

# Automated Synthesis: Functional, Reactive and Beyond

S. Akshay<sup>\*1</sup>, Bernd Finkbeiner<sup>\*2</sup>, Kuldeep S. Meel<sup>\*3</sup>,  
Ruzica Piskac<sup>\*4</sup>, and Arijit Shaw<sup>†5</sup>

- 1 Indian Institute of Technology Bombay – Mumbai, IN. [akshayss@cse.iitb.ac.in](mailto:akshayss@cse.iitb.ac.in)
- 2 CISPA – Saarbrücken, DE. [finkbeiner@cispa.de](mailto:finkbeiner@cispa.de)
- 3 University of Toronto, CA. [meel@comp.nus.edu.sg](mailto:meel@comp.nus.edu.sg)
- 4 Yale University – New Haven, US. [ruzica.piskac@yale.edu](mailto:ruzica.piskac@yale.edu)
- 5 Chennai Mathematical Institute, IN & University of Toronto, CA.  
[if.arijit@gmail.com](mailto:if.arijit@gmail.com)

---

## Abstract

This report summarizes the program of Dagstuhl Seminar 24171 on “Automated Synthesis: Functional, Reactive and Beyond”. The seminar brought together researchers working on different aspects of functional synthesis and investigated its relationship with reactive synthesis. Through multiple expository tutorials, diverse technical talks, and multiple open discussion sessions, the seminar crystallized the current challenges for theory and tools in this area and opened fresh directions towards new applications.

**Seminar** April 21–26, 2024 – <https://www.dagstuhl.de/24171>

**2012 ACM Subject Classification** Computing methodologies; Theory of computation → Logic  
**Keywords and phrases** automated synthesis, boolean functions, knowledge representations, reactive synthesis, SAT/SMT solvers

**Digital Object Identifier** 10.4230/DagRep.14.4.85


## 1 Executive Summary

*S. Akshay (Indian Institute of Technology Bombay – Mumbai, IN)*

*Bernd Finkbeiner (CISPA – Saarbrücken, DE)*

*Kuldeep S. Meel (University of Toronto, CA)*

*Ruzica Piskac (Yale University – New Haven, US)*

**License**  Creative Commons BY 4.0 International license  
© S. Akshay, Bernd Finkbeiner, Kuldeep S. Meel, and Ruzica Piskac

In Dagstuhl Seminar 24171, we brought together researchers working in various aspects of automated functional synthesis. This diverse topic encompasses areas ranging from Boolean variants to quantified variants, automated reasoning for general theories, program synthesis, and more. One particular focus was on finding synergies between functional and reactive synthesis communities and investigating the deep connections between these two areas.

On the first day, we started with two introductory tutorials: one on Boolean functional synthesis and another on reactive synthesis, setting the agenda for the entire seminar. This was succeeded by technical presentations on definability and dependency in quantified Boolean formulas. The second day included a tutorial on automated reasoning and synthesis, with an emphasis on theories extending beyond Boolean (e.g., SMT), followed by discussions

---

\* Editor / Organizer

† Editorial Assistant / Collector



Except where otherwise noted, content of this report is licensed under a Creative Commons BY 4.0 International license

Automated Synthesis: Functional, Reactive and Beyond, *Dagstuhl Reports*, Vol. 14, Issue 4, pp. 85–107

Editors: S. Akshay, Bernd Finkbeiner, Kuldeep S. Meel, and Ruzica Piskac



DAGSTUHL  
REPORTS

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

on quantitative properties. On the third day, we organized a special session with other tool competition organizers to assess the feasibility of a competition or track dedicated to functional synthesis.

The remaining days were filled with diverse technical talks that fell into two categories. The first category included talks that delved deeper into specific aspects of functional synthesis, reactive/LTL synthesis, and specific problems within these fields. The second category introduced new applications or connections, such as quantum applications and functional programming. Discussions during and beyond these talks were further explored in different open and problem sessions. Some of the identified and discussed problems were:

1. How to formalize the Boolean functional synthesis problem at the heart of reactive synthesis? Various problem formulations were discussed, and some benchmarks were created.
2. Can we go beyond Boolean theories and synthesize programs and functions for general SMT? What bottlenecks do we face?
3. How can we find synergy between automated functional synthesis and synthesis using transformers? Specifically, what is the meeting ground between machine learning and inductive program synthesis techniques, functional synthesis, and automated reasoning?
4. Can the successful lens of knowledge representations and compilations for model counting and Boolean functional synthesis be extended to other settings?
5. Can we synthesize quantum circuits from specifications, thus leading to a theory of automated reasoning for quantum systems?
6. Can reactive synthesis over finite traces utilize techniques developed in automated functional synthesis?

These were among the prominent topics discussed, but the list is by no means exhaustive. Several bottlenecks were identified, such as the need for growth within the community developing these tools before establishing a proper competition. Additionally, there was a recognized necessity for broader and more extensive discussions on benchmarks.

Overall, the seminar fostered a collaborative spirit among theoreticians, tool developers, and experts across different aspects of automated functional synthesis. The seminar was also attended by a large number of early career researchers, postdoctoral fellows, and graduate students who also participated enthusiastically throughout the seminar. The shared optimism generated during this seminar has laid a strong foundation for future advancements. We advocate for the continuation of these valuable discussions and propose organizing further meetings of a similar nature to build on the momentum gained and to explore new frontiers in automated functional synthesis.

In the remainder of this report, we provide the abstracts of all the talks, as well as discussion sessions held during the seminar. We thank all the speakers and attendees for their active participation and look forward to attending and organizing more such events in the future.

## 2 Table of Contents

### Executive Summary

<i>S. Akshay, Bernd Finkbeiner, Kuldeep S. Meel, and Ruzica Piskac</i> . . . . .	85
--	----

### Overview of Talks

To Assume, Or Not To Assume <i>Ashwani Anand</i> . . . . .	89
LTLf Model Checking <i>Suguman Bansal</i> . . . . .	90
Formal XAI via Syntax-Guided Synthesis <i>Katrine Bjørner</i> . . . . .	90
Programming by example for end user tasks and the use of LLMs <i>José Cambronero</i> . . . . .	91
Boolean Functional Synthesis: A Quick Tour <i>Supratik Chakraborty</i> . . . . .	91
Symbolic Fixpoint Techniques for Logical LTL Games <i>Deepak D'Souza</i> . . . . .	92
On the compilation of non-CNF systems of constraints (or, your weekly dose of knowledge compilation) <i>Alexis de Colnet</i> . . . . .	92
Synthesis of Infinite-State Reactive Systems (and why it needs functional synthesis for theories beyond Boolean) <i>Rayna Dimitrova</i> . . . . .	92
A Semi-Gentle Introduction to Reactive Synthesis <i>Rüdiger Ehlers</i> . . . . .	93
On Dependent Variables in Reactive Synthesis <i>Dror Fried</i> . . . . .	93
Exploring Connections between Automated Reasoning and Synthesis <i>Mikoláš Janota</i> . . . . .	94
Stochastic Boolean Satisfiability: Recent Developments and Connection to Functional Synthesis <i>Jie-Hong Roland Jiang</i> . . . . .	95
The Unreasonable Effectiveness of Classical Automated Reasoning in Quantum Computing <i>Alfons Laarman and Jingyi Mei</i> . . . . .	96
Reactive Synthesis modulo Theories using Abstraction Refinement <i>Benedikt Maderbacher</i> . . . . .	97
Boosting Definability Bipartition Computation using SAT Witnesses <i>Pierre Marquis</i> . . . . .	98
A Flock of Birds: Owl & Strix <i>Tobias Megendorfer</i> . . . . .	98

