


# 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).

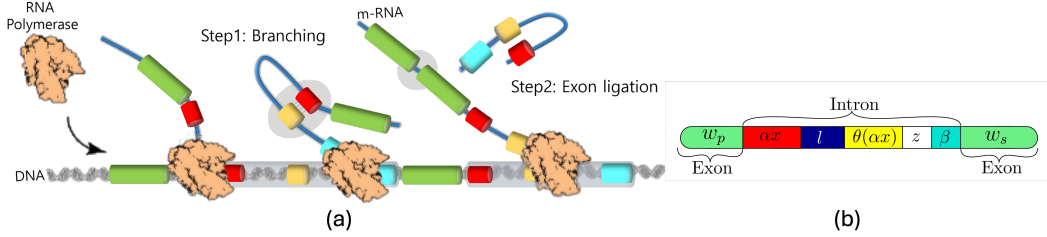
Editors: Josie Schaeffer and Fei Zhang; Article No. 5; pp. 5:1–5:22

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 oritadami 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).

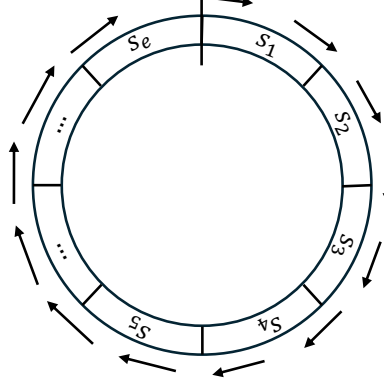
► **Theorem 5.** Let  $A = (Q, \Sigma, q_1, \delta, F)$  be some NFA. There exists a word  $w \in \Sigma^*$  and a properties tuple for log-hairpin deletion  $S = (\Sigma, \theta, c, C, n, T, m)$  as defined before such that  $L(A) = [w^\omega]_{\xrightarrow{S}^{\uparrow \log}}$ .

**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.

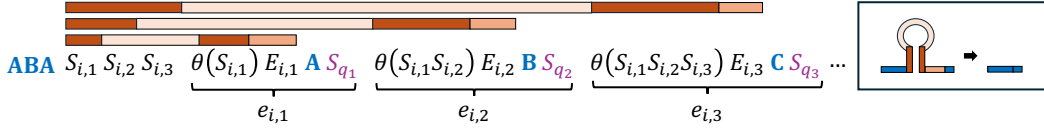
We continue with the definition of the context-set  $C$  over the words defined above. For each  $i \in [o]$ , we assume  $(S_{q_i}, E_{q_i}) \in C$ . These are contexts responsible for jumping between different  $s_i$  and  $s_j$  using hairpin deletion. In addition to  $i \in [o]$ , for each  $j \in [k]$  (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.

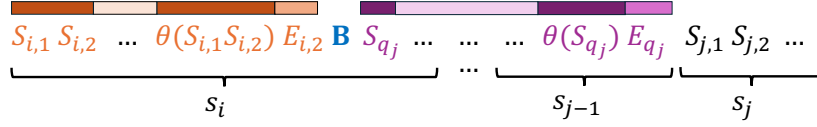
**Intuition.** Transcription starts in  $s_1$  and moves around  $w^\omega$  over and over again (see Figure 2). Every time the transcription is at the beginning of one of the factors  $s_i$ ,  $i \in [o]$ , a nondeterministic choice of one of the encoded transitions occurs. By the selection of one of 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. ◀

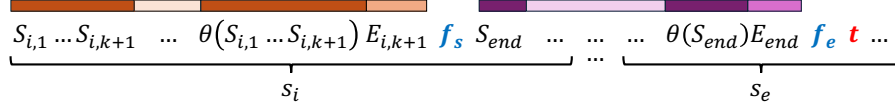
For this construction to work, we clearly needed some assumptions regarding the hairpin deletion model to simulate co-transcriptional splicing. These are especially the assumptions that we are able to do this in a parallel manner (modeling the independence of subsequent co-transcriptional deletions) 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  $(S_{q_m}, E_{q_m}) \in C$ , 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.*

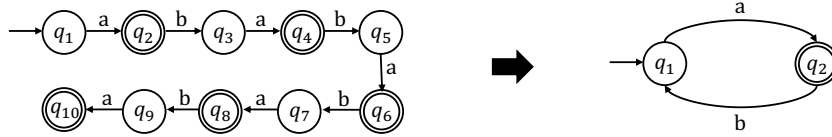
Next, in the encoding of NFAs onto templates for hairpin deletion proposed in Theorem 5, different states are encoded consecutively. So, simulating a transition from an earlier encoded state to a later encoded state still occurs on a single repetition of the template. Hence, we could also impose an order  $<_Q$  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  $L(A) \subseteq [w^\omega]_{\text{p}\uparrow\text{-log}_{S'}}$  given a constructive argument. Then, we show that  $[w^\omega]_{\text{p}\uparrow\text{-log}_{S'}} \subseteq L(A)$  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 ( $L(A) \subseteq [w^\omega]_{\text{p}\uparrow\text{-log}_{S'}}$ ).** 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.

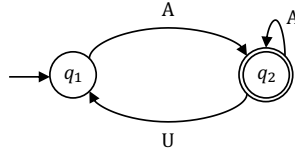
For all contexts  $(S_{i,1} \dots S_{i,j}, E_{i,j}) \in C$ , we see that they are bordered by either AA or UU, followed and preceded by either C or G. For all  $S_{i,1} \dots S_{i,j}$ , we know that after the first AA, there occur  $i$  many C's. We notice that there is only one position in the word 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.