) Therefore, we utilize the method of representing each bit of the binary encoding of  $n$  as follows. We represent each 0 by 00, and each 1 by a 01. We build our encoding in this way, and to indicate the end of the sequence, after all bits have been encoded this way, we append two consecutive 1s. In this way, a tile set growing over the north of the terminal assembly and reading the encoding will be able to detect when it has read the final position.

Now with the encoded string  $e_0, e_1, \dots, e_v$ , we convert that into a series of temperatures over  $\Gamma$ . Because there are no blockers for tiles in the column containing the seed, that column grows at any temperature in  $\Gamma$ , as all north/south glues are of strength  $\tau_{max}$ . This means that no separate temperature phase is necessary to account for the placement of the “S” symbol and the first temperature phase is used to grow the column of the first encoded value of  $n$ .

Additionally, the east-facing, output glues of the seed column use the same labels as a column that grew at the second highest temperature so that new glues do not need to be designed for tiles to attach to that column. Thus, for the first, and then each subsequent, encoded value we simply iterate over the values  $e_0, e_1, \dots, e_v$  and examine the `column_transition_map` keeping track of which temperature  $t_i$  was used to grow the previous column (pretending it was the second largest temperature if that was the seed column). For each  $e_m$ , We find the unique  $t_j$  such that `column_transition_map`( $t_i, t_j$ ) =  $e_m$ . We then append that  $t_j$  as the next temperature in the sequence. Iterating through the full encoded string, the resulting temperature sequence is the exact sequence that will cause the universal number encoder to build an assembly of columns expressing the encoded value of  $n$  in its northern glues. (For details of a few examples of systems, please see Section A.1 of the Appendix.)

## 4 Using Temperature Changes to Steer Assembly Through Shapes

The most common goal in the design of a self-assembling system is for the resulting system to deterministically form into assemblies of a single, target shape. However, the goal of accurate shape-building must usually be balanced with competing factors such as minimizing the number of unique tile types required and the scale factor at which the shape is formed. In this section, we present a *universal shape builder* consisting of a fixed tile set, blocker set, seed tile, and temperature set that is capable of being utilized to build any arbitrary, finite (and necessarily connected) shape. That is, given an arbitrary shape  $S$ , a temperature sequence exists that causes the BlockTAM system composed of these components to self-assemble a scaled version of  $S$ , i.e.  $S^c$ . If  $S$  has a Hamiltonian path, our construction assembles  $S$  at scale factor 3. If not, our construction assembles it at scale factor 6. It has valid temperature set  $\{2, 3, 4\}$ , consists of 145 tile types with glues of strengths 1, 2, 3, and 4, and has 16 strength-3 and 16 strength-4 blockers. There is a single designated tile type for the seed.

We first explain the algorithm that takes as input an arbitrary shape  $S$  and outputs a path through it. We then explain the tile set and how it is capable of following such a path. Finally, we demonstrate how that path can be converted into a series of temperatures to drive the universal shape constructor along it.

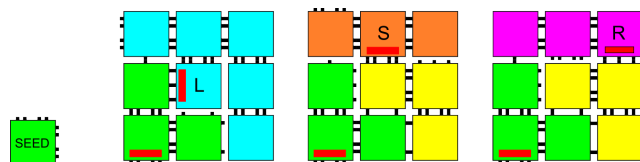
### 4.1 Finding a “turtle graphics” path

In [21] it was shown how, beginning with an arbitrary input shape  $S$ ,  $S$  can be scaled by a factor of two to  $S^2$ , and then a Hamiltonian path through  $S^2$  can be created that visits each point in  $S^2$  exactly once. (Note that any shape  $S$  that already has a Hamiltonian path through it does not require the scaling by two, but we’ll assume that, in the worst case for an arbitrary  $S$ , the scaling is required.) Using the technique of [21] we can convert that path into a series of “turtle graphics” instructions. That is, starting on the first point of the path, we simply follow a series of instructions taken from the set  $\{\text{Straight}, \text{Right}, \text{Left}\}$  (which we’ll abbreviate as  $\{S, R, L\}$ ) to visit every point on the path (and thus in  $S^2$ ) exactly once. Let  $p = |S|$  be the number of locations, or *pixels*, in the original shape  $S$ . The output of this step is an ordered list  $L \in \{S, R, L\}^{4p-1}$ , that is, a list of  $4p - 1$  directions that allow one to begin at one point in  $S^2$  and visit every other (since each pixel in  $S$  is composed of 4 in  $S^2$ , and we don’t need an instruction to get to the pixel from which the path begins).

## 4.2 The universal tile set

We now describe the tile set and blockers of the universal shape builder by explaining how they are able to follow such a path. The assembly that follows  $L$  is composed of *macrotiles* that are  $3 \times 3$  squares of tiles. That is, each pixel of  $S^2$  is represented by a  $3 \times 3$  square of tiles (yielding an overall scale factor of 6 from  $S$ ). Starting from the seed tile and with a temperature phase of  $\tau = 4$ , the first macrotile grows the *base* portion of a macrotile, which is composed of 3 tiles (that can be seen as green in the macrotiles in Figure 5). The macrotile then becomes terminal, until the next temperature phase begins. A phase of 3 causes the blue tiles to become unblocked and attach, finishing the macrotile and orienting the output glue toward the left. A phase of 2 causes the blue and pink tiles to be blocked but allows the yellow tiles to attach, after which the macrotile becomes terminal. Then, a next temperature phase of  $\tau = 4$  allows the orange tiles to become unblocked and attach, exposing the macrotile's output glue straight ahead. A phase of  $\tau = 3$  would instead cause the orange tiles to be blocked and the pink tiles to be unblocked and thus attach. They expose a right-facing output glue for the macrotile. Refer to Figure 5 to see that all portions of macrotile growth are fully stabilized before the temperature can increase.

The result of the temperature phases described are that the macrotile completes, is fully stable at all temperatures (at the end of each intermediate temperature phase and the final), and exposes exactly one output glue of strength 4 that can initiate the growth of the next macrotile. While growth of the next macrotile begins with a temperature phase of  $\tau = 4$ , note that in the case of the direction being  $S$ , the final phase of growth of the current macrotile is also at  $\tau = 4$ . In this case, the base portion of the next macrotile forms during the same phase that the current macrotile completes. This is not a problem and is easily accounted for when designing our temperature sequence.



**Figure 5** The seed tile (left) and 3 basic macrotiles of the universal shape builder. Glue that have blockers are indicated by red rectangles. Each macrotile first grows its base portion, composed of the same three tiles (green) in each. This growth occurs at temperature 4, and once the three green tiles have attached, the assembly is terminal. The first macrotile grows from the seed tile as its bottom-left tile. The others begin with the bottom left green tile attaching to a previously formed macrotile. After the base, which is common to each of the macrotiles, is terminal, a new temperature phase begins to allow subsequent growth. If the new temperature is 3, the blue tiles attach, effecting a left turn and completing the macrotile, and the assembly becomes terminal at that temperature. If it is 2, then the three yellow tiles attach, before growth halts. Then, a new temperature phase begins. If it is temperature 4, the orange tiles attach to cause the next macrotiles to grow straight ahead. If it is 3, the pink tiles attach for a right turn. In the cases where the yellow tiles attach, note that two of them attach to the bottom-right green tile, each with a strength-2 bond. This creates a minimum cut of total strength 4 in the binding graph meaning that the yellow tiles are stably attached even at the highest temperature, which is 4. When the blue and pink tiles attach with initial strength-3 bonds, they each also grow so that their leftmost tiles form a single-strength bond with the upper green tile to again guarantee a minimum cut of total strength 4. Therefore, in all three scenarios,  $S$ ,  $R$ , and  $L$ , by the end of the temperature phases that complete the growth of the macrotile, all portions are 4-stable.

The final, and relatively straightforward, aspect of the construction is necessary to handle growth of macrotiles following one or more  $R$  or  $L$  turns. For each clockwise rotation of 90, 180, and 270 degrees, we simply create a new copy of each tile making up any of the three versions of the macrotile. We give each copy new glue labels that are specific for that rotation, with the only glue labels that are shared between tiles intended for different rotations being the four glues on the exteriors. These glue labels are designed so that each macrotile output glue is specific to the input glue (i.e. the glue exposed by the green tile) of the correctly oriented version. Since the output locations for  $R$  and  $L$  turns are not symmetrically located on the macrotile exterior, it is also necessary to make a reflected version of the basic macrotile, and copies of that for each of the rotations. (An example of the full set of rotations and reflections can be seen in Figure 7 in the Appendix.) We use an additional glue prefix of “ $x$ ” for all of the labels of glues on tiles of the reflected copies so that tiles intended for a macrotile of one rotation and/or reflection are not able to bind inside of macrotiles with different rotations and/or reflections, but again we make sure that each rotated and/or reflected macrotile’s output glue labels correctly match those of the properly rotated and/or reflected macrotile input glues to match their output directions (relative to their orientations). (A complete simple example can be seen in Figure 8 of the Appendix.)

### 4.3 Generating the temperature sequence

From the above description, it is clear that the universal shape builder can correctly follow an arbitrary turtle graphics path. Thus, all that remains is to show how a shape’s path can be turned into a proper temperature sequence. This is also quite trivial, as we have already seen which temperature sequences are necessary for each movement of  $S$ ,  $R$ , or  $L$ . Since we require that every step is either  $S$ ,  $R$ , or  $L$ , if the path through  $S^2$  starts with a backward step we simply rotate the shape so that the first step is  $S$ ,  $R$ , or  $L$ , which doesn’t change the shape since shapes are invariant up to rotation. The first temperature is 4, allowing the base of the first macrotile to grow. Then, for  $L$  a 3 is appended; for  $S$ , a 2 and 4 are appended; and for  $R$  2 and 3. To initiate the next macrotile, if the last direction was  $L$  or  $R$ , we append 4, but if it was  $S$  we don’t need to do that. The final main consideration is for handling reflected macrotiles. We keep track of each turn that uses the  $L$  version of the macrotile, as that causes a flip between reflected and non-reflected versions, and note that, when in a reflected macrotile it is an  $R$  turn that uses the blue  $L$  tiles and flips the reflection status. Finally, to make sure that the final macrotile is complete, after a  $\tau = 4$  phase to grow its base, we include a  $\tau = 3$  phase to cause it to complete. (The code used to compute the temperature sequence can be seen in Listing 2 in the Appendix.) We’ve made software freely available that allows one to draw a path, save it as a BlockTAM system [13], then simulate it in a web-based simulator [8]. (Figure 9 of the Appendix shows a screenshot.)

Thus, we have demonstrated a universal shape builder. While only requiring 145 tile types and 32 blockers, it requires an average of just over 2 temperature phases per pixel of  $S^2$  (or  $S$  if scaling isn’t required). While infeasible for experimental implementations of large shapes, the design techniques may provide tools useful for other and improved constructions.

---

### References

- 1 Gagan Aggarwal, Michael H. Goldwasser, Ming-Yang Kao, and Robert T. Schweller. Complexities for generalized models of self-assembly. In *Proceedings of ACM-SIAM Symposium on Discrete Algorithms*, 2004.
- 2 Florent Becker, Daniel Hader, and Matthew J Patitz. Strict self-assembly of discrete self-similar fractals in the abstract tile assembly model. In *Proceedings of the 2025 Annual ACM-SIAM Symposium on Discrete Algorithms (SODA’25), New Orleans, USA*, pages 2387–2466. SIAM, 2025. doi:10.1137/1.9781611978322.80.

- 3 Junghuei Chen and Nadrian C Seeman. Synthesis from dna of a molecule with the connectivity of a cube. *Nature*, 350(6319):631–633, 1991.
- 4 Qi Cheng, Gagan Aggarwal, Michael H. Goldwasser, Ming-Yang Kao, Robert T. Schweller, and Pablo Moisset de Espanés. Complexities for generalized models of self-assembly. *SIAM Journal on Computing*, 34:1493–1515, 2005. doi:10.1137/S0097539704445202.
- 5 Constantine Evans, Angel Cervera Roldan, Trent Rogers, and Damien Woods. Tile blockers as a simple motif to control self-assembly: kinetics and thermodynamics. Unpublished.
- 6 Ashwin Gopinath, Evan Miyazono, Andrei Faraon, and Paul WK Rothemund. Engineering and mapping nanocavity emission via precision placement of DNA origami. *Nature*, 2016.
- 7 Daniel Hader, Aaron Koch, Matthew J. Patitz, and Michael Sharp. The impacts of dimensionality, diffusion, and directedness on intrinsic universality in the abstract tile assembly model. In Shuchi Chawla, editor, *Proceedings of the 2020 ACM-SIAM Symposium on Discrete Algorithms, SODA 2020, Salt Lake City, UT, USA, January 5-8, 2020*, pages 2607–2624. SIAM, 2020. doi:10.1137/1.9781611975994.159.
- 8 Daniel Hader and Matthew J. Patitz. WebTAS (beta): A browser-based simulator for tile-assembly models, 2025. URL: [http://self-assembly.net/software/WebTAS\\_beta/WebTAS\\_beta-latest/](http://self-assembly.net/software/WebTAS_beta/WebTAS_beta-latest/).
- 9 Yonggang Ke, Luvena L Ong, William M Shih, and Peng Yin. Three-dimensional structures self-assembled from DNA bricks. *Science*, 338(6111):1177–1183, 2012.
- 10 James I. Lathrop, Jack H. Lutz, Matthew J. Patitz, and Scott M. Summers. Computability and complexity in self-assembly. *Theory Comput. Syst.*, 48(3):617–647, 2011. doi:10.1007/s00224-010-9252-0.
- 11 James I. Lathrop, Jack H. Lutz, and Scott M. Summers. Strict self-assembly of discrete Sierpinski triangles. *Theoretical Computer Science*, 410:384–405, 2009. doi:10.1016/J.TCS.2008.09.062.
- 12 Dionis Mineev, Christopher M Wintersinger, Anastasia Ershova, and William M Shih. Robust nucleation control via crisscross polymerization of highly coordinated DNA slats. *Nature Communications*, 12(1):1–9, 2021.
- 13 Matthew J. Patitz and Trent Rogers. Blocked TAM wiki page and software downloads, 2025. URL: [http://self-assembly.net/wiki/index.php/Blocked\\_Tile\\_Assembly\\_Model\\_\(BlockTAM\)](http://self-assembly.net/wiki/index.php/Blocked_Tile_Assembly_Model_(BlockTAM)).
- 14 Matthew J. Patitz and Scott M. Summers. Self-assembly of decidable sets. *Natural Computing*, 10(2):853–877, 2011. doi:10.1007/s11047-010-9218-9.
- 15 Trent Rogers, Constantine Evans, and Damien Woods. Controlling self-assembly with blockers: programming nanostructure size with temperature. Unpublished.
- 16 Trent Rogers, Constantine Evans, and Damien Woods. Controlling self-assembly with blockers: tunable nucleation in growth conditions. Unpublished.
- 17 Paul W. K. Rothemund. Folding DNA to create nanoscale shapes and patterns. *Nature*, 440(7082):297–302, March 2006. doi:10.1038/nature04586.
- 18 Paul W. K. Rothemund and Erik Winfree. The program-size complexity of self-assembled squares (extended abstract). In *STOC '00: Proceedings of the thirty-second annual ACM Symposium on Theory of Computing*, pages 459–468, Portland, Oregon, United States, 2000. ACM. doi:10.1145/335305.335358.
- 19 Nadrian C Seeman. Nucleic acid junctions and lattices. *Journal of theoretical biology*, 99(2):237–247, 1982.
- 20 David Soloveichik and Erik Winfree. Complexity of self-assembled shapes. *SIAM Journal on Computing*, 36(6):1544–1569, 2007. doi:10.1137/S0097539704446712.
- 21 Scott M. Summers. Reducing tile complexity for the self-assembly of scaled shapes through temperature programming. *Algorithmica*, 63(1-2):117–136, June 2012. doi:10.1007/s00453-011-9522-5.
- 22 Ning Wang, Chang Yu, Tingting Xu, Dan Yao, Lingye Zhu, Zhifa Shen, and Xiaoying Huang. Self-assembly of dna nanostructure containing cell-specific aptamer as a precise drug delivery system for cancer therapy in non-small cell lung cancer. *Journal of Nanobiotechnology*, 20(1):1–19, 2022.

- 23 Erik Winfree. *Algorithmic Self-Assembly of DNA*. PhD thesis, California Institute of Technology, June 1998.
- 24 Damien Woods, David Doty, Cameron Myhrvold, Joy Hui, Felix Zhou, Peng Yin, and Erik Winfree. Diverse and robust molecular algorithms using reprogrammable dna self-assembly. *Nature*, 567(7748):366–372, 2019. doi:10.1038/S41586-019-1014-9.

## A

 Technical Appendix

This Technical Appendix contains additional technical details related to the results in the main body of the paper.

■ **Listing 1** Code used to generate the transition mapping of temperature pairs to encoded symbols for the universal number encoder.

```
def get_column_transition_map(temp_list, base, final_temp):
    temp_list = sorted(temp_list)
    column_transition_map = {}

    for temp1 in temp_list:
        if temp1 != final_temp:
            count = 0
            for temp2 in temp_list:
                if temp1 != temp2:
                    if count < base:
                        column_transition_map[(temp1,temp2)] = count
                        count += 1
                    else:
                        column_transition_map[(temp1,temp2)] = 'X'

    return column_transition_map
```

### A.1 Number encoder examples

We now provide examples with a few details about universal number encoders for two different sets of temperatures and how they each encode a pair of numbers.

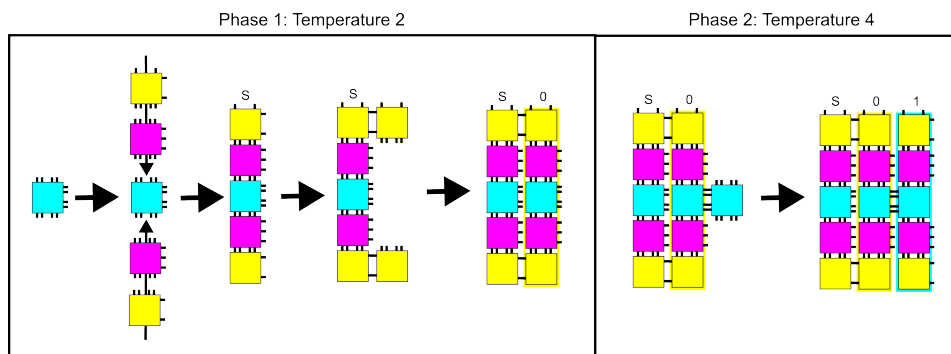
Let  $\Gamma_1 = \{2, 3, 4\}$  and  $\Gamma_2 = \{2, 3, 4, 5\}$  be two sets of temperatures. Using the construction, the universal number encoders produced have the characteristics listed in the first two rows of Table 1. Additionally, Figure 6 demonstrates how the first three columns of the first encoder would grow for the encoded value of  $100_{10}$ , and Figures 3 and 4 show their terminal assemblies.

■ **Table 1** Comparisons of universal number encoding systems.

Temperatures	Number Tile Types	Column Height	Base	Temps(100)	Temps(10000)
2,3,4	35	5	2	16	30
2,3,4,5	80	8	2	8	15
2,3,4,5,6	170	10	3	6	10
2,3,4,5,6,7	364	14	4	5	8

### A.2 Additional details of the universal shape builder

Here we provide a few supplementary figures for the universal shape builder construction.



■ **Figure 6** Depiction of the growth during the first 2 temperature phases of the universal number encoder for temperatures  $\{2, 3, 4\}$  when encoding the number  $100_{10}$ , whose full encoding is the string “S0101000001000011” which requires the temperature sequence  $[2, 4, 2, 4, 2, 3, 2, 3, 2, 4, 2, 3, 2, 3, 4, 3]$ . The growth of phase 1, at temperature 2, begins from the seed tile and terminates once the first column (which will ultimately be the leftmost of the assembly) and second column are complete and terminal (at that phase’s temperature). (Note that phase 1 is a special case in which two columns grow, but during all other phases only a single additional column grows.) Phase 2 begins from those two completed columns, and in this example its temperature is 4, so only tiles with strength-4 west-facing (input) glues are unblocked during this phase and thus able to attach, allowing the third column to then complete the growth that begins from the blue tile. (Note that the only glues that have blockers are west-facing glues of columns.). The terminal assembly of phase 2 consists of the first three completed columns of which each pair is bound together by exactly the necessary number of glues of the respective strengths to be stable at the highest system temperature, in this case 4.

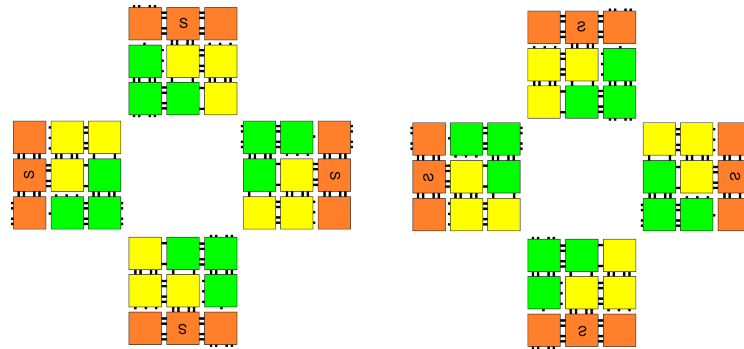
■ **Listing 2** Code used to generate the temperature sequence for a turtle graphics path

```
def get_temperature_sequence_for_path(path):
    temperature_sequence = []
    B = [4]
    L = [3]
    R = [2, 3]
    S = [2, 4]

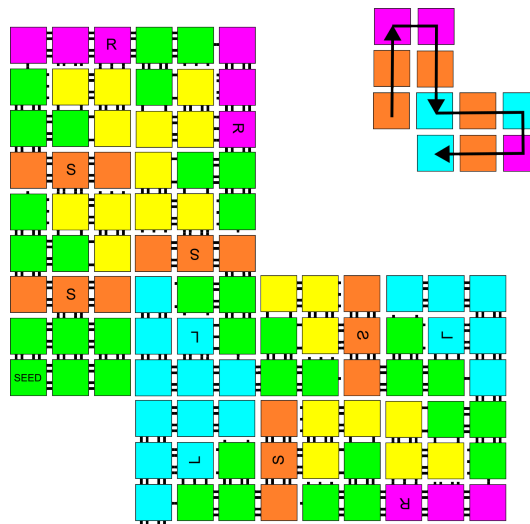
    temperature_sequence += B
    b_reflected = False
    for i in range(len(path)):
        move = path[i]
        if move == 'S':
            temperature_sequence += S
        elif move == 'R':
            if b_reflected:
                temperature_sequence += L
                b_reflected = False
            else:
                temperature_sequence += R
        elif move == 'L':
            if b_reflected:
                temperature_sequence += R
            else:
                temperature_sequence += L
                b_reflected = True
        if (move in ['L', 'R']):
            temperature_sequence += B
    temperature_sequence += L
    return temperature_sequence
```



## 8:20 Leveraging Blockers for Temperature Programming



**Figure 7** Macrotiles used in the universal shape builder. (Left) The four rotated versions of the *S* type of macrotile, (Right) the four rotated versions of the reflected type of macrotile.



**Figure 8** Example shape building example.



**Figure 9** An example of a (poorly drawn) turtle-shaped turtle graphics path drawn, saved, and then simulated in the BlockTAM.

# Synchronous Versus Asynchronous Tile-Based Self-Assembly

Florent Becker ✉ 

Laboratoire d'Informatique Fondamentale d'Orléans (UR4022), Université d'Orléans, France

Phillip Drake ✉ 

University of Arkansas, Fayetteville, AR, USA

Matthew J. Patitz ✉ 

University of Arkansas, Fayetteville, AR, USA

Trent A. Rogers ✉ 

University of Arkansas, Fayetteville, AR, USA

---

## Abstract

In this paper we study the relationship between mathematical models of tile-based self-assembly which differ in terms of the synchronicity of tile additions. In the standard abstract Tile Assembly Model (aTAM), each step of assembly consists of a single tile being added to an assembly. At any given time, each location on the perimeter of an assembly to which a tile can legally bind is called a frontier location, and for each step of assembly one frontier location is randomly selected and a tile is added. In the Synchronous Tile Assembly Model (syncTAM), at each step of assembly every frontier location simultaneously receives a tile.

Our results show that while directed, non-cooperative syncTAM systems are capable of universal computation (while directed, non-cooperative aTAM systems are known not to be), and they are capable of building shapes that can't be built within the aTAM, the non-cooperative aTAM is also capable of building shapes that can't be built within the syncTAM even cooperatively. We show a variety of results that demonstrate the similarities and differences between these two models.

**2012 ACM Subject Classification** Theory of computation → Models of computation

**Keywords and phrases** self-assembly, noncooperative self-assembly, models of computation, tile assembly systems

**Digital Object Identifier** 10.4230/LIPIcs.DNA.31.9

**Supplementary Material Software:** [http://self-assembly.net/software/syncTAM\\_examples/syncTAM\\_examples-source-2025-04-22-21-40-28.zip](http://self-assembly.net/software/syncTAM_examples/syncTAM_examples-source-2025-04-22-21-40-28.zip)

**Funding** *Florent Becker*: Research made at a public, state-funded university, thanks to the guarantee of full academic freedom and perennial research funding.

*Phillip Drake*: This author's work was supported in part by NSF grant 2329908.

*Matthew J. Patitz*: This author's work was supported in part by NSF grant 2329908.

## 1 Introduction

At the macroscale, humans have mastered the construction of structures with desired spatial and temporal properties. This mastery has provided us with not only entertainment, but also the tools necessary for our survival. As civilization shifts towards a more virtual world and the threats to our survival evolve, a mastery of matter at the macroscale is no longer sufficient. Unfortunately, controlled interaction with the universe at the nanoscale presents significant hurdles. We face major difficulties in observing the universe at the nanoscale, let alone manipulating matter at the this scale. Just as axiomatic models (such as the natural or real numbers) have greatly aided in our understanding (and manipulating) of the universe at the macroscale, axiomatic models of self-assembly have proven successful



© Florent Becker, Phillip Drake, Matthew J. Patitz, and Trent A. Rogers;  
licensed under Creative Commons License CC-BY 4.0

31st International Conference on DNA Computing and Molecular Programming (DNA 31).

Editors: Josie Schaeffer and Fei Zhang; Article No. 9; pp. 9:1–9:21

Leibniz International Proceedings in Informatics



LIPICs Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

in allowing us to understand (and manipulate) the universe at the nanoscale. Indeed, such axiomatic models have already enabled the design and implementation of a variety of futuristic nanotechnologies which have a wide range of uses: molecular computers [28], drug delivery [26], and lithographically-patterned electronic/photonic devices [13].

Here, we focus on tile-based, axiomatic models of self-assembly. In these models, the atomic components are square “tiles” which have a “glue” on each of their edges. Growth begins from an initial “seed” assembly and proceeds via single tile attachments. On one end of the spectrum, these single tile attachments occur asynchronously and nondeterministically in a model called the abstract Tile Assembly Model (aTAM). And, on the other end of the spectrum, these single tile attachments occur completely synchronously so that every “frontier” location receives a tile at every step. The fully synchronous model has yet to be studied, so in this paper we introduce a formalization of this model called the synchronous tile assembly model (syncTAM). The syncTAM provides a formal framework for studying the power and limitations of fully-synchronous systems and gives us insight into how we can leverage these synchronization mechanisms to imbue our experimental systems with more power than their asynchronous counterparts.

Perhaps surprisingly, the aTAM gives rise to a variety of complex behaviors. For example, the aTAM is known to be computationally universal [27], intrinsically universal [6], and capable of assembling complex, fractal structures [2]. But, it also has its limitations. For example, the aTAM is limited in the types of interesting, infinite shapes it can assemble [2] and the class of directed, temperature-1 aTAM systems is incapable of computing [19]. Here we investigate whether the syncTAM can overcome some of the limitations of the aTAM and whether a fully-synchronous attachment regime introduces its own set of limitations.

Current ubiquitous technology is capable of physically realizing approximations of asynchronous models of self-assembly, such as the aTAM, but it is unclear if a fully synchronous model can be physically realized in an efficient manner. Inefficient implementations of more synchronous models currently exist. For example, in [25], experimentalists demonstrated the ability to implement a “staged” system where a target structure is assembled in stages by assembling different parts of the structure in separate test tubes and then combining the results to obtain a new substructure which feeds into the next stage. This staging can be thought of as a synchronization mechanism since (if we wait long enough) every assembly will be terminal before it is combined with others. To implement a fully synchronous model using this approach a stage is required for every step making this approach so tedious and laborious it is infeasible.

This work seeks to answer the following question: Is exploring ways to efficiently implement a more synchronous model (than the existing models which are able to be physically realized) a worthy endeavor? We seek to answer this question by comparing the relative powers of an asynchronous model of self-assembly (the aTAM) to a synchronous model of self-assembly (the syncTAM) in terms of shape-building power and computational power.

While the asynchronous mode of tile attachment is the default for self-assembly literature, the synchronous mode of update is the default update model for cellular automata. Fatès, Régnault, Schabanel and others [5, 9, 10, 23] have shown that introducing asynchronism in even the simplest cellular automata vastly alters their behavior. Likewise, Paulevé and Sené [22] show the importance of the choice of update mode in boolean networks. Due to its synchronous nature and the fact that a tile attachment depends on the surrounding neighbors, the class of syncTAM systems can be thought of as a restricted class of freezing cellular automata (CA). Indeed, the class of freezing CA only allows states to increase (according to some ordering), so restricting this class to only allow a single state change and a radius-1

Von Neumann neighborhood yields a class which roughly corresponds to syncTAM systems – it is a slightly more permissive superclass of the syncTAM. Consequently, the results from these models are transitive which provides an interesting mathematical motivation for the study of SyncTAM systems.

Just as comparing asynchronous and synchronous models has been a central theme in cellular automata, our results seek to compare the powers and limitations of the syncTAM with respect to the aTAM. To this end, we begin with formally defining the models used in this paper in Section 2, as well as the formal notions we use to compare the models. Section 3 states the elementary connections between the behavior of the same system in the aTAM versus the syncTAM. In Section 4, we show that there exists shapes that can be assembled in the temperature-1 syncTAM which cannot be assembled at any scale factor in the aTAM (at any temperature). Likewise, that section exposes a set of shapes which can be assembled nondeterministically by an aTAM system but cannot be assembled at any scale factor by a nondeterministic syncTAM system. In Section 5, we show that like the directed aTAM, the directed syncTAM is computationally universal, but unlike the directed, temperature-1 aTAM [19], the syncTAM is computationally universal at temperature 1. To show this, we show that the any temperature-2 compact zig-zag system in the aTAM, can be simulated by a temperature-1 syncTAM system. And, since the class of aTAM temperature-2 compact zig-zag systems is computationally universal [1], the class of temperature-1 syncTAM systems is computationally universal. While previous results about computing at temperature 1 in the aTAM or adjacent models led to very poorly connected assemblies [4, 11, 12, 16, 21], we also show in Section 5.2 that the syncTAM can compute using more well-connected structures.

To make it easier to understand and visualize our results, we have created a web-based simulator for the syncTAM [15] and have implemented all of the systems used in our results [8]. These can all be accessed from [http://self-assembly.net/wiki/index.php/Synchronous\\_Tile\\_Assembly\\_Model\\_\(syncTAM\)](http://self-assembly.net/wiki/index.php/Synchronous_Tile_Assembly_Model_(syncTAM)).

## 2 Preliminary Definitions and Models

In this section we define the models and terminology used throughout the paper.

### 2.1 The abstract Tile-Assembly Model

We work within the abstract Tile-Assembly Model [27] in 2 and 3 dimensions. The abbreviation *aTAM* refers to the 2D model. These definitions are borrowed from [14] and we note that [24] and [17] are good introductions to the model for unfamiliar readers.

Let  $\mathbb{N}$  be the set of nonnegative integers, for  $l, u \in \mathbb{Z}$ , let  $\llbracket l, u \rrbracket = \{k \in \mathbb{Z} \mid l \leq k < u\}$ . For  $n \in \mathbb{N}$ , let  $[n] = \llbracket 0, n \rrbracket = \{0, 1, \dots, n-2, n-1\}$ .

Fix  $\Sigma$  to be some alphabet with  $\Sigma^*$  its finite strings. A *glue*  $g \in \Sigma^* \times \mathbb{N}$  consists of a finite string *label* and non-negative integer *strength*. There is a single glue of strength 0, referred to as the *null* glue. A *tile type* is a tuple  $t \in (\Sigma^* \times \mathbb{N})^4$ , thought of as a unit square with a glue on each side. A *tile set* is a finite set of tile types. We always assume a finite set of tile types, but allow an infinite number of copies of each tile type to occupy locations in the  $\mathbb{Z}^2$  lattice, each called a *tile*. A *glue set*  $G_T$  is the set of all glues associated with a tile set  $T$ . Given a direction  $d \in \{N, E, S, W\} = \mathcal{D}$  and a tile set  $T$ ,  $G_{T,d}$  is the set of all glues on the edges of tiles in direction  $D$  and  $g_{t,d}$  is the glue  $g$  on tile  $t$  in direction  $D$ . We write  $\text{str}_t(d)$  to denote the strength of  $g_{t,d}$ .

Given a tile set  $T$ , a *configuration* is an arrangement (possibly empty) of tiles in the lattice  $\mathbb{Z}^2$ , i.e. a partial function  $\alpha : \mathbb{Z}^2 \dashrightarrow T$ . Two adjacent tiles in a configuration *interact*, or are *bound* or *attached*, if the glues on their abutting sides are equal (in both label and

strength) and have positive strength. Each configuration  $\alpha$  induces a *binding graph*  $B_\alpha$  whose vertices are those points occupied by tiles, with an edge of weight  $s$  between two vertices if the corresponding tiles interact with strength  $s$ . An *assembly* is a configuration whose domain (as a graph) is connected and non-empty. The *shape* of  $X \subset \mathbb{Z}^2$  is  $\{X + (x, y) \mid x, y \in \mathbb{Z}\}$ , i.e. the set of all its translations. The shape of an assembly  $\alpha$  is the shape  $S_\alpha$  of its domain: two assemblies have the same shape if they have the same domain modulo translation. An assembly  $\alpha$  contains a shape  $S$  if there is  $s \in S$  such that  $s \subset \text{dom } \alpha$ . For some  $\tau \in \mathbb{Z}^+$ , an assembly  $\alpha$  is  $\tau$ -*stable* if every cut of  $B_\alpha$  has weight at least  $\tau$ , i.e. a  $\tau$ -stable assembly cannot be split into two pieces without separating bound tiles whose shared glues have cumulative strength  $\tau$ . Given two assemblies  $\alpha, \beta$ , we say  $\alpha$  is a *subassembly* of  $\beta$  (denoted  $\alpha \sqsubseteq \beta$ ) if  $\text{dom } \alpha \subseteq \text{dom } \beta$  and for all  $p \in \text{dom } \alpha$ ,  $\alpha(p) = \beta(p)$  (i.e., they have tiles of the same types in all locations of  $\text{dom } \alpha$ ).

A *tile-assembly system* (TAS) is a triple  $\mathcal{T} = (T, \sigma, \tau)$ , where  $T$  is a tile set,  $\sigma$  is a finite  $\tau$ -stable assembly called the *seed assembly*, and  $\tau \in \mathbb{Z}^+$  is called the *binding threshold* (a.k.a. *temperature*). If the seed  $\sigma$  consists of a single tile, i.e.  $|\sigma| = 1$ , we say  $\mathcal{T}$  is *singly-seeded*. Given a TAS  $\mathcal{T} = (T, \sigma, \tau)$  and two  $\tau$ -stable assemblies  $\alpha$  and  $\beta$ , we say that  $\alpha$   $\mathcal{T}$ -*produces*  $\beta$  *in one step* (written  $\alpha \rightarrow_1^\mathcal{T} \beta$ ) if  $\alpha \sqsubseteq \beta$  and  $|S_\beta \setminus S_\alpha| = 1$ . That is,  $\alpha \rightarrow_1^\mathcal{T} \beta$  if  $\beta$  differs from  $\alpha$  by the addition of a single tile. The  $\mathcal{T}$ -*frontier* is the set  $\partial^\mathcal{T} \alpha = \bigcup_{\alpha \rightarrow_1^\mathcal{T} \beta} S_\beta \setminus S_\alpha$  of locations in which a tile could  $\tau$ -stably attach to  $\alpha$ . When  $\mathcal{T}$  is clear from context we simply refer to these as the *frontier* locations.

We use  $\mathcal{A}^T$  to denote the set of all assemblies of tiles in tile set  $T$ . Given a TAS  $\mathcal{T} = (T, \sigma, \tau)$ , a sequence of  $k \in \mathbb{Z}^+ \cup \{\infty\}$  assemblies  $\alpha_0, \alpha_1, \dots$  over  $\mathcal{A}^T$  is called a  $\mathcal{T}$ -*assembly sequence* if, for all  $1 \leq i < k$ ,  $\alpha_{i-1} \rightarrow_1^\mathcal{T} \alpha_i$ . The *result*  $\lim \alpha$  of an assembly sequence  $\alpha$  is the unique limiting assembly of the sequence. For finite assembly sequences, this is the final assembly; whereas for infinite assembly sequences, this is the assembly consisting of all tiles from any assembly in the sequence. We say that  $\alpha$   $\mathcal{T}$ -*produces*  $\beta$  (denoted  $\alpha \rightarrow^\mathcal{T} \beta$ ) if there is a  $\mathcal{T}$ -assembly sequence starting with  $\alpha$  whose result is  $\beta$ . We say  $\alpha$  is  $\mathcal{T}$ -*producible* if  $\sigma \rightarrow^\mathcal{T} \alpha$  and write  $\mathcal{A}[\mathcal{T}]$  to denote the set of  $\mathcal{T}$ -producible assemblies. We say  $\alpha$  is  $\mathcal{T}$ -*terminal* if  $\alpha$  is  $\tau$ -stable and there exists no assembly that is  $\mathcal{T}$ -producible from  $\alpha$ . We denote the set of  $\mathcal{T}$ -producible and  $\mathcal{T}$ -terminal assemblies by  $\mathcal{A}_\square[\mathcal{T}]$ . If  $|\mathcal{A}_\square[\mathcal{T}]| = 1$ , i.e., there is exactly one terminal assembly, we say that  $\mathcal{T}$  is *directed*. When  $\mathcal{T}$  is clear from context, we may omit  $\mathcal{T}$  from notation.

Given TAS  $\mathcal{T}$ , we define  $\mathcal{S}_\square[\mathcal{T}] = \{S_\alpha \mid \alpha \in \mathcal{A}_\square[\mathcal{T}]\}$  as the set of all shapes of terminal assemblies in  $\mathcal{T}$ . Clearly, if  $\mathcal{T}$  is directed  $|\mathcal{S}_\square[\mathcal{T}]| = 1$ . However, even if  $\mathcal{T}$  is not directed it may be the case that  $|\mathcal{S}_\square[\mathcal{T}]| = 1$  (see [3] for an example). In such a case we call the system *shape directed*.

Given  $S$ , a connected set of points in  $\mathbb{Z}^2$ , we define a version of  $S$  *scaled by factor*  $c$ , and denote it by  $c \cdot S$ , as  $c \cdot S = \{(x, y) \in \mathbb{Z}^2 \mid (\lfloor \frac{x}{c} \rfloor, \lfloor \frac{y}{c} \rfloor) \in S\}$ . Intuitively,  $c \cdot S$  is a version of  $S$  expanded by replacing each point of  $S$  by a  $c \times c$  square of points. Likewise, for a set  $A$  of shapes,  $c \cdot A$  is defined as  $\{c \cdot S \mid S \in A\}$ .

### 2.1.1 Zig-zag Tile Assembly Systems

We now provide the definition for a subset of aTAM systems which are known as compact zig-zag systems (originally defined in [4]). These systems have very simple dynamics (the tiles have uniform input and output sides) yet are capable of universal computation – for every Turing machine, there exists a compact zig-zag system which simulates it [4]. Due to their simple dynamics and Turing Universality, these systems are an ideal target for systems with

simple mechanisms to simulate to demonstrate they are Turing Universality. This approach has become a paradigm in axiomatic, tile-based self-assembly [4, 11, 12, 16, 21], and we follow this same approach in Section 5.

The following definitions of zig-zag, and compact zig-zag systems are almost identical to those described in [21].  $\mathcal{T} = (T, \sigma, \tau)$  is a zig-zag tile assembly system provided that (1)  $\mathcal{T}$  is directed, (2) there is a single  $\mathcal{T}$ -assembly sequence  $\alpha \in \mathcal{T}$ , with  $\mathcal{A}_{\square}[\mathcal{T}] = \{\alpha\}$ , and (3) no tile initially binds to an assembly with its north glue. More intuitively, a zig-zag tile assembly system is a system which grows to the left or right, grows up some amount, and then continues growth again to the left or right. Again, as defined in [20], a tile assembly system  $\mathcal{T} = (T, \sigma, \tau)$  is a compact zig-zag tile assembly system iff  $\mathcal{A}_{\square}[\alpha] = \{\alpha\}$  and for every  $x \in \text{dom } \alpha$ ,  $\text{str}_{\alpha(x)}(N) + \text{str}_{\alpha(x)}(S) < 2\tau$  and  $\text{str}_{\alpha(x)}(W) + \text{str}_{\alpha(x)}(E) < 2\tau$ . Informally, this can be thought of as a zig-zag tile assembly system where two  $\tau$ -strength tile attachments in a row cannot occur in the same direction. This means a row can never extend by more than one tile over the previous row, and the system is only able to travel upwards one tile at a time before being required to zig-zag again.

## 2.2 The Synchronous Tile Assembly Model

The Synchronous Tile Assembly Model (syncTAM) is the variant of the aTAM where at every step of assembly, rather than a single location of the frontier being randomly selected to receive a tile, *all* frontier locations simultaneously receive a tile each. As the frontier of a system can grow arbitrarily large, the number of tiles added per assembly step may also grow arbitrarily large. As in the aTAM, if any single frontier location allows for  $\tau$ -strength binding of tiles of multiple types, one of those types is non-deterministically chosen for each such location.

We formally define the syncTAM in the same way as the aTAM with the exception of the definition of a single step in the assembly sequence. Given a syncTAS  $\mathcal{S} = (T, \sigma, \tau)$  and two  $\tau$ -stable assemblies  $\alpha$  and  $\beta$ , we say that  $\alpha$   *$\mathcal{S}$ -produces  $\beta$  in one step*, written  $\alpha \rightarrow_1^{\mathcal{S}} \beta$ , if  $\alpha \sqsubseteq \beta$  and  $\text{dom}(\beta) \setminus \text{dom}(\alpha) = \partial^{\mathcal{S}}(\alpha)$ . Note that like in the aTAM, we define  $\partial^{\mathcal{S}}(\alpha)$  to be the set of locations where a tile can  $\tau$ -stably attach to  $\alpha$ . The syncTAM inherits all other definitions and notation from the aTAM.

To distinguish between a tile assembly system (a.k.a. TAS) in the aTAM versus one in the syncTAM, we will use the term *syncTAS* to refer to a TAS in the syncTAM.

## 3 Similarities between the aTAM and the syncTAM

In this section we show that directed and shape-directed aTAM systems behave equivalently in the syncTAM. That is, given a tile set  $T$ , seed  $\sigma$ , and temperature  $\tau$ , if  $(T, \sigma, \tau)$  is used as an aTAM system and that system is directed (resp. shape directed), then when  $(T, \sigma, \tau)$  is used as a syncTAM system the same unique terminal assembly (resp. assembly shape) is produced. Although these are relatively straightforward results, they demonstrate the types of systems, namely those with reduced nondeterminism, which are invariant to the differing dynamics of the models.

► **Lemma 1.** *Let  $\mathcal{S} = (T, \sigma, \tau)$  be a syncTAM system and  $\mathcal{T} = (T, \sigma, \tau)$  be an aTAM system composed of the same components. If  $\alpha$  and  $\alpha'$  are assemblies of  $\mathcal{S}$  such that  $\alpha \rightarrow_{\mathcal{S}}^1 \alpha'$ , then  $\alpha \rightarrow_{\mathcal{T}} \alpha'$ .*

**Proof.** Let  $\mathcal{S} = (T, \sigma, \tau)$  be a syncTAM system and  $\mathcal{T} = (T, \sigma, \tau)$  be an aTAM system composed of the same components. Furthermore, assume that  $\alpha$  and  $\alpha'$  are assemblies of  $\mathcal{S}$  such that  $\alpha \rightarrow_{\mathcal{S}}^1 \alpha'$ .

Let  $F$  be the set of pairs consisting of a frontier location of  $\alpha$  and a tile that can  $\tau$ -stably attach to  $\alpha$  at that location such that for every pair  $(p, t) \in F$ ,  $\alpha'(p) = t$ . Intuitively,  $F$  is the set of all tiles (a tile type and position) such that adding the tiles of  $F$  to  $\alpha$  yields  $\alpha'$ . Our construction of  $F$  ensures that all the positions contained in the pairs are distinct (that is, no two pairs contain the same position), and every tile can attach  $\tau$  stably to  $\alpha$ . Consequently, we can construct a valid  $\mathcal{T}$ -assembly sequence from  $\alpha$  to  $\alpha'$  through single tile attachments from the tiles in  $F$ . Indeed, let  $(f_i)_{i=1}^n$  be an enumeration of the set  $F$ . Define  $\alpha_0 = \alpha$  and  $\alpha_i = \alpha_{i-1} \cup f_i$ . Then  $\alpha_0 \rightarrow \alpha_1 \rightarrow \dots \rightarrow \alpha_n$  is a valid assembly sequence since all tiles can  $\tau$ -stably attach and none prevent the others from binding, and  $\alpha_n = \alpha'$ . ◀

Lemma 1 shows that each single step of syncTAM assembly can be matched by a set of steps of an equivalent aTAM system. Thus, beginning from the same seed assembly, successive applications of that lemma show that any assembly that is producible in a syncTAM system is also producible in the equivalent aTAM system.

► **Lemma 2.** *Let  $\mathcal{T} = (T, \sigma, \tau)$  be an arbitrary directed (resp. shape directed) aTAM system. Then the syncTAM system  $\mathcal{S} = (T, \sigma, \tau)$ , composed of the same components, is also directed (resp. shape directed) and terminally produces exactly the same singular terminal assembly (resp. singular shape for all terminal assemblies) as  $\mathcal{T}$ .*

**Proof.** Let  $\mathcal{T} = (T, \sigma, \tau)$  be an arbitrary directed aTAM system, and let  $\mathcal{S} = (T, \sigma, \tau)$  be composed of the same components. By definition,  $\mathcal{T}$  has exactly one terminal assembly which we denote by  $\alpha$ . We claim that this is also the only terminal assembly of  $\mathcal{S}$ . Suppose, for the sake of contradiction, this was not the case. That is, suppose  $\alpha' \in \mathcal{A}_{\square}[\mathcal{S}]$  and  $\alpha' \neq \alpha$ . Let  $\alpha_i$  be a  $\mathcal{T}$ -assembly sequence such that  $\alpha_0 = \sigma$  and  $\lim \alpha_i = \alpha$  and let  $\alpha'_i$  be a  $\mathcal{S}$ -assembly sequence such that  $\alpha'_0 = \sigma$  and  $\lim \alpha'_i = \alpha'$ . Then it follows from Lemma 1 that  $\alpha'_i$  is a valid  $\mathcal{T}$  assembly sequence which contradicts our assumption  $\mathcal{T}$  is directed.

The argument for the shape directed version of the lemma is similar, but instead of assuming (for the sake of contradiction) that  $\alpha' \neq \alpha$  as above, we assume for the sake of contradiction that  $S_{\alpha} \neq S_{\alpha'}$ . The rest of the argument remains the same. ◀

Thus, the two lemmas presented here show that systems with limited nondeterminism produce the same assemblies (and/or shapes) in both the aTAM and the syncTAM. This completely changes in the general case: the additional power, and constraints, imposed by the global synchronization of the syncTAM also provide for separations between the models, which we now present in Sections 4 and 5. This culminates in Theorem 8 which demonstrates a shape  $S$  such that no aTAM system can assemble  $S$ , but there exists a syncTAM system which assembles  $S$ .

## 4 Separating the aTAM and the syncTAM by Shape Production

The difference between the aTAM and the syncTAM is manifest in the shapes an sets of shape each model is able to assemble. This section exhibits a set of shapes and a shape which are exclusive to one of the models, respectively the aTAM and the syncTAM. Much like how we ignore constants when studying the runtime of algorithms, often in tile-based self-assembly the size of a “pixel” in the shape is discounted. In our results, it will not just be the shape which is unobtainable in the “wrong” model, but all of its scaled versions. Likewise, while in each case, the “right” model can assemble those shapes without collaboration (i.e. at temperature 1), no amount of collaboration can help the wrong model in obtaining them.



## 4.1 Sets of Shapes that need the Asynchronism of the aTAM

The synchronism of the syncTAM can act as a constraint, making it impossible for a single syncTAS to match the dynamics of the more asynchronous aTAM. Namely, we exhibit a simple aTAM system that can create an infinite set of different shapes, while any syncTAS must be constrained to only forming a subset of them.

Impossibility proofs for the aTAM often rely on the Window Movie Lemma [20] (an overview of which can be found in Section A.2). Alas, it does not hold in the syncTAM, as the delay between successive attachments along the window can be used by either side to avoid being “fooled” by the change in the other. On the other hand, in the syncTAM, positions cannot remain attachable for an arbitrarily long time, which prevents the delaying tactics the aTAM can use. Hence, the fooling lemmas for the syncTAM differ from those of the aTAM. The following two lemmas capture this and will prove sufficient for the purpose of this paper.

The first limit of syncTAM systems is that they cannot leave positions attachable for an infinitely long time (and this holds for *any* infinite assembly sequence, which differs from the aTAM).

► **Lemma 3** (Completion). *Let  $\mathcal{S}$  be a syncTAM system. For any infinite assembly sequence  $\alpha$  of  $\mathcal{S}$ ,  $\lim \alpha$  is a terminal production:  $\lim \alpha \in \mathcal{A}_{\square}[\mathcal{S}]$ .*

**Proof.** Assume  $z$  is a frontier location in  $\lim \alpha$ . There is a subset  $N$  of the neighbors of  $z$  such that a tile could attach  $\tau$ -stably to the tiles of  $\lim \alpha$  within  $N$ . Let  $t$  be the last time a tile was attached in  $N$ . Then  $z$  belongs to the frontier of  $\alpha_t$ , has been filled at time  $t + 1$  and cannot be empty in  $\lim \alpha$ .

Thus, the frontier of  $\lim \alpha$  is empty and  $\lim \alpha$  is a terminal production. ◀

The second limitation of syncTAM systems is that if their productions can avoid some patterns for an arbitrarily long time, then they have a terminal production avoiding those patterns.

► **Lemma 4** (Compacity). *Let  $F$  be a set of finite shapes of  $\mathbb{Z}^2$ . Let  $\mathcal{S}$  be a syncTAM system with, for each  $k \in \mathbb{N}$ , a production  $P_k \in \mathcal{A}[\mathcal{S}]$  producible in  $k$  assembly steps that does not contain  $F$ .*

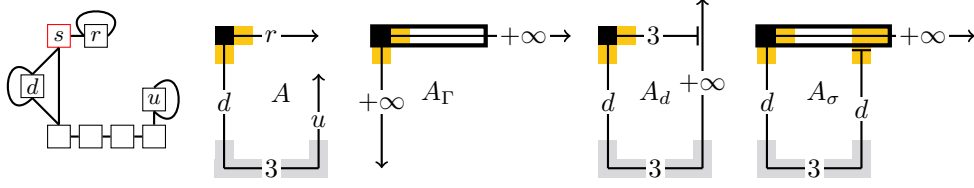
*Then there is a terminal production  $T \in \mathcal{A}_{\square}[\mathcal{S}]$  such that  $\text{dom}(T)$  does not contain any element of  $F$ .*

**Proof.** The attachment sequence  $(\beta_i)_{i \geq 0}$  of  $\mathcal{S}$  (defined recursively as follows) is such that from each  $\beta_i$ , infinitely many  $P_i$  can be  $\mathcal{S}$ -produced:

- $\beta_0$  is the seed assembly, from which all the  $P_i$  can be produced,
- for each  $i$ , since  $\beta_i$  is a finite production, it has only finitely many descendants (productions  $d$  such that  $\beta_i \rightarrow_1^{\mathcal{S}} d$ ). Since  $\beta_i$   $\mathcal{S}$ -produces infinitely productions  $P_j$ , one of its descendants must also produce infinitely many  $P_j$ ; pick that descendant to be  $\beta_{i+1}$ .

By Lemma 3,  $T = \lim \beta$  is a terminal production, yet it cannot contain any element of  $F$ . If it did contain an  $X \in F$ , since  $X$  is finite, there would be a finite  $f$  such that  $\beta_f$  contains  $X$  and that  $\beta_f$  would not be compatible with any of the  $P_k$ . ◀

► **Theorem 5.** *There is an aTAM system  $\mathcal{A}$  such that for any syncTAM system  $\mathcal{S}$  and scale factor  $c \geq 1$ ,  $\mathcal{S}_{\square}[\mathcal{S}] \neq c \cdot \mathcal{S}_{\square}[\mathcal{A}]$ .*



■ **Figure 1** Left, the tile types of the temperature 1 aTAM System  $\mathcal{A}$ . There is a line between two tile edges whenever their glue match, edges without incident lines have a 0-strength glue. The seed is the red  $s$  tile. Each of the tiles  $r, d, u$  can grow arbitrarily long tile segments, respectively to the right, upwards and downwards. To the right, a typical production  $A(u, d, r)$  of  $\mathcal{A}$ , then the three sets of terminal assemblies of  $\mathcal{A}$ :  $A_\Gamma$ ,  $A_d(d)$  and  $A_\sigma(d)$ . The yellow parts at the top of the productions are the copies of  $\Gamma$ , the gray parts at the bottom, the copies of  $U_4$ , and the encircled part at the top is the copy of  $B_5$ .

**Proof.** Let  $\mathcal{A}$  be the aTAM system of Figure 1. As depicted in Figure 1, let  $A(d, r, u)$  for  $d \geq 0, r \geq 0, u \geq -4$  be the subset of  $\mathbb{Z}^2$  defined by:

$$A(d, r, u) = (\llbracket 0, r \rrbracket \times 0) \cup (0 \times \llbracket -d, 0 \rrbracket) \cup (\llbracket 0, \min(u+4, 4) \rrbracket \times \{-d\}) \cup (\{3\} \times \llbracket -d, u-d \rrbracket)$$

The shape of any assembly of  $\mathcal{A}$  is of the form  $A(d, r, u)$  for some values of  $d, r$  and  $u$ . The quantity  $u$  starts from negative values so as to correspond to the number of  $u$  tiles of the assembly, as shown on Figure 1. The “−3rd, −2nd, −1st and 0th  $u$  tiles” are the 4 tiles at the bottom of the assembly between the  $d$  and  $u$  arms. The values of  $d, r$  and  $u$  are not independent:

- when  $d = 0, u = -4$ : the bottom part can only grow after the lower arm has grown
- only one of  $u > d$  and  $r > 3$  can hold at once, because of the competition between the two arms for position  $(3, 0)$ .

The shapes of the terminal productions of  $\mathcal{A}$  are of one of the following forms:

$$\begin{aligned} A_\Gamma &= A(+\infty, +\infty, -3) \\ A_d(d) &= A(d, 3, +\infty) \\ A_\sigma(d) &= A(d, +\infty, d) \end{aligned}$$

► **Remark 6.** Having glanced at Figure 1, it may seem straightforward that  $\mathcal{A}$  cannot be simulated by a syncTAS  $\mathcal{S}$ , assuming  $\mathcal{S}$  proceeds like  $A$ , with two concurrent arms starting from the top-left: the horizontal arm cannot wait for a downwards arm which may never be coming back to the meeting point; yet, it can never turn to the north without knowing that the other arm will come back. The remainder of the proof essentially makes that argument robust against  $\mathcal{S}$  placing its seed elsewhere, using its scale factor to synchronize the two arms, or any other shenanigans.  $\lrcorner$

Back to the proof, consider the following shapes:

- $\Gamma = \{(0, 0), (0, -1), (1, 0)\}$ ,
- $U_4 = \{(0, 1), (3, 1)\} \cup (\{0\} \times \llbracket 0, 3 \rrbracket)$ , a 4-tile wide U,
- $B_5 = (\llbracket 0, 4 \rrbracket \times \{0\})$ , a 5-tile horizontal bar.

Any terminal production of  $\mathcal{A}$  must contain  $\Gamma$ , and either  $B_5$  or a translated copy of  $U_4$ .

Assume now there is a syncTAM system  $\mathcal{S}$  and an integer  $c$  such that  $\mathcal{S}_\square[\mathcal{S}] = c \cdot \mathcal{S}_\square[\mathcal{A}]$ . Then for any terminal production  $T_\mathcal{S}$  of  $\mathcal{S}$ , there is a terminal production  $T_\mathcal{A}$  of  $\mathcal{A}$  and a vector  $\vec{\tau}$  such that  $\text{dom } T_\mathcal{S} = c \cdot \text{dom } T_\mathcal{A} + \vec{\tau}$ . Note that while  $c$  is the same for all productions,  $\vec{\tau}$  may be different for each of them.

All terminal productions of  $\mathcal{S}$  must have a translated copy of  $c \cdot \Gamma$ , and a translated copy of either  $c \cdot U_4$  or  $c \cdot B_5$ . By compactity (Lemma 4), there is an integer  $t(\{c \cdot \Gamma\})$  such that any production at time  $t \geq t(\{c \cdot \Gamma\})$  has  $(c \cdot \Gamma) + \vec{\tau}$  as a subset of its shape for some vector  $\vec{\tau} \in \mathbb{Z}^2$ . Likewise, there is an integer  $t(\{c \cdot U_4, c \cdot B_5\})$  such that any production at time  $t \geq t(\{c \cdot U_4, c \cdot B_5\})$  has either  $(c \cdot U_4) + \vec{\tau}$  or  $(c \cdot B_5) + \vec{\tau}$  as a subset of its shape for some vector  $\vec{\tau} \in \mathbb{Z}^2$ .

Let  $T = \max(t(\{c \cdot \Gamma\}), t(\{c \cdot U_4, c \cdot B_5\}))$ , the shape of any production  $P$  of  $\mathcal{S}$  at time  $T$  contains: a copy of  $c \cdot \Gamma$ , and a copy of either  $c \cdot U_4$  or  $c \cdot B_5$ . Because  $P$  was produced in time  $T$ , the distance between its copy of  $c \cdot \Gamma$  and its copy of  $c \cdot B_5$  or  $c \cdot U_4$  is less than  $2T + 1$ .

Let  $d = \lceil \frac{2T+1}{c} \rceil + 1$ . The terminal production  $A_d(d)$  of  $\mathcal{A}$  contains neither a copy of  $B_5$ , nor a copy of  $U_4$  within distance  $\frac{2T+1}{c}$  of its copy of  $\Gamma$ : its unique  $\Gamma$  is at  $(0, 0)$ , while the leftmost tile of its  $U_4$  is at  $(0, -d)$ . Hence, there is no production of  $\mathcal{S}$  at time larger than  $T$  whose shape is contained in  $c \cdot A_d(d)$ . Therefore, there is no final (infinite) production of  $\mathcal{S}$  with shape  $c \cdot A_d(d)$ , which is a contradiction.  $\blacktriangleleft$

## 4.2 A Shape Demanding the Synchronization of the syncTAM to Self-assemble

This section exhibits a simple (infinite) shape which cannot be uniquely self-assembled in the asynchronous aTAM, while it can be assembled at temperature 1 by a relatively simple syncTAS. Essentially, the synchronous nature of the syncTAM allows for a (shape) directed syncTAS that creates a single shape using an infinite series of rows of tiles growing from opposite directions that are always guaranteed to meet near a central location, while the asynchronous aTAM cannot guarantee that both sides will always grow at the same speed and therefore the meeting locations may differ and the aTAM system must also be able to produce other, non-target shapes.

Consider the syncTAM system  $\mathcal{S}$  of Figure 2a, and let  $V$  be the following subset of  $\mathbb{Z}^2$ , as depicted on that figure:

$$V = \{(0, 0)\} \cup L \cup R \cup \bigcup_{k>0} (B_k^- \cup T_k^- \cup B_k^+ \cup T_k^+)$$

where:

**the left branch** is the set  $L = \{(-x, x), (-x - 1, x) | x \in \mathbb{N}\}$ ,

**the right branch** is the set  $R = \{(x, x), (x + 1, x) | x \in \mathbb{N}\}$ ,

**the left bar at height  $k$**  is the set  $B_k^- = \{(x, 4k) | x \in \llbracket -4k + 1, 1 \rrbracket\}$ , and

**the right bar at height  $k$**  is the set  $B_k^+ = \{(x, 4k + 1) | x \in \llbracket 0, 4k \rrbracket\}$

**the left thumbs** form the set  $T_k^- = \{(2x + 1, 4k + 1) | x \in \llbracket -2k, 0 \rrbracket\}$

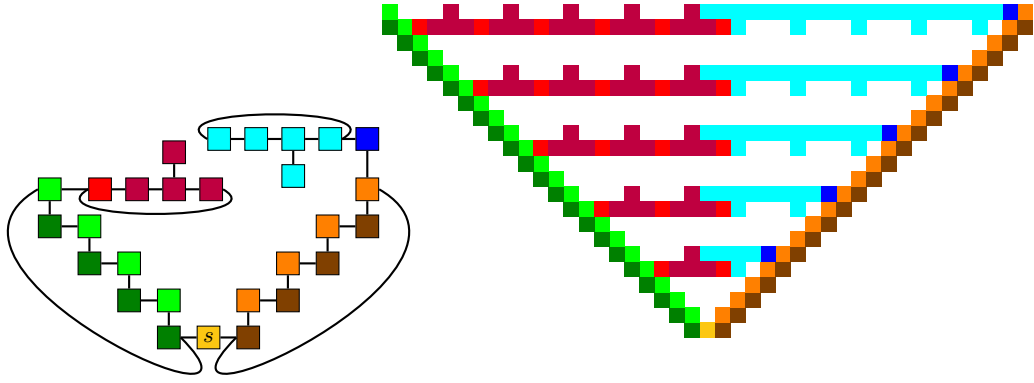
**the right thumbs** form the set  $T_k^+ = \{(2x, 4k) | x \in \llbracket 0, 2k \rrbracket\}$

► **Remark 7.** The system  $\mathcal{S}$  is directed and  $V$  is the shape of its unique terminal assembly.  $\dashv$

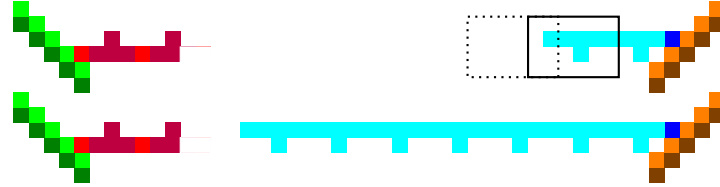
► **Theorem 8.** *There is no TAS system which uniquely assembles  $V$  at any scale factor: for all TAS systems  $\mathcal{A}$  and  $c \geq 1$ ,  $\mathcal{S}_{\square}[\mathcal{A}] \neq c \cdot \mathcal{S}_{\square}[\mathcal{S}]$ .*

**Proof.** Consider an integer  $c$  and a TAS system  $\mathcal{A}$  with  $g$  different glues. Given a producible assembly  $A$  in  $\mathcal{A}$  and a position  $z \in \mathbb{Z}^2$ , we say that  $A$  has entered  $p$  if  $c \cdot \{z\} \cap \text{dom } A \neq \emptyset$ .

As in [18], define a *glue movie* on a set of edges  $W$  of the grid to be the order of placement, position and glue type for each glue that appears along the window  $W$  in an assembly sequence. There are at most  $p = 2(4c)!(4c)^g$  possible glue movies of  $\mathcal{A}$  on a window of size  $4c$ . Indeed, there are  $(4c)^g$  potential scenarios for the tuple of glue types that lie along one



(a) The tiles of the temperature 1 syncTAM system  $\mathcal{S}$  (the seed is the yellow  $s$  tile), and a portion of the infinite shape  $V$  assembled by  $\mathcal{S}$ . Growth begins from the seed tile at the bottom and proceeds with the diagonal “arms” growing upward to the left and right, and with red rows growing right from the left arm, and blue rows growing left from the right arm, at every 4th step. The green part is  $L$ , the orange and brown part is  $R$ , the blue parts are the  $B_k^+ \cup T_k^+$ , while the red parts are the  $B_k^- \cup T_k^-$ .



(b) Using pumping in the aTAM on the two windows depicted at the top to produce an illegal production at the bottom: the right part has grown past the middle.

■ **Figure 2** A syncTAM system assembling a shape that cannot be obtained in the aTAM.

side of the window, and there are  $(4c)!$  permutations of this tuple, and we get the 2 in the beginning of the expression from the fact that we must consider both sides of the windows. Assume without loss of generality that  $p > 4$  and thus  $p$  is even.

Consider an assembly sequence  $(\alpha)_{i=0}^\infty$  of  $\mathcal{A}$  reaching a terminal production; the shape of  $\lim \alpha$  is  $c \cdot V$  translated by a vector  $\vec{\tau} \in \mathbb{Z}^2$ . From now on, all coordinates are translated by  $-\vec{\tau}$ , so that  $\text{dom}(\lim \alpha) = c \cdot V$ .

Let  $t^-$  be the minimal  $t$  such that there are  $p + 3$  different positions of  $B_{4p+2}^-$  where  $\alpha_t$  has entered. Likewise,  $t^+$  is the time where  $p + 3$  different positions of  $B_{4p+2}^+$  have been entered by  $\alpha_{t^+}$ . Assume for now that  $t^+ < t^-$ , and let  $A = \alpha_{t^+}$ .

Pose  $A_0 = A$ , and  $x_0^{\max} = c(4p+2) - 2$ . Let  $W_x$  be a rectangle of height  $4c$ , width  $p+1$  with its lower-left corner at  $(x, (16p+7)c)$ . In  $A_0$ , on any window  $W_x$  with  $x \in [x_0^{\max} - p - 1, x_0^{\max}]$  there are no tiles touching the left, upper and lower edges of  $W_x$ , while there must be at least a tile inside  $W_x$ . Thus, there are only  $p$  different window movies possible for all windows  $W_x$ , while there are  $p + 1$  such windows. There must be  $a, b$  such that the Window Movie Lemma of [18] applies between  $W_a$  and  $W_b$ . This begets a production  $A_1$  in which at most  $p + 3$  positions have been entered in  $B_{4p+2}^-$ , while  $p + 4$  positions have been entered in  $B_{4p+2}^+$ . This process can be iterated by posing  $x_i^{\max} = x_0^{\max} - i$ , until  $i = 3cp$ . In  $A_{3cp}$ , the right bar from the right extends past the midpoint, and the resulting shape is not a subset of  $V$ . We have obtained a terminal shape that is not included in  $V$  and a contradiction.

If  $t^- < t^+$ , the reasoning is the same, with  $x$  varying at iteration  $i$  between  $x_i^{\min}$  and  $x_i^{\min} + p + 1$ , with  $x_i^{\min} = -x_i^{\max}$ . Then  $A_{3p}$  extends straight past the midpoint. ◀

## 5 Separating the aTAM and the syncTAM by $\tau = 1$ Computation

In this section we show that directed, temperature-1 syncTAM systems are computationally universal (while this is not the case for directed, temperature-1 aTAM systems [19]). To prove this, we show that any temperature-2, compact zig-zag aTAM system can be simulated by a temperature-1 syncTAM system. The result then follows from a previous result: For every Turing machine  $M$ , there exists a temperature-2 compact zig-zag aTAM system which simulates it [4]. This approach (often used to show a class of temperature-1 systems are computationally universal [4, 11, 12, 16, 21]) benefits from the fact that rigorous definitions are already in place defining simulation between the necessary models. We end the section by showcasing a class of computationally universal temperature-1 syncTAM systems wherein each row is connected to the next by two north-south attachments, a level of connectivity not seen in previous simulations of TMs by non-cooperative tile assembly systems.

### 5.1 Computation at Temperature 1 in the syncTAM

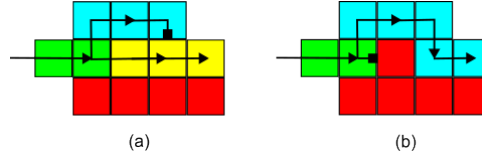
We first give a high-level overview of how a temperature-2, compact zig-zag aTAM system  $\mathcal{T}$  simulates a Turing machine  $M$ . At a high level,  $\mathcal{T}$  simulates  $M$  by exposing glues on the north of assembly which encode the symbols on the tape of  $M$ . One time step of a TM is simulated by growing a row which “zigs” (a row which grows to the right) and then a row which “zags” (a row which grows to the left). If the head transitions to the right during the step, this occurs during the growth of the “zig” row. A left transition, is performed during the growth of the “zag” row. In either case, the cooperative placement of tiles (that is, the requirement that a tile binds with two strength-1 glues) allows for information about the head of the state to be propagated through the east and west glues while information about the symbols stored in the tape is propagated through the south and north glues. See part (a) of Figure 5 for an example which highlights the movement of the head as an aTAM system simulates a Turing machine. For a more in depth explanation of how compact zig-zag systems simulate Turing machines, see [4].

The generally used technique (employed in [4, 11, 12, 16, 21]) is to get one of the two necessary inputs of a simulated cooperative binding via a glue attachment, and the other via carefully regulated growth around some sort of hindrance provided by geometry or glue repulsion. The subsets of tiles that perform the simulation of a single cooperative binding are referred to as *bit-readers* and *bit-writers*, i.e. tiles that grow a portion of an assembly capable of logically reading or writing bit values.

► **Theorem 9.** *Let  $\mathcal{T} = (T, \sigma, 2)$  be an arbitrary compact zig-zag aTAM system. There exists a syncTAS system  $\mathcal{S} = (S, \sigma', 1)$  such that  $\mathcal{S}$  simulates  $\mathcal{T}$  at scale factor  $O(\log |G_{T,N}|)$*

The system  $\mathcal{S}$  simulates  $\mathcal{T}$  by assembling  $n \times m$  blocks of tiles, which we call macrotiles. These macrotiles map to tiles in  $T$  under some macrotile representation function,  $R$ . We design  $\mathcal{S}$  so that it behaves exactly like  $\mathcal{T}$  under the macrotile representation function.

It is straightforward to design  $\mathcal{S}$  to simulate the strength-2 attachments in  $\mathcal{T}$ , but simulating cooperative binding attachments (attachments where a tile binds with two strength-1 glues) in  $\mathcal{T}$  using  $\mathcal{S}$  is much less obvious. Suppose that we want to simulate the attachment of a tile which binds cooperatively via its west and south glues. At a high-level, our construction uses gadgets to encode the northern strength-1 glues on a preceding row as geometry. Information about the strength-1 east glues being simulated by the macrotile to the west is presented as a glue on the macrotile boundary. From this glue, gadgets assemble which “read” the existing geometry below. Details about how these gadgets read and write



■ **Figure 3** An example of writing and reading a bit in the syncTAM. The red tiles represent the bit-writers. (a) A value of 0 was first written by the red tiles. Later, growth from the west causes placement of the green tiles. The rightmost green tile initiates the growth of the yellow and blue paths. Since the yellow path is shorter it always arrives at the second yellow location first, blocking placement of a blue tile later. The yellow path proceeds, encoding that a value of 0 was read. (b) A value of 1 is encoded in the geometry of the red tiles. The northernmost red tile blocks the growth of the yellow path allowing the blue path to complete instead, encoding that a value of 1 was read.

the geometry is shown in Figure 3. The end result of this is the exposure of a glue which is unique to the simulated glue from below and the simulated glue from the preceding tile in the row. Consequently, this glue allows a tile to attach that causes the macrotile to map to the correct tile  $t$  under the representation function, and it allows for a chain of bit-writers which encode information about the north glue of  $t$  to grow into the neighboring macrotile region to the north. After the bit-writers are assembled, a path of tiles grow to expose a glue corresponding to the west glue of  $t$  on the boundary of the next macrotile region.

We now provide a sketch of the proof.

**Proof.** Let  $\mathcal{T} = (T, \sigma, 2)$  be a compact zig-zag TAS. We now describe how to construct a syncTAS  $\mathcal{S} = (S, \sigma_s, 1)$  which simulates  $\mathcal{T}$  (under the definition of simulation formally defined in Section A.1). Note that under the definition of simulation, the domain of a macrotile is square. Our construction is designed to work with a notion of simulation that allows for rectangular macrotile domains, but we could design our tile set to pad the lengths of the paths that it grows to account for square macrotile domains.

We construct  $\mathcal{S}$  so that the macrotiles over  $\mathcal{S}$  consist of translations of subassemblies, also referred to as gadgets, called *bit-writers*, *bit-readers* and *translators*. We define *0-bit-writers* and *1-bit-writers* to be any translation of the red subassemblies shown in Figure 3. We refer to these collectively as just *bit-writers*. A bit-writer encodes (in binary) the label of a strength 1, north glue of a tile in  $\mathcal{T}$  so that it can be “read” by a *bit-reader* contained in the macrotile region above it which will enable  $\mathcal{S}$  to determine which tile the macrotile should map to in  $\mathcal{T}$ . Gadgets called *0-bit-readers* and *1-bit-readers* “read” this information (accomplished through growing exactly one of two paths depending on the geometry). We define these to be translations of the non-red subassemblies shown in Figure 3. Note that each of these gadgets grow from the same tile, but the existing geometry of the bit-writers (which were guaranteed to have previously completed growth) give rise to exactly one of the two subassemblies growing. A *translator* gadget is a translation of a subassembly that maps to a tile, denote this tile  $t'$ , which binds with a strength 2 glue. This subassembly is simply a path of tiles that grows from the “beginning” of a macrotile to the “beginning” of the next macrotile. The exact beginning and end position of this path depends on the location of the strength 2 glue on  $t'$  and the locations of its output glue. In the case that the tile to which a translator gadget belongs contains a strength-1 glue to its north, the translator may contain a chain of bit-writers in order to express this glue.

We now describe the idea behind bit-writers and bit-readers. At a high level, bit-readers attempt to grow two paths of tiles, say a 0-path and a 1-path. The bit-writers are designed to block exactly one of these paths, leaving the “surviving” path to constitute a read of its

relevant bit. To accomplish this behavior, we design the 1-path to be exactly one tile longer than the 0-path, so if a previously assembled 0-bit-writer does not block the 0-path, then it will be one tile “ahead” of the 1-path, and thus block the 1-path, hence, a 0 is read. However, if the 1-bit-writer blocks the 0-path, then the 1-path will not be blocked, and a 1 is read. Figure 3 shows the details of this mechanism. The glues in these surviving paths then “know” the bit that was encoded in the geometry by the bit-writer. It can use this knowledge for subsequent tasks such as assembling its own bit-writers.

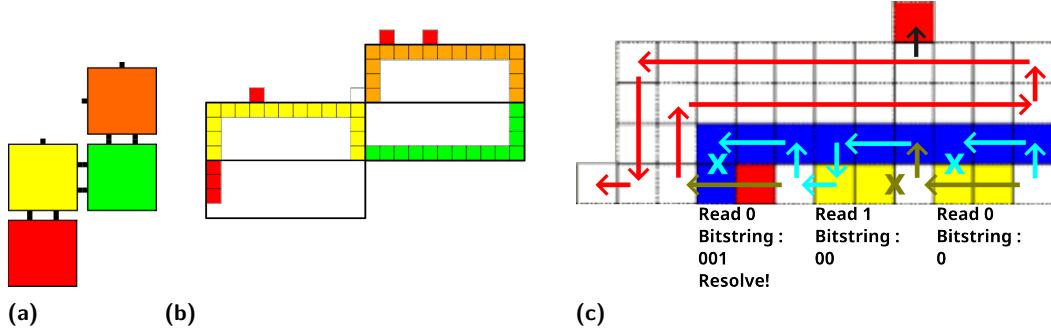
We now describe the desired behavior of  $\mathcal{S}$ , and then we describe how we can construct the tile set  $\mathcal{S}$  to obtain this desired behavior. We describe the behavior of  $\mathcal{S}$  in two scenarios. The first scenario we consider is the scenario where  $\mathcal{S}$  needs to assemble a macrotile that maps to a tile (under the macrotile representation function) that attaches cooperatively. The second scenario we consider is the scenario where  $\mathcal{S}$  is growing a macrotile that “simulates” the attachment of a tile in  $\mathcal{T}$  via a strength-2 attachment.

In the following discussion, let  $\vec{t} = ((x', y'), t)$ , for  $t \in T$ , be a tile (i.e. a placed instance of a tile type) in a producible assembly of  $\mathcal{T}$ , and let  $REG(x, y)$  be the macrotile region which maps to the tile at location  $(x, y)$  (in  $\mathcal{T}$ ) under the macrotile representation function. That is  $REG(x, y)$  is the set  $\{(i, j) | (mx \leq i \leq m(x + 1)) \text{ and } my \leq j \leq m(y + 1)\}$ . Then, intuitively,  $REG(x', y')$  is the region which maps to  $\vec{t}$ ,  $REG(x' - 1, y')$  is the region which maps to the tile to the west of  $\vec{t}$ , call this tile  $\vec{t}_w$ , and  $REG(x', y' - 1)$  is the region which maps to the tile to the south of  $\vec{t}$ , call this tile  $\vec{t}_s$ .

In the first case, assume  $\vec{t}$  is a tile in a producible assembly of  $\mathcal{T}$  which binds into the assembly cooperatively using two strength-1 glues. Without loss of generality, we assume the two “input” glues  $\vec{t}$  uses to bind to the assembly are on its west and south. The case where they are on its east and south is symmetric. We now describe how  $\mathcal{S}$  leverages bit-writers and bit-readers to assemble a macrotile in the macrotile region  $R$  that maps to  $\vec{t}$  under the macrotile representation function. Before a bit-reader begins assembling in a macrotile region, our construction of  $\mathcal{S}$  ensures a chain of bit-writers has assembled in the southern region of the macrotile. The bit-writers are chained together such that the last tile in a bit-writer exposes a glue on its east so that another bit-writer can begin growing from this glue. This chain of bit-writers is assembled during the growth of the macrotile to the south, and it encodes the label of the north glue of  $\vec{t}_s$  in binary. For each strength 1 glue on the west of a tile in  $T$ , say  $g_w$ , there is a unique chain of bit-readers that grows from a glue that corresponds to  $g_w$ . So, a chain of bit-readers specific to the east glue of  $\vec{t}_w$  starts assembling from the southwest most corner of  $REG(x', y')$ . Each bit-reader in this chain is unique to the binary sequence read up to that point, so that by the time the bit-reading chain is assembled, there is a tile placed at the end of the bit-reading chain that is specific to the east glue of  $\vec{t}_w$  and the north glue of  $\vec{t}_s$ . It is at this point that the macrotile representation function begins mapping the subassembly contained in  $REG(x', y')$  to  $\vec{t}$ . A new path of tiles begins assembling from this glue which grows bit-writers in the macrotile region to its north, that is the region  $R(x', y' + 1)$ , encoding the north output glue of  $\vec{t}$ . Then a path grows from the end of this chain of bit-writers to place a glue on the border of the macrotile region directly to the west of  $R$ , that is the region  $R(x' + 1, y')$ , which is specific to the east output glue of  $\vec{t}$ . The process then repeats.

Note that  $\log |G_{T,N}|$  bits are necessary to express the label of a strength-1 north glue in  $\mathcal{T}$ . As such, each macrotile that simulates a tile that has a strength-1 glue on its south must include  $\log |G_{T,N}|$  bit-readers, and each macrotile that expresses a strength-1 north glue must grow  $\log |G_{T,N}|$  bit writers. Therefore, with a bit-reader/writer length of 4, our macrotiles are of scale  $4 \times O(\log |G_{T,N}|)$ .





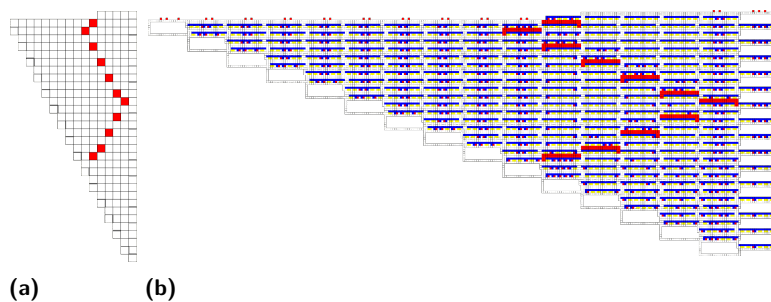
**Figure 4** (a): A simple assembly producible by a compact zig-zag system, which contains all potential tiles that bind into the assembly via a strength-2 glue in a compact zig-zag system, subject to reflection. (b) A temperature-1 syncTAM system consisting of macrotiles that simulate the behavior of the assembly in (a). Each macrotile is enclosed with a black rectangle, and assembly starts from the bottom-left red tile, concluding with the placement of the white tile (which attaches to the orange macrotile). Note that in the case that the yellow tile were to contain a strength-2 glue to its west, then the red macrotile would be on the east of its macrotile location. (c) A macrotile that may bind to the orange, and yellow macrotiles, shown in (b). The chain of bit readers are colored blue and yellow, with 0-paths denoted by gold arrows, and 1-paths denoted by light blue arrows. Arrows indicate connected glues: the side of a tile at the tail of the arrow attaches to the abutting side of the next tile that the arrow points to, and tiles connect sequentially along the arrow. X's denote glues to which a tile may be placed, but is blocked as a result of a tile that had previously been placed. For the sake of clarity, each bit which is read is associated with a label to its south, displaying the current bit string which either be encoded in the subsequent bit-reader, or used to resolve to a tile  $t \in \mathcal{T}$ .

The second case we consider is where  $\vec{t}$  is a tile in a producible assembly of  $\mathcal{T}$  that binds into the assembly via a strength-2 glue. In this case, the simulator simply grows a path of tiles (which are unique to that glue and, hence, uniquely determine the tile in  $\mathcal{T}$  the macrotile needs to map to under the representation function) to place a glue at the beginning of the next macrotile region so that the growth process can repeat. This path may include a chain of bit-writers, in the case that the tile exhibits a strength-1 glue to its north. The exact path that is grown depends on the exact locations of the input and output glues on  $\vec{t}$ .

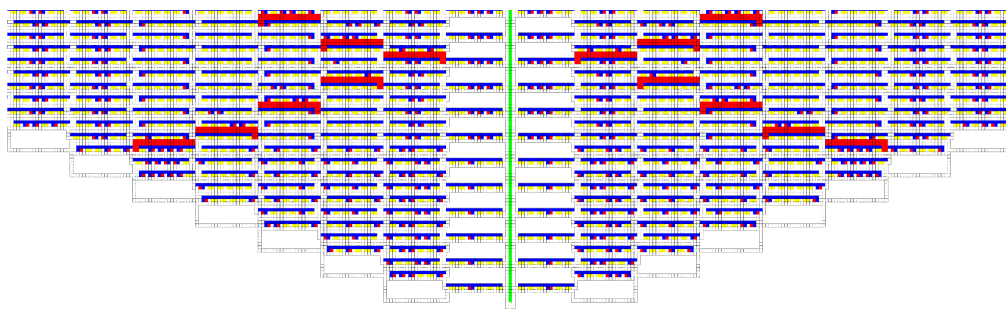
Compact zig-zag systems are singly-seeded (consequently their seed tile exposes a strength-2 glue if growth proceeds) by definition, so to form the seed for our simulator, we simply place a tile so that it exposes a glue that begins the simulation process described above.

We now describe the tiles in  $S$  that lead to the desired behavior described above. For each east/west glue in  $T$ , we add tiles to  $S$  with unique glues that assemble a chain of bit-readers. Note that the glues for each bit-reader in the chain are distinct and we add tiles for each position in the chain which assemble a 0-bit-reader and a 1-bit-reader. In addition, for each tile in  $T$ , we add unique tiles to  $S$  that grow the bit-writer corresponding to the label of the strength-1 north glue (if it exists) and grows a path to the macrotile boundary exposing a glue corresponding to the east/west glue. An example macrotile, including both a chain of bit-readers and bit-writers is displayed in Figure 4c. Finally, for each translator we add tiles to  $S$  that assemble the path of tiles composing the translator and have unique glues that do not interact with any other gadgets in  $\mathcal{S}$ . The tiles required to construct the translators of an example compact zig-zag system are displayed in Figure 4b.

The above construction correctly simulates  $\mathcal{T}$  provided that a bit-reader never attempts to grow into a macrotile region without a bit-writer chain. Note that provided a strength-1 glue is not exposed at the end of a row in  $\mathcal{T}$ , we have constructed  $\mathcal{S}$  to essentially consist of



**Figure 5** (a): An example compact zig-zag one-way infinite Turing machine simulation by an aTAM system. (b) The compact zig-zag system shown in (a) simulated by a temperature-1 syncTAS.



**Figure 6** A snapshot of the syncTAS system in Figure 5 modified to exhibit greater assembly connectedness. The assembly is shown alongside its reflected counterpart, growing from a single-tile seed, with a tile 'stitching' both sides together (shown in green).

one long path of tiles so that the bit-writer chain must be assembled before a bit-reading gadget reaches the macrotile region. To ensure no strength-1 glue is exposed at the end of a row in  $\mathcal{T}$ , we modify  $\mathcal{T}$  so that the north glues of the tiles at the end of each row are marked with a special symbol. We leverage this special glue to further modify  $\mathcal{T}$  so that no strength-1 glue is ever exposed at the end of the row. Consequently, no bit-reader will ever attempt to “read” an empty macrotile region and the construction is correct.

An example compact zig-zag aTAM system, along with a syncTAM system that simulates it is displayed in Figure 5. Along with this, a script which may be used to convert an arbitrary compact zig-zag TAS to a syncTAS may be found at [8]. ◀

► **Corollary 10.** *For every standard Turing machine  $M$  and input  $w$ , there exists a temperature-1 syncTAM system that simulates  $M$  on  $w$ .*

This follows directly from Lemma 7 of [4] (i.e. that, given an arbitrary Turing machine  $M$  and input  $w$ , there exists a compact zig-zag that simulates  $M$  on  $w$ ), and Theorem 9.

## 5.2 Greater Assembly Connectedness at Temperature 1

As previously mentioned, a variety of tile-based, temperature-1 models of self-assembly are computationally universal. Discrimination between types of tiles representing the correct input values from two different locations can be easily done using multiple glue bonds in a temperature-2 system. However, in a temperature-1 system this discrimination must be performed using some sort of hindrance that prevents incorrect tile types from binding while still always allowing the correct tile type to bind. This hindrance (often provided by geometry,

as with the “blocker” tile in our construction for Corollary 10) allows for algorithmically correct growth but necessarily results in assemblies with extremely sparse connectivity. The resulting have a binding graph that is a single-tile-wide path that zig-zags and folds back over itself, row by row, throughout the entire assembly. Each row, of increasingly great length, is attached by a single glue bond to the row below it. This results in an assembly that would be extremely “floppy” if physically realized. In all others models where such temperature-1 computation is achieved, this floppiness appears unavoidable since any exposed glue that may be intended to bind to a later-growing portion of the assembly and provide greater stability could, at any time, initiate its own growth (which may or may not agree with correct later growth). However, the precise synchronization enabled by the syncTAM can ensure that two separately-growing portions of an assembly of the same size can be guaranteed to complete at the exact same time, and we can leverage this to provide greater connectivity.

In order to provide this greater connectivity, we must ensure that each row which is placed in the zig-zag TM grows in from *both* directions, this means that the binding graph will have two paths for each zig / zag of the system, which converge, and diverge in repeating intervals. To accomplish this, the compact zig-zag TM construction displayed in Figure 5 may be reflected along the y axis, and connected to the east, and west of a single-tile seed such that the two identical simulations assemble in parallel.

Due to the synchronous nature of the syncTAM, both identical simulations will arrive at a single-tile wide unoccupied column during the *exact* same step. All macrotiles which contain a side abutting this unoccupied column only belong to the tiles which grow on the far-east edge of the original aTAM system, and thus only appear abutting the unoccupied column. As such, those tiles may have the side abutting the unoccupied column modified such that they contain a unique “connecting” glue, to which a single “connector” tile type may connect, “stitching” both sides of the assembly together.

As a result of the connector tile stitching both sub-assemblies together along each row, each row will therefore be connected to the row previous by at least two north-south attachments, one on the original TM simulation, and one on its reflection. Consequentially, every row is connected to the previous row by two or more glues. A snapshot of this connected assembly is shown in Figure 6. As with the section previous, a script which takes as input a TM definition, and outputs a more connected syncTAS which simulates it may be found at [8].

---

## References

- 1 Gagan Aggarwal, Michael H. Goldwasser, Ming-Yang Kao, and Robert T. Schweller. Complexities for generalized models of self-assembly. In *Proceedings of ACM-SIAM Symposium on Discrete Algorithms*, 2004.
- 2 Florent Becker, Daniel Hader, and Matthew J. Patitz. *Strict Self-Assembly of Discrete Self-Similar Fractals in the abstract Tile Assembly Model*, pages 2387–2466. SIAM, 2025. doi:10.1137/1.9781611978322.80.
- 3 Nathaniel Bryans, Ehsan Chiniforooshan, David Doty, Lila Kari, and Shinnosuke Seki. The power of nondeterminism in self-assembly. *Theory of Computing*, 9:1–29, 2013. doi:10.4086/toc.2013.v009a001.
- 4 Matthew Cook, Yunhui Fu, and Robert T. Schweller. Temperature 1 self-assembly: Deterministic assembly in 3D and probabilistic assembly in 2D. In *SODA 2011: Proceedings of the 22nd Annual ACM-SIAM Symposium on Discrete Algorithms*. SIAM, 2011. doi:10.1137/1.9781611973082.45.
- 5 Isabel Donoso Leiva, Eric Goles, Martín Ríos-Wilson, and Sylvain Sené. Asymptotic (a)synchronism sensitivity and complexity of elementary cellular automata. In *Proceedings of LATIN’24*, volume 14579 of *LNCS*, pages 272–286. Springer, March 2024. doi:10.1007/978-3-031-55601-2\_18.

- 6 David Doty, Jack H. Lutz, Matthew J. Patitz, Robert T. Schweller, Scott M. Summers, and Damien Woods. The tile assembly model is intrinsically universal. In *Proceedings of the 53rd Annual IEEE Symposium on Foundations of Computer Science*, FOCS 2012, pages 302–310, 2012. doi:10.1109/FOCS.2012.76.
- 7 Phillip Drake, Daniel Hader, and Matthew J Patitz. Simulation of the abstract tile assembly model using crisscross slats. In *30th International Conference on DNA Computing and Molecular Programming (DNA 30)(2024)*, pages 3–1. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2024.
- 8 Phillip Drake and Matthew J. Patitz. Synchronous TAM wiki page and software downloads, 2025. URL: [http://self-assembly.net/wiki/index.php/Synchronous\\_Tile\\_Assembly\\_Model\\_\(syncTAM\)](http://self-assembly.net/wiki/index.php/Synchronous_Tile_Assembly_Model_(syncTAM)).
- 9 Nazim Fatès. A guided tour of asynchronous cellular automata. In Jarkko Kari, Martin Kutrib, and Andreas Malcher, editors, *Cellular Automata and Discrete Complex Systems*, pages 15–30, Berlin, Heidelberg, 2013. Springer Berlin Heidelberg. doi:10.1007/978-3-642-40867-0\_2.
- 10 Nazim Fatès, Damien Regnault, Nicolas Schabanel, and Éric Thierry. Asynchronous behavior of double-quiescent elementary cellular automata. In José R. Correa, Alejandro Hevia, and Marcos Kiwi, editors, *LATIN 2006: Theoretical Informatics*, pages 455–466, Berlin, Heidelberg, 2006. Springer Berlin Heidelberg.
- 11 Sándor P. Fekete, Jacob Hendricks, Matthew J. Patitz, Trent A. Rogers, and Robert T. Schweller. Universal computation with arbitrary polyomino tiles in non-cooperative self-assembly. In *Proceedings of the Twenty-Sixth Annual ACM-SIAM Symposium on Discrete Algorithms (SODA 2015), San Diego, CA, USA, January 4-6, 2015*, pages 148–167, 2015. doi:10.1137/1.9781611973730.12.
- 12 Oscar Gilbert, Jacob Hendricks, Matthew J. Patitz, and Trent A. Rogers. Computing in continuous space with self-assembling polygonal tiles. In *Proceedings of the Twenty-Seventh Annual ACM-SIAM Symposium on Discrete Algorithms (SODA 2016), Arlington, VA, USA January 10-12, 2016*, pages 937–956, 2016.
- 13 Ashwin Gopinath, Evan Miyazono, Andrei Faraon, and Paul WK Rothmund. Engineering and mapping nanocavity emission via precision placement of DNA origami. *Nature*, 2016.
- 14 Daniel Hader, Aaron Koch, Matthew J. Patitz, and Michael Sharp. The impacts of dimensionality, diffusion, and directedness on intrinsic universality in the abstract tile assembly model. In Shuchi Chawla, editor, *Proceedings of the 2020 ACM-SIAM Symposium on Discrete Algorithms, SODA 2020, January 5-8, 2020*, pages 2607–2624, Salt Lake City, UT, USA, 2020. SIAM. doi:10.1137/1.9781611975994.159.
- 15 Daniel Hader and Matthew J. Patitz. WebTAS (beta): A browser-based simulator for tile-assembly models, 2025. URL: [http://self-assembly.net/software/WebTAS\\_beta/WebTAS\\_beta-latest/](http://self-assembly.net/software/WebTAS_beta/WebTAS_beta-latest/).
- 16 Jacob Hendricks, Matthew J. Patitz, Trent A. Rogers, and Scott M. Summers. The power of duples (in self-assembly): It’s not so hip to be square. *Theoretical Computer Science*, 743:148–166, 2018. doi:10.1016/J.TCS.2015.12.008.
- 17 James I. Lathrop, Jack H. Lutz, and Scott M. Summers. Strict self-assembly of discrete Sierpinski triangles. *Theoretical Computer Science*, 410:384–405, 2009. doi:10.1016/J.TCS.2008.09.062.
- 18 Pierre-Étienne Meunier, Matthew J. Patitz, Scott M. Summers, Guillaume Theyssier, Andrew Winslow, and Damien Woods. Intrinsic universality in tile self-assembly requires cooperation. In *Proceedings of the ACM-SIAM Symposium on Discrete Algorithms (SODA 2014), (Portland, OR, USA, January 5-7, 2014)*, pages 752–771, 2014. doi:10.1137/1.9781611973402.56.
- 19 Pierre-Étienne Meunier and Damien Regnault. Directed Non-Cooperative Tile Assembly Is Decidable. In Matthew R. Lakin and Petr Šulc, editors, *27th International Conference on DNA Computing and Molecular Programming (DNA 27)*, volume 205 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 6:1–6:21, Dagstuhl, Germany, 2021. Schloss Dagstuhl – Leibniz-Zentrum für Informatik. doi:10.4230/LIPIcs.DNA.27.6.

- 20 Pierre-Étienne Meunier and Damien Woods. The non-cooperative tile assembly model is not intrinsically universal or capable of bounded turing machine simulation. In *Proceedings of the 49th Annual ACM SIGACT Symposium on Theory of Computing*, STOC 2017, pages 328–341, New York, NY, USA, 2017. ACM. doi:10.1145/3055399.3055446.
- 21 Matthew J. Patitz, Robert T. Schweller, and Scott M. Summers. Exact shapes and Turing universality at temperature 1 with a single negative glue. In Luca Cardelli and William M. Shih, editors, *DNA Computing and Molecular Programming - 17th International Conference, DNA 17, Pasadena, CA, USA, September 19-23, 2011. Proceedings*, volume 6937 of *Lecture Notes in Computer Science*, pages 175–189. Springer, 2011. doi:10.1007/978-3-642-23638-9\_15.
- 22 Loïc Paulevé and Sylvain Sené. Boolean networks and their dynamics: the impact of updates. (Chapter) *Systems biology modelling and analysis: formal bioinformatics methods and tools*, pages 173–250, November 2022. doi:10.1002/9781119716600.ch6.
- 23 Damien Regnault, Nicolas Schabanel, and Éric Thierry. Progresses in the analysis of stochastic 2d cellular automata: A study of asynchronous 2d minority. *Theoretical Computer Science*, 410(47):4844–4855, 2009. doi:10.1016/j.tcs.2009.06.024.
- 24 Paul W. K. Rothmund and Erik Winfree. The program-size complexity of self-assembled squares (extended abstract). In *STOC '00: Proceedings of the thirty-second annual ACM Symposium on Theory of Computing*, pages 459–468, Portland, Oregon, United States, 2000. ACM. doi:10.1145/335305.335358.
- 25 Grigory Tikhomirov, Philip Petersen, and Lulu Qian. Fractal assembly of micrometre-scale DNA origami arrays with arbitrary patterns. *Nature*, 552(7683):67–71, 2017.
- 26 Ning Wang, Chang Yu, Tingting Xu, Dan Yao, Lingye Zhu, Zhifa Shen, and Xiaoying Huang. Self-assembly of dna nanostructure containing cell-specific aptamer as a precise drug delivery system for cancer therapy in non-small cell lung cancer. *Journal of Nanobiotechnology*, 20(1):1–19, 2022.
- 27 Erik Winfree. *Algorithmic Self-Assembly of DNA*. PhD thesis, California Institute of Technology, June 1998.
- 28 Damien Woods, David Doty, Cameron Myhrvold, Joy Hui, Felix Zhou, Peng Yin, and Erik Winfree. Diverse and robust molecular algorithms using reprogrammable dna self-assembly. *Nature*, 567(7748):366–372, 2019. doi:10.1038/S41586-019-1014-9.

## A Technical Appendix

This Technical Appendix contains additional technical definitions and details related to the proofs in the main body of the paper.

### A.1 Simulation Definition Details

In this section, we present the formal definitions of simulation between aTAM and syncTAM systems.

The definition of the syncTAM leads itself to a simulation definition that is equivalent to the definition of intrinsic simulation for a TAS simulating another TAS. In large part, our definitions come from [7].

From this point on, let  $\mathcal{T} = (T_{\mathcal{T}}, \sigma_{\mathcal{T}}, \tau_{\mathcal{T}})$  be a TAS, and let  $\mathcal{S} = (T_{\mathcal{S}}, \sigma_{\mathcal{S}}, \tau_{\mathcal{S}})$  be some syncTAS that simulates  $\mathcal{T}$ . For each such simulation we fix a positive integer  $m$  called the *scale factor* with the intention that  $m \times m$  blocks of locations from  $\mathcal{S}$  map to individual locations in  $\mathcal{T}$ . In other words, simulation happens at scale so that by “zooming out” by a factor of  $m$ , the assemblies in  $\mathcal{S}$  resemble corresponding assemblies in  $\mathcal{T}$ .