Pre-condition and Program Synthesis for Polynomial Specifications over Integers <i>Govind Rajanbabu, S. Akshay, and Supratik Chakraborty</i> . . . . .	99
Synthesis of Semantic Actions in Attribute Grammars <i>Subhajit Roy</i> . . . . .	99
Reactive Program Synthesis Modulo LLM Code Generation <i>Mark Santolucito</i> . . . . .	100
A Short Introduction to Inductive Functional Programming <i>Ute Schmid</i> . . . . .	100
Oracle-Guided Inductive Synthesis, Learning Theory, and LLMs <i>Sanjit A. Seshia</i> . . . . .	101
An Approximate Skolem Function Counter <i>Arijit Shaw</i> . . . . .	101
Counterexample-Guided DQBF Solving <i>Friedrich Slivovsky</i> . . . . .	102
A problem with functional synthesis <i>Mate Soos, Supratik Chakraborty, and Kuldeep S. Meel</i> . . . . .	102
Reactive synthesis via parity and Rabin games <i>K. S. Thejaswini</i> . . . . .	103
On the Power of LTLf in Reactive Synthesis <i>Shufang Zhu</i> . . . . .	103
<b>Working groups</b>	
Benchmarking (and LLMs) <i>José Cambroneiro, Johannes Klaus Fichte, Benedikt Maderbacher, Tobias Meggen-</i> <i>dorfer, Ruzica Piskac, and Mark Santolucito</i> . . . . .	104
Developing a Computational Theory for Learning Functions from Relations <i>Kuldeep S. Meel, Dror Fried, Alfons Laarman, Sanjit A. Seshia, and Mate Soos</i> . .	105
<b>Panel discussions</b>	
Reactive Synthesis and Model Counting Competitions <i>Guillermo A. Pérez and Johannes Klaus Fichte</i> . . . . .	105
<b>Participants</b> . . . . .	107

### 3 Overview of Talks

#### 3.1 To Assume, Or Not To Assume

Ashwani Anand (*MPI-SWS – Kaiserslautern, DE*)

**License** © Creative Commons BY 4.0 International license  
© Ashwani Anand

**Joint work of** Ashwani Anand, Kaushik Mallik, Satya Prakash Nayak, Anne-Kathrin Schmuck

**Main reference** Ashwani Anand, Anne-Kathrin Schmuck, Satya Prakash Nayak: “Contract-Based Distributed Logical Controller Synthesis”, in Proc. of the 27th ACM International Conference on Hybrid Systems: Computation and Control, HSCC 2024, Hong Kong SAR, China, May 14-16, 2024, pp. 11:1–11:11, ACM, 2024.

**URL** <https://doi.org/10.1145/3641513.3650123>

Reactive synthesis techniques assume that the environment acts adversarially. However, in many real-life scenarios, the environment might not work antagonistically. We solve the problem of automatically computing a new class of environment assumptions in two-player turn-based finite graph games which characterize an “adequate cooperation” needed from the environment to allow the system player to win [1]. Given an  $\omega$ -regular winning condition  $\Phi$  for the system player, we compute an  $\omega$ -regular assumption  $\Psi$  for the environment player, such that (i) every environment strategy compliant with  $\Psi$  allows the system to fulfill  $\Phi$  (sufficiency), (ii)  $\Psi$  can be fulfilled by the environment for every strategy of the system (implementability), and (iii)  $\Psi$  does not prevent any cooperative strategy choice (permissiveness).

For parity games, which are canonical representations of  $\omega$ -regular games, we present a polynomial-time algorithm for the symbolic computation of adequately permissive assumptions and show that our algorithm runs faster and produces better assumptions than existing approaches – both theoretically and empirically. To the best of our knowledge, for  $\omega$ -regular games, we provide the first algorithm to compute sufficient and implementable environment assumptions that are also permissive.

In the second part of the talk, we apply the lessons learned to strategies computation [2], and negotiations between multiple agents [3].

#### References

- 1 Ashwani Anand, Kaushik Mallik, Satya Prakash Nayak, and Anne-Kathrin Schmuck. “Computing Adequately Permissive Assumptions for Synthesis.” In *Tools and Algorithms for the Construction and Analysis of Systems*, edited by Sriram Sankaranarayanan and Natasha Sharygina, 211–228. Cham: Springer Nature Switzerland, 2023.
- 2 Ashwani Anand, Satya Prakash Nayak, and Anne-Kathrin Schmuck. “Synthesizing Permissive Winning Strategy Templates for Parity Games.” In *Computer Aided Verification – 35th International Conference, CAV 2023, Paris, France, July 17-22, 2023, Proceedings, Part I*, edited by Constantin Enea and Akash Lal, 13964:436–458. Lecture Notes in Computer Science. Springer, 2023. [https://doi.org/10.1007/978-3-031-37706-8\\_22](https://doi.org/10.1007/978-3-031-37706-8_22).
- 3 Ashwani Anand, Anne-Kathrin Schmuck, and Satya Prakash Nayak. “Contract-Based Distributed Logical Controller Synthesis.” In *Proceedings of the 27th ACM International Conference on Hybrid Systems: Computation and Control*, 1–11. HSCC ’24. New York, NY, USA: Association for Computing Machinery, 2024. <https://doi.org/10.1145/3641513.3650123>.

### 3.2 LTLf Model Checking

*Suguman Bansal (Georgia Institute of Technology – Atlanta, US)*

**License**  Creative Commons BY 4.0 International license  
© Suguman Bansal

**Joint work of** Suguman Bansal, Yong Li, Lucas M. Tabajara, Moshe Y. Vardi, Andrew Wells

**Main reference** Suguman Bansal, Yong Li, Lucas M. Tabajara, Moshe Y. Vardi, Andrew M. Wells: “Model Checking Strategies from Synthesis over Finite Traces”, in Proc. of the Automated Technology for Verification and Analysis – 21st International Symposium, ATVA 2023, Singapore, October 24-27, 2023, Proceedings, Part I, Lecture Notes in Computer Science, Vol. 14215, pp. 227–247, Springer, 2023.

**URL** [https://doi.org/10.1007/978-3-031-45329-8\\_11](https://doi.org/10.1007/978-3-031-45329-8_11)

The innovations in reactive synthesis from Linear Temporal Logics over finite traces (LTLf) will be amplified by the ability to verify the correctness of the strategies generated by LTLf synthesis tools. This motivates our work on LTLf model checking. LTLf model checking, however, is not straightforward. The strategies generated by LTLf synthesis may be represented using terminating transducers or non-terminating transducers where executions are of finite-but-unbounded length or infinite length, respectively. For synthesis, there is no evidence that one type of transducer is better than the other since they both demonstrate the same complexity and similar algorithms.

In this work, we show that for model checking, the two types of transducers are fundamentally different. Our central result is that LTLf model checking of non-terminating transducers is exponentially harder than that of terminating transducers. We show that the problems are EXPSpace-complete and PSpace-complete, respectively. Hence, considering the feasibility of verification, LTLf synthesis tools should synthesize terminating transducers. This is, to the best of our knowledge, the first evidence to use one transducer over the other in LTLf synthesis.

### 3.3 Formal XAI via Syntax-Guided Synthesis

*Katrine Bjørner (New York University, US)*

**License**  Creative Commons BY 4.0 International license  
© Katrine Bjørner

**Joint work of** Katrine Bjørner, Samuel Judson, Filip Cano Córdoba, Drew Goldman, Nicholas Shoemaker, Ruzica Piskac, Bettina Könighofer

**Main reference** Katrine Bjørner, Samuel Judson, Filip Cano Córdoba, Drew Goldman, Nicholas Shoemaker, Ruzica Piskac, Bettina Könighofer: “Formal XAI via Syntax-Guided Synthesis”, in Proc. of the Bridging the Gap Between AI and Reality – First International Conference, AISoLA 2023, Crete, Greece, October 23-28, 2023, Proceedings, Lecture Notes in Computer Science, Vol. 14380, pp. 119–137, Springer, 2023.

**URL** [https://doi.org/10.1007/978-3-031-46002-9\\_7](https://doi.org/10.1007/978-3-031-46002-9_7)

We propose a novel application of syntax-guided synthesis to find symbolic representations of a model’s decision-making process, designed for easy comprehension and validation by humans. Our approach takes input-output samples from complex machine learning models, such as deep neural networks, and automatically derives interpretable mimic programs. A mimic program precisely imitates the behavior of an opaque model over the provided data. We discuss various types of grammars that are well-suited for computing mimic programs for tabular and image input data.

Our experiments demonstrate the potential of the proposed method: we successfully synthesized mimic programs for neural networks trained on the MNIST and the Pima Indians diabetes data sets. All experiments were performed using the SMT-based cvc5 synthesis tool.

### 3.4 Programming by example for end user tasks and the use of LLMs

*José Cambronero (Microsoft – Redmond, US)*

**License** © Creative Commons BY 4.0 International license  
© José Cambronero

**Joint work of** José Cambronero, Mukul Singh, Gust Verbruggen, Sumit Gulwani, Vu Le

Programming by example (PBE) allows users with little to no formal computation experience to carry out tasks by providing simple demonstrations (e.g. input/output-based examples). In practice, PBE has found considerable industrial uptake, particularly in end-user environments like spreadsheet software (e.g. Microsoft Excel, Google Sheets). In this talk, I'll present a recent project on learning data-dependent formatting rules in Excel from examples. We'll then discuss how PBE in this domain can be extended to also incorporate multimodal specifications, by supporting use of natural language. Using this as a segue into combining symbolic and neural methods, I'll discuss recent work from the field that uses LLMs and may provide ideas for nice collaborations between formal reasoning and popular LLM-based approaches.

### 3.5 Boolean Functional Synthesis: A Quick Tour

*Supratik Chakraborty (Indian Institute of Technology Bombay – Mumbai, IN)*

**License** © Creative Commons BY 4.0 International license  
© Supratik Chakraborty

**Joint work of** S. Akshay, Aman Bansal, Supratik Chakraborty, Priyanka Golia, Kuldeep Meel, Subhajit Roy, Preey Shah, Shetal Shah, Friedrich Slivovsky

Given a Boolean relational specification  $\varphi(X, Y)$  over input variables  $X$  and output variables  $Y$ , Boolean functional synthesis concerns finding Skolem functions  $F(X)$  for  $Y$  such that  $\exists Y \varphi(X, Y)$  is semantically equivalent to  $\varphi(X, F(X))$ . In this talk, we introduce the problem, survey some earlier results and then take a deeper dive into two solution approaches that have shown promise in recent years. Specifically, we discuss the guess-check-repair paradigm for synthesizing Skolem functions, and also present a knowledge compilation based approach for Boolean functional synthesis. Finally, we conclude with some perspectives on future research in this area. The talk is based on work reported in [1, 2, 3, 4, 5].

#### References

- 1 S. Akshay, Supratik Chakraborty, Shubham Goel, Sumith Kulal, Shetal Shah: Boolean Functional Synthesis: Hardness and Practical Algorithms, *Formal Methods Syst. Des.* 57(1): 53-86 (2021).
- 2 S. Akshay, S. Chakraborty, S. Shah: Tractable Representations for Boolean Functional Synthesis, *Annals of Mathematics and Artificial Intelligence*, 1-46, 2023.
- 3 Preey Shah, Aman Bansal, S. Akshay, Supratik Chakraborty: A Normal Form Characterization for Efficient Boolean Skolem Function Synthesis, *LICS 2021*: 1-13.
- 4 Priyanka Golia, Friedrich Slivovsky, Subhajit Roy, Kuldeep S. Meel: Engineering an Efficient Boolean Functional Synthesis Engine, *ICCAD 2021*: 1-9.
- 5 Priyanka Golia, Subhajit Roy, Kuldeep S. Meel: Manthan: A Data-Driven Approach for Boolean Function Synthesis, *CAV (2) 2020*: 611-633.

### 3.6 Symbolic Fixpoint Techniques for Logical LTL Games

*Deepak D’Souza (Indian Institute of Science – Bangalore, IN)*

**License** © Creative Commons BY 4.0 International license  
© Deepak D’Souza

**Joint work of** Deepak D’Souza, Stanly John Samuel, Komondoor V. Raghavan

**Main reference** Stanly Samuel, Deepak D’Souza, Raghavan Komondoor: “Symbolic Fixpoint Algorithms for Logical LTL Games”, in Proc. of the 38th IEEE/ACM International Conference on Automated Software Engineering, ASE 2023, Luxembourg, September 11-15, 2023, pp. 698–709, IEEE, 2023.

**URL** <https://doi.org/10.1109/ASE56229.2023.00212>

We consider the problem of synthesizing strategies in logically-specified infinite-state two-player games with LTL winning conditions. We lift classical fixpoint algorithms for synthesizing strategies in finite-states games, to our setting. Our evaluation of these algorithms show that they compare well with earlier techniques based on template-based logical synthesis and abstraction-refinement, on benchmarks from the literature.

This is joint work with Stanly Samuel and K V Raghavan.

### 3.7 On the compilation of non-CNF systems of constraints (or, your weekly dose of knowledge compilation)

*Alexis de Colnet (TU Wien, AT)*

**License** © Creative Commons BY 4.0 International license  
© Alexis de Colnet

Knowledge compilers often take as inputs a CNF formula and construct an equivalent Boolean circuit with specific properties. Generally, the size of the output circuit increases exponentially. However, for some families of CNF formulas, one can exploit the structure of the formulas to compile them efficiently. In this talk, I first give a general overview of knowledge compilation and of the circuits that knowledge compilers construct. Then, I present results on the compilation of non-CNF inputs. Seeing CNF as systems of constraints, where every constraint is a clause, I explain how positive results on the compilation of CNF with a certain structure can be extended to more general systems of constraints.