In the context of  $\mathcal{S}$ , we call each  $m \times m$  block of locations in  $\mathbb{Z}^2$  a *macrotile location* under the convention that the origin  $(0, 0) \in \mathbb{Z}^2$  occupies the south-westernmost location in one of the  $m \times m$  blocks. That is, macrotiles locations are squares with southwest corners



of the form  $(cx, cy)$  and northeast corners of the form  $(m(x+1)-1, m(y+1)-1)$  where  $x, y \in \mathbb{Z}$ . Given an assembly  $\alpha' \in \mathcal{A}[\mathcal{S}]$ , we use the notation  $\mathcal{M}_{x,y}^c[\alpha']$  to refer to the set of tiles in  $\alpha'$  which occupy locations in the macrotile location whose southwest corner is  $(cx, cy)$ . This set of tiles is referred to as the *macrotile* located at  $(x, y)$ . Note that in this context, we keep track of the tiles that occupy a macrotile with coordinates relative to  $(x, y)$  so that two macrotiles which differ only by a translation are identical. We denote the set of all possible macrotiles of scale factor  $m$  made from tiles in  $T_{\mathcal{S}}$  as  $\mathcal{M}^m[\mathcal{S}]$ .

A partial function  $R : \mathcal{M}^m[T_{\mathcal{S}}] \rightarrow T_{\mathcal{T}}$  is called a *macrotile representation function* from  $T_{\mathcal{S}}$  to  $T_{\mathcal{T}}$  if, for any macrotiles  $\alpha, \beta \in \mathcal{M}^m[T_{\mathcal{S}}]$  where  $\alpha \sqsubseteq \beta$  and  $\alpha \in \text{dom} R$ , then  $R(\alpha) = R(\beta)$ . In other words, a macrotile representation function may not map a macrotile to an individual tile type in  $T_{\mathcal{T}}$ , but if it does, then any additional tile attachments do not change how the macrotile is mapped under  $R$ . In the context of some  $\mathcal{S}$ -assembly sequence, a macrotile is said to *resolve* to a tile type  $t \in T_{\mathcal{T}}$  when an assembly maps under  $R$  to  $t$ , but the prior assembly is not in the domain of  $R$ .

From a macrotile representation function  $R$ , a function  $R^* : \mathcal{A}^{T_{\mathcal{S}}} \rightarrow \mathcal{A}^{T_{\mathcal{T}}}$ , called the *assembly representation function*, is induced which maps entire assemblies in  $\mathcal{S}$  to assemblies in  $\mathcal{T}$ . This function is defined by applying the function  $R$  to each macrotile location containing slats. We also use the notation  $R^{*-1}(\alpha)$  to refer to the *producible pre-image* of an assembly  $\alpha \in \mathcal{A}^T$ . That is,  $R^{*-1}(\alpha) = \{\alpha' \in \mathcal{A}[\mathcal{S}] \mid R^*(\alpha') = \alpha\}$  so that  $R^{*-1}(\alpha)$  includes every  $\mathcal{S}$ -producible assembly mapping to  $\alpha$  under  $R^*$ . We say that an assembly  $\alpha' \in \mathcal{A}^{\mathcal{S}}$  maps cleanly to an assembly  $\alpha \in \mathcal{A}^T$  under  $R$  if  $R(\alpha') = \alpha$  and no slats in  $\alpha'$  occupy any macrotile whose location is not adjacent (not including diagonally) to a resolved macrotile. Formally, if  $\alpha'$  maps cleanly to  $\alpha$ , then for each non-empty macrotile  $\mathcal{M}_{x,y}^m[\alpha']$ , there exists some vector  $(u, v) \in \mathbb{Z}^2$  such that  $\|(u, v)\| \leq 1$  so that  $\mathcal{M}_{x+u, y+v}^m[\alpha'] \in \text{dom} R$ .<sup>1</sup> The definition of *maps cleanly* requires that any non-empty but unresolved macrotile has a resolved macrotile to its N, E, S, or W, thus ensuring that the growth of tiles is only occurring in macrotile locations that map to locations in  $\mathcal{T}$  adjacent to tiles, and thus with some potential to receive a tile.

► **Definition 11** (Equivalent Productions). *We say that  $\mathcal{S}$  and  $\mathcal{T}$  have equivalent productions (under  $R$ ) and we write  $\mathcal{S} \Leftrightarrow \mathcal{T}$  if the following conditions hold:*

1.  $\{R^*(\alpha') \mid \alpha' \in \mathcal{A}[\mathcal{S}]\} = \mathcal{A}[\mathcal{T}]$ .
2.  $\{R^*(\alpha') \mid \alpha' \in \mathcal{A}_{\square}[\mathcal{S}]\} = \mathcal{A}_{\square}[\mathcal{T}]$ .
3. for all  $\alpha' \in \mathcal{A}[\mathcal{S}]$ ,  $\alpha'$  maps cleanly to  $R^*(\alpha')$ .

► **Definition 12** (Follows). *We say that  $\mathcal{T}$  follows  $\mathcal{S}$  (under  $R$ ), and we write  $\mathcal{T} \dashv^{\mathcal{S}} \beta'$ , for some  $\alpha', \beta' \in \mathcal{A}[\mathcal{S}]$ , implies that  $R^*(\alpha') \rightarrow^{\mathcal{T}} R^*(\beta')$ .*

► **Definition 13** (Models). *We say that  $\mathcal{S}$  models  $\mathcal{T}$  (under  $R$ ), written  $\mathcal{S} \models_R \mathcal{T}$  if for every  $\alpha \in \mathcal{A}[\mathcal{T}]$ , there exists a non-empty subset  $\Pi_{\alpha} \subseteq R^{*-1}(\alpha)$ , such that for all  $\beta \in \mathcal{A}[\mathcal{T}]$  where  $\alpha \rightarrow^{\mathcal{T}} \beta$ , the following conditions are satisfied:*

1. for every  $\alpha' \in \Pi_{\alpha}$ , there exists  $\beta' \in R^{*-1}(\beta)$  such that  $\alpha' \rightarrow^{\mathcal{S}} \beta'$
2. for every  $\alpha'' \in R^{*-1}(\alpha)$  and  $\beta'' \in R^{*-1}(\beta)$  where  $\alpha'' \rightarrow^{\mathcal{S}} \beta''$ , there exists  $\alpha' \in \Pi_{\alpha}$  such that  $\alpha' \rightarrow^{\mathcal{S}} \alpha''$ .

In Definition 13 above, the set  $\Pi_{\alpha}$  is defined to be a set of assemblies representing  $\alpha$  from which it is still possible to produce assemblies representing all possible  $\beta$  producible from  $\alpha$ . Informally, the first condition specifies that all assemblies in  $\Pi_{\alpha}$  can produce some

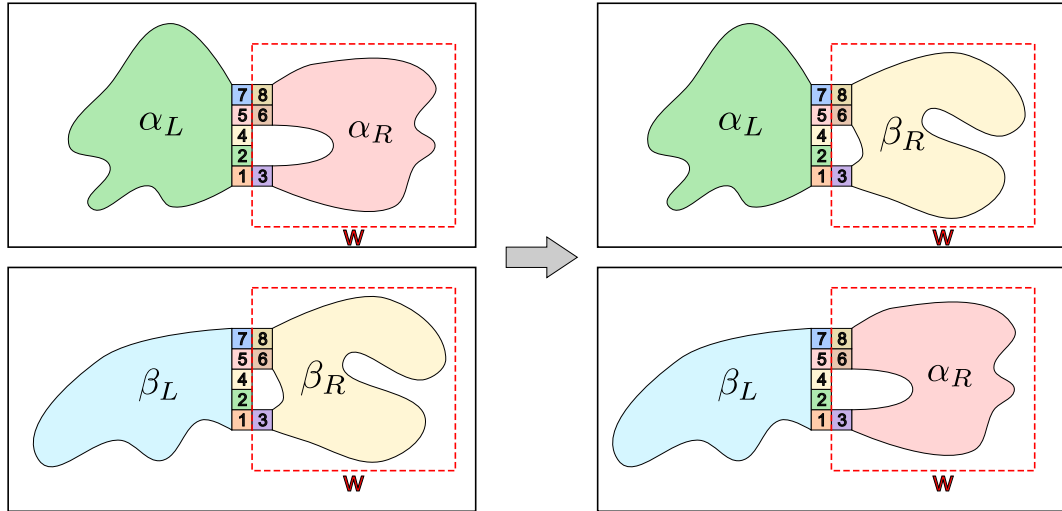
<sup>1</sup> Note that this definition requires that the seed assembly  $\sigma_{\mathcal{S}}$  resolves to at least one tile in  $\mathcal{T}$ . This will suffice for the constructions of this paper, but if needed this definition can be relaxed to specifically allow for the seed assembly to grow before resolving.

assembly representing any  $\beta$  producible from  $\alpha$ , while the second condition specifies that any assembly  $\alpha''$  representing  $\alpha$  that may produce an assembly representing  $\beta$  is producible from an assembly in  $\Pi_\alpha$ . In this way,  $\Pi_\alpha$  represents a set of the earliest possible representations of  $\alpha$  where no commitment has yet been made regarding the next simulated assembly. Requiring the existence of such a set  $\Pi_\alpha$  for every producible  $\alpha$  ensures that non-determinism is faithfully simulated. That is, the simulation cannot simply “decide in advance” which tile attachments will occur.

► **Definition 14 (Simulates).** *Given an aTAM system  $\mathcal{T} = (T, \sigma, 2)$ , a syncTAM system  $\mathcal{S} = (S, \sigma_S, c)$ , and a macrotile representation function  $R$  from  $\mathcal{S}$  to  $\mathcal{T}$ , we say that  $\mathcal{S}$  intrinsically simulates  $\mathcal{T}$  (under  $R$ ) if  $\mathcal{S} \Leftrightarrow_R \mathcal{T}$  (they have equivalent productions),  $\mathcal{T} \dashv_R \mathcal{S}$  and  $\mathcal{S} \models_R \mathcal{T}$  (they have equivalent dynamics).*

## A.2 Window Movie Lemma

The Window Movie Lemma is a powerful tool often used in impossibility results for tile-based self-assembly models. Originally introduced in [18], it allow us to perform surgery on an assembly sequence to obtain a new producible assembly (assuming that some conditions are met). In its simplest form, the lemma states that if there are two enclosed areas of the plane that have the same “shape” and the same glues appear in the same order at the same position relative to these two enclosed shapes (that is, the same window movie), then the subassemblies produced in the two enclosed areas are interchangeable. See Figure 7 for a graphical representation of this simple case of the Window Movie Lemma.



■ **Figure 7** An illustration of the window movie lemma. The squares represent tiles and the color of the tile is an indication of the glue type the tile exposes along the window  $w$  (represented by a red dotted box). The numbers on the tile represent the relative order in which the glues contained on the tile appear on the window as the assembly sequence is played forward. The Window Movie Lemma tells us that since  $\alpha_L \cup \alpha_R$  is producible (shown top left) and  $\beta_L \cup \beta_R$  is producible (show bottom left) and because the assembly sequence produces the same window movie sequence along  $w$ , the assemblies shown on the right are also producible.

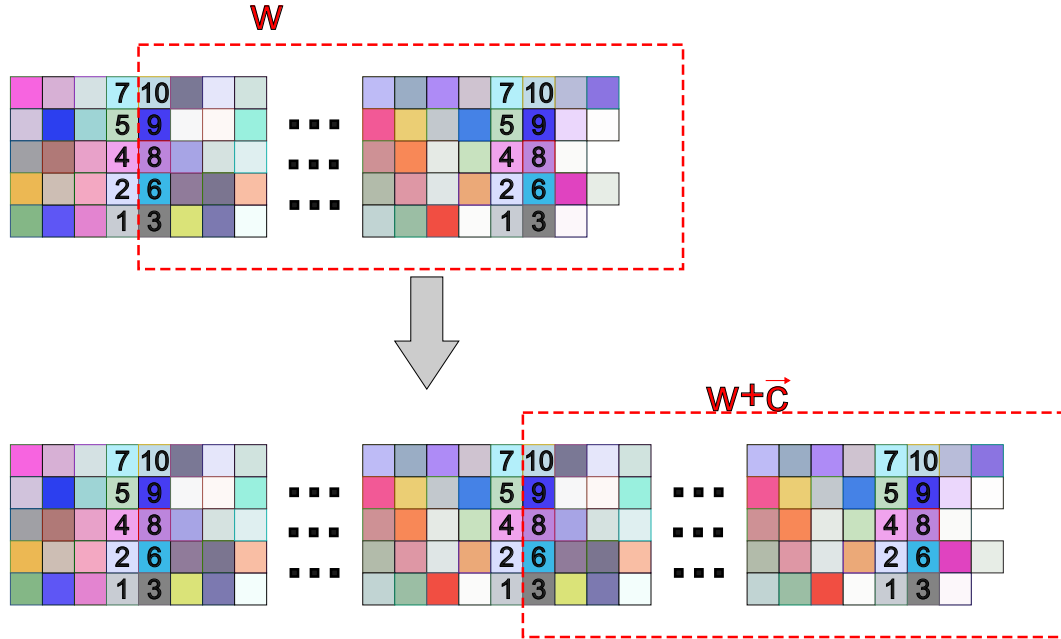
The above surgical procedure can be extended in order to pump assemblies. Indeed, if there is a fixed width subassembly of sufficient length, then there is some window movie which repeats. We can then pump the subassembly between these two window movies indefinitely. See Figure 8 for a graphical representation showing how assemblies can be pumped via the Window Movie Lemma.



The following definitions are taken from [18], and replicated here for completeness. A *window* is an edge cut which partitions the lattice graph ( $\mathbb{Z}^2$  in 2D or  $\mathbb{Z}^3$  in 3D) into two regions. Given some window  $w$  and some assembly sequence  $\vec{\alpha}$  in a TAS  $\mathcal{T}$ , a *window movie*  $M$  is defined to be the ordered sequence of glues presented along  $w$  by tiles in  $\mathcal{T}$  during the assembly sequence  $\vec{\alpha}$ . Informally, the window  $w$  is a cut dividing two regions of tile locations and  $M$  is constructed by recording the glues which appear on the cut and their relative order as the assembly sequence  $\vec{\alpha}$  is played forward. More formally, a *window movie* is the sequence  $M_w^{\vec{\alpha}} = \{(v_i, g_i)\}$  of pairs of grid graph vertices  $v_i$  and glues  $g_i$ , given by order of appearance of the glues along window  $w$  during  $\vec{\alpha}$ . Furthermore, if  $k$  glues appear along  $w$  during the same assembly step in  $\vec{\alpha}$ , then these glues appear contiguously and are listed in lexicographical order of the unit vectors describing their orientation in  $M_w^{\vec{\alpha}}$ .

### Window Movie Lemma

Let  $\vec{\alpha} = \{\alpha_i\}$  and  $\vec{\beta} = \{\beta_i\}$  be assembly sequences in TAS  $\mathcal{T}$  and let  $\alpha, \beta$  be the result assemblies of each respectively. Let  $w$  be a window that partitions  $\alpha$  into two configurations  $\alpha_L$  and  $\alpha_R$  and let  $w' = w + \vec{c}$  be a translation of  $w$  that partitions  $\beta$  into two configurations  $\beta_L$  and  $\beta_R$  (with  $\alpha_L$  and  $\beta_L$  being the configurations containing their respective seed tiles). Furthermore define  $M_w^{\vec{\alpha}}$  and  $M_{w'}^{\vec{\beta}}$  to be the window movies for  $\vec{\alpha}, w$  and  $\vec{\beta}, w'$  respectively. Then if  $M_w^{\vec{\alpha}} = M_{w'}^{\vec{\beta}}$ , the assemblies  $\alpha_L \cup \beta'_R$  and  $\beta'_L \cup \alpha_R$  (where  $\beta'_L = \beta_L - \vec{c}$  and  $\beta'_R = \beta_R - \vec{c}$ ) are also producible.



**Figure 8** An illustration of how the Window Movie Lemma can be leveraged to pump assemblies. The squares represent tiles and the color of the tile is an indication of the glue type the tile exposes along the window  $w$  (represented by a red dotted box). The numbers on the tile represent the relative order in which the glues contained on the tile appear on the window as the assembly sequence is played forward. The top assembly shows a fixed width ribbon grown sufficiently long. By the pigeon hole principle, there must exist a window that has the same window movie when translated to two distinct positions along the ribbon (as indicated by the same colored tiles with the same numbers on them occurring twice in the ribbon). Then, by the Window Movie Lemma, the assembly between these two translations of the window can be repeated indefinitely. This is illustrated in the bottom portion of the figure which shows that the portion of the ribbon between repeating window movies can be also be assembled beginning where the last repeating window movie is located.



# Computing and Bounding Equilibrium Concentrations in Athermic Chemical Systems

Hamidreza Akef  

The University of Texas at Austin, TX, USA

Minki Hhan  

The University of Texas at Austin, TX, USA

David Soloveichik  

The University of Texas at Austin, TX, USA

---

## Abstract

Computing equilibrium concentrations of molecular complexes is generally analytically intractable and requires numerical approaches. In this work we focus on the polymer-monomer level, where indivisible molecules (monomers) combine to form complexes (polymers). Rather than employing free-energy parameters for each polymer, we focus on the athermic setting where all interactions preserve enthalpy. This setting aligns with the strongly bonded (domain-based) regime in DNA nanotechnology when strands can bind in different ways, but always with maximum overall bonding – and is consistent with the saturated configurations in the Thermodynamic Binding Networks (TBNs) model. Within this context, we develop an iterative algorithm for assigning polymer concentrations to satisfy detailed-balance, where on-target (desired) polymers are in high concentrations and off-target (undesired) polymers are in low. Even if not directly executed, our algorithm provides effective insights into upper bounds on concentration of off-target polymers, connecting combinatorial arguments about discrete configurations such as those in the TBN model to real-valued concentrations. We conclude with an application of our method to decreasing leak in DNA logic and signal propagation. Our results offer a new framework for design and verification of equilibrium concentrations when configurations are distinguished by entropic forces.

**2012 ACM Subject Classification** Theory of computation → Models of computation; Theory of computation → Design and analysis of algorithms

**Keywords and phrases** Equilibrium concentrations, Thermodynamic Binding Networks, Monomer-polymer model, Detailed balance

**Digital Object Identifier** 10.4230/LIPIcs.DNA.31.10

**Funding** *Hamidreza Akef*: Support was provided by Schmidt Sciences.

*Minki Hhan*: Support was provided by Schmidt Sciences.

*David Soloveichik*: Support was provided by Schmidt Sciences, Department of Energy award DE-SC0024467, and National Science Foundation SemiSynBio III: GOALI award.

**Acknowledgements** We thank Joshua Petrack for valuable discussions and insights.

## 1 Introduction

In general, chemical equilibria of complex chemical systems are not analytically solvable and numerical tools are required for analysis. Such tools include NUPACK [17] for thermodynamic analysis of nucleic-acid systems, as well as more abstract platforms that support domain-level abstraction and free energy specification [12] including via rule-based modeling [11], and software for computing steady-state concentrations of chemical reaction networks [7, 10]. However, engineers often want a deeper understanding of the equilibrium than what analytically opaque numerical calculations can provide. Moreover, we often seek to understand infinite classes of designs such as logic circuits constructed from gate modules or parameterized constructions. For example, there is a growing body of work on leak in DNA-based systems,



© Hamidreza Akef, Minki Hhan, and David Soloveichik;  
licensed under Creative Commons License CC-BY 4.0

31st International Conference on DNA Computing and Molecular Programming (DNA 31).

Editors: Josie Schaeffer and Fei Zhang; Article No. 10; pp. 10:1–10:19

Leibniz International Proceedings in Informatics



LIPICs Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

exhibiting a family of schemes parameterized by a “redundancy parameter” meant to decrease leak arbitrarily at the cost of additional system components [13, 5, 15, 1, 16, 14]. It is proven that producing off-target species necessarily decreases the overall number of separate complexes – the change controlled by the redundancy parameter – and thus incurs an entropic penalty. However, due to system complexity, the relationship between this rigorously proven thermodynamic unfavorability and the actual concentrations of off-target species often remains implicit.

While properties of equilibria of abstract coupled chemical reactions have been extensively explored in the chemical reaction network theory literature (e.g., [6], Chapter 14, for detailed-balance equilibria), and full explicit-parameter schemes (e.g., [9]) can characterize the entire space of equilibria, these analytical approaches have severe limitations. For large systems the resulting explicit formulas are unwieldy and offer no effective guidance on how to choose the parameters so that off-target concentrations remain below a desired bound.

In this paper we are interested in the following problem: Our chemical species are complexes (termed polymers) made up of indivisible units called monomers. We assume there are finitely many possible polymers, and among these we are given a set of on-target species that we want to have desired equilibrium concentrations, and all other off-target species to have some sufficiently low equilibrium concentrations. Our task is to determine a consistent (detailed-balance) equilibrium, and therefore the concentrations of the monomers that would lead to this equilibrium. Note that our interest in setting equilibrium concentrations rather than solving for them (based on initial concentrations or total monomer concentrations, for example) is misaligned with most computational approaches. For instance, it can be seen as the reverse problem to NUPACK 3’s **concentrations** tool which takes the concentrations of the monomers (strands) and returns the corresponding equilibrium concentrations of the polymers (complexes).

While having the flexibility to set monomer concentrations may appear to simplify the problem in the same way that finding *some* detailed-balance equilibrium is easier than computing the one consistent with input monomer concentrations, the complexity comes from simultaneously ensuring that off-target polymers remain in low concentration. Consider the natural approach of taking logarithms of all concentrations, thereby converting the detailed-balance equations (balancing each reaction) into a linear system amenable to standard linear solvers. When we fix the concentrations of on-target polymers, the remaining (off-target) concentrations are typically under-determined. The linear system then describes an unbounded affine subspace, and there is no obvious way to extract upper bounds on off-target species’ concentrations without leaving the linear framework. Our solution is an iterative algorithm that assigns concentrations to off-target polymers in decreasing order of concentration, ensuring that each off-target species remains below a desired threshold concentration when possible. Importantly, terminating the algorithm at any iteration still provides valid upper bounds for all remaining off-target polymers.

In the most general formulation of the monomer-polymer equilibrium concentration problem, the polymer free energies can be assigned arbitrarily incorporating binding strength, geometric constraints, etc. In this work, we focus on the simpler *athermic* case rather than tackling the problem in its full generality. Our allowed polymers are such that all possible reactions between them are enthalpy-neutral. This model is consistent with systems of strong fully-complementary DNA domains in which domain-level bonds can only switch binding partners but not de-hybridize. Thermodynamic Binding Network (TBN) [5, 2] saturated configurations capture this condition, but our setting is more general without a built-in notion of domains (binding sites).

The main result of this paper is Algorithm 1 and Theorem 5.4 showing how starting with desired concentrations of on-target polymers (already in detailed balance), we can set the concentrations of off-target polymers to satisfy detailed balance and thus thermodynamic equilibrium. In Section 4 we explain the apparent difficulties in balancing reactions which our approach needed to overcome.

If, rather than computing exact concentrations of off-target polymers, it is sufficient to bound them, then we refer the reader to Section 6. In Section 7 we apply our framework to the analysis of systems in the TBN model specifically, connecting the combinatorial notions of stability and entropy loss in the TBN model to equilibrium concentrations. In Section 8, we show applications of our method to the analysis of a simple TBN AND gate, as well as a parameterized family of signal propagation systems (translator cascades) from prior work. For the translator cascade, we argue that tuning concentrations of on-target polymers according to our framework is essential for leak to decrease exponentially with the redundancy parameter. We conclude with a discussion of future work (Section 9), including a formulation of new combinatorial conditions in the TBN model to make our framework easily applicable.

## 2 Model

Let  $\mathbb{N}$  denote the set of nonnegative integers. Given a finite set  $\mathcal{A}$ , we define  $\mathbb{N}^{\mathcal{A}}$  as the set of functions  $f : \mathcal{A} \rightarrow \mathbb{N}$ .

A multiset  $\mathcal{M}$  over the finite set  $\mathcal{A}$  is described by its *counting function*  $f_{\mathcal{M}} \in \mathbb{N}^{\mathcal{A}}$ , where for each element  $a \in \mathcal{A}$ , the value  $f_{\mathcal{M}}(a)$  indicates how many times  $a$  appears in  $\mathcal{M}$ . We often write  $\mathcal{M} \in \mathbb{N}^{\mathcal{A}}$  to mean that  $\mathcal{M}$  is a multiset over  $\mathcal{A}$ , and we denote the count of  $a \in \mathcal{A}$  in  $\mathcal{M}$  by  $\mathcal{M}[a]$ . The notation  $a \in \mathcal{M}$  means that  $\mathcal{M}[a] \geq 1$ . The *cardinality* of a multiset  $\mathcal{M}$ , denoted  $|\mathcal{M}|$ , is the total number of elements in the multiset  $|\mathcal{M}| = \sum_{a \in \mathcal{A}} \mathcal{M}[a]$ . For two multisets  $\mathcal{M}$  and  $\mathcal{M}'$  over  $\mathcal{A}$ , we define their union  $\mathcal{M} + \mathcal{M}'$  as the multiset whose counting function is the pointwise sum  $(\mathcal{M} + \mathcal{M}')(a) = \mathcal{M}[a] + \mathcal{M}'[a]$  for all  $a \in \mathcal{A}$ . For example, let  $\mathcal{M} = \{a, a, b, c\}$ . Then,  $\mathcal{M}[a] = 2$ ,  $\mathcal{M}[b] = 1$ ,  $\mathcal{M}[c] = 1$ , and  $\mathcal{M}[d] = 0$  for all  $d \notin \{a, b, c\}$ . Also, the cardinality of  $\mathcal{M}$  is  $|\mathcal{M}| = 2 + 1 + 1 = 4$ . Finally, note that  $\mathcal{M}$  could also be written as the union  $\{a, b\} + \{a, c\}$ , for example.

The linear combination of multisets with nonnegative integers is defined analogously: For multisets  $\mathcal{M}_1, \dots, \mathcal{M}_n$  and  $a_1, \dots, a_n \in \mathbb{N}$ ,  $\sum_{i=1}^n a_i \cdot \mathcal{M}_i$  corresponds to the counting function  $\sum_{i=1}^n a_i \cdot \mathcal{M}_i[a]$ . Let  $\mathcal{M}_1, \mathcal{M}_2 \in \mathbb{N}^{\mathcal{S}}$  be two multisets over the same set  $\mathcal{S}$ . The difference  $\mathcal{M}_1 - \mathcal{M}_2$  is defined as the multiset  $\mathcal{M} \in \mathbb{N}^{\mathcal{S}}$  such that for every  $a \in \mathcal{S}$ ,  $(\mathcal{M}_1 - \mathcal{M}_2)[a] = \mathcal{M}_1[a] - \mathcal{M}_2[a]$  provided that  $\mathcal{M}_1[a] \geq \mathcal{M}_2[a]$  for all  $a \in \mathcal{S}$ .

We also define the intersection of a multiset with a *set*. Given a multiset  $\mathcal{M}$  over  $\mathcal{A}$  and a subset  $\mathcal{S} \subset \mathcal{A}$ , the intersection  $\mathcal{M} \cap \mathcal{S}$  is a *multiset* over  $\mathcal{A}$  defined by  $(\mathcal{M} \cap \mathcal{S})[a] = \mathcal{M}[a]$  if  $a \in \mathcal{S}$ , otherwise  $(\mathcal{M} \cap \mathcal{S})[a] = 0$ . For instance,  $\{a, a, b, c\} \cap \{a, c\} = \{a, a, c\}$ .

The main object of this paper is an abstract model of monomers and polymers, motivated by systems in DNA nanotechnology. This model captures how simple indivisible molecules (monomers) combine to form complexes (polymers) under specific physical and chemical constraints.

► **Definition 2.1.** Let  $\Psi^0$  be a finite set of monomers, and  $\Psi \subseteq \mathbb{N}^{\Psi^0}$  be a finite set of polymers over these monomers, where each polymer  $P \in \Psi$  is a multiset of monomers.

Let  $\mathbf{x}^0 \in (0, 1)^{\Psi^0}$  represent the vector of concentrations for all monomers, and let  $\mathbf{x} \in (0, 1)^{\Psi}$  represent the vector of concentrations for all polymers (also called configuration). The relationship between monomer and polymer concentrations is governed by mass conservation.

Specifically, we require

$$\mathbf{x}^0 = \mathbf{A} \cdot \mathbf{x} \quad (1)$$

where  $\mathbf{A} \in \mathbb{N}^{|\Psi^0| \times |\Psi|}$  is a matrix such that each entry  $A_{ij}$  specifies the number of monomers of type  $i$  in polymer  $j$ .

For example, the polymer  $P = \{m_1, m_1, m_2, m_3\}$  contains two copies of  $m_1$ , one of  $m_2$ , and one of  $m_3$ . Note that we will be interested in cases where the set of polymers of interest  $\Psi$  is a finite (proper) subset of all possible polymers over  $\Psi^0$ .

In DNA nanotechnology the monomers are typically DNA strands with different sequences. Polymers are analogous to a multistranded DNA structure composed of multiple DNA strands. We use the term polymer rather than “complex” in order to better emphasize their composition from monomers and to be consistent with the TBN literature.

To model the equilibrium behavior of such systems, we use the free energy formulation in the notation of Dirks et al. [4], where the equilibrium concentrations are obtained by minimizing the following free energy function (corresponding to the pseudo-Helmholtz free energy used throughout chemical reaction network theory literature [8, 6]):

$$\mathbf{g}(\mathbf{x}) = \sum_{P \in \Psi} x_P (\log x_P - \log \Omega_P - 1) \quad (2)$$

where  $x_P$  denotes the concentration of polymer  $P$ , and  $\Omega_P$  is its partition function corresponding to the exponential of the polymer’s negative free energy. The minimization is subject to the mass conservation constraint given in Equation (1).

In this work, we focus on *athermic* systems where all interactions are equally favored enthalpically. Thus we assume that  $\sum_{P \in \Psi} x_P \cdot \log \Omega_P$  is constant for all configurations  $\mathbf{x}$  satisfying mass conservation (Equation (1)), yielding a simplified cost function that is entirely entropic:

$$g(\mathbf{x}) = \sum_{P \in \Psi} x_P (\log x_P - 1). \quad (3)$$

This function serves as the objective of our optimization problem. Its minimizer under the constraint of Equation (1) corresponds to the equilibrium concentration of polymers.

Understanding the equilibrium requires us to formalize how polymers can transform into one another. We do so by defining reconfigurations and the reactions they induce.

► **Definition 2.2.** *Two multisets of polymers  $\mathcal{M}_1, \mathcal{M}_2 \in \mathbb{N}^\Psi$  are reconfigurations of each other, written  $\mathcal{M}_1 \cong \mathcal{M}_2$ , if for every monomer  $m \in \Psi^0$ , the total count of  $m$  is the same in  $\mathcal{M}_1$  as in  $\mathcal{M}_2$ ; i.e.,  $\sum_{P \in \Psi} \mathcal{M}_1[P] \cdot P[m] = \sum_{P \in \Psi} \mathcal{M}_2[P] \cdot P[m]$ . Whenever  $\mathcal{M}_1 \cong \mathcal{M}_2$ , we also define reaction  $\mathcal{M}_1 \rightarrow \mathcal{M}_2$ .*

We occasionally use Greek alphabets to denote the reactions. Note that while  $\cong$  is a binary relation, a reaction  $\alpha : \mathcal{M}_1 \rightarrow \mathcal{M}_2$  is an ordered pair representing a directed transformation between the multisets. Our arguments will refer to *reactants* (left-hand side) and *products* (right-hand side) of particular reactions.<sup>1</sup>

An important property of the minimum of Equations (2) and (3) is that it satisfies detailed balance over reactions [8, 6]. For us (Equation (3)), for any single reaction  $\alpha : \mathcal{M}_1 \rightarrow \mathcal{M}_2$ , the equilibrium concentrations satisfy:

$$\prod_{P \in \mathcal{M}_1} x_P^{\mathcal{M}_1[P]} = \prod_{P \in \mathcal{M}_2} x_P^{\mathcal{M}_2[P]}.$$

<sup>1</sup> All reactions are of course reversible, but we treat each direction as a separate reaction.

We say that a reaction  $\alpha$  is balanced when this equality holds. This notion of balance will be central to our characterization of equilibrium concentrations formed by on-target polymers, which we explore next.

The fact that balance leads to the minimum of the free energy is well-established, and we present its sketch in Section B for the completeness of the paper.

### 3 Characterizing On-target Polymers

We now define the set  $\mathcal{S}$  of on-target polymers – intuitively, these are the high-concentration polymers whose equilibrium concentration we set as input to our algorithm.