### 3.8 Synthesis of Infinite-State Reactive Systems (and why it needs functional synthesis for theories beyond Boolean)

*Rayna Dimitrova (CISPA – Saarbrücken, DE)*

**License** © Creative Commons BY 4.0 International license  
© Rayna Dimitrova

**Joint work of** Philippe Heim, Rayna Dimitrova

**Main reference** Philippe Heim, Rayna Dimitrova: “Solving Infinite-State Games via Acceleration”, Proc. ACM Program. Lang., Vol. 8(POPL), pp. 1696–1726, 2024.

**URL** <https://doi.org/10.1145/3632899>


Infinite-state games are a commonly used model for the synthesis of reactive systems with unbounded data domains. Symbolic methods for solving such games need to be able to construct intricate arguments to establish the existence of winning strategies. Furthermore, the synthesis of the resulting reactive system implementations necessitates the use of functional synthesis for theories beyond Boolean. In this talk, I will present a recent symbolic approach



for the synthesis of infinite-state reactive systems, called attractor acceleration, which employs ranking arguments to improve the convergence of symbolic game-solving algorithms. I will then discuss the application and the challenges for functional synthesis in this context.

### 3.9 A Semi-Gentle Introduction to Reactive Synthesis


*Rüdiger Ehlers (TU Clausthal, DE)*

License  Creative Commons BY 4.0 International license  
© Rüdiger Ehlers

Reactive synthesis is the process of computing correct-by-construction finite-state controllers from temporal logic specifications. In this tutorial, we have a look at the basic concepts that underlie current reactive synthesis approaches. We discuss the topic on a fairly technical level in order to highlight the connections to functional and Boolean synthesis.

### 3.10 On Dependent Variables in Reactive Synthesis


*Dror Fried (The Open University of Israel – Ra’anana, IL)*

License  Creative Commons BY 4.0 International license  
© Dror Fried

Given a Linear Temporal Logic (LTL) formula over input and output variables, reactive synthesis requires us to design a deterministic Mealy machine that gives the values of outputs at every time step for every sequence of inputs, such that the LTL formula is satisfied. In this paper, we investigate the notion of dependent variables in the context of reactive synthesis. Inspired by successful pre-processing techniques in Boolean functional synthesis, we define dependent variables in reactive synthesis as output variables that are uniquely assigned, given an assignment to all other variables and the history so far. We describe an automata-based approach for finding a set of dependent variables. Using this, we show that dependent variables are surprisingly common in reactive synthesis benchmarks. Next, we develop a novel synthesis framework that exploits dependent variables to construct an overall synthesis solution. By implementing this framework using the widely used library Spot, we show that reactive synthesis that exploits dependent variables can solve some problems beyond the reach of existing techniques. Furthermore, we observe that among benchmarks with dependent variables, if the count of non-dependent variables is low ( $\leq 3$  in our experiments), our method outperforms state-of-the-art tools for synthesis.

### 3.11 Exploring Connections between Automated Reasoning and Synthesis

Mikoláš Janota (Czech Technical University – Prague, CZ)

License  Creative Commons BY 4.0 International license  
© Mikoláš Janota

In this talk we explore the connections between synthesis and automated reasoning techniques. Generally, synthesis, from logic perspective, is formalized as solving a formula of the following form.

$$\exists f \forall \mathbf{x}. P[f, \mathbf{x}]$$

where  $P$  is a predicate parametrized by a vector of variables  $\mathbf{x}$  and an unknown function  $f$ . Typically, the (second order) quantifier  $\exists f$  it is omitted; in particular in Satisfiability Modulo Theories (SMT), where  $f$  functions are implicitly quantified existentially.

Many approaches use solvers in a black-box fashion by assuming a certain *template* for  $f$ , such as linear, quadratic etc. [9]. Then, the synthesis problem is formulated as an SMT problem that search as for the parameters (coefficients) of the template. Interestingly, such approach can also be used to search for *all* the possible  $f$  [1].

Some approaches integrate more deeply with the solver. A powerful technique is *deskolemization* of  $f$  (as inverse of skolemization), which is possible, if  $f$  is always applied to the same tuple of arguments everywhere in  $P$ . In the literature, such specifications are referred to as *single-invocation properties* [5, 8]. An example of such property would be  $\forall x_1 x_2. f(x_1, x_2) > x_1 \wedge f(x_1, x_2) > x_2$ , which would be deskolemized as  $\forall x_1 x_2 \exists z. z > x_1 \wedge z > x_2$ .

For deskolemized version of the specification,  $f$  can be that synthesized by *quantifier elimination* (QE) if the formula is in a theory that admits QE, such as linear real/integer arithmetic [7, 2], cf. [5, 6]. Alternatively, Reynolds et al. [8] synthesize  $f$  by inspecting the SMT refutation (proof). Hozzová et al. [3] synthesize  $f$  in the setting of first order logic (FOL), again from the proof, which relies on explicit axiomatization of any theory that may be used.

Specifications going beyond the single-invocation property fragment maybe tackled by embedding the language of possible solutions into the solver as than algebraic datatype [8]. More recent research shows that refutations containing mathematical induction enable synthesizing recursive functions [4].


#### References

- 1 Chad E. Brown, Mikoláš Janota, and Mirek Olšák. Symbolic computation for all the fun. In *Satisfiability Checking and Symbolic Computation*, 2024. <https://ceur-ws.org/Vol-3717/paper6.pdf>.
- 2 David C Cooper. Theorem proving in arithmetic without multiplication. *Machine intelligence*, 7(91-99):300, 1972.
- 3 Petra Hozzová, Laura Kovács, Chase Norman, and Andrei Voronkov. Program synthesis in saturation. In *CADE*, 2023.
- 4 Petra Hozzová, Daneshvar Amrollahi, Márton Hajdu, Laura Kovács, Andrei Voronkov, and Eva Maria Wagner. Synthesis of recursive programs in saturation. In *International Joint Conference on Automated Reasoning IJCAR*, 2024.
- 5 Swen Jacobs and Viktor Kuncak. Towards complete reasoning about axiomatic specifications. In *Verification, Model Checking, and Abstract Interpretation*, 2011.
- 6 Viktor Kuncak, Mikael Mayer, Ruzica Piskac, and Philippe Suter. Software synthesis procedures. *Communications of the ACM*, 55(2):103–111, February 2012.

- 7 Rüdiger Loos and Volker Weispfenning. Applying linear quantifier elimination. *Comput. J.*, 36(5):450–462, 1993.
- 8 Andrew Reynolds, Viktor Kuncak, Cesare Tinelli, Clark W. Barrett, and Morgan Deters. Refutation-based synthesis in SMT. *Formal Methods Syst. Des.*, 55(2):73–102, 2019.
- 9 Saurabh Srivastava, Sumit Gulwani, and Jeffrey S. Foster. Template-based program verification and program synthesis. *Int. J. Softw. Tools Technol. Transf.*, 15(5-6):497–518, 2013.

### 3.12 Stochastic Boolean Satisfiability: Recent Developments and Connection to Functional Synthesis

*Jie-Hong Roland Jiang (National Taiwan University – Taipei, TW)*

License  Creative Commons BY 4.0 International license  
© Jie-Hong Roland Jiang

Joint work of Jie-Hong Roland Jiang, Pei-Wei Chen, Che Cheng, Yu-Wei Fan, Cheng-Han Hsieh, Yu-Ching Huang, Nian-Ze Lee, Yun-Rong Luo, Christoph Scholl, Kuan-Hua Tu, Hao-Ren Wang, Yen-Shi Wang

Stochastic Boolean Satisfiability (SSAT) generalizes quantified Boolean formulas (QBFs) by allowing quantification over random variables. It is often referred to as games against nature and has applications in making decisions or optimizing under uncertainty. This talk will introduce SSAT, its recent developments, and its connection to Boolean functional synthesis.

#### References

- 1 Yu-Wei Fan, Jie-Hong R. Jiang: Unifying Decision and Function Queries in Stochastic Boolean Satisfiability, AAAI 2024: 7995-8003.
- 2 Yun-Rong Luo, Che Cheng, Jie-Hong R. Jiang: A Resolution Proof System for Dependency Stochastic Boolean Satisfiability, *J. Autom. Reason.* 67(3): 26 (2023).
- 3 Che Cheng, Jie-Hong R. Jiang: Lifting (D)QBF Preprocessing and Solving Techniques to (D)SSAT, AAAI 2023: 3906-3914.
- 4 Yu-Wei Fan, Jie-Hong R. Jiang: SharpSSAT: A Witness-Generating Stochastic Boolean Satisfiability Solver, AAAI 2023: 3949-3958.
- 5 Jie-Hong R. Jiang: Second-Order Quantified Boolean Logic, AAAI 2023: 4007-4015.
- 6 Cheng-Han Hsieh, Jie-Hong R. Jiang: Encoding Probabilistic Graphical Models into Stochastic Boolean Satisfiability, *IJCAI* 2022: 1834-1842.
- 7 Hao-Ren Wang, Kuan-Hua Tu, Jie-Hong R. Jiang, Christoph Scholl: Quantifier Elimination in Stochastic Boolean Satisfiability, *SAT* 2022: 23:1-23:17.
- 8 Pei-Wei Chen, Yu-Ching Huang, Jie-Hong R. Jiang: A Sharp Leap from Quantified Boolean Formula to Stochastic Boolean Satisfiability Solving, AAAI 2021: 3697-3706.
- 9 Nian-Ze Lee, Jie-Hong R. Jiang: Dependency Stochastic Boolean Satisfiability: A Logical Formalism for NEXPTIME Decision Problems with Uncertainty, AAAI 2021: 3877-3885.
- 10 Nian-Ze Lee, Yen-Shi Wang, Jie-Hong R. Jiang: Solving Exist-Random Quantified Stochastic Boolean Satisfiability via Clause Selection, *IJCAI* 2018: 1339-1345.
- 11 Nian-Ze Lee, Yen-Shi Wang, Jie-Hong R. Jiang: Solving Stochastic Boolean Satisfiability under Random-Exist Quantification, *IJCAI* 2017: 688-694.

### 3.13 The Unreasonable Effectiveness of Classical Automated Reasoning in Quantum Computing

*Alfons Laarman (Leiden University, NL) and Jingyi Mei (Leiden University, NL)*

**License** © Creative Commons BY 4.0 International license

© Alfons Laarman and Jingyi Mei

**Joint work of** Jingyi Mei, Arend-Jan Quist, Alejandro Villoria, Sebastiaan Brand, Dimitrios Thanos, Tim Coopmans, Alfons Laarman

**Main reference** Jingyi Mei, Tim Coopmans, Marcello M. Bonsangue, Alfons Laarman: “Equivalence Checking of Quantum Circuits by Model Counting”, in Proc. of the Automated Reasoning – 12th International Joint Conference, IJCAR 2024, Nancy, France, July 3-6, 2024, Proceedings, Part II, Lecture Notes in Computer Science, Vol. 14740, pp. 401–421, Springer, 2024.

**URL** [https://doi.org/10.1007/978-3-031-63501-4\\_21](https://doi.org/10.1007/978-3-031-63501-4_21)

In this talk, we will show that existing classical automated reasoning methods perform exceedingly well for computationally hard problems in quantum computing and physics. In particular, we demonstrate a linear-length #SAT encoding of the simulation and equivalence checking of universal quantum circuits. An implementation of this method, called Quokka#, outcompetes other state-of-the-art approaches using an off-the-shelf #SAT solver that supports negative weights (GPMC). While decision diagrams offer a viable alternative, we unveil their inherent limitations stemming from their inability to represent the prevalent stabilizer states. This limitation is particularly noteworthy considering the efficient classical simulatability of circuits generating such states. To address this constraint, we introduce Local Invertible Map Decision Diagrams (LIMDDs), which offer exponential improvements in succinctness compared to the combination of stabilizer formalism and existing decision diagrams. Finally, we illustrate how these findings hold relevance beyond quantum computing by translating them back to the domain of quantum physics. To achieve this, we build upon Darwiche and Marquis’ seminal “knowledge compilation map” approach, by pioneering a knowledge compilation map for quantum information. This map juxtaposes various decision diagrams against tensor networks and Boltzmann machines, two formalisms extensively utilized in physics to address quantum-hard problems such as simulating many-body systems and determining their ground energy. Our results underscore the significant potential of existing automated reasoning methods in both quantum computing and physics domains.


#### References

- 1 Coecke, B., Duncan, R.: Interacting Quantum Observables: Categorical Algebra and Diagrammatics. *New Journal of Physics* 13(4), 043016 (Apr 2011), <http://arxiv.org/abs/0906.4725>, arXiv:0906.4725 [quant-ph]
- 2 Vrudhula, S.B.K., Pedram, M., Lai, Y.T.: Edge Valued Binary Decision Diagrams, pp. 109–132. Springer US (1996)
- 3 Burgholzer, L., Wille, R.: Advanced equivalence checking for quantum circuits. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* 40(9), 1810–1824 (2020)
- 4 Zulehner, A., Wille, R.: Advanced simulation of quantum computations. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* 38(5), 848–859 (2019)
- 5 Tafertshofer, P., Pedram, M.: Factored edge-valued binary decision diagrams. *Formal Methods in System Design* 10(2), 243–270 (1997)
- 6 Miller, D.M., Thornton, M.A.: QMDD: A decision diagram structure for reversible and quantum circuits. *36th International Symposium on Multiple-Valued Logic (ISMVL’06)* pp. 30–30 (2006)
- 7 Viamontes, G.F., Markov, I.L., Hayes, J.P.: Quantum circuit simulation. Springer Science and Business Media (2009)