► **Definition 3.1.** Let  $\mathcal{S} \subseteq \Psi$  be a set of polymers, and let  $\mu : \mathcal{S} \rightarrow (0, 1]$  be a function assigning a concentration exponent to each polymer in  $\mathcal{S}$ . We say that  $\mathcal{S}$  is on-target with concentration exponents  $\mu$  if:

1. For every polymer  $P \in \Psi$ , there are multisets of polymers  $\mathcal{M} \in \mathbb{N}^{\mathcal{S}}$  and  $\mathcal{M}' \in \mathbb{N}^{\Psi}$  where  $\mathcal{M}'$  contains  $P$  such that  $\mathcal{M} \cong \mathcal{M}'$ .
2. For any two multisets  $\mathcal{M}_1, \mathcal{M}_2 \in \mathbb{N}^{\mathcal{S}}$ , if  $\mathcal{M}_1 \cong \mathcal{M}_2$ , then their concentration exponents are equal  $\mu(\mathcal{M}_1) = \mu(\mathcal{M}_2)$ . Here, the concentration exponent of a multiset  $\mathcal{M}$  is defined as  $\mu(\mathcal{M}) = \sum_{P \in \mathcal{M}} \mathcal{M}[P] \cdot \mu(P)$ .

While presented in terms of restricting  $\mathcal{S}$ , another (perhaps more apropos) interpretation of the first condition is that we are restricting  $\Psi$  to be only those polymers that can be obtained via a reconfiguration of polymers over  $\mathcal{S}$ . The interpretation of the second condition is that every reaction among polymers in  $\mathcal{S}$  is in detailed balance. More precisely:

► **Remark 3.2.** Let  $0 < c < 1$ , and suppose every on-target polymer  $P \in \mathcal{S}$  has concentration  $c^{\mu(P)}$ . Then every reaction  $\mathcal{M}_1 \rightarrow \mathcal{M}_2$ , where  $\mathcal{M}_1, \mathcal{M}_2 \in \mathbb{N}^{\mathcal{S}}$ , satisfies  $c^{\mu(\mathcal{M}_1)} = c^{\mu(\mathcal{M}_2)}$  and thus the product of the concentrations of the left-hand side polymers is equal to the product of the concentrations of the right-hand side polymers.<sup>23</sup>

If  $\mathcal{S} = \Psi$  then we are done, being guaranteed that all reactions are in detailed balance. However, the case of interest is where we do not know the concentration exponents of all polymers – making it the goal of this paper to compute them or bound them to ensure that the equilibrium concentration of off-target polymers is small.

We now define canonical reactions, which are the reactions with reactants from  $\mathcal{S}$ . These reactions can generate polymers outside of  $\mathcal{S}$  from polymers in  $\mathcal{S}$ . Each canonical reaction has key quantities we call imbalance and novelty which will play a key role in our algorithm.

► **Definition 3.3.** For an on-target set  $\mathcal{S}$  with  $\mu$ , a reaction  $\alpha : \mathcal{M}_1 \rightarrow \mathcal{M}_2$  is called canonical if all its reactants are over  $\mathcal{S}$  (i.e.,  $\mathcal{M}_1 \in \mathbb{N}^{\mathcal{S}}$ ). Then the imbalance of  $\alpha$  is defined as  $k(\alpha) := \mu(\mathcal{M}_1) - \mu(\hat{\mathcal{M}}_2)$ , where  $\hat{\mathcal{M}}_2 := \mathcal{M}_2 \cap \mathcal{S}$ . The novelty of  $\alpha$  is defined by  $l(\alpha) := |\mathcal{M}_2 - \hat{\mathcal{M}}_2|$ .

Given the definition of multiset subtraction, the novelty  $l(\alpha)$  is always non-negative.

Intuitively, the imbalance of the reaction represents how far from detailed balance it is if we consider only the polymers whose concentrations are known. The novelty is the number of off-target (new) polymers produced by the same reaction.

<sup>2</sup> Note that we use symbol  $\mu$  for concentration exponents because of the direct parallel to the standard notion of chemical potential, which is expressed with logarithmic terms of concentration: chemical potential  $\mu = \mu^\circ + RT \ln(x)$ , where  $x$  is the mole fraction.

<sup>3</sup> Our base concentration  $c$  is smaller than one because the units of concentration are mole fractions – the ratio of polymers to solvent molecules, and the derivation of the energy function Equation (2) only holds in the regime of less polymer than solvent [4].



Intuitively, a large  $k(\alpha)$  means a larger bias of the reaction toward its reactants, i.e., elements of  $\mathcal{S}$ , prior to assigning concentration exponents to off-target polymers. This is desired since it implies less pressure to make off-target polymers and gives us more room to maneuver in assigning concentrations to them.

Intuitively, a larger novelty  $l(\alpha)$  means that the canonical reaction is more entropically favored to its products. Note that the term  $\sum_{P \in \Psi} x_P \log x_P$  in Equation (3) is minimized when the concentration is more “spread out” over the different species (like Shannon entropy), and thus there is an entropic force to generate new polymers. A large  $l(\alpha)$  is thus undesired. As justified in the subsequent results, the ratio  $k(\alpha)/l(\alpha)$  captures the effective tradeoff between imbalance and novelty.

Note that summing canonical reactions gives another canonical reaction. While on the one hand this leads to infinitely many canonical reactions, on the other hand, this additive property allows us to analyze the set of all canonical reactions using a Hilbert basis of *finite size*. We explain how to employ the Hilbert basis in our algorithm to avoid the infinity of canonical reactions in Section A.

The notion of stability of  $\mathcal{S}$ , defined below, captures the idea that on-target polymers are in higher concentration compared to off-target polymers. Recall that concentration exponents  $\mu$  are  $\leq 1$  for polymers in  $\mathcal{S}$  (Definition 3.1). As shown later by Theorem 5.4, the following definition ensures that all concentration exponents of polymers outside of  $\mathcal{S}$  are greater than 1 by our algorithm. Since the base concentration  $c < 1$ , this implies that off-target polymers are in smaller concentration.

► **Definition 3.4.** *The on-target set  $\mathcal{S}$  with  $\mu$  is called stable if, for every canonical reaction  $\alpha$  with  $l(\alpha) \neq 0$ , the ratio  $k(\alpha)/l(\alpha) > 1$ .*

While our formalism allows different concentration exponents for different polymers in  $\mathcal{S}$ , nearly all insight can be obtained from the simple case of uniform on-target concentration exponents:

► **Definition 3.5.** *The on-target set  $\mathcal{S}$  with  $\mu$  is called uniform if every polymer  $P \in \mathcal{S}$  has concentration exponent  $\mu(P) = 1$ .*

## 4 Why Balancing is Nontrivial

Our main goal is to ensure that in equilibrium the concentration of on-target polymers is much higher than the others. This section presents two examples illustrating why this is nontrivial.

Consider a uniform set of on-target polymers  $\mathcal{S} = \{A, B, C, D\}$ , aiming at the equilibrium concentration  $x_A = x_B = x_C = x_D = c$ . For two off-target polymers  $P_1$  and  $P_2$ , suppose that we figured out a canonical reaction

$$\alpha : 2A + B + C \rightarrow P_1 + P_2.$$

This reaction is in detailed balance if and only if  $x_{P_1} \cdot x_{P_2} = x_A^2 \cdot x_B \cdot x_C = c^4$ . At this point, it is unclear how to balance  $x_{P_1}$  and  $x_{P_2}$  without inspecting the other reactions: For example, the reaction  $\alpha$  is balanced by assigning  $x_{P_1} = x_{P_2} = c^2$ , but another canonical reaction  $A + 2B + 2D \rightarrow 2P_1$  would not be if it exists.

The following example exhibits a different potential problem. While Remark 3.2 shows that reactions entirely within  $\mathcal{S}$  are balanced, it is not clear that given our choice of concentration exponents  $\mu$  for  $\mathcal{S}$ , the reactions involving polymers outside of  $\mathcal{S}$  could be balanced at all. Suppose there are two reactions

$$\beta : A + B \rightarrow P_3 \text{ and } \gamma : B + C + D \rightarrow 2P_3.$$

These reactions intuitively say that the off-target polymer  $P_3$  is non-interacting to other polymers in these reactions so that the above problem of balancing the concentrations over off-target polymers is absent.<sup>4</sup> However, balancing the reaction  $\beta$  suggests the concentration  $x_{P_3} = c^2$  at equilibrium, but the reaction  $\gamma$  gives  $x_{P_3} = c^{1.5}$ . In other words, it is unclear if the concentrations of each of the polymers in  $\mathcal{S}$  can be the same or even close to each other.

Despite these difficulties, our main result guarantees that the configuration on on-target polymers we demand is in equilibrium without the above issue, and can be *extended* to the configuration over all polymers at equilibrium, and ensures that the off-target polymers remain in very low concentration at equilibrium. Intuitively, we prove that there exists a canonical reaction for which we can assign concentrations to product polymers without creating conflicts with other reactions, and we provide a method to find this reaction. Moreover, as we will see later, these assignments occur in order of decreasing concentration allowing us to restrict the concentration of off-target species.

## 5 Main Result: Concentration of Off-target Polymers

For the remainder, we fix  $\Psi^0$ ,  $\Psi$ , and a particular set of on-target polymers  $\mathcal{S}$  with concentration exponents  $\mu$ .

To systematically assign concentration exponents to all polymers, we will organize them into hierarchical groups called levels. The process begins with a designated set of polymers  $\mathcal{S}$ , the on-target polymers, which are assigned initial concentration exponents via  $\mu : \mathcal{S} \rightarrow \mathbb{R}^+$ .

All other polymers, called off-target polymers, will be partitioned into level sets  $\mathcal{S}_1, \mathcal{S}_2, \dots$ , constructed inductively based on how these polymers appear in certain canonical reactions. At each level  $i$ , we will compute a scalar value  $\mu_i$  that serves as the concentration exponent for all polymers newly added at that level.

In this way, we gradually extend the initial function  $\mu$  to a global function  $\bar{\mu} : \Psi \rightarrow \mathbb{R}^+$  that assigns a concentration exponent to every polymer in the system. This inductive process and the precise definitions of  $\mu_i$ ,  $\mathcal{S}_i$ , and  $\bar{\mu}$  will be described in detail below.

► **Definition 5.1.** Given  $\mathcal{S}_0, \dots, \mathcal{S}_{i-1}$ , assume  $\bar{\mu}(P)$  is assigned for any  $P \in \bigcup_{j=0}^{i-1} \mathcal{S}_j$ . For a canonical reaction  $\alpha : \mathcal{M}_1 \rightarrow \mathcal{M}_2$ , we define  $\hat{\mathcal{M}}_2 := \mathcal{M}_2 \cap \left( \bigcup_{j=0}^{i-1} \mathcal{S}_j \right)$ . The  $i$ th-level imbalance of  $\alpha$  is defined as  $k_i(\alpha) := \bar{\mu}(\mathcal{M}_1) - \bar{\mu}(\hat{\mathcal{M}}_2)$ , and the  $i$ th-level novelty by  $l_i(\alpha) := |\mathcal{M}_2| - |\hat{\mathcal{M}}_2|$ .

While  $\hat{\mathcal{M}}_2$  technically depends on the level index  $i$ , we suppress this dependency in the notation for simplicity; it will be clear from the context that it is updated at each level. Additionally, by the definition of a canonical reaction, we have  $\bar{\mu}(\mathcal{M}_1) = \mu(\mathcal{M}_1)$  in the expression above.

► **Definition 5.2.** Let  $\mu_i = \min_{\alpha} \{k_i(\alpha)/l_i(\alpha)\}$  where the minimum is taken over all canonical reactions  $\alpha$  with  $l_i(\alpha) \neq 0$ .<sup>5</sup> The canonical reactions achieving the minimum are termed  $i$ -levelizing canonical reactions. The  $i$ th level set  $\mathcal{S}_i$  is defined as the set of all polymers  $P$  appearing in any  $i$ -levelizing reactions not already in  $\bigcup_{j=0}^{i-1} \mathcal{S}_j$ . Every polymer  $P \in \mathcal{S}_i$  is assigned concentration exponent  $\bar{\mu}(P) := \mu_i$ .

At the heart of this inductive construction is the requirement that each  $i$ -levelizing reaction maintains balance with respect to the assigned concentration exponents. That is, for every reaction  $\alpha : \mathcal{M}_1 \rightarrow \mathcal{M}_2$  used to define level  $\mathcal{S}_i$ , the assigned exponents ensure the

<sup>4</sup> Looking ahead, these non-interacting polymers are indeed easier to assign concentrations to as shown in Section 5.1.

<sup>5</sup> The minimum can actually be taken over a finite subset of canonical reactions using Hilbert basis. We refer the reader to Section A for more detail.

reaction remains detailed balanced,  $c^{\bar{\mu}(\mathcal{M}_1)} = c^{\bar{\mu}(\mathcal{M}_2)}$ . This holds because each polymer in  $\mathcal{M}_2$  that already appeared in previous levels contributes its already-defined exponent, while newly introduced polymers in  $\mathcal{S}_i$  all receive the same value  $\mu_i$ . Decomposing  $\mathcal{M}_2$  into previously leveled polymers  $\hat{\mathcal{M}}_2$  and new polymers in  $\mathcal{S}_i$ , we obtain:

$$\bar{\mu}(\mathcal{M}_1) = \bar{\mu}(\hat{\mathcal{M}}_2) + k_i(\alpha) = \bar{\mu}(\hat{\mathcal{M}}_2) + \mu_i \cdot l_i(\alpha) = \bar{\mu}(\hat{\mathcal{M}}_2) + \mu_i \cdot (|\mathcal{M}_2| - |\hat{\mathcal{M}}_2|) = \bar{\mu}(\mathcal{M}_2),$$

ensuring that the total concentration exponent on both sides of the reaction is equal. This confirms that the assignment of  $\mu_i$  for polymers in level  $\mathcal{S}_i$  is consistent with the balance requirements dictated by the underlying reactions.

The full procedure for determining the level sets  $\mathcal{S}_i$  and the corresponding concentration exponents  $\mu_i$  is formally specified in Definitions 5.1 and 5.2 and carried out through the step-by-step process described in Algorithm 1. This algorithm inductively builds the extended concentration exponent function  $\bar{\mu}$  by identifying polymers that can be leveled at each step and assigning them an appropriate exponent value based on their participation in canonical reactions.

While the algorithm itself does not explicitly state a stopping condition, its termination is ensured by the structure of the level construction process. Each time a new level  $\mathcal{S}_i$  is defined, it includes at least one polymer not present in any previous level. Since the set of all polymers  $\Psi$  is finite by assumption, only finitely many levels can be introduced. As a result, the inductive process must terminate after assigning a level and concentration exponent to each polymer, thus completing the definition of the extended function  $\bar{\mu}$ .

For us, canonical reactions represent the simplest meaningful interactions and serve as building blocks for more complex behavior. Their central role for our results is motivated by the following lemma:

► **Lemma 5.3.** *Let  $\mathbf{x}^0 \in (0, 1)^{\Psi^0}$  be a vector of concentrations of the monomers. If all canonical reactions are balanced at configuration  $\mathbf{x} \in (0, 1)^{\Psi}$ , then the cost function  $g(\mathbf{x})$  is minimum subject to  $\mathbf{A} \cdot \mathbf{x} = \mathbf{x}^0$ .*

**Proof.** We will show that any arbitrary reaction can be decomposed into a canonical reaction. Consequently, if detailed balance holds for all canonical reactions, it follows that detailed balance holds for all reactions. Per Section B, the cost function  $g$  reaches its minimum – under the constraint  $\mathbf{A} \cdot \mathbf{x} = \mathbf{x}^0$  – when all reactions are balanced. This confirms the statement of the lemma.

Consider an arbitrary non-canonical reaction  $\alpha : \mathcal{M}_1 + P \rightarrow \mathcal{M}_2$  with  $P \notin \mathcal{S}$  where  $\mathcal{M}_1 + P$  denotes the union of two multisets  $\mathcal{M}_1$  and  $\{P\}$ . According to the first condition of Definition 3.1, there exists a canonical reaction  $\beta : \mathcal{M}'_1 \rightarrow \mathcal{M}'_2 + P$  that produces  $P$ . Now, apply  $\beta$  and  $\alpha$  sequentially on the reactants  $\mathcal{M}_1 + \mathcal{M}'_1$ . This gives rise to a new reaction:

$$\gamma : \mathcal{M}_1 + \mathcal{M}'_1 \rightarrow \mathcal{M}_1 + (\mathcal{M}'_2 + P) = (\mathcal{M}_1 + P) + \mathcal{M}'_2 \rightarrow \mathcal{M}_2 + \mathcal{M}'_2.$$

Therefore, the reaction  $\alpha$ , when catalyzed by  $\mathcal{M}'_2$  (i.e., adding  $\mathcal{M}'_2$  to reactants and products) and using the inverse of the canonical reaction  $\beta$ , can be replaced by the reaction  $\gamma$ , which involves fewer reactant polymers outside of  $\mathcal{S}$  than  $\alpha$  did originally:

$$(\mathcal{M}_1 + P) + \mathcal{M}'_2 = \mathcal{M}_1 + (\mathcal{M}'_2 + P) \rightarrow \mathcal{M}_1 + \mathcal{M}'_1 \rightarrow \mathcal{M}_2 + \mathcal{M}'_2.$$

By repeating this procedure,  $\alpha$  decomposes into a canonical reaction. This concludes the proof. ◀

With this groundwork in place, we now state our main theorem, which characterizes the equilibrium distribution of the entire polymer system, both on-target and off-target, in terms of their assigned levels.

► **Theorem 5.4.** *Let  $\mathcal{S}$  be the stable set of on-target polymers with concentration exponents  $\mu : \mathcal{S} \rightarrow (0, 1]$ . For the extended concentration exponent  $\bar{\mu} : \Psi \rightarrow \mathbb{R}^+$  generated by Algorithm 1 and for any  $0 < c < 1$ , there are monomer concentrations  $\mathbf{x}^0 \in (0, 1)^{\Psi^0}$  such that the configuration  $\mathbf{x} \in (0, 1)^{\Psi}$  with each polymer  $P \in \Psi$  at concentration  $c^{\bar{\mu}(P)}$  is the minimum of  $g(\mathbf{x})$  subject to  $\mathbf{A} \cdot \mathbf{x} = \mathbf{x}^0$  (i.e., Equations (1) and (3)). Furthermore,  $\bar{\mu}(P) \geq \mu_1 > 1$  for all  $P \notin \mathcal{S}$ .*

In this configuration, every off-target polymer has strictly lower concentration than any on-target polymer. This is because concentrations scale exponentially with the extended exponent  $\bar{\mu}(P)$ , and off-target polymers are assigned strictly larger exponents than the shared minimal value  $\mu$  of the on-target set  $\mathcal{S}$ . As a result, for  $0 < c < 1$ , off-target concentrations are exponentially suppressed.

■ **Algorithm 1** Calculating level sets and concentration exponents. The repeat loop will terminate because there are finitely many polymers. The Hilbert basis implementation avoiding the infinite set  $\Lambda$  is discussed in Section A.

---

```

1: Input: A set of level-0 polymers  $\mathcal{S}_0 = \mathcal{S}$  with  $\mu(\cdot)$  and all canonical reactions  $\Lambda$ 
2: Output: Sets of level- $i$  polymers  $\mathcal{S}_i$  and concentration exponents  $\bar{\mu}(\cdot)$ 
3: Set  $i \leftarrow 0$ ,  $\mathcal{S}_j \leftarrow \emptyset$  for all  $j \in \mathbb{N}$ 
4: repeat
5:    $i \leftarrow i + 1$ 
6:   for each canonical reaction  $\alpha : \mathcal{M}_1 \rightarrow \mathcal{M}_2 \in \Lambda$  do
7:      $\hat{\mathcal{M}}_2 \leftarrow$  the multiset of all level  $\leq (i - 1)$  polymers in  $\mathcal{M}_2$ .
8:      $k_i(\alpha) \leftarrow \bar{\mu}(\mathcal{M}_1) - \bar{\mu}(\hat{\mathcal{M}}_2)$  and  $l_i(\alpha) \leftarrow |\mathcal{M}_2| - |\hat{\mathcal{M}}_2|$ 
9:     Compute  $\mu_i = \min_{\alpha} \{k_i(\alpha)/l_i(\alpha)\}$  where min is taken over  $\alpha \in \Lambda$  with  $l_i(\alpha) \neq 0$ .
10:    for each  $\alpha : \mathcal{M}_1 \rightarrow \mathcal{M}_2 \in \Lambda$  such that  $\mu_i = k_i(\alpha)/l_i(\alpha)$  do
11:      Append  $\alpha$  to the set of  $i$ -levelizing reactions
12:      Append all polymers  $P$  in  $\mathcal{M}_2$  that are not in  $\hat{\mathcal{M}}_2$  to  $\mathcal{S}_i$  and assigns  $\bar{\mu}(P) = \mu_i$ 
13: until All polymers are included in  $\cup_{j=0}^i \mathcal{S}_j$ 
14: return The sets  $\mathcal{S}_i$  and the concentration exponents  $\bar{\mu}(\cdot)$ 

```

---

**Proof of Theorem 5.4.** We use a proof by contradiction to show that each canonical reaction must be balanced in the configuration with  $[P] = c^{\bar{\mu}(P)}$ , which suffices to conclude the proof thanks to Lemma 5.3.

Suppose that there exists a canonical reaction  $\alpha : \mathcal{M}_1 \rightarrow \mathcal{M}_2$ . We consider two cases  $\bar{\mu}(\mathcal{M}_1) < \bar{\mu}(\mathcal{M}_2)$  or  $\bar{\mu}(\mathcal{M}_1) > \bar{\mu}(\mathcal{M}_2)$ . Recall all polymers are leveled.

**Case 1.**  $\bar{\mu}(\mathcal{M}_1) < \bar{\mu}(\mathcal{M}_2)$ . Let  $t$  be the top level among the polymers in  $\mathcal{M}_2$ . Note that  $\bar{\mu}(P) = \mu_t$  for each  $P \in \mathcal{S}_t$ . Since  $\mu_t$  is defined as the minimum of  $k_t(\alpha')/l_t(\alpha')$  over all canonical reactions  $\alpha' \in \Lambda$  with  $l_t(\alpha') \neq 0$ , we must have:

$$\mu_t \leq \frac{k_t(\alpha)}{l_t(\alpha)} = \frac{\bar{\mu}(\mathcal{M}_1) - \bar{\mu}(\hat{\mathcal{M}}_2)}{l_t(\alpha)} < \frac{\bar{\mu}(\mathcal{M}_2) - \bar{\mu}(\hat{\mathcal{M}}_2)}{l_t(\alpha)} = \mu_t$$

which is a contradiction.

**Case 2.**  $\bar{\mu}(\mathcal{M}_1) > \bar{\mu}(\mathcal{M}_2)$ . Let  $\beta_P$  be the levelizing reaction including  $P$  as product for each polymer  $P \in \mathcal{M}_2$ . Consider a canonical reaction  $\sum_P \mathcal{M}_2[P] \cdot \beta_P : \mathcal{M}'_1 \rightarrow \mathcal{M}'_2 + \mathcal{M}_2$  obtained by summing  $\mathcal{M}_2[P]$  copies of  $\beta_P$  which includes  $\mathcal{M}_2$  as products.<sup>6</sup> By applying  $\mathcal{M}_2 \rightarrow \mathcal{M}_1$  to this reaction, we have a canonical reaction  $\beta : \mathcal{M}'_1 \rightarrow \mathcal{M}'_2 + \mathcal{M}_2 \rightarrow \mathcal{M}'_2 + \mathcal{M}_1$  such that

$$\bar{\mu}(\mathcal{M}'_1) = \bar{\mu}(\mathcal{M}'_2) + \bar{\mu}(\mathcal{M}_2) < \bar{\mu}(\mathcal{M}'_2) + \bar{\mu}(\mathcal{M}_1).$$

Therefore this case is reduced to **Case 1**, leading once again to a contradiction.

The “Furthermore” part is proven in Corollary 6.4. ◀

## 5.1 Non-Interacting Off-target Polymers

Our main theorem implies that if the on-target polymers  $\mathcal{S}$  and the corresponding concentration exponents  $\mu$  are chosen appropriately, the equilibrium *exists* with the extended concentration exponents  $\bar{\mu}$  consistent with  $\mu$ . In this section, we show that  $\bar{\mu}$  exponents for *non-interacting* off-target polymers  $P$  can be assigned in a particularly easy way. Non-interacting off-target polymers are those that can be independently made by a canonical reaction:

► **Corollary 5.5.** *Let  $\mathcal{S}$  be a stable set of on-target polymers. Suppose that an off-target polymer  $P \notin \mathcal{S}$  is a product of a canonical reaction  $\alpha_P : \mathcal{M}_P \rightarrow \mathcal{M}'_P + P$  for some multisets  $\mathcal{M}_P$  and  $\mathcal{M}'_P$  from  $\mathcal{S}$ .<sup>7</sup> Then  $\bar{\mu}(P) = \mu(\mathcal{M}_P) - \mu(\mathcal{M}'_P)$ .*

**Proof.** By Theorem 5.4, the reaction  $\alpha_P$  is in detailed balance,  $c^{\bar{\mu}(\mathcal{M}_P)} = c^{\bar{\mu}(\mathcal{M}'_P + P)}$ , and we take the logarithm of both sides. ◀

The same idea extends further. At any point of the execution of the algorithm, suppose that the extended concentration exponents of the set of polymers  $\bar{\mathcal{S}}$  have been already assigned. For a non-interacting polymer  $P$  outside of  $\bar{\mathcal{S}}$  with the reaction  $\alpha_P : \mathcal{M}_P \rightarrow \mathcal{M}'_P + P$  for multisets  $\mathcal{M}_P$  and  $\mathcal{M}'_P$  from  $\bar{\mathcal{S}}$ , we can assign  $\bar{\mu}(P) = \bar{\mu}(\mathcal{M}_P) - \bar{\mu}(\mathcal{M}'_P)$ . This assignment is valid by the same reason to the corollary.

► **Example 5.6.** Consider the set of monomers and polymers given by  $\Psi^0 = \{a, b, c\}$  and  $\Psi = \{A, B, C, X, Y, Z\}$ ; where  $A = \{a, a\}$ ,  $B = \{a, b\}$ ,  $C = \{c\}$ ,  $X = \{a, a, a, b\}$ ,  $Y = \{b, b, c, c\}$ , and  $Z = \{b, b, c, c, c\}$ . Consider the uniform on-target set  $\mathcal{S} = \{A, B, C\}$ . Instead of inspecting all canonical reactions, we can focus on three canonical reactions

$$\alpha : A + B \rightarrow X, \quad \beta : 3B + 2C \rightarrow X + Y, \quad \gamma : A + 2B + 3C \rightarrow X + Z.$$

Here  $X$  is the non-interacting off-target polymer in  $\alpha$ , thus  $\bar{\mu}(X) = 2$ . After assigning  $\bar{\mu}(X)$ ,  $Y$  and  $Z$  become non-interacting off-target polymers in  $\beta$  and  $\gamma$ , respectively, so that we derive  $\bar{\mu}(Y) = 3$  and  $\bar{\mu}(Z) = 4$ .

## 6 Framework for Upper-Bounding Off-target Polymers

Often we are only interested in an upper bound on the concentration of an off-target undesired polymer. In this case, rather than generating its exact equilibrium concentration via Algorithm 1, we can more efficiently compute an upper bound by narrowing our focus to reactions that directly produce  $P$  instead of examining the full set of canonical reactions. This leads to a simpler surrogate quantity that approximates  $\bar{\mu}(P)$  from below.

<sup>6</sup> Here we use the linear combination of reactions in a standard way; for example, for  $\alpha : \mathcal{N}_1 \rightarrow \mathcal{N}_2$  and  $\beta : \mathcal{N}'_1 \rightarrow \mathcal{N}'_2$ , the reaction  $2 \cdot \alpha + \beta$  denotes the reaction  $2\mathcal{N}_1 + \mathcal{N}'_1 \rightarrow 2\mathcal{N}_2 + \mathcal{N}'_2$ .

<sup>7</sup> Note that there may be other reactions involving  $P$  with other off-target polymers.

► **Definition 6.1.** For  $P \notin \cup_{j=0}^{i-1} \mathcal{S}_j$ , let  $\tilde{\mu}_i(P) = \min_{\alpha} \{k_i(\alpha)/l_i(\alpha)\}$  where the minimum is taken over all canonical reactions  $\alpha$  that include  $P$  as a product.

Since  $P$  has not yet been leveled by construction, any such reaction  $\alpha$  producing  $P$ , obviously involves at least one unlevelized polymer, ensuring  $l_i(\alpha) \neq 0$ .

Intuitively,  $\tilde{\mu}_i(P)$  captures a conservative estimate of the exponent with which polymer  $P$  can grow in concentration at level  $i$ , based only on reactions that actually produce  $P$ . Since it is defined as a minimum over a subset of possible reactions, it is easier to compute than the full concentration exponent  $\bar{\mu}(P)$ , yet is still meaningful for analysis:

► **Theorem 6.2.** For any polymer  $P \notin \cup_{j=0}^{i-1} \mathcal{S}_j$ ,  $\bar{\mu}(P) \geq \tilde{\mu}_i(P) \geq \mu_i$ .

► **Lemma 6.3.** For a canonical reaction  $\alpha$  with  $l_{i+1}(\alpha) \neq 0$ , it holds that  $k_i(\alpha)/l_i(\alpha) \leq k_{i+1}(\alpha)/l_{i+1}(\alpha)$ .

**Proof.** Note that  $\mu_i \leq k_i(\alpha)/l_i(\alpha)$  by definition of  $\mu_i$ . Let  $n$  be the count of level- $i$  product polymers in  $\alpha$ , which must be smaller than  $l_{i+1}(\alpha)$ . Then we have

$$\frac{k_{i+1}(\alpha)}{l_{i+1}(\alpha)} = \frac{k_i(\alpha) - n\mu_i}{l_i(\alpha) - n} = \frac{k_i(\alpha) - n\mu_i}{l_i(\alpha) - n} \geq \frac{k_i(\alpha) - n(k_i(\alpha)/l_i(\alpha))}{l_i(\alpha) - n} = \frac{k_i(\alpha)}{l_i(\alpha)}. \quad \blacktriangleleft$$

**Proof of Theorem 6.2.** Since all polymers will eventually be leveled, there exists a  $j$ -levelizing canonical reaction  $\alpha$  that includes  $P$  as a product for some  $j \geq i$ . Then,  $\tilde{\mu}_i(P) \leq k_i(\alpha)/l_i(\alpha) \leq k_j(\alpha)/l_j(\alpha) = \bar{\mu}(P)$  where the first inequality follows from Definition 6.1, and the second follows from Lemma 6.3.

Furthermore, the value  $\tilde{\mu}_i(P)$  is computed by taking the minimum of  $k_i(\alpha)/l_i(\alpha)$  only over canonical reactions that produce  $P$  (and have  $l_i(\alpha) \neq 0$ ). In contrast,  $\mu_i$  is defined as the minimum over all canonical reactions, regardless of whether they produce  $P$  or not. In other words, the set of reactions considered when computing  $\tilde{\mu}_i(P)$  is a subset of those considered for  $\mu_i$ , making the search space for  $\tilde{\mu}_i(P)$  more restricted. Since a minimum over a smaller set cannot be smaller than the minimum over a larger set, we conclude that  $\tilde{\mu}_i(P) \geq \mu_i$ . ◀

To summarize,  $\tilde{\mu}_i(P)$  serves as a computationally efficient lower bound on  $\bar{\mu}(P)$ . When used alongside Theorem 5.4, this upper bounds the equilibrium concentration of  $P$ .

The usefulness of Theorem 6.2 extends beyond individual estimates. It contributes to structural insights about level assignments during the iterations of our algorithm. Specifically, after leveling up through  $\mathcal{S}_{i-1}$ , we know that any unlevelized polymer  $P \notin \cup_{j=0}^{i-1} \mathcal{S}_j$  satisfies  $\bar{\mu}(P) \geq \mu_i > \mu_{i-1}$ . This inequality ensures that polymers awaiting level assignment must exhibit smaller concentrations. In particular, because the first level imbalance  $k_1$  and novelty  $l_1$  are precisely identical to  $k$  and  $l$  defined in Definition 3.3, we derive  $\bar{\mu}(P) \geq \mu_i \geq \min_{\alpha} \{k(\alpha)/l(\alpha)\} > 1$  and the following corollary.

► **Corollary 6.4.**  $\bar{\mu}(P) > 1$  for all  $P \notin \mathcal{S}$ .

## 7 Concentrations of Polymers in the TBN Model

While the TBN model is combinatorial in nature, quantifying over discrete (saturated) configurations, at the end we are most often interested in determining real-valued concentrations, which are accessible to (bulk) wet-lab experiment. The framework developed in this paper helps to bridge this gap between combinatorics of discrete configurations and concentrations.<sup>8</sup>

<sup>8</sup> Of course, much of the heavy lifting in bridging this gap is done by the derivation [4] of the cost function  $g(\mathbf{x})$ , but our work expands on it beyond numerical simulation.

Consider a saturated (i.e., maximally bonded) configuration  $\mathcal{M}$  in the TBN model. Quantified over all saturated reconfigurations  $\mathcal{M}' \cong \mathcal{M}$ , the key quantity of interest in the TBN model is the “entropy” of  $\mathcal{M}'$ , defined as the number of polymers in  $\mathcal{M}'$  (i.e.,  $|\mathcal{M}'|$ ). The TBN model defines the “stable” configurations to be those that have maximum entropy among all saturated configurations.

Corresponding to the TBN literature [5], a multiset of polymers  $\mathcal{M}$  is called *TBN-stable* if any reaction  $\mathcal{M} \rightarrow \mathcal{M}'$  has  $|\mathcal{M}| \geq |\mathcal{M}'|$ . We say that a set  $\mathcal{S}$  of polymers is *TBN-stability closed* if every multiset  $\mathcal{M} \in \mathbb{N}^{\mathcal{S}}$  is TBN-stable and further any reaction  $\mathcal{M} \rightarrow \mathcal{M}'$  where  $\mathcal{M}'$  contains a polymer outside of  $\mathcal{S}$  (i.e.,  $P \in \mathcal{M}'$  for some  $P \notin \mathcal{S}$ ) satisfies  $|\mathcal{M}| > |\mathcal{M}'|$ . In other words, producing a polymer outside of a TBN-stability closed set  $\mathcal{S}$  costs entropy. The following lemma connects our notion of stability of on-target polymers (Definition 3.4) to TBN-stability.

► **Lemma 7.1.** *If  $\mathcal{S}$  is TBN-stability closed and satisfies condition (1) of Definition 3.1, then  $\mathcal{S}$  is on-target with  $\mu(P) = 1$  for all  $P \in \mathcal{S}$  (i.e., uniform) and stable.*

**Proof.** We first check  $\mathcal{S}$  is on-target with  $\mu$ . For a reaction  $\alpha : \mathcal{M}_1 \rightarrow \mathcal{M}_2$  where  $\mathcal{M}_1$  and  $\mathcal{M}_2$  are multisets over  $\mathcal{S}$ ,  $\mu(\mathcal{M}_1) = |\mathcal{M}_1| \geq |\mathcal{M}_2| = \mu(\mathcal{M}_2)$ . Applying the same argument to  $\alpha' : \mathcal{M}_2 \rightarrow \mathcal{M}_1$  yields  $\mu(\mathcal{M}_1) \leq \mu(\mathcal{M}_2)$ . Combining the two gives  $\mu(\mathcal{M}_1) = \mu(\mathcal{M}_2)$ , proving  $\mathcal{S}$  with  $\mu$  is an on-target set.

Now we will prove that uniform on-target  $\mathcal{S}$  is stable. Consider a canonical reaction  $\beta : \mathcal{M}'_1 \rightarrow \mathcal{M}'_2$  with  $l(\beta) > 0$ . By definition, we have  $k(\beta) = \mu(\mathcal{M}'_1) - \mu(\mathcal{M}'_2) = |\mathcal{M}'_1| - |\mathcal{M}'_2|$  and  $l(\beta) = |\mathcal{M}'_2| - |\mathcal{M}'_1|$ . Therefore, the condition for  $\mathcal{S}$  being stable, namely  $k(\beta)/l(\beta) > 1$ , is implied by  $|\mathcal{M}'_1| > |\mathcal{M}'_2|$ . ◀

Thus, the polymers in the TBN-stability closed set  $\mathcal{S}$  represent the intended “high-concentration” polymers, while everything outside of  $\mathcal{S}$  is considered undesired (off-target).

Let canonical reaction  $\alpha$  be  $\mathcal{M} \rightarrow \mathcal{M}'$ ; we define  $|\mathcal{M}| - |\mathcal{M}'|$  to be the *entropy loss*  $e(\alpha)$  of the reaction  $\alpha$ . Recall that during the first iteration of our algorithm, novelty  $l(\alpha)$  is the number of off-target polymers generated in canonical reaction  $\alpha$ . The imbalance  $k(\alpha)$  in the first iteration can be understood in terms of the entropy loss of the reaction: the decrease in the number of polymers of a reaction is exactly  $e(\alpha) = k(\alpha) - l(\alpha)$ . Thus to have an upper bound on the concentration of off-target polymers via Theorem 6.2, it is sufficient to find the smallest ratio  $e(\alpha)/l(\alpha)$  of any reaction:

► **Corollary 7.2.** *Let set  $\mathcal{S} \subseteq \Psi$  be a TBN-stability closed set of polymers. Let  $\mu_1 = \min_{\alpha} \{e(\alpha)/l(\alpha)\} + 1$  minimized over all reactions  $\alpha : \mathcal{M} \rightarrow \mathcal{M}'$  where  $\mathcal{M} \in \mathbb{N}^{\mathcal{S}}$ ,  $\mathcal{M}' \in \mathbb{N}^{\Psi}$  and  $l(\alpha) > 0$ . For any  $0 < c < 1$ , there are monomer concentrations  $\mathbf{x}^0 \in (0, 1)^{\Psi^0}$  and configuration  $\mathbf{x} \in (0, 1)^{\Psi}$  that minimizes  $g(\mathbf{x})$  subject to  $\mathbf{A} \cdot \mathbf{x} = \mathbf{x}^0$  where  $\mathbf{x}$  satisfies: each polymer  $P \in \mathcal{S}$  has concentration exactly  $c$ , and each polymer  $P \in \Psi \setminus \mathcal{S}$  has concentration not more than  $c^{\mu_1}$ .*

In other words, if  $\mathcal{S}$  is a TBN-stability closed set of polymers then we can assign concentration  $c$  to each of them (uniform assignment). Then we consider the “worst” way to generate any polymers outside of  $\mathcal{S}$  (i.e., off-target): the way that has the smallest entropy loss and the largest novelty. The ratio of the entropy loss to novelty gives us an upper bound on the concentration exponent  $\mu_1$  of any off-target polymer, bounding its concentration by  $c^{\mu_1}$ . The smallest entropy loss to generate off-target polymers is already the key item of interest in the TBN literature. Thus the above corollary helps to bootstrap concentration bounding arguments via existing arguments based on entropy loss.



## 8 Example Applications

In this section we show several applications of our mathematical tools in the analysis of existing systems of interest in the DNA molecular programming literature. We base our arguments on previous results proving the entropy loss (quantity  $e(\alpha)$  of Corollary 7.2) of the systems in producing off-target (leak) species.

We note that while existing literature succeeds in characterizing entropy loss via TBN-like combinatorial arguments, more work is needed to develop similarly rigorous combinatorial arguments on novelty (quantity  $l(\alpha)$ ); see also the discussion in Section 9. In the examples to follow, we claim to have identified the worst-case canonical reactions – i.e., having the least  $e(\alpha)/l(\alpha)$  or  $k(\alpha)/l(\alpha)$  ratio – without proof.

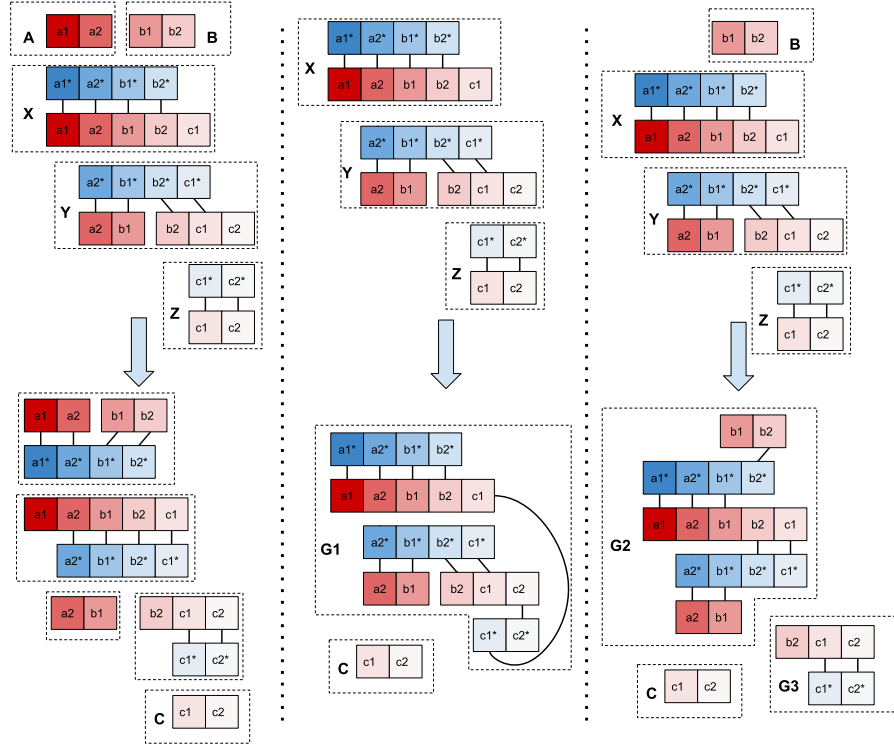
We first consider the TBN AND gate introduced in prior work [5, 2] and recently experimentally realized [14]. Figure 1 (left) shows the desired functionality of the AND gate in which inputs  $A$  and  $B$  cooperate to produce  $C$ . We are interested in bounding the concentration of  $C$  (leak) that can be produced in the absence of inputs  $A$  and  $B$ . Phrased in our terminology, combinatorial TBN arguments have shown that  $\mathcal{S} = \{X, Y, Z\}$  is TBN-stability closed, as well as  $\{X, Y, Z, A\}$  and  $\{X, Y, Z, B\}$  where one of the two inputs is present. This implies that any canonical reaction  $\alpha$  producing  $C$  has entropy loss  $e(\alpha) \geq 1$ . However, such arguments do not directly connect entropy loss to leak concentrations, justifying the need for a tool like our Corollary 7.2.

To apply our framework to the TBN AND gate, we use Corollary 7.2. In the absence of both inputs, take  $\mathcal{S} = \{X, Y, Z\}$ . We claim that the worst-case canonical reaction  $\alpha$  producing  $C$  is  $X + Y + Z \rightarrow G_1 + C$  shown in Figure 1 (middle). This reaction has entropy loss  $e(\alpha) = 1$  and novelty  $l(\alpha) = 2$  since  $G_1$  and  $C$  are outside of  $\mathcal{S}$ . Thus for any base concentration  $0 < c < 1$ , there is an equilibrium with  $[X] = [Y] = [Z] = c$  where the leak concentration is  $[C] \leq c^{1.5}$ .

Similarly, consider the case of having input  $B$  but no input  $A$ . We take  $\mathcal{S} = \{X, Y, Z, B\}$  and claim<sup>9</sup> that the worst-case canonical reaction  $\alpha$  producing  $C$  is  $X + Y + Z + B \rightarrow G_2 + G_3 + C$  as shown in Figure 1 (right). This reaction has entropy loss  $e(\alpha) = 1$  and novelty  $l(\alpha) = 3$  since  $G_2$ ,  $G_3$ , and  $C$  are outside of  $\mathcal{S}$ , yielding the equilibrium leak concentration of  $[C] \leq c^{1.33}$ . Our analysis thus provides concrete polynomial upper bounds on leak, consistent with the qualitative expectation of the TBN model that leak should become comparatively negligible in the limit of decreasing  $c$ . The bound is smaller when both inputs are absent than when  $B$  is present.

As the next example, we consider the “leakless” DNA strand displacement system previously theoretically and experimentally studied [15]. That work focuses on a family of “translator” modules that convert an input strand to an output strand of independent sequence. The family is parameterized by the redundancy parameter  $N$  defined as the number of bound domains in each fuel polymer  $F_i$ , such that the number of domains in each signal  $X_i$  is  $N + 1$ . Leak is expected to decrease with decreasing overall concentration (as for the AND gate), as well as with increasing redundancy  $N$ . The decrease of leak with increasing  $N$  was confirmed by experiment, at least for small  $N$ . The reactions of the translator with  $N = 3$  are shown in Figure 2.

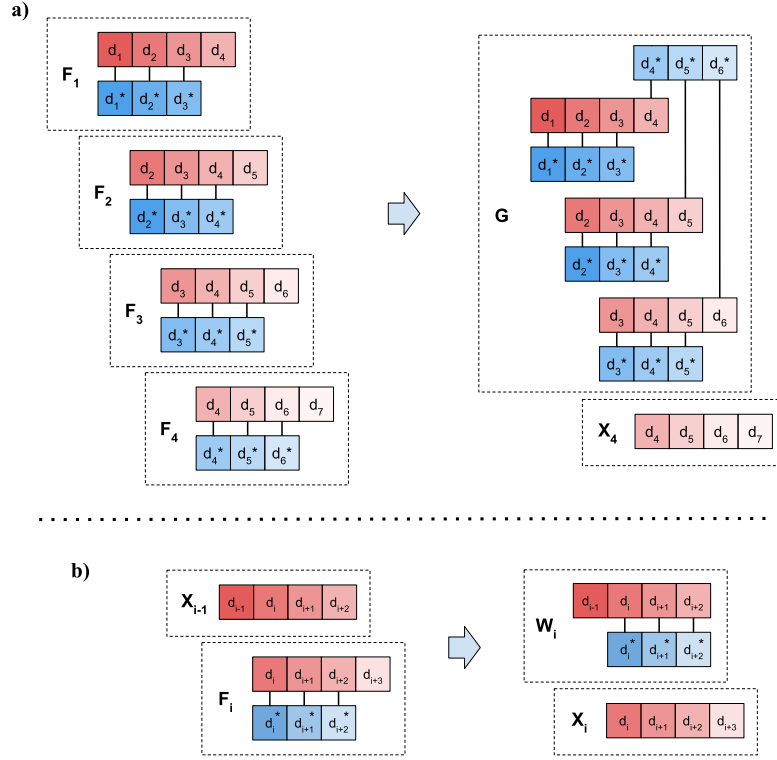
<sup>9</sup> While we do not provide a formal proof that this reaction is worst-case, our confidence is based on the observation that we cannot increase the novelty further without increasing the entropy loss in proportion. For example, we can combine two copies of reaction  $\alpha$  to yield reaction  $\alpha'$  but then  $e(\alpha') = 2e(\alpha)$  and  $l(\alpha') = 2l(\alpha)$ , maintaining the same ratio.



■ **Figure 1** A TBN module implementing an AND gate. The presence of polymers  $A$  and  $B$  represents input 1, while their absence represents input 0. (Left) The intended reaction pathway generates the output polymer  $C$  when both  $A$  and  $B$  are present. (Middle) When both  $A$  and  $B$  are absent, the worst-case canonical reaction is  $\alpha : X + Y + Z \rightarrow G_1 + C$  producing an erroneous output  $C$  (leak). The entropy loss and novelty of the reaction are  $e(\alpha) = 1$  and  $l(\alpha) = 2$ , resulting in the concentration upper bound of  $C$  of  $c^{1.5}$  via Corollary 7.2. (Right) If only one input is present ( $B$ ), then the worst-case canonical reaction is  $\beta : B + X + Y + Z \rightarrow G_2 + G_3 + C$  with  $e(\beta) = 1$  and  $l(\beta) = 3$ , yielding the upper bound concentration of  $C$  of  $c^{1.33}$ .

To apply our framework, we consider the system without toeholds, driven solely by entropy; with long domains alone, we are in the enthalpy neutral (athermic) regime. Prior work focused on the case where the system was prepared in a state with only fuel polymers ( $F_i$ ), all at equal concentration, and zero initial concentration of waste polymers ( $W_i$ ). Let on-target set be  $\mathcal{S} = \{F_i, W_i \mid i = 1, \dots, n\}$  with uniform  $\mu = 1$ . Rephrasing in our terminology, ref. [13] proved that any canonical reaction  $\alpha : \mathcal{M} \rightarrow \mathcal{M}'$  where  $X_i \in \mathcal{M}'$  has entropy loss  $e(\alpha) = N - 1$ , and this fact was used to argue that by increasing  $N$  we can make leak arbitrarily small. We now show that considering novelty in addition to entropy loss makes this argument problematic, and suggest an alternative parameter setting to arbitrarily decrease leak.

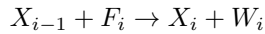
Consider a cascade of two translators. Importantly, the number of fuels for a single translator module is  $N + 1$ ; i.e.,  $X_i$  and  $X_j$  overlap in sequence for  $i < j < i + N + 1$  but are completely sequence-independent for  $j = i + N + 1$ . We claim that the worst-case leak pathway  $\alpha$  is where the first translator leaks resulting in the triggering of the second translator. This pathway generates  $l(\alpha) = N + 3$  new off-target polymers: 1 for the leaked upstream translator (all fuels coming together),  $N + 1$  for the triggered fuels of the second translator, and 1 for the output. Thus  $e(\alpha)/l(\alpha) = (N - 1)/(N + 3)$ . By Corollary 7.2, the



■ **Figure 2** Translator cascade with redundancy parameter  $N = 3$ . Polymer  $X_0$  serves as input and polymer  $X_n$  as output signal. The leak pathway with  $N + 1$  fuels coming together to generate a signal  $X_{N+1}$  in the absence of input is described in (a). Part (b) describes the intended reaction for each  $i$ ; iterating them for  $i = 1, \dots, N + 1$  produces  $X_{N+1}$  given input  $X_0$ . Note that if several translators are composed, then  $X_{N+1}$  is the input to the downstream translator and once the leak signal is generated via the pathway shown in (a) it can propagate via intended reactions to the output  $X_n$  of the last translator.

concentration of the leak product in the uniform setting is bounded above by  $c^{4/3}$  for  $N = 3$ . The bound tightens to  $c^2$  with increasing  $N$ ; however, it does not get arbitrarily small. This suggests that we do not decrease equilibrium leak concentration arbitrarily by increasing “redundancy”  $N$  because while the entropy loss increases, the novelty increases as well.

To arbitrarily decrease leak, we propose to use positive initial concentrations of the waste polymers  $W_i$ . The fuel polymers are denoted  $F_1, \dots, F_n$ , and the waste polymers produced after translator triggering are  $W_1, \dots, W_n$ . Let  $X_0$  represent the input and  $X_n$  represent the output. We have  $n = 2(N + 1)$  for two translators composed together. This cascade of length  $n$  proceeds through the following reactions described in Figure 2:



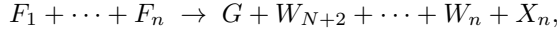
for  $i = 1, \dots, n$ .

We first consider the triggered system (with-input) showing at least a constant fraction of the signal is propagated to the end as  $X_n$ . We then focus on the case without input and bound the leak. Note that this analysis utilizes Theorem 5.4 directly rather than Corollary 7.2 because we will have different concentration exponents  $\mu$  for  $F_i$  and  $W_i$  in  $\mathcal{S}$ .

For the triggered (with-input) system we define our on-target set as  $\mathcal{S} = \{F_i, W_i, X_0, X_i \mid i = 1, \dots, n\}$ . We assign concentrations to the on-target polymers as follows: All fuel concentrations are equal to  $2c$ , and all waste concentrations are equal to  $c$ . These concentrations correspond to concentration exponents  $\mu(F_i) = 1 + \log_c 2$  and  $\mu(W_i) = 1$ . Let the concentration of the output in the final layer  $X_n$  be  $y$  and assign balancing concentrations of the other  $X_i$ . Since  $[F_i]/[W_i] = 2$ , we have  $\sum_{i=0}^n [X_i] < 2y$ , meaning that more than half of the total signal ( $X_i$ ) is at the output layer.

Now, we investigate the system without input. Let  $\mathcal{S} = \{F_i, W_i \mid i = 1, \dots, n\}$ . For the situation to properly correspond to the with-input case, we need to ensure that all monomer concentrations are the same between the two cases, except removing the monomer corresponding to input  $X_0$ . Rather than thinking about specific monomers, we start in the with-input case and conceptually run reactions  $X_i + W_i \rightarrow X_{i-1} + F_i$  to completely push all  $X_i$  to  $X_0$ , and then remove  $X_0$  from the system.<sup>10</sup> Since the total amount of all  $X_i$  is less than  $2y$  in the with-input case, this results in:  $[F_i] < 2c + 2y$  and  $[W_i] > c - 2y$ . These correspond to  $\mu(F_i) > \log_c(2c + 2y)$  and  $\mu(W_i) < \log_c(c - 2y)$ .

Recall that redundancy  $N$  results in entropy penalty  $N - 1$ . We claim that the reaction with the smallest imbalance-novelty ratio (i.e., worst-case) is reaction  $\beta$ :



where  $G$  is the “large polymer” formed after all  $F_1, \dots, F_N$  displace the top strand from  $F_{N+1}$ , and  $X_n$  is the leak output. The imbalance of this reaction is:

$$k(\beta) = \sum_{i=1}^n \mu(F_i) - \sum_{i=N+2}^n \mu(W_i) > n \log_c(2c + 2y) - \frac{n}{2} \log_c(c - 2y). \quad (4)$$

We can ensure that  $k(\beta)$  is at least a constant fraction of  $n$  for small enough  $c$ . For example, if we let  $y \leq c/4$ , then  $k(\beta) \geq n/4$  for any  $c \leq 0.0064$ . The novelty is independent of  $n$ :  $l(\beta) = 2$  since  $G$  and  $X_n$  are not in  $\mathcal{S}$ . Therefore, the imbalance-novelty ratio  $k(\beta)/l(\beta)$  of the worst case reaction is at least  $n/8$ , which increases linearly with  $N$  (recall  $n = 2(N + 1)$  for two translators composed together). Applying Theorem 5.4 leads to leak concentration of at most  $c^{n/8} = c^{1/4 + N/4}$ . This upper bound<sup>11</sup> implies smaller-than  $c$  concentration of leak for  $N \geq 4$ , with the leak exponentially decreasing for larger  $N$ .

To summarize, by increasing redundancy  $N$  in the appropriate regime, we maintain the property that a constant fraction of the input is converted to output in the with-input case, while arbitrarily (exponentially in  $N$ ) decreasing leak in the without-input case.

## 9 Discussion

Our results suggest a few important directions for future work. Given the central role of worst-case canonical reactions – i.e., canonical reactions with the lowest imbalance-novelty ratio (for Algorithm 1 and Theorem 5.4) or entropy loss-novelty ratio (Corollary 7.2) – it is important to develop formal techniques to prove that a given canonical reaction is indeed worst-case overall, at least or for a particular off-target polymer (for Section 6). Note that while combinatorial techniques in prior work in the TBN model have focused on proving

<sup>10</sup> More precisely, this ensures that the concentrations of monomers making up on-target polymers are the same between the with-input and without-input case (other than  $X_0$ ).

<sup>11</sup> Note that this upper bound is loose because of inequalities such as Equation (4).

entropy loss, more work is needed to study the ratio directly, making our framework more easily applicable. While we believe the canonical reactions highlighted in Section 8 are indeed worst-case, the argument is informal.

Another promising avenue of research is to establish a more direct link between a polymer's monomer composition and its equilibrium concentration. Our current framework is effectively reaction-centric, inferring concentrations based on how polymers transform into one another. An alternative approach could be to derive concentration bounds directly from the structural properties of the off-target polymers, such as their size (monomer count) or degree of overlap with one another (multiset difference). Nonetheless, we hope that a variety of structure-based results could be proven based on a reduction to our canonical reaction framework.

Finally, this work has focused exclusively on the athermic case, where all molecular interactions are enthalpically neutral. While this is a reasonable and useful abstraction for systems with strong, saturated bonds, many real-world molecular systems, including many popular in DNA molecular programming, involve a range of binding strengths and enthalpic effects (e.g., from toehold binding). Extending our algorithmic framework to incorporate user-specified  $\Delta G$ 's for each polymer could significantly broaden its applicability, and although this would complicate our algorithm, we do not anticipate any insurmountable difficulties.

## References

- 1 Keenan Breik, Cameron Chalk, David Doty, David Haley, and David Soloveichik. Programming substrate-independent kinetic barriers with thermodynamic binding networks. *IEEE/ACM Trans. Comput. Biol. Bioinformatics*, 18(1):283–295, 2021. doi:10.1109/TCBB.2019.2959310.
- 2 Keenan Breik, Chris Thachuk, Marijn Heule, and David Soloveichik. Computing properties of stable configurations of thermodynamic binding networks. *Theor. Comput. Sci.*, 785:17–29, 2019. doi:10.1016/j.tcs.2018.10.027.
- 3 W. Bruns, B. Ichim, C. Söger, and U. von der Ohe. Normaliz. Algorithms for rational cones and affine monoids. Available at <https://www.normaliz.uni-osnabrueck.de>.
- 4 Robert M Dirks, Justin S Bois, Joseph M Schaeffer, Erik Winfree, and Niles A Pierce. Thermodynamic analysis of interacting nucleic acid strands. *SIAM review*, 49(1):65–88, 2007. doi:10.1137/060651100.
- 5 David Doty, Trent A. Rogers, David Soloveichik, Chris Thachuk, and Damien Woods. Thermodynamic binding networks. In *DNA Computing and Molecular Programming, 23rd International Conference, DNA 23*, pages 249–266, 2017. doi:10.1007/978-3-319-66799-7\_16.
- 6 Martin Feinberg. *Foundations of chemical reaction network theory*. Springer, 2019.
- 7 Stefan Hoops, Sven Sahle, Ralph Gauges, Christine Lee, Jürgen Pahle, Natalia Simus, Mudita Singhal, Liang Xu, Pedro Mendes, and Ursula Kummer. COPASI—a complex pathway simulator. *Bioinformatics*, 22(24):3067–3074, 2006. doi:10.1093/BIOINFORMATICS/BTL485.
- 8 Fritz Horn and Roy Jackson. General mass action kinetics. *Archive for rational mechanics and analysis*, 47:81–116, 1972.
- 9 Matthew D Johnston, Stefan Müller, and Casian Pantea. A deficiency-based approach to parametrizing positive equilibria of biochemical reaction systems. *Bulletin of Mathematical Biology*, 81:1143–1172, 2019.
- 10 A. M. M. Leal. Reaktoro: An open-source unified framework for modeling chemically reactive systems, 2015. URL: <https://reaktoro.org>.
- 11 John AP Sekar, Justin S Hogg, and James R Faeder. Energy-based modeling in BioNetGen. In *2016 IEEE international conference on bioinformatics and biomedicine (BIBM)*, pages 1460–1467. IEEE, 2016.
- 12 Elie Soloveichik, Leo Orshansky, Cameron Chalk, and Boya Wang. Concentrat.io, 2025. Accessed 25 June 2025. URL: <https://concentrat.io/>.

- 13 Chris Thachuk, Erik Winfree, and David Soloveichik. Leakless DNA strand displacement systems. In Andrew Phillips and Peng Yin, editors, *DNA Computing and Molecular Programming*, pages 133–153, Cham, 2015. Springer International Publishing. doi:10.1007/978-3-319-21999-8\_9.
- 14 Boya Wang, Cameron Chalk, David Doty, and David Soloveichik. Molecular computation at equilibrium via programmable entropy. *bioRxiv*, 2025. doi:10.1101/2024.09.13.612990.
- 15 Boya Wang, Chris Thachuk, Andrew D. Ellington, Erik Winfree, and David Soloveichik. Effective design principles for leakless strand displacement systems. *Proc. Natl. Acad. Sci. USA*, 115(52):E12182–E12191, 2018. doi:10.1073/pnas.1806859115.
- 16 Boya Wang, Chris Thachuk, and David Soloveichik. Speed and correctness guarantees for programmable enthalpy-neutral DNA reactions. *ACS Synthetic Biology*, 12(4):993–1006, 2023.
- 17 Joseph N Zadeh, Conrad D Steenberg, Justin S Bois, Brian R Wolfe, Marshall B Pierce, Asif R Khan, Robert M Dirks, and Niles A Pierce. NUPACK: analysis and design of nucleic acid systems. *Journal of computational chemistry*, 32(1):170–173, 2011. doi:10.1002/JCC.21596.

## A

 Hilbert Basis Implementation

This section recalls some results on Hilbert bases and explains how to use the Hilbert basis in Algorithm 1. While this section makes the main theorem mathematically rigorous, most readers can safely skip this section.

The main contents of this section are (1) the termination of Algorithm 1 and (2) the use of minimum (versus infimum) in Definition 5.2. We address both concerns by showing that, although there are infinitely many canonical reactions, we can always restrict our attention to a finite subset of them in our analysis and algorithm.

For integral vectors  $\mathbf{v}_1, \dots, \mathbf{v}_m \in \mathbb{Z}^n$ , the set  $C = \{\sum_{i=1}^m b_i \mathbf{v}_i : b_1, \dots, b_m \geq 0\}$  is called a (rational polyhedral) cone, which is also known to be described by a system of inequalities  $C = \{\mathbf{v} : \mathbf{B} \cdot \mathbf{v} \leq 0\}$  for some matrix  $\mathbf{B} \in \mathbb{Z}^{l \times n}$ . It is known that the set  $C \cap \mathbb{Z}^n$  has a *finite* subset  $\mathcal{H}(C) = \{\mathbf{h}_1, \dots, \mathbf{h}_t\}$ , called *Hilbert basis* of  $C$ , that generates  $C \cap \mathbb{Z}^n$  with non-negative integer coefficients, that is, for any  $\mathbf{v} \in C \cap \mathbb{Z}^n$  there are  $a_1, \dots, a_t \in \mathbb{N}$  such that  $\mathbf{v} = \sum_{i=1}^t a_i \mathbf{h}_i$ .

The set of canonical reactions  $\Lambda$  can be precisely described in terms of a rational polyhedral cone. For a canonical reaction  $\alpha : \mathcal{M}_1 \rightarrow \mathcal{M}_2$ , we define a vector

$$\mathbf{v}_\alpha = (\mathcal{M}_1[P] - \mathcal{M}_2[P])_{P \in \Psi} \in \mathbb{Z}^{|\Psi|}$$

capturing the stoichiometric change of concentrations due to reaction  $\alpha$ . Note that  $\mathcal{M}_1[P'] - \mathcal{M}_2[P'] = -\mathcal{M}_2[P'] \leq 0$  for  $P' \notin \mathcal{S}$ , thus  $\mathbf{v}_\alpha \cdot \mathbf{e}_{P'} \leq 0$  for  $\mathbf{e}_{P'} = (\delta_{PP'})_{P \in \Psi}$  for the delta function  $\delta_{ij} = 1$  iff  $i = j$ . Definition 2.2 ensures that this vector must satisfy the condition  $\mathbf{A} \cdot \mathbf{v}_\alpha = 0$ . Combining these, the cone

$$C^\mathcal{S} = \{\mathbf{v}_\alpha \in \mathbb{R}^{|\Psi|} : \mathbf{A} \cdot \mathbf{v}_\alpha \geq 0 \text{ and } \mathbf{A} \cdot \mathbf{v}_\alpha \leq 0 \text{ and } \mathbf{v}_\alpha \cdot \mathbf{e}_P \leq 0 \text{ for all } P \notin \mathcal{S}\}$$

characterizes the canonical relations:  $C^\mathcal{S} \cap \mathbb{Z}^{|\Psi|}$  is the set of vectors  $\mathbf{v}_\alpha$ . Therefore, there exists a finite set of canonical reactions  $H$  that corresponds to the Hilbert basis  $\mathcal{H}(C^\mathcal{S})$ .

The following lemma implies that for the purposes of our algorithm and analysis, we can focus on the Hilbert basis  $H$  of canonical reactions.

► **Lemma A.1.** *Let  $\alpha : \mathcal{M}_1 \rightarrow \mathcal{M}_2$  and  $\beta : \mathcal{N}_1 \rightarrow \mathcal{N}_2$  be canonical reactions with  $l_i(\alpha), l_i(\beta) \neq 0$ . Then, for any  $a, b \in \mathbb{N}$ , it holds that*

$$\frac{k_i(a \cdot \alpha + b \cdot \beta)}{l_i(a \cdot \alpha + b \cdot \beta)} \geq \min \left( \frac{k_i(\alpha)}{l_i(\alpha)}, \frac{k_i(\beta)}{l_i(\beta)} \right). \quad (5)$$

*The equality holds only when  $a = 0$ ,  $b = 0$ , or  $k_i(\alpha)/l_i(\alpha) = k_i(\beta)/l_i(\beta)$ .*

**Proof.** It is not hard to see that  $k_i$  and  $l_i$  are linear, i.e.,  $k_i(a \cdot \alpha + b \cdot \beta) = a \cdot k_i(\alpha) + b \cdot k_i(\beta)$  and  $l_i(a \cdot \alpha + b \cdot \beta) = a \cdot l_i(\alpha) + b \cdot l_i(\beta)$ . Then, Equation (5) is identical to the mediant inequality, which states that for  $p, q, r, s \geq 0$  with  $q, s \neq 0$  it holds that  $\min(p/q, r/s) \leq (p+r)/(q+s)$ . ◀

Consider a canonical reaction  $\alpha = a_1 \cdot \eta_1 + \dots + a_t \cdot \eta_t$  for  $a_1, \dots, a_t > 0$  and  $\eta_1, \dots, \eta_t \in H$ . If  $\alpha$  is  $i$ -levelizing, then the equality  $k_i(\alpha)/l_i(\alpha) = \mu_i = \min_{j=1, \dots, t} (k_i(\eta_j)/l_i(\eta_j))$  must hold, and the equality condition above ensures that  $\mu_i = k_i(\eta_j)/l_i(\eta_j)$  for all  $j = 1, \dots, t$ . In other words, the set of  $i$ -levelizing reactions is

$$\left\{ \sum_{j=1}^t a_j \cdot \eta_j : a_1, \dots, a_t \in \mathbb{N} \right\} \text{ where } \{\eta_1, \dots, \eta_t\} \text{ is the set of } i\text{-levelizing reactions in } H.$$

This allows us to inspect the minimum over the finite set  $H$  instead of  $\Lambda$  in Definition 5.2. Similarly, as the Hilbert basis can be computed in finite time and implemented in [3], we can run Algorithm 1 with guaranteed termination by computing the minimum over  $H$ .

## B Detailed Balance and Equilibrium

Recall that  $\mathbf{A} \in \mathbb{N}^{|\Psi^0| \times |\Psi|}$  is the matrix such that each entry  $A_{ij}$  specifies the number of monomers of type  $i$  in polymer  $j$ , and that  $\mathbf{A} \cdot \mathbf{x} = \mathbf{x}^0$  (Equation (1)) captures the mass-conservation constraint.

► **Theorem B.1.** *Let  $\mathbf{x}^0 \in (0, 1)^{\Psi^0}$  be a fixed vector of monomer concentrations. If all reactions are balanced at the configuration  $\mathbf{x} \in (0, 1)^{\Psi}$  of polymer concentrations, then the cost function  $g(\mathbf{x})$  is minimum subject to  $\mathbf{A} \cdot \mathbf{x} = \mathbf{x}^0$ .*

**Proof.** (Sketch) The function  $g$  is strictly convex since its Hessian  $H$  is positive definite (specifically diagonal with  $H_{jj} = 1/x_j > 0$ ). Strict convexity of  $g$  implies that the local minimum of  $g$  becomes the unique (global) minimum.

We associate a vector  $\mathbf{v}_\alpha \in \mathbb{Z}^{|\Psi|}$  with every reaction  $\alpha$ , capturing the net stoichiometric effect of reaction  $\alpha$ .<sup>12</sup> It is straightforward to show that the function  $g$  along with the direction of  $\mathbf{v}_\alpha$  has zero derivative at  $\mathbf{x}$  if and only if the reaction  $\alpha$  is balanced at  $\mathbf{x}$ . More explicitly, for  $\alpha : \mathcal{M}_1 \rightarrow \mathcal{M}_2$ , the  $P$ -th entry of the vector  $\mathbf{v}_\alpha$  is  $(\mathcal{M}_1[P] - \mathcal{M}_2[P])$ . The directional derivative  $D_{\mathbf{v}_\alpha} g(\mathbf{x}) = \sum_P (\mathcal{M}_1[P] - \mathcal{M}_2[P]) \log x_P = 0$  implies that  $\prod_{P \in \mathcal{M}_1} x_P^{\mathcal{M}_1[P]} = \prod_{P \in \mathcal{M}_2} x_P^{\mathcal{M}_2[P]}$  holds, which means the reaction  $\alpha$  is balanced.

The set  $\{\mathbf{x} : \mathbf{A} \cdot \mathbf{x} = \mathbf{x}^0\}$  is spanned by the vectors  $\mathbf{v}_\alpha$  for all reactions by Definition 2.2. Therefore, if all reactions are balanced at  $\mathbf{x}$ , then any directional derivative at  $\mathbf{x}$  vanishes and  $\mathbf{x}$  is a critical point, which is the unique minimum. ◀

<sup>12</sup>This is the same vector  $\mathbf{v}_\alpha$  as defined in Section A. For example,  $(1, -1, 0, \dots)$  corresponds to  $X_1 \rightarrow X_2$  for  $\Psi = \{X_1, X_2, \dots\}$ .






# Leakless Polymerase-Dependent Strand Displacement Systems

Zoë Evelyn Mōhalakealoha Derauf  

Paul G. Allen School of Computer Science & Engineering,  
University of Washington, Seattle, WA, USA

Chris Thachuk<sup>1</sup>  

Paul G. Allen School of Computer Science & Engineering,  
University of Washington, Seattle, WA, USA

---

## Abstract

A grand challenge facing molecular programmers is the rational development of fast, robust, and isothermal architectures akin to “chemical central processing units” that can sense (bio-)chemical signals from their environment, perform complex computation, and orchestrate a physical response *in situ*. DNA strand displacement systems (DSDs) remain a compelling candidate, but are hampered by spurious reaction pathways that lead to incorrect output. DSDs that utilize the systematic *leakless* motif can be made arbitrarily robust at the cost of increasing redundancy and network size (scaling), and thus a degradation of kinetic performance. Another class of architectures utilize DNA hybridization, extension, and signal production of entirely sequestered outputs via strand-displacing polymerases (SDPs) that have resulted in impressive demonstrations; however, they face similar challenges of aberrant behavior such as mis-priming by incorrect signals. Our work introduces a unified polymerase-dependent toehold-mediated strand displacement (PD-TMSD) architecture that integrates the programmed specificity of DSDs with the unique advantages of SDPs. This unification enables systems that can be made arbitrarily robust, at any concentration range, without increasing network size. We propose a number of gate designs and composition rules to compute arbitrary Boolean functions, emulate arbitrary chemical reaction networks, and explore time-bounded probabilistic computation made possible by certain classes of SDPs. Our theoretical exploration is backed by preliminary experimental demonstrations. This contribution was inspired by the belief that molecular programming can meet or exceed the complexity exhibited in biology if we embrace its best understood molecular machinery and couple it with systematic design principles built upon a strong theoretical foundation.

**2012 ACM Subject Classification** Applied computing → Chemistry; Computer systems organization → Molecular computing

**Keywords and phrases** DNA strand displacement, strand-displacing polymerases, molecular computation, energy barriers, kinetics

**Digital Object Identifier** 10.4230/LIPIcs.DNA.31.11

**Funding** Supported by a Faculty Early Career Development Award from NSF (CCF 2143227).

**Acknowledgements** We thank Enos Kline for helpful discussions about properties and behaviors of different strand-displacing polymerases.

## 1 Introduction

DNA strand displacement (DSD) has emerged as a powerful and versatile mechanism for programming molecular interactions and constructing dynamic molecular circuits and nanoscale devices. In these systems, information is carried by DNA strands that interact through predictable and programmable Watson-Crick base pairing, enabling cascades of

---

<sup>1</sup> Corresponding author



© Zoë Evelyn Mōhalakealoha Derauf and Chris Thachuk;  
licensed under Creative Commons License CC-BY 4.0

31st International Conference on DNA Computing and Molecular Programming (DNA 31).

Editors: Josie Schaeffer and Fei Zhang; Article No. 11; pp. 11:1–11:19

Leibniz International Proceedings in Informatics



LIPICs Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

strand displacement reactions that form the basis of molecular computation [26]. The appeal of DNA strand displacement lies in its universality and scalability: any arbitrary reaction network can, in principle, be emulated by a suitable collection of strand displacement reactions, enabling the construction of dynamic reaction networks, logic circuits, oscillators, amplifiers, and neural networks [4, 20, 19, 23, 36, 30]. Despite the success of these systems, a key practical challenge has become evident. As these systems grow in complexity so-called *leak* reactions, in which outputs are spuriously produced in the absence of correct input, remain a persistent limitation. These typically result from unintended partial strand interactions or branch migration events, and can be exacerbated by poor DNA quality and fraying at the ends of DNA helices. Such leakage accumulates and severely degrades the reliability of large-scale DNA circuits. This not only limits circuit fidelity and scalability, but also poses major issues for applications like biosensing and diagnostics, where accurate circuit output remains paramount. The elimination or mitigation of leak has thus been recognized as a crucial component in scaling up DNA-based computation.

Strategies to mitigate leak include thresholding [23, 19], shadow cancellation circuits [28], and the systematic use of clamps at duplex ends [34]. The introduction of the “leakless” architecture marked a major step forward, offering the potential for arbitrarily low leak through the introduction of redundant domains [32, 35]. However, as the guarantee to leak suppression increases, so too does the required redundancy. This results in systems that require deeper reaction pathways and additional intermediates to propagate a signal. Moreover, without the use of systematic mismatches [11], sequence overlap between species introduces unproductive reactions via partial hybridization, leading to occlusion of necessary fuel complexes and propagating signals in the intended reaction pathway. This added complexity can complicate sequence design (to avoid crosstalk between partially overlapping domains) and slows productive reaction kinetics.

In parallel with the developments in enzyme-free DNA circuits, researchers have explored enzyme-assisted DNA computing architectures. Enzyme-assisted schemes bring catalytic speed-ups and active turnover of components, offering the potential for fast, biochemically efficient circuit operation. The PEN DNA toolbox, for example, combines a polymerase, an exonuclease, and a nicking enzyme to produce programmable chemical reaction networks (CRNs) [17, 1]. This strategy enabled the first practical realization of enzyme-assisted DNA dynamical circuits including synthetic DNA oscillators [17, 7] and bistable switchable memory circuits [16], demonstrating that DNA circuits could exhibit autonomous dynamic behaviors like those in biological networks. The versatility of the PEN toolbox has since been demonstrated in a wide range of applications, including isothermal digital detection of microRNAs [8], and spatial pattern formation in agent-based systems [9], establishing it as a robust and modular platform for dynamic molecular programming. More recently, synthetic molecular switches regulated by nicking endonucleases and polymerases have been built that enable enzymatically controlled ON/OFF switching of sticky end cohesion for structural reconfiguration of DNA-based materials [10]. However, reliance on multiple enzymatic components introduces complexity in design due to sequence constraints from enzyme recognition sites and reaction tuning, in addition to increased experimental complexity and potential for unintended reactions such as off-target nicking, incomplete degradation of DNA intermediates, and unintended cross-reactivity among circuit elements.

Simpler, enzyme-assisted architectures that rely on a single enzyme – most notably, strand-displacing DNA polymerases – to drive DNA circuits have also been explored. By harnessing the extension and strand-displacement activity of enzymes such as *Bst* or *Bsu* DNA polymerase, it is possible to design logic gates in which an input primer strand hybridizes to a gate and triggers polymerase-mediated extension, displacing a pre-annealed or previously

synthesized output strand. This displaced strand can then interact with downstream gates or reporter complexes. Notable implementations include polymerase-driven logic gates and catalytic amplifiers [27, 25, 31, 24]. Polymerase-mediated systems and primer exchange reactions have also been leveraged for programmable autonomous synthesis of single-stranded DNA and the construction of molecular event recorders that track the order in which distinct RNA inputs are detected [12]. Despite these advantages, such systems often suffer from reduced sequence discrimination and increased leak due to mispriming and nonspecific polymerase extension. Because polymerization can initiate from any hybridized strand with a free 3'-hydroxyl group, even partial hybridization or input mispairing – where an input is only partially complementary to an exposed domain – can trigger extension and result in spurious output. Furthermore, some architectures employing this motif are vulnerable to unproductive reaction pathways where the order in which inputs bind and are extended is critical to correct function – if correct inputs bind in an equally likely yet unintended order then the gate can deactivate thus preventing proper signal propagation.

This work introduces the first architecture that couples the robustness of leakless toehold-mediated DNA strand displacement (TMSD) with the enzymatic advantages afforded by polymerase-mediated strand displacement (PSD). Each gate complex integrates the specificity of TMSD – to discriminate, thermodynamically and kinetically, between correct and incorrect input – with the speed and strand displacing power of PSD – to produce sequence orthogonal output of arbitrary length, gated by a verified TMSD event. Leveraging PSD allows for the complete burial, in double-stranded form, of sequence unique to sequestered output. In contrast, every gate complex in an enzyme-free strand displacement system can bury at most a “toehold” length of each output sequence, relative to the displacing input(s).

By addressing leak both at the level of domain architecture and through enzymatic control, we circumvent common pitfalls leading to spurious reactions such as polymerase mispriming, branch migration-induced errors, and reaction breathing at elevated temperatures. In doing so, we also overcome key kinetic limitations of previous leakless systems caused by their required sequence redundancy – distributed across overlapping domains in multiple layers of related complexes – thereby eliminating an entire class of unproductive side-reactions and obviating the need to scale circuit size. Our system thus preserves the high specificity and robustness properties of prior (enzyme-free) leakless designs while leveraging the kinetic and scaling efficiencies made possible by strand displacing polymerases.

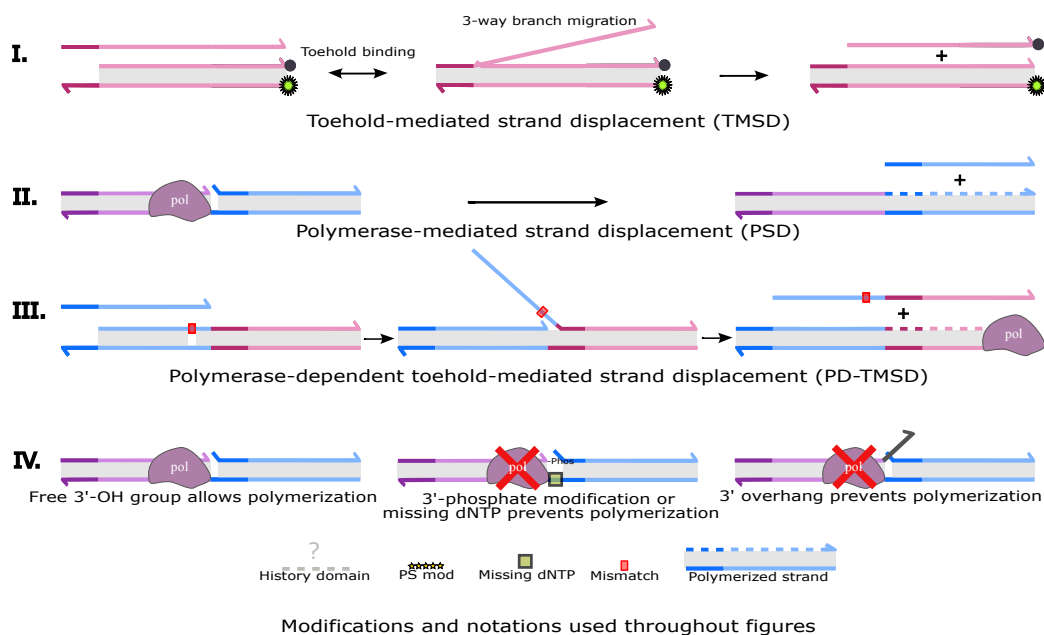
This contribution, arriving a decade after the original leakless architecture was proposed, extends the core design principles into an enzymatic domain, enabling a new class of leakless and modular molecular systems [32]. We believe this represents a significant advance in the design of molecularly programmable systems, enabling more scalable and reliable circuit construction and paving the way for larger and more efficient DNA-based CRNs.

## Summary of contributions

This paper introduces a hybrid architecture, **polymerase-dependent toehold-mediated strand displacement (PD-TMSD)**, that integrates the precise, programmable control of DNA strand displacement systems (DSDs) with the kinetic advantages of strand-displacing polymerases (SDPs). The key contributions of this work are:

- We define a new class of molecular gates that leverage both toehold-mediated strand displacement and polymerase-mediated extension to achieve robust, leak-resistant signal propagation at arbitrary concentrations, without increasing network size.
- We formalize a set of design rules and polymerase extension constraints – including blocking via 3'-modifications, mismatch inhibition, and dNTP omission – to systematically control enzymatic activity within molecular circuits.

## 11:4 Leakless Polymerase-Dependent Strand Displacement Systems



**Figure 1** Overview of mechanisms and notations used throughout this work.

- We propose scalable composition strategies for PD-TMSD gates to compute arbitrary Boolean functions, emulate chemical reaction networks (CRNs), and implement time-bounded probabilistic computation.
- We provide a visual and theoretical framework for analyzing PD-TMSD systems, accompanied by preliminary experimental results (see Appendix).

Taken together, these contributions bridge the gap between two powerful yet previously disjoint paradigms – DSD and SDP – and lay the groundwork for molecular architectures that combine theoretical rigor with practical implementation.

## 2 Preliminaries

Figure 1 provides an overview of the key methodologies and modifications used throughout this work, with panels (I)–(IV) illustrating the core mechanisms and design principles used in our system.

**I. Toehold-Mediated Strand Displacement (TMSD).** Toehold-mediated strand displacement (TMSD) is a widely used mechanism in dynamic DNA nanotechnology. In TMSD, an incoming strand (the *invader*) binds to an exposed single-stranded region (the *toehold*). This interaction initiates branch migration, allowing the invader to displace an *incumbent* strand that is initially hybridized to the complementary domain.

**II. Polymerase Strand Displacement (PSD).** Polymerase-mediated strand displacement involves a strand-displacing DNA polymerase that extends a DNA primer by synthesizing a complementary strand along a template. During this process, the polymerase displaces downstream DNA strands hybridized to the same template, effectively replacing them with the newly synthesized strand.

**III. Polymerase-Dependent Toehold-Mediated Strand Displacement (PD-TMSD).** PD-TMSD is introduced in this work and combines the specificity of TMSD with the catalytic extension activity of a strand-displacing DNA polymerase. In this hybrid mechanism, toehold binding initiates a partial displacement into a complex and polymerase extension from a free 3'-OH group drives completion.

**IV. Polymerase Extension Rules and Diagram Notation.** Polymerase extension proceeds only from strands that terminate in a free 3'-OH group. Polymerization is blocked in the following cases:

- if the 3' end is modified with a phosphate group
- if a missing dNTP is required to complement a position of the template downstream of the primer (denoted by a light green box)
- if the 3' end is a mismatched overhang with the template

The following notational conventions are used throughout:

- **Dotted light gray lines with question marks** mark *history domains*.
- **Solid black regions** denote *clamps*. Each clamp domain is unique to its respective gate.
- Each signal is color-coded, with the *toehold domain* shown in a darker shade and the *long domain* in a lighter shade of the same color.
- **Yellow stars** denote phosphorothioate-modified backbone linkages, which resist exonuclease degradation (see Section 6.2).
- **Red boxes** indicate *mismatches*, which are used to help inputs “lock in” to their target complex and slow the rate that the incumbent can displace them back off, allowing time for the polymerase to find the free 3' end.
- **Colored dotted lines** represent strands synthesized via polymerization.

These definitions and visual conventions serve as the foundation for interpreting the molecular designs and experimental results presented throughout this paper.

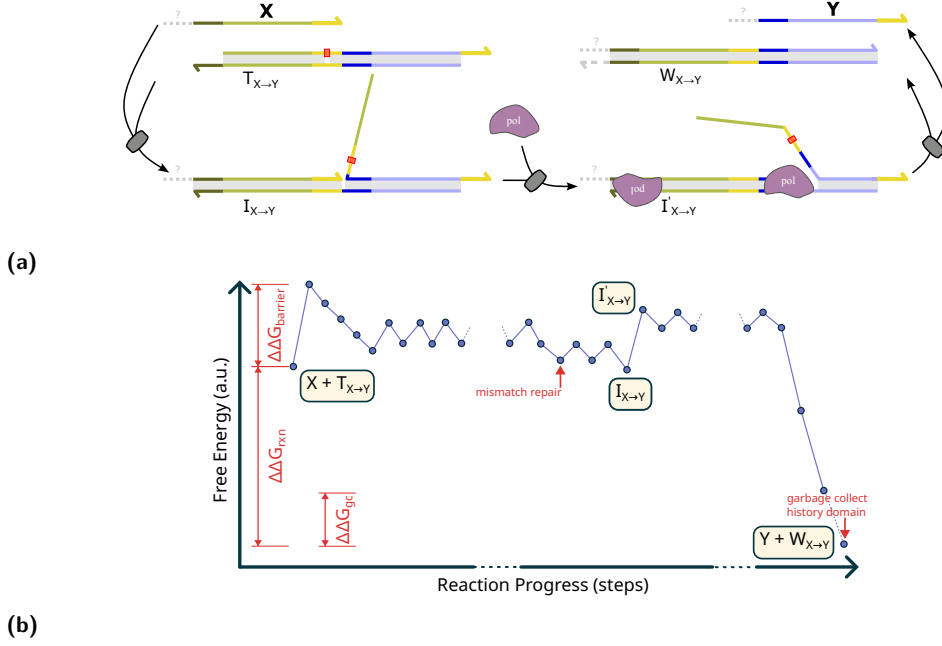
### 3 A simple motif for polymerase-dependent strand displacement

The main contribution of this work is the synthesis of two successful molecular computing architectures via the simple motif depicted as a translator gate, from input X to output Y, in Figure 2. The motif is composed of two logical steps:

#### (i) Input discrimination via TMSD

As in normal TMSD, incorrect inputs quickly fail during their invasion attempts due to the thermodynamic *and kinetic* penalty incurred when replacing a complementary sequence with one that is not. Input discrimination via displacement is typically absent in architectures that utilize strand-displacing polymerases. The reliance on thermodynamic discrimination alone among all possible hybridization events can limit the scalability in these other architectures, since a lack of kinetic sequence discrimination in the face of a growing number of sequences will lead to increased rates of mis-priming, subsequent polymerization extension, and thus spurious output production. In this section and in Section 4, we adopt a standardized structure for signal strands, each strand consisting of a unique identity sequence, with the toehold domain shown in a darker shade and the long domain in a lighter shade, followed by a universal domain depicted in yellow. This universal domain is identical across all signal strands. Although this domain could be made unique for each signal, logical correctness is still preserved when the same sequence is used across strands.

## 11:6 Leakless Polymerase-Dependent Strand Displacement Systems



**Figure 2** A translator gate converting an input signal  $X$  to a sequence orthogonal output signal  $Y$  using a single instance of the polymerase-dependent toehold-mediated strand displacement (PD-TMSD) motif. (a) The intended reaction pathway. (b) The intuitive-energy landscape [29].

### (ii) Polymerase-mediated strand displacement

Only after successful consumption of an input via TMSD, and only if its 3' end is paired with the template and contains a free hydroxyl group can polymerase extension and polymerase-mediated strand displacement of the output signal occur.

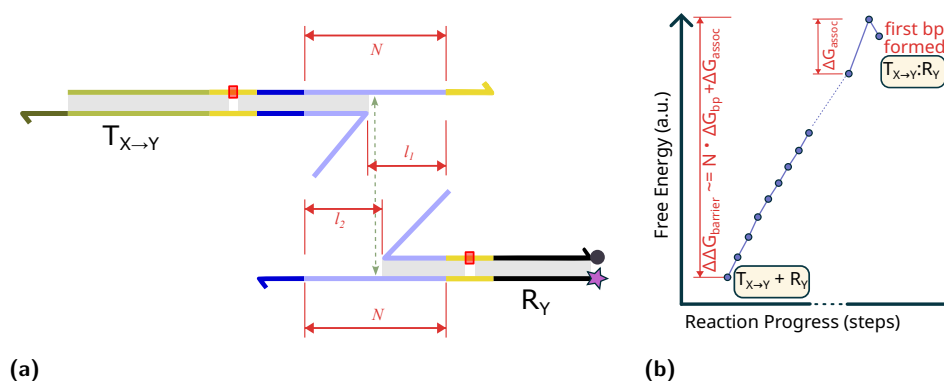
## 3.1 An arbitrarily high barrier to “leak” between gates

Remarkably, a single complex can produce an output signal that is sequence orthogonal to its input while guaranteeing an arbitrarily high energy barrier to leak. The intuition for this claim is depicted in Figure 3a: if a signal domain has not yet been produced then it is only found sequestered in double-stranded regions of gates thus necessitating an energy barrier to leak proportional to fraying open an entire long domain (see Figure 3b).

The robustness of this architecture is possible due to the complete sequestration of output signal domains in double stranded regions of gates. This point can be subtle. It is the systematic composition of this motif, into larger gates and then into entire circuits or reaction networks, that maintain an invariant on the barrier to leak. While our focus in this paper is on the description of the architecture, in the complete absence of input, it is a straightforward exercise to arrive at the following result. (We introduce **Join** and **Fork** gates in Section 4.)

► **Theorem 1.** *Given an arbitrary composition of PD-TMSD (activatable) translator, **Join** and **Fork** gates, with long domains of length  $N$ , then in the absence of input the production of a first output must overcome an energy barrier proportional to  $\Theta(N)$ .*





**Figure 3** Leak between two fuel complexes. (a) Initiation requires that a long-domain worth of base pairs fray open prior to the formation of the first intermolecular base pair. (b) The energy barrier to leak scales with  $N$ , the chosen length for long-domains.  $\Delta G_{assoc}$  is the entropic cost of combining two complexes into one [5] and  $\Delta G_{bp}$  is the average free energy of a base pair.

### 3.2 Activatable translator gates

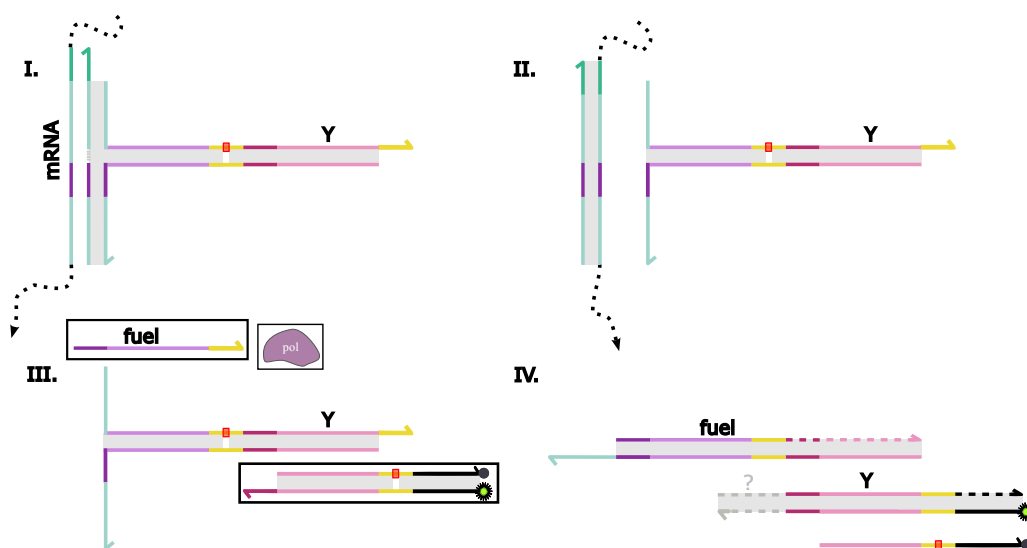
The simple translator motif can be extended into a multi-armed complex that remains inactive until a specific signal is present to “deprotect” the gate, at which point it becomes functionally identical to the original translator. This conditional gating strategy enables the system to remain dormant, sensing and collecting input signals without triggering downstream reactions until the final readout step. As shown in Figure 4, this approach is particularly well-suited for long-term applications such as environmental monitoring – where inputs may accumulate over days or weeks – and fuel strands, dNTPs and polymerase can be withheld from the reaction mixture until readout.

### 3.3 Activatable reactions and modulating rate constants

The PD-TMSD architecture supports engineered systems that require control of relative rate constants. As with TMSD, PD-TMSD reaction rate constants can be modulated by the careful design of toehold strengths and the introduction of secondary structure features into the branch migration domain such as mismatch introduction and mismatch replacement [15].

Additionally, the rate of the polymerase extension portion of a PD-TMSD reaction can also be modulated via sequence design and choice of dNTP concentrations. It is common to design signal strands over a three-letter code, reserving a fourth nucleotide that could be used as a rare or missing base to introduce conditional behavior. For example, one or more complements to a rare dNTP could be incorporated into the template strand of a gate, as illustrated in Figure 1 part IV. By changing the available concentration of the rare dNTP, the rate at which the polymerase is able to progress through that region can be tuned.

To prevent any polymerization through gates containing the rare dNTP complement, the corresponding dNTP could be omitted entirely from the reaction mixture, effectively disabling the associated reaction pathways. If desired, these pathways can later be activated by simply spiking in the missing dNTP, restoring gate function and enabling polymerase extension. The use of a missing dNTP to precisely halt polymerization has been previously demonstrated as an effective control mechanism in other contexts [12, 10].



**Figure 4** An activatable translator for long term environmental monitoring or detection of mRNA. Gates consume input mRNAs before fuel, reporter, and polymerase is added for quick readout.

## 4 Boolean circuits from leakless Join and Fork gates

In this section we introduce a leakless **Join** gate, capable of AND logic, and a leakless **Fork** gate capable of fanout. By design these gates naturally compose via the common PD-TMSD motif to realize arbitrary Boolean circuits that utilize dual-rail inputs and common gate outputs to emulate OR logic in the usual manner.<sup>2</sup> The signals in this section are compatible with the translator gates from Section 3 as they utilize the same domain structure for signal strands: a unique toehold domain, a unique long domain, and a common universal domain.

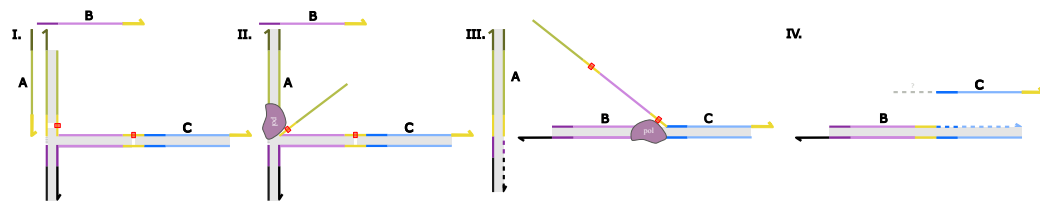
### 4.1 Leakless Join

Figure 5a introduces a three-stranded **Join** gate, a natural extension of the translator architecture and similar in style to the activatable translator. This design implements a conditional logic where the output **C** is only produced when both inputs **A** and **B** are present, and avoids sequence overlap between the two inputs – a desirable property for scalable systems.

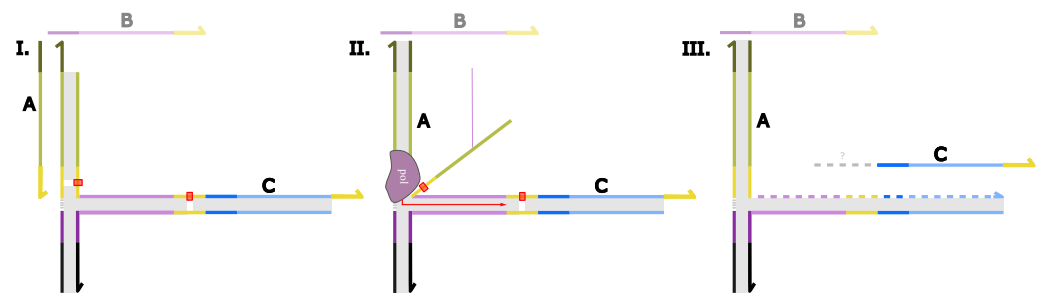
The gate operates as follows: input **A** first binds via TMSD to the exposed arm of the gate. This triggers polymerase-mediated extension, which displaces the *protector* strand. This displacement reveals a previously sequestered internal toehold, enabling input **B** to bind via TMSD, be extended by polymerase, and displace output **C**. In theory, this architecture realizes a leakless join operation, where both inputs are required sequentially for productive output.

However, in practice, we observed complications that undermine the theoretical leaklessness. Prior work [33] has shown that strand-displacing polymerases can exhibit template switching or “track jumping” in the absence of single-stranded binding proteins (SSBs), particularly when polymerizing through regions of secondary structure. In our gate, this

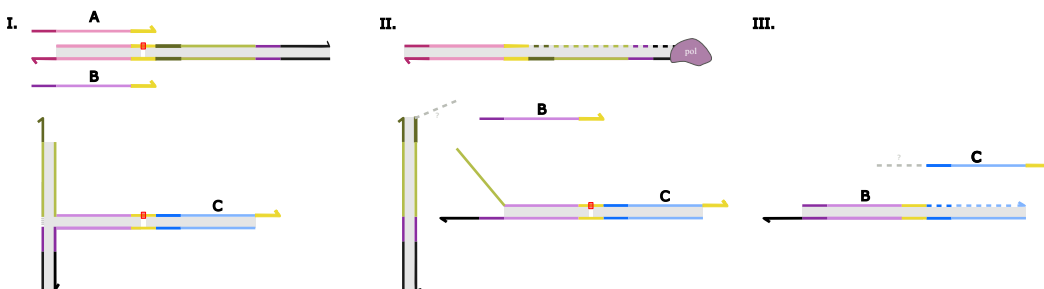
<sup>2</sup> For example,  $C := \text{OR}(A, B)$  can be realized by the parallel reactions  $A \rightarrow C$  and  $B \rightarrow C$ .



(a) A multi-arm join gate that is *leakless* in theory.

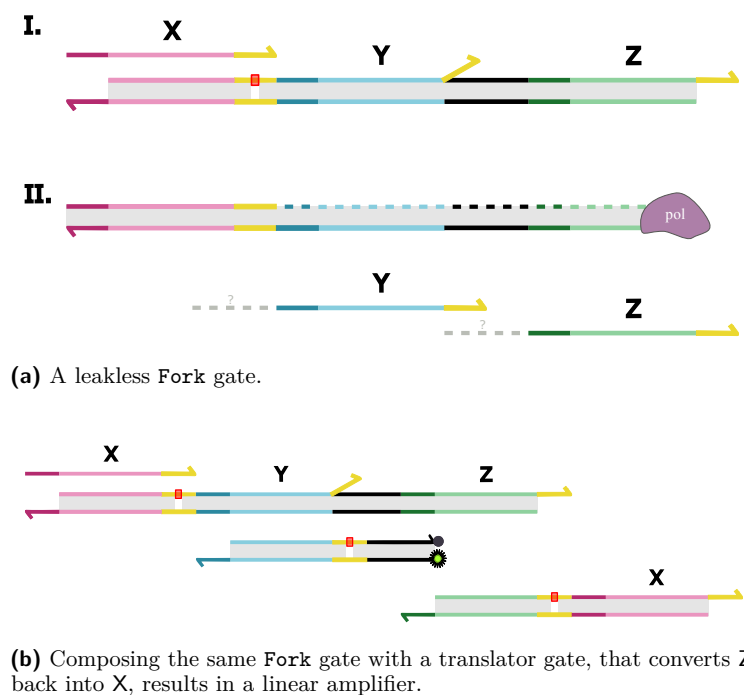


(b) Off-target “track jumping” activity of polymerase can lead to spurious output of C, absent input B. Here input B is faded out to denote its absence.



(c) Leakless Join that eliminates potential for “track jumping”.

■ **Figure 5** Join gates that are leakless in theory and practice.



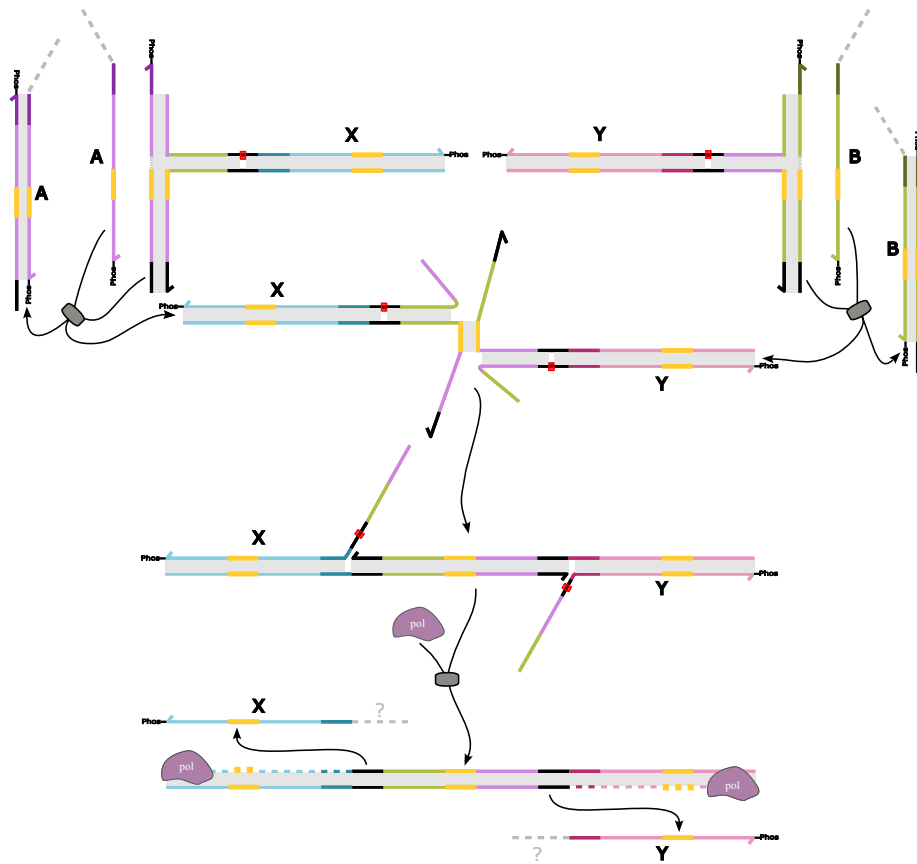
■ **Figure 6** Signal fanout and amplification using leakless Fork and translator gates.

behavior results in polymerase switching from the intended path to the adjacent duplex arm, producing spurious output C even without input A (Figure 5b). Thus, despite its appealing design, the gate is not functionally leakless under experimental conditions (data not shown).

To address the issue of track jumping, we added a specialized PD-TMSD translator gate (Figure 5c) for correct computation. In this design, input A binds the translator gate that produces an output strand which can remove the entire *protecting* strand on the three-stranded complex through TMSD. This eliminates the possibility of the polymerase choosing to follow an incorrect strand as the template. Following this deprotection event, an internal toehold is revealed that allows input B to bind and extend, resulting in the production of output C. Functionally, this gate resembles the activatable translator introduced in Section 3.2, where signal-dependent deprotection enables conditional activation, here utilized to implement join logic. This structural similarity highlights the modularity of the PD-TMSD motif and its adaptability to different computational tasks.

## 4.2 Leakless Fork

Figure 6a presents a leakless Fork gate that converts a single input X into two outputs, Y and Z. When input X binds to the fork gate via TMSD, it is extended by a polymerase, subsequently displacing two signal strands Y and Z. The overhang of the universal domain following signal Y has a free 3' end; however, it is sufficiently long and a designed to mismatch with the template in order to prevent spurious initiation of polymerase extension. The Fork gate can be generalized to produce an arbitrary number of outputs by extending the template strand to accommodate additional bound signals to be released via PD-TMSD initiated by a single input.



■ **Figure 7** *Pas de deux* (PDD) bimolecular reaction gate pathway.

By adding a single additional translator gate that converts output Z back into input X, signal fanout can be transformed into a linear signal amplifier (Figure 6b). This implementation enables signal amplification using only one polymerase step per gate, and two per cycle, using a minimal set of components.

## 5 Emulating arbitrary chemical reaction networks

While our previously proposed **Join** gate and **Fork** gate seemingly compose to implement the bimolecular reaction  $A + B \rightarrow X + Y$  that is not the case. In particular, the **Join** gate can irreversibly consume its first input in the complete absence of the second. While this behavior is fine for feedforward networks, dynamical systems such as oscillators require reactants to be (irreversibly) consumed, and then products produced, only when all reactants are present. In the remainder of this section we focus on a gate that implements *rateless* bimolecular reactions, but point the reader to Section 3.3 for a discussion of how rate constants can be programmed with these gates.

### *Pas de deux* bimolecular reaction gate

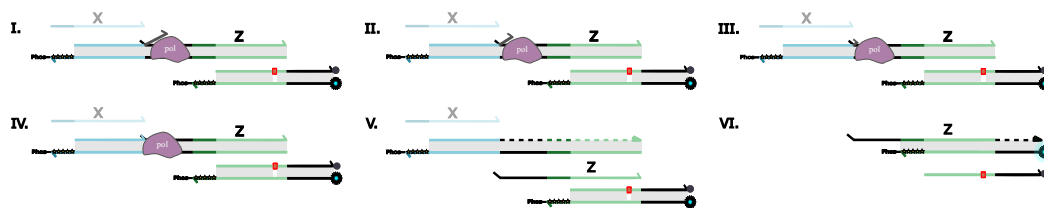
We introduce the “*pas de deux*” (PDD) gate.<sup>3</sup> Like its namesake, it exhibits symmetry and elegant coordination of a pair as they act alone and then in a choreographed sequence to achieve their collective aim. Although the sequence of events in our system unfolds slightly out of order compared to a *grand pas de deux*, the underlying motifs remain the same. Each input first interacts with its respective gate, stripping off the *protector* strand via toehold-mediated strand exchange (TMSE) – this step mirrors the solo variations of male and female dancers, each showcasing their individual abilities. The combination of 3′ phosphate modifications on inputs and toeholds to prevent polymerization along with TMSE at this stage, render these reactions reversible. Only when both inputs are present can the two deprotected components of the gates come together – analogous to the adagio when dancers join in a coordinated, slow duet. In the final stage of the reaction, the polymerase is able to find and extend from the two free 3′ hydroxyl groups, displacing both output strands in a final, coordinated motion – this mirrors the coda where both dancers re-emerge with renewed energy, and the performance culminates in a dazzling, synchronized flourish.

## 6 Practical considerations and exploitations

### 6.1 Improving PSD kinetics with mismatches

To facilitate more efficient strand displacement by Klenow fragment (exo-) polymerase, we introduced intentional mismatches into the duplex regions downstream of input binding. Unlike more processive polymerases or those operating at elevated temperatures – where decreased duplex stability can aid in duplex unwinding – Klenow operates at relatively low temperatures (25–37°C) and lacks intrinsic helicase activity, making polymerase-mediated strand displacement more challenging. In natural systems, single-stranded binding proteins (SSBs) have been shown to enhance strand displacement activity by binding and stabilizing the displaced strand, preventing its rehybridization to the template, thus enabling continuous polymerase progression [18]. However, because our system uses single-stranded DNA as signal strands, the inclusion of SSBs is incompatible, as they would bind these functional signals and interfere with downstream reactions. In the absence of such stabilizing factors, the displaced strand can readily rehybridize to the template, creating a kinetic barrier to polymerase progression. By incorporating mismatches or bulges into the displaced strand, we reduce its hybridization stability and effectively slow the reverse reaction, lowering the energy barrier for forward strand displacement synthesis. A similar strategy has been observed in biological systems; for example, HIV-1 reverse transcriptase leverages single unpaired nucleotides or bulged structures within its RNA genome to promote strand displacement and enhance polymerase processivity through otherwise inhibitory duplex regions [13]. Consistent with this, Srinivas *et al.* (2013) showed that branch migration initiation can be a key rate-limiting step in strand displacement and that tuning the stability of the incumbent strand’s interactions with the template can modulate these interactions to favor forward displacement [29]. This design strategy provides a simple and effective means to improve polymerization kinetics in systems where traditional accessory proteins cannot be used.

<sup>3</sup> In classical ballet, “*pas de deux*” (French for “step of two”) is a duet typically performed by a male and female dancer in concert. A *grand pas de deux* follows the structure of an entrée, a slow partnered adagio, individual solos for each dancer, and culminates in a fast virtuosic coda danced in unison.



**Figure 8** A molecular fuse gate utilizing the  $3' \rightarrow 5'$  exonuclease activity of Klenow fragment polymerase. Signal X is shown faded out to denote its absence.

## 6.2 Exploitations of proof-reading polymerases

While the majority of this paper focuses on systems built with the exonuclease-deficient Klenow fragment (Klenow  $\text{exo}^-$ ), our architecture is readily adaptable to polymerases that retain  $3' \rightarrow 5'$  exonuclease activity, such as the full Klenow fragment polymerase. Before exploring the new behaviors this enables, we first address several important design considerations necessary for working with proofreading polymerases.

In systems using strand displacing polymerases with an exonuclease domain, all exposed  $3'$  overhangs – including toeholds – must be protected from degradation. This is typically achieved using phosphorothioate (PS) backbone modifications, which render the strand resistant to exonuclease activity. Once these protective modifications are in place, the exonuclease activity of the polymerase can be harnessed as a feature rather than a threat. In particular, we can exploit the gradual degradation of single-stranded overhangs by exonuclease activity.

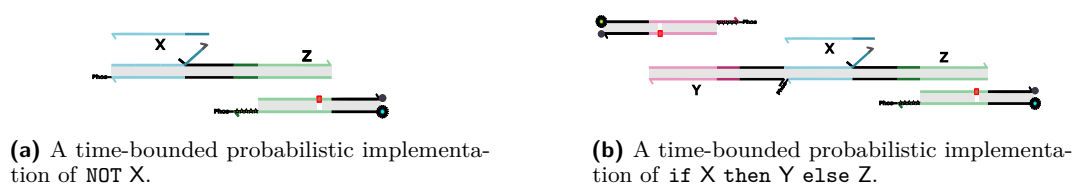
As the polymerase chews back mismatched or unpaired regions from the  $3'$  end of a strand, it will eventually reveal a perfectly matched primer-template duplex. At this point, the enzyme can initiate extension. This delayed onset of polymerization effectively functions as a molecular timer, with the length of the single-stranded region acting as a programmable delay: the longer the overhang, the longer the delay before polymerization can proceed.

Timed delay elements are increasingly important in molecular systems that require sequential, transient, or conditional behavior. Prior work has demonstrated timer circuits and sequential release mechanisms using DNA strand displacement [6, 22, 14], as well as enzymatic timers for pulse generation and orthogonal delays [2, 3, 21]. However, many of these designs depend on auxiliary structures, multi-step cascades, or reaction-specific configurations, which can limit composability and generalizability. Our approach offers a simple and scalable alternative: delay elements based on a single enzyme and tunable overhangs. We use this principle to build a new class of timing-enabled molecular gates, that highlight how proofreading polymerases, when used strategically, can unlock compact and programmable temporal control in DNA-based molecular computation.

### A molecular “fuse”

Here we introduce the molecular fuse gate (Figure 8), where polymerization is triggered after a tunable delay set by the length of the gray single-stranded overhang. In the absence of input X, the proofreading polymerase begins degrading the overhang from the  $3'$  end via its exonuclease activity. If sufficient time passes and the overhang is completely digested, the polymerase will reach a fully paired primer-template junction and initiate extension. This results in the production of an output that can be routed into another reaction gate or reporter complex, so fluorescent output can be observed.





■ **Figure 9** Time-bounded probabilistic gates.

In contrast, when input  $X$  is present, it binds to the overhang and displaces the primer strand via standard strand displacement, allowing for immediate polymerase extension – functionally behaving like a conventional translator gate. The fuse gate thus acts as a *timed checkpoint*: if a particular input is not detected within a programmable time window, determined by the overhang length, a default output is produced. This behavior enables built-in responses in molecular circuits that require temporal decision-making in the absence of expected inputs.

### Time-bounded probabilistic NOT gates and thresholds

Figure 9a shows a probabilistic NOT gate that operates within a bounded time window. In the absence of input  $X$ , the proofreading polymerase gradually degrades the long single-stranded overhang from the 3' end. Once fully degraded, polymerase extension proceeds, displacing an output strand that can trigger a fluorescent reporter or other downstream gate. In the presence of input  $X$ , the overhang serves as a toehold, enabling  $X$  to bind and displace the primer strand before it can be extended. This inhibits production of output  $Z$ , acting as a logical NOT gate.

However, this behavior is temporally constrained: the  $3' \rightarrow 5'$  exonuclease activity will eventually begin degrading not just the spacer domain (gray), but also the toehold for  $X$ . As the toehold shortens, the probability that  $X$  can successfully bind and displace the primer decreases. Thus, the likelihood of output production increases over time – even in the presence of  $X$  – leading to a time-bounded probabilistic inversion. Early in the time course,  $X$  has a high chance of binding and blocking signal propagation; later, as its binding site erodes, the system effectively “commits” to the NOT operation being bypassed.

Figure 9b extends the probabilistic NOT gate architecture by introducing a second output arm, resulting in a time-bounded If-Then-Else gate. In this system, the polymerase can generate one of two outputs –  $Y$  or  $Z$  – depending on the presence or absence of input  $X$ . In the absence of  $X$ , the exonuclease slowly degrades the long single-stranded overhang. Once fully processed, polymerase extension proceeds, leading to the production of output  $Z$ , which can be routed to a specific reporter (*e.g.*, a fluorescent channel) or downstream gate. This mirrors the behavior of the fuse and NOT gates described earlier. In the presence of  $X$ , the overhang instead serves as a toehold, allowing  $X$  to bind and displace the primer strand. This both prevents the generation of  $Z$  and triggers the release of output  $Y$  through strand displacement, enabling an alternate reporter or reaction pathway. The resulting behavior can be logically described as: If  $X$ , then output  $Y$ ; else output  $Z$ . As with the NOT gate, this decision is time-sensitive – if  $X$  fails to bind before its toehold is degraded then the system defaults to producing  $Z$ .

## 7 Discussion

We presented a versatile molecular programming framework (PD-TMSD) that integrates toehold-mediated strand displacement with the enzymatic activity of strand-displacing polymerases to realize an unprecedented level of robustness in a compact, programmable molecular implementation. This system is capable of implementing arbitrary Boolean circuits and emulating arbitrary chemical reaction networks. While polymerase-based systems expand the design space for molecular logic and signal processing, they also introduce practical constraints and benefits that must be considered for real-world applications. We explored these trade-offs in novel uses of time-dependent probabilistic gates. In practice, unlike enzyme-free systems, these enzyme-driven architectures depend on the activity and stability of the polymerase. Many commonly used polymerases are not stable at ambient temperatures and have limited functional lifetimes. Although some thermostable variants exist, most require refrigeration or freezing to preserve activity, and not all polymerases are amenable to lyophilization. As a result, systems that rely on polymerase function – particularly for biosensing or diagnostic applications – may require a cold-chain infrastructure, which can limit deployment in resource-limited or field-based environments. Despite these limitations, the modularity, composability, and kinetic control offered by PD-TMSD systems open new possibilities for building robust, programmable molecular systems.

## References

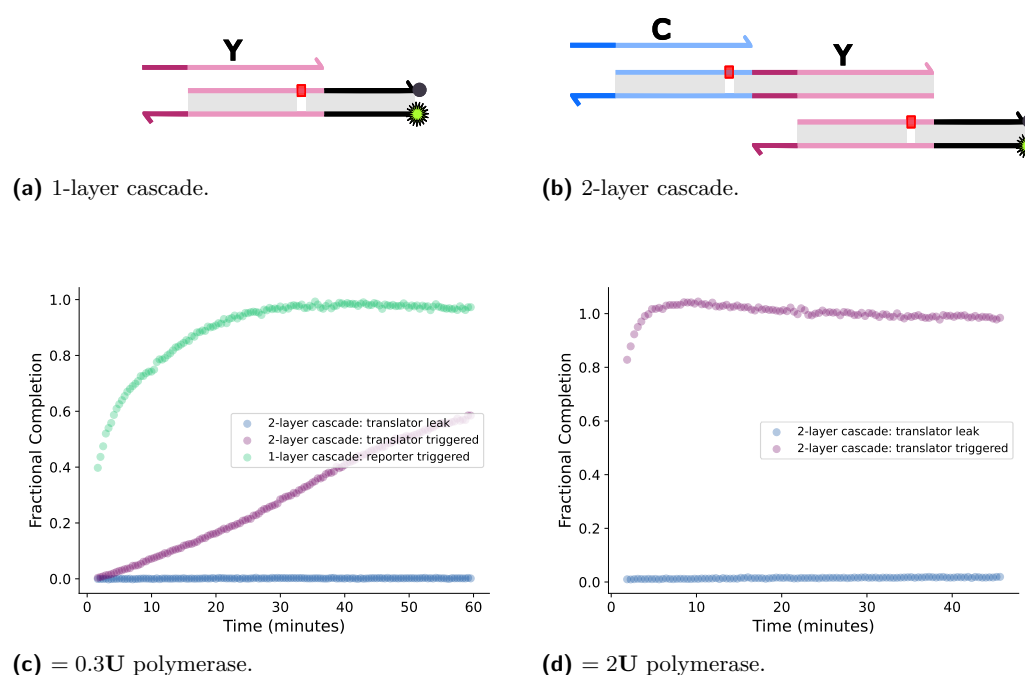
- 1 Alexandre Baccouche, Kevin Montagne, Adrien Padirac, Teruo Fujii, and Yannick Rondelez. Dynamic DNA-toolbox reaction circuits: a walkthrough. *Methods*, 67(2):234–249, 2014.
- 2 Juliette Bucci, Patrick Irmisch, Erica Del Grosso, Ralf Seidel, and Francesco Ricci. Orthogonal enzyme-driven timers for DNA strand displacement reactions. *Journal of the American Chemical Society*, 144(43):19791–19798, 2022.
- 3 Juliette Bucci, Patrick Irmisch, Erica Del Grosso, Ralf Seidel, and Francesco Ricci. Timed pulses in DNA strand displacement reactions. *Journal of the American Chemical Society*, 145(38):20968–20974, 2023.
- 4 Kevin M Cherry and Lulu Qian. Scaling up molecular pattern recognition with DNA-based winner-take-all neural networks. *Nature*, 559(7714):370–376, 2018.
- 5 Robert M Dirks, Justin S Bois, Joseph M Schaeffer, Erik Winfree, and Niles A Pierce. Thermodynamic analysis of interacting nucleic acid strands. *SIAM review*, 49(1):65–88, 2007. doi:10.1137/060651100.
- 6 Joshua Fern, Dominic Scalise, Angelo Cangialosi, Dylan Howie, Leo Potters, and Rebecca Schulman. DNA strand-displacement timer circuits. *ACS Synthetic Biology*, 6(2):190–193, 2017.
- 7 Teruo Fujii and Yannick Rondelez. Predator–prey molecular ecosystems. *ACS Nano*, 7(1):27–34, 2013.
- 8 Guillaume Gines, Roberta Menezes, Kaori Nara, Anne-Sophie Kirstetter, Valerie Taly, and Yannick Rondelez. Isothermal digital detection of micrornas using background-free molecular circuit. *Science Advances*, 6(4):eaay5952, 2020.
- 9 Guillaume Gines, AS Zadorin, J-C Galas, Teruo Fujii, A Estevez-Torres, and Y Rondelez. Microscopic agents programmed by DNA circuits. *Nature Nanotechnology*, 12(4):351–359, 2017.
- 10 Hong Kang, Yuexuan Yang, and Bryan Wei. Synthetic molecular switches driven by DNA-modifying enzymes. *Nature Communications*, 15(1):3781, 2024.
- 11 Tiernan Kennedy, Cadence Pearce, and Chris Thachuk. Fast and robust strand displacement cascades via systematic design strategies. In *28th International Conference on DNA Computing and Molecular Programming (DNA 28)(2022)*, pages 1–1. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2022.

- 12 Jocelyn Y Kishi, Thomas E Schaus, Nikhil Gopalkrishnan, Feng Xuan, and Peng Yin. Programmable autonomous synthesis of single-stranded DNA. *Nature Chemistry*, 10(2):155–164, 2018.
- 13 Christian Lanciault and James J Champoux. Single unpaired nucleotides facilitate HIV-1 reverse transcriptase displacement synthesis through duplex RNA. *Journal of Biological Chemistry*, 279(31):32252–32261, 2004.
- 14 Anna P Lapteva, Namita Sarraf, and Lulu Qian. DNA strand-displacement temporal logic circuits. *Journal of the American Chemical Society*, 144(27):12443–12449, 2022.
- 15 Robert RF Machinek, Thomas E Ouldrige, Natalie EC Haley, Jonathan Bath, and Andrew J Turberfield. Programmable energy landscapes for kinetic control of DNA strand displacement. *Nature communications*, 5(1):5324, 2014.
- 16 Kevin Montagne, Guillaume Gines, Teruo Fujii, and Yannick Rondelez. Boosting functionality of synthetic DNA circuits with tailored deactivation. *Nature Communications*, 7(1):13474, 2016.
- 17 Kevin Montagne, Raphael Plasson, Yasuyuki Sakai, Teruo Fujii, and Yannick Rondelez. Programming an in vitro DNA oscillator using a molecular networking strategy. *Molecular systems biology*, 7(1):466, 2011.
- 18 Ismael Plaza-GA, Kateryna M Lemishko, Rodrigo Crespo, Thinh Q Truong, Laurie S Kaguni, Francisco J Cao-García, Grzegorz L Ciesielski, and Borja Ibarra. Mechanism of strand displacement DNA synthesis by the coordinated activities of human mitochondrial DNA polymerase and SSB. *Nucleic Acids Research*, 51(4):1750–1765, 2023.
- 19 Lulu Qian and Erik Winfree. Scaling up digital circuit computation with DNA strand displacement cascades. *Science*, 332(6034):1196–1201, 2011.
- 20 Lulu Qian, Erik Winfree, and Jehoshua Bruck. Neural network computation with DNA strand displacement cascades. *Nature*, 475(7356):368–372, 2011. doi:10.1038/NATURE10262.
- 21 Qinze Rong, Yingnan Deng, Fangzhou Chen, Zhe Yin, Lingfei Hu, Xin Su, and Dongsheng Zhou. Polymerase-based signal delay for temporally regulating DNA involved reactions, programming dynamic molecular systems, and biomimetic sensing. *Small*, 20(35):2400142, 2024.
- 22 Dominic Scalise, Moshe Rubanov, Katherine Miller, Leo Potters, Madeline Noble, and Rebecca Schulman. Programming the sequential release of DNA. *ACS Synthetic Biology*, 9(4):749–755, 2020.
- 23 Georg Seelig, David Soloveichik, David Yu Zhang, and Erik Winfree. Enzyme-free nucleic acid logic circuits. *Science*, 314(5805):1585–1588, 2006.
- 24 Shalin Shah, Tianqi Song, Xin Song, Ming Yang, and John Reif. Implementing arbitrary CRNs using strand displacing polymerase. In *DNA Computing and Molecular Programming: 25th International Conference, DNA 25, Seattle, WA, USA, August 5–9, 2019, Proceedings 25*, pages 21–36. Springer, 2019. doi:10.1007/978-3-030-26807-7\_2.
- 25 Shalin Shah, Jasmine Wee, Tianqi Song, Luis Ceze, Karin Strauss, Yuan-Jyue Chen, and John Reif. Using strand displacing polymerase to program chemical reaction networks. *Journal of the American Chemical Society*, 142(21):9587–9593, 2020.
- 26 David Soloveichik, Georg Seelig, and Erik Winfree. DNA as a universal substrate for chemical kinetics. *Proceedings of the National Academy of Sciences*, 107(12):5393–5398, 2010.
- 27 Tianqi Song, Abeer Eshra, Shalin Shah, Hieu Bui, Daniel Fu, Ming Yang, Reem Mokhtar, and John Reif. Fast and compact DNA logic circuits based on single-stranded gates using strand-displacing polymerase. *Nature Nanotechnology*, 14(11):1075–1081, 2019.
- 28 Tianqi Song, Nikhil Gopalkrishnan, Abeer Eshra, Sudhanshu Garg, Reem Mokhtar, Hieu Bui, Harish Chandran, and John Reif. Improving the performance of DNA strand displacement circuits by shadow cancellation. *ACS Nano*, 12(11):11689–11697, 2018.
- 29 Niranjana Srinivas, Thomas E Ouldrige, Petr Šulc, Joseph M Schaeffer, Bernard Yurke, Ard A Louis, Jonathan PK Doye, and Erik Winfree. On the biophysics and kinetics of toehold-mediated DNA strand displacement. *Nucleic Acids Research*, 41(22):10641–10658, 2013.

- 30 Niranjan Srinivas, James Parkin, Georg Seelig, Erik Winfree, and David Soloveichik. Enzyme-free nucleic acid dynamical systems. *Science*, 358(6369):eaal2052, 2017.
- 31 Haomiao Su, Jinglei Xu, Qi Wang, Fuan Wang, and Xiang Zhou. High-efficiency and integrable DNA arithmetic and logic system based on strand displacement synthesis. *Nature Communications*, 10(1):5390, 2019.
- 32 Chris Thachuk, Erik Winfree, and David Soloveichik. Leakless DNA strand displacement systems. In *DNA Computing and Molecular Programming: 21st International Conference, DNA 21, Boston and Cambridge, MA, USA, August 17-21, 2015. Proceedings 21*, pages 133–153. Springer, 2015. doi:10.1007/978-3-319-21999-8\_9.
- 33 Enrique Viguera, Danielle Canceill, and S Dusko Ehrlich. Replication slippage involves DNA polymerase pausing and dissociation. *The EMBO journal*, 2001.
- 34 Boya Wang, Chris Thachuk, Andrew D Ellington, and David Soloveichik. The design space of strand displacement cascades with toehold-size clamps. In *DNA Computing and Molecular Programming: 23rd International Conference, DNA 23, Austin, TX, USA, September 24–28, 2017, Proceedings 23*, pages 64–81. Springer, 2017. doi:10.1007/978-3-319-66799-7\_5.
- 35 Boya Wang, Chris Thachuk, Andrew D Ellington, Erik Winfree, and David Soloveichik. Effective design principles for leakless strand displacement systems. *Proceedings of the National Academy of Sciences*, 115(52):E12182–E12191, 2018.
- 36 David Yu Zhang and Georg Seelig. Dynamic DNA nanotechnology using strand-displacement reactions. *Nature Chemistry*, 3(2):103–113, 2011.

## A Supplementary Information

### A.1 Preliminary experimental demonstration



**Figure 10** Shallow cascades with varying concentrations of polymerase. Translator, reporter, and inputs were always at  $1 \times 100 \text{ nM}$  and polymerase concentration was at  $0.03 \text{ U}/\mu\text{L}$  or  $0.2 \text{ U}/\mu\text{L}$ .  $10 \mu\text{L}$  reactions were performed in NEB r2.1 buffer at  $37^\circ\text{C}$ .

We performed two preliminary experiments with 1-layer and 2-layer cascades to validate the feasibility of our PD-TMSD architecture in achieving fast, programmable signal propagation while maintaining minimal leak – properties critical for building scalable molecular computing architectures. Note that this experimental architecture is modified from the proposed motif in the main paper, with signal strands lacking the universal domain following a unique toehold and long domain. In the single-layer cascade, fluorescent output is produced through the toehold-mediated strand displacement of trigger Y into the reporter complex, followed by the polymerase-mediated strand displacement from the 3' end of the trigger through the clamp region, resulting in the quencher strand being displaced and observed fluorescence (Figure 10a). The two-layer cascade introduces an additional translator upstream of the reporter (Figure 10b). Here, the initial trigger C goes through the translator via the same PD-TMSD mechanism, producing trigger Y which can then interact with the reporter.

Fractional completion is used as a normalized measure of reporter signal in our experiments. A value of 1.0 corresponds to the maximum fluorescence from a fully triggered reporter, while the value of 0.0 corresponds to a fully quenched reporter.

We first evaluated the systems at low polymerase concentration (0.3 units of Klenow fragment exo-). The 1-layer cascade exhibited a reaction halftime of approximately 2.5 mins, reaching completion within 30 minutes. As expected, the 2-layer cascade experienced a kinetic slowdown with a halftime of approximately 50 minutes, reflecting the additional step introduced by the translator (Figure 10c). When the polymerase concentration was increased to 2 units, we observed a dramatic acceleration in the 2-layer cascade with the reaction completing in less than 5 minutes and a half-time that was complete prior to first fluorescence measurement (Figure 10d). This tunable speed demonstrates a key advantage of PD-TMSD, reaction kinetics – up to a certain point – can be tuned through enzyme concentration, enabling rapid signal propagation without need for architectural redesign.

Importantly, the improvement in kinetics of the triggered cases did not come at the cost of increased leak in the cases lacking input. Even at elevated polymerase concentrations, the background signal in the absence of input remained minimal over the observed time period. This confirms the compatibility of toehold-mediated and polymerase-driven strand displacement, enabling the construction of molecular systems that are both fast and robust. These preliminary experiments demonstrate that our underlying theoretical framework provides a strong foundation for the construction of larger and more sophisticated architectures.

## A.2 DNA sequences

Strand ID	Sequence
Signal-Y	CCTATCCACTCTCACCT
Quencher-top	CCACTCTCATCTTACACATTCCAAA/3BHQ <sub>1</sub> /
Reporter-bottom	/5ATTO488N/TTTGGAATGTGTAAGGTGAGAGTGGATAGG
Signal-C	CCACACTTCTCTTCTCC
C-Y-translator-top	CTTCTCTTCACCCCTATCCACTCTCACCT
C-Y-translator-bottom	AGGTGAGTGTGGAGAGGGGAGAAGAGAAGTGTGG

## A.3 DNA sequence design and gate preparation

DNA sequences were designed using NUPACK dna04 model at 25°C, 50mM Na<sup>+</sup>, and 12.5mM Mg<sup>2+</sup>. All signal strands were designed to be unstructured, over the A, T, C alphabet, and orthogonal to each other in order to minimize occlusion. Strands were ordered from Integrated

DNA Technologies, dried and unpurified at 100 nmole scale. Single strands were dPAGE purified in a 15% polyacrylamide gel with a 5% stacking gel using a Hoefer SE6000X Chroma Deluxe Electrophoresis unit heated to 55°C.

To prepare gates, strands were annealed with 1.2× excess top strand in 1× TEM buffer (12.5 mM MgCl<sub>2</sub>, 10 mM Tris, 1 mM EDTA, pH 8.0). Annealing protocol was executed in an Eppendorf Mastercycler Nexus thermocycler, heating to 90°C over two minutes then cooling by 1°C/min to 20°C.

Complexes were PAGE purified in a 12% polyacrylamide gel with 6% stacking gel in 1× TAE/MgCl<sub>2</sub> buffer (40 mM Tris-acetate, 1mM EDTA, 12.5 mM MgCl<sub>2</sub>, pH 8.0) using a Hoefer SE6000X Chroma Deluxe Electrophoresis unit. Following purification, desired band was excised from gel, cut into strips, and eluted into 1× TEMT (12.5 mM MgCl<sub>2</sub>, 10 mM Tris, 1 mM EDTA, 0.01% Tween, pH 8.0) buffer overnight. Following elution, stock concentrations were estimated from A260 measurements on a Nanodrop One.

#### A.4 Fluorescence spectroscopy

All experiments were conducted on a Biotek Cytation 5 microplate reader with a bottom read and 100 gain. The excitation and emission wavelengths for used for ATTO488 were 500 nm and 525 nm, respectively.

Individual components for each experiment were combined using an ECHO 525 Acoustic Liquid Handler (Beckman Coulter) into Corning® 384 well plates. Each experimental condition presented was prepared in duplicate or triplicate. After transfers from ECHO were complete, the plates were vortexed and centrifuged. Plates were read within 2 minutes of coming out of ECHO.