- 8 Mei, J., Coopmans, T., Bonsangue, M., Laarman, A. (2024, July). Equivalence checking of quantum circuits by model counting. In International Joint Conference on Automated Reasoning (pp. 401-421). Cham: Springer Nature Switzerland.
- 9 Mei, J., Bonsangue, M., Laarman, A. (2024). Simulating Quantum Circuits by Model Counting. to appear in CAV 2024, available as arXiv preprint arXiv:2403.07197.
- 10 Quist, A.J., Laarman, A.: Optimizing quantum space using spooky pebble games. In: International Conference on Reversible Computation. pp. 134–149. Springer (2023)
- 11 Thanos, D., Coopmans, T., Laarman, A.: Fast equivalence checking of quantum circuits of Clifford gates. In: Andr e, ´E., Sun, J. (eds.) Automated Technology for Verification and Analysis. pp. 199–216. Springer Nature Switzerland, Cham (2023)
- 12 Villoria, A., Basold, H., Laarman, A.: Enriching diagrams with algebraic operations. arXiv preprint arXiv:2310.11288 (2023)
- 13 Vinkhuijzen, L., Coopmans, T., Elkouss, D., Dunjko, V., Laarman, A.: LIMDD: A decision diagram for simulation of quantum computing including stabilizer states. Quantum 7, 1108 (2023), <https://doi.org/10.22331/q-2023-09-11-1108>
- 14 Vinkhuijzen, L., Coopmans, T., Laarman, A.: A knowledge compilation map for quantum information. arXiv preprint arXiv:2401.01322 (2024), <https://doi.org/10.48550/arXiv.2401.01322>
- 15 Vinkhuijzen, L., Grurl, T., Hillmich, S., Brand, S., Wille, R., Laarman, A.: Efficient implementation of LIMDDs for quantum circuit simulation. In: International Symposium on Model Checking of Software (SPIN) (2023)
- 16 Brand, S., Coopmans, T., Laarman, A.: Quantum graph-state synthesis with SAT. Proceedings of the 14th International Workshop on Pragmatics of SAT (2023)
- 17 Rennela, M., Brand, S., Laarman, A., Dunjko, V.: Hybrid divide-and-conquer approach for tree search algorithms. Quantum 7, 959 (2023)

### 3.14 Reactive Synthesis modulo Theories using Abstraction Refinement

*Benedikt Maderbacher (TU Graz, AT)*

**License**  Creative Commons BY 4.0 International license  
 © Benedikt Maderbacher

**Joint work of** Benedikt Maderbacher, Roderick Bloem


**Main reference** Benedikt Maderbacher, Roderick Bloem: “Reactive Synthesis Modulo Theories using Abstraction Refinement”, in Proc. of the 22nd Formal Methods in Computer-Aided Design, FMCAD 2022, Trento, Italy, October 17-21, 2022, pp. 315–324, IEEE, 2022.

**URL** [https://doi.org/10.34727/2022/ISBN.978-3-85448-053-2\\_38](https://doi.org/10.34727/2022/ISBN.978-3-85448-053-2_38)

Temporal stream logic modulo theories (TSL-T) is used to specify the behavior of infinite state reactive systems. We present a refinement based synthesis method that works using LTL synthesis and SMT solving. First, a LTL underapproximation is computed and given to a LTL synthesis tool. In case this is unrealizable the created counter-strategy is analyzed for inconsistencies with the theory. New assumptions and predicates are added to the specification to rule out the counter-strategy and the LTL synthesis is run again. If the problem becomes realizable a program satisfying the original specification is extracted.

### 3.15 Boosting Definability Bipartition Computation using SAT Witnesses

*Pierre Marquis (University of Artois/CNRS – Lens, FR)*

**License**  Creative Commons BY 4.0 International license  
 © Pierre Marquis

**Joint work of** Jean-Marie Lagniez, Pierre Marquis

**Main reference** Jean-Marie Lagniez, Pierre Marquis: “Boosting Definability Bipartition Computation Using SAT Witnesses”, in Proc. of the Logics in Artificial Intelligence – 18th European Conference, JELIA 2023, Dresden, Germany, September 20-22, 2023, Proceedings, Lecture Notes in Computer Science, Vol. 14281, pp. 697–711, Springer, 2023.

**URL** [https://doi.org/10.1007/978-3-031-43619-2\\_47](https://doi.org/10.1007/978-3-031-43619-2_47)

Bipartitioning the set of variables  $\text{Var}(\Sigma)$  of a propositional formula  $\Sigma$  w.r.t. definability consists in pointing out a bipartition  $\langle I, O \rangle$  of  $\text{Var}(\Sigma)$  such that  $\Sigma$  defines the variables of  $O$  (outputs) in terms of the variables in  $I$  (inputs), i.e., for every  $o \in O$ , there exists a formula  $\Phi_o$  over  $I$  such that  $o \Leftrightarrow \Phi_o$  is a logical consequence of  $\Sigma$ . The existence of  $\Phi_o$  given  $o$ ,  $I$ , and  $\Sigma$  is a coNP-complete problem, and as such, it can be addressed in practice using a SAT solver. From a computational perspective, definability bipartitioning has been shown as a valuable preprocessing technique for model counting, a key task for a number of AI problems involving probabilities. To maximize the benefits offered by such a preprocessing, one is interested in deriving subset-minimal bipartitions in terms of input variables, i.e., definability bipartitions  $\langle I, O \rangle$  such that for every  $i \in I$ ,  $\langle I \setminus \{i\}, O \cup \{i\} \rangle$  is not a definability bipartition. We show how the computation of subset-minimal bipartitions can be boosted by leveraging not only the decisions furnished by SAT solvers (as done in previous approaches), but also the SAT witnesses (models and cores) justifying those decisions.

### 3.16 A Flock of Birds: Owl & Strix

*Tobias Meggendorfer (Lancaster University Leipzig, DE)*

**License**  Creative Commons BY 4.0 International license  
 © Tobias Meggendorfer

**Joint work of** Tobias Meggendorfer, Javier Esparza, Jan Křetínský, Michael Luttenberger, Salomon Sickert, Philipp J. Meyer

**Main reference** Javier Esparza, Jan Křetínský, Salomon Sickert: “A Unified Translation of Linear Temporal Logic to  $\omega$ -Automata”, J. ACM, Vol. 67(6), pp. 33:1–33:61, 2020.

**URL** <https://doi.org/10.1145/3417995>

In this talk, I briefly outline the theoretical and practical advances that together form the foundation of Owl and Strix, state-of-the-art tools for LTL to automata translation and LTL synthesis, respectively.

This includes a large body of work, a small selection follows:

- Unified translation (JACM): <https://doi.org/10.1145/3417995>
- One theorem to rule them all (LICS): <https://doi.org/10.1145/3209108.3209161>
- Owl tool paper: [https://doi.org/10.1007/978-3-030-01090-4\\_34](https://doi.org/10.1007/978-3-030-01090-4_34)
- Strix tool paper: [https://doi.org/10.1007/978-3-319-96145-3\\_31](https://doi.org/10.1007/978-3-319-96145-3_31)

The small list above is the culmination of about a dozen papers, see the respective cites within for more details.

### 3.17 Pre-condition and Program Synthesis for Polynomial Specifications over Integers

*Govind Rajanbabu (Indian Institute of Technology Bombay – Mumbai, IN), S. Akshay (Indian Institute of Technology Bombay – Mumbai, IN), Supratik Chakraborty (Indian Institute of Technology Bombay – Mumbai, IN)*

**Joint work of** Govind Rajanbabu, S. Akshay, Amir Kafshdar Goharshady, Supratik Chakraborty, Harshit Jitendra Motwani, Sai Teja Varanasi

**License** © Creative Commons BY 4.0 International license  
© Govind Rajanbabu, S. Akshay, and Supratik Chakraborty

**Main reference** S. Akshay, Supratik Chakraborty, Amir Kafshdar Goharshady, R. Govind, Harshit J. Motwani, Sai Teja Varanasi: “Automated Synthesis of Decision Lists for Polynomial Specifications over Integers”, in Proc. of the LPAR 2024: Proceedings of 25th Conference on Logic for Programming, Artificial Intelligence and Reasoning, Port Louis, Mauritius, May 26-31, 2024, EPiC Series in Computing, Vol. 100, pp. 484–502, EasyChair, 2024.

**URL** <https://doi.org/10.29007/NJPH>

In this talk, we will look at the problem of synthesizing both the program and pre-condition, when the post-condition is given as Boolean combination of polynomial inequalities and variables take integral values over a bounded region. The problem does not have a sub-exponential time procedure under Exponential Time Hypothesis. We will discuss an approach that is more efficient than naive enumeration by exploiting results from algebraic geometry.

### 3.18 Synthesis of Semantic Actions in Attribute Grammars

*Subhajit Roy (Indian Institute of Technology Kanpur, IN)*

**License** © Creative Commons BY 4.0 International license  
© Subhajit Roy

**Joint work of** Subhajit Roy, Pankaj Kumar Kalita, Miriyala Jeevan Kumar

**Main reference** Pankaj Kumar Kalita, Miriyala Jeevan Kumar, Subhajit Roy: “Synthesis of Semantic Actions in Attribute Grammars”, in Proc. of the 22nd Formal Methods in Computer-Aided Design, FMCAD 2022, Trento, Italy, October 17-21, 2022, pp. 304–314, IEEE, 2022.

**URL** [https://doi.org/10.34727/2022/ISBN.978-3-85448-053-2\\_37](https://doi.org/10.34727/2022/ISBN.978-3-85448-053-2_37)

Attribute grammars allow the association of semantic actions to the production rules in context-free grammars, providing a simple yet effective formalism to define the semantics of a language. However, drafting the semantic actions can be tricky and a large drain on developer time. In this work, we propose a synthesis methodology to automatically infer the semantic actions from a set of examples associating strings to their meanings. We also propose a new coverage metric, derivation coverage. We use it to build a sampler to effectively and automatically draw strings to drive the synthesis engine. We build our ideas into our tool, PĀNINI, and empirically evaluate it on twelve benchmarks, including a forward differentiation engine, an interpreter over a subset of Java bytecode, and a mini-compiler for C language to two-address code. Our results show that PĀNINI scales well with the number of actions to be synthesized and the size of the context-free grammar, significantly outperforming simple baselines.

### 3.19 Reactive Program Synthesis Modulo LLM Code Generation

*Mark Santolucito (Barnard College, Columbia University – New York, US)*

**License** © Creative Commons BY 4.0 International license  
© Mark Santolucito

**Main reference** Raven Rothkopf, Hannah Tongxin Zeng, Mark Santolucito: “Enforcing Temporal Constraints on Generative Agent Behavior with Reactive Synthesis”, CoRR, Vol. abs/2402.16905, 2024.

**URL** <https://doi.org/10.48550/ARXIV.2402.16905>

Temporal logics are powerful tools that are widely used for the synthesis and verification of reactive systems. The recent progress on Large Language Models (LLMs) has the potential to make the process of writing such specifications more accessible. However, writing specifications in temporal logics remains challenging for all but the most expert users. A key question in using LLMs for temporal logic specification engineering is to understand what kind of guidance is most helpful to the LLM and the users to easily produce specifications. Looking specifically at the problem of reactive program synthesis, we explore the impact of providing an LLM with guidance on the separation of control and data-making explicit for the LLM what functionality is relevant for the specification, and treating the remaining functionality as an implementation detail for a series of pre-defined functions and predicates. We present a benchmark set and find that this separation of concerns improves specification generation. Our benchmark provides a test set against which to verify future work in LLM generation of temporal logic specifications.

### 3.20 A Short Introduction to Inductive Functional Programming

*Ute Schmid (Universität Bamberg, DE)*

**License** © Creative Commons BY 4.0 International license  
© Ute Schmid

**Main reference** Sumit Gulwani, José Hernández-Orallo, Emanuel Kitzelmann, Stephen H. Muggleton, Ute Schmid, Benjamin G. Zorn: “Inductive programming meets the real world”, Commun. ACM, Vol. 58(11), pp. 90–99, 2015.

**URL** <https://doi.org/10.1145/2736282>

Inductive functional programming, also called inductive program synthesis, addresses the problem of learning (mostly recursive) functional programs from input/output examples. An related area of research is inductive logic programming (ILP). IP is a type of machine learning because programs (models) are synthesized by inductive generalisation. In contrast to statistical and neural approaches to machine learning, IP approaches typically only need a small number of training examples. Since learned models are represented in form of programs, IP belongs to the group of interpretable machine learning approaches. In the talk, I will give an introduction to inductive functional programming and also present basic concepts of ILP. Furthermore, I will point out how IP can be combined with Deep Learning Architectures for explainability.



### 3.21 Oracle-Guided Inductive Synthesis, Learning Theory, and LLMs

*Sanjit A. Seshia (University of California – Berkeley, US)*

License © Creative Commons BY 4.0 International license  
© Sanjit A. Seshia

More than a decade ago, I described how many problems in formal methods are effectively addressed through reduction to synthesis, including the synthesis of proof artifacts during verification, and synthesis within solvers such as for theory lemmas and quantifier instantiation. Additionally, I observed how an inductive, data-driven approach to synthesis is often very effective. In this talk, I review these ideas, which are also summarized in [1]. I also describe how they enable one to develop learning-theoretic foundations for synthesis, leading to the frameworks of formal inductive synthesis and oracle-guided inductive synthesis (OGIS), with initial theoretical results reported in [2]. Finally, I note how synthesis with large language models (LLMs) is but a special case of oracle-guided synthesis where the LLM forms an untrusted but often effective oracle for searching over large expression (program) spaces. I describe how we can use this oracle-guided synthesis view to formulate an approach to verified code transpilation with LLMs, which beats all conventional approaches to verified code transpilation – initial results are presented in [3].

#### References

- 1 Sanjit A. Seshia. Combining Induction, Deduction, and Structure for Verification and Synthesis. Proceedings of the IEEE, 103(11):2036–2051, 2015. Conference version in DAC 2012.
- 2 Susmit Jha and Sanjit A. Seshia. A Theory of Formal Synthesis via Inductive Learning. Acta Informatica, 54(7):693–726, 2017.
- 3 Sahil Bhatia, Jie Qiu, Sanjit A. Seshia and Alvin Cheung. Can LLMs Perform Verified Lifting of Code? Technical Report No. UCB/EECS-2024-11, EECS Department, UC Berkeley, March 2024.

### 3.22 An Approximate Skolem Function Counter

*Arijit Shaw (Chennai Mathematical Institute, IN & University of Toronto, CA)*

License © Creative Commons BY 4.0 International license  
© Arijit Shaw

**Joint work of** Arijit Shaw, Brendan Juba, Kuldeep S. Meel

**Main reference** Arijit Shaw, Brendan Juba, Kuldeep S. Meel: “An Approximate Skolem Function Counter”, CoRR, Vol. abs/2312.12026, 2023.

**URL** <https://doi.org/10.48550/ARXIV.2312.12026>

Motivated by the recent development of scalable approaches to Boolean function synthesis, we study the problem of counting Boolean functions: given a Boolean specification between a set of inputs and outputs, count the number of functions of inputs such that the specification is met. This stands in relation to our problem analogously to the relationship between Boolean satisfiability and the model counting problem. Yet, counting Skolem functions poses considerable new challenges. From the complexity-theoretic standpoint, counting Skolem functions is not only  $\#P$ -hard; it is quite unlikely to have an FPRAS (Fully Polynomial Randomized Approximation Scheme) as the problem of even synthesizing one Skolem function remains challenging, even given access to an NP oracle. The primary contribution of this work is the first algorithm, SkolemFC, that computes an estimate of the number of Skolem functions.

SkolemFC relies on technical connections between counting functions and propositional model counting: our algorithm makes a linear number of calls to an approximate model counter and computes an estimate of the number of Skolem functions with theoretical guarantees. Moreover, we show that Skolem function count can be approximated through a polynomial number of calls to a SAT oracle. Our prototype displays impressive scalability, handling benchmarks comparably to state-of-the-art Skolem function synthesis engines, even though counting all such functions ostensibly poses a greater challenge than synthesizing a single function.

### References

- 1 Arijit Shaw, Brendan Juba, and Kuldeep S. Meel. *An Approximate Skolem Function Counter*. Proceedings of the AAAI Conference on Artificial Intelligence, volume 38, number 8, pages 8108–8116, 2024.

## 3.23 Counterexample-Guided DQBF Solving

*Friedrich Slivovsky (University of Liverpool, GB)*

**License** © Creative Commons BY 4.0 International license  
© Friedrich Slivovsky

**Joint work of** Franz-Xaver Reichl, Friedrich Slivovsky, Stefan Szeider

**Main reference** Franz-Xaver Reichl, Friedrich Slivovsky, Stefan Szeider: “Certified DQBF Solving by Definition Extraction”, in Proc. of the Theory and Applications of Satisfiability Testing – SAT 2021 – 24th International Conference, Barcelona, Spain, July 5-9, 2021, Proceedings, Lecture Notes in Computer Science, Vol. 12831, pp. 499–517, Springer, 2021.

**URL** [https://doi.org/10.1007/978-3-030-80223-3\\_34](https://doi.org/10.1007/978-3-030-80223-3_34)

In a Dependency Quantified Boolean Formula (DQBF), each existentially quantified variable is annotated with a dependency set consisting of universally quantified variables. A model of a DQBF consists of functions that correctly assign values to the existentially quantified variables based on the values of the universally quantified variables they depend on. Determining whether a DQBF has a model is NEXP-complete, and DQBFs can naturally express a range of synthesis problems. This talk presents an algorithm for finding a model of a DQBF that iteratively refines a candidate model based on counterexamples. It covers techniques that are crucial to make this approach work well in practice, such as identifying unique Skolem functions by propositional definition extraction, and finding local repairs of invalid functions.

## 3.24 A problem with functional synthesis

*Mate Soos (University of Toronto, CA), Supratik Chakraborty (Indian Institute of Technology Bombay – Mumbai, IN), and Kuldeep S. Meel (University of Toronto, CA)*

**License** © Creative Commons BY 4.0 International license  
© Mate Soos, Supratik Chakraborty, and Kuldeep S. Meel


There is an issue we should address related to functional synthesis. Its definition is incomplete. It talks about inputs and outputs only – but many of the variables are in fact don’t cares. It is easily imaginable that many users don’t need the skolem function for a number of internal variables that are not inputs. However, current systems have to create a skolem function for these, too, potentially wasting the end user’s resources. In my opinion, this should to be changed, allowing users to give a dontcare set.

So, if e.g. there are 100 variables, and the user sets 1..50 as inputs, they should be able to say that they are only interested in the skolem function of variable 100 – and if that involves variables 51..99 then of course their skolem functions, too. But if, for example, variable 100 is not connected in any way, shape or form, to variable 66, then there is absolutely no point in creating the skolem function for variable 66. It would be nothing but a waste of the end user’s resources. We should focus on what the end users want – and I’m quite sure they only want specific variable(s)’ skolem functions, not everything that isn’t an input.

Let’s discuss. Obviously this new formulation can gracefully simulate the original problem, simply set  $\text{dotcare} = \emptyset$ . But I am pretty sure that once users start using functional synthesis, their  $\text{dotcare}$  set will be quite large.

### 3.25 Reactive synthesis via parity and Rabin games


*K. S. Thejaswini (IST Austria – Klosterneuburg, AT)*

License  Creative Commons BY 4.0 International license  
© K. S. Thejaswini

To solve the reactive synthesis problem from LTL specifications or non-deterministic Buchi automata, there are two common approaches: either reduce it to the solving a parity game or solving a Rabin game. We discuss some algorithms to solve these games.

### 3.26 On the Power of LTLf in Reactive Synthesis

*Shufang Zhu (University of Oxford, GB)*

License  Creative Commons BY 4.0 International license  
© Shufang Zhu  
Joint work of Shufang Zhu, Geguang Pu, Moshe Y. Vardi, Jianwen Li, Lucas M. Tabajara, Giuseppe De Giacomo, Antonio Di Stasio, Marta Kwiatkowska, Pian Yu

Reactive synthesis emerges as a trustworthy-by-design technique in developing verifiably correct autonomous AI systems. This talk puts a particular focus on reactive synthesis of Linear Temporal Logic on finite traces (LTLf). LTLf, on the one hand, allows for specifying a rich set of temporally extended specifications, and on the other hand, focuses on finite traces, which makes it particularly suitable for specifying tasks of autonomous AI systems. Note that autonomous AI systems will not get stuck accomplishing a task for all their lifetime, but only for a finite (but unbounded) number of steps. In this talk, I will present an overview of key advancements in LTLf synthesis, highlighting its scalability and potential in complex scenarios. These results base on a so-called DFA-technology, which essentially takes the maximal simplicity of reasoning about efficiently constructed deterministic finite word automaton (DFA) of the LTLf objective. The goal of this talk is to engage researchers in automated synthesis, encouraging further advances on the scalability and applicability of LTLf synthesis.

#### References

- 1 Shufang Zhu, Lucas M. Tabajara, Jianwen Li, Geguang Pu, Moshe Y. Vardi: Symbolic LTLf Synthesis. IJCAI 2017: 1362-1369
- 2 Shufang Zhu, Giuseppe De Giacomo, Geguang Pu, Moshe Y. Vardi: LTLf Synthesis with Fairness and Stability Assumptions. AAAI 2020: 3088-3095

- 3 Giuseppe De Giacomo, Antonio Di Stasio, Moshe Y. Vardi, Shufang Zhu: Two-Stage Technique for LTLf Synthesis Under LTL Assumptions. KR 2020: 304-314
- 4 Shufang Zhu, Lucas M. Tabajara, Geguang Pu, Moshe Y. Vardi: On the Power of Automata Minimization in Temporal Synthesis. GandALF 2021: 117-134
- 5 Giuseppe De Giacomo, Antonio Di Stasio, Lucas M. Tabajara, Moshe Y. Vardi, Shufang Zhu: Finite-Trace and Generalized-Reactivity Specifications in Temporal Synthesis. IJCAI 2021: 1852-1858
- 6 Pian Yu, Shufang Zhu, Giuseppe De Giacomo, Marta Kwiatkowska, Moshe Y. Vardi: The Trembling-Hand Problem for LTLf Planning. To appear at IJCAI2024

## 4 Working groups

### 4.1 Benchmarking (and LLMs)

*José Cambroneró (Microsoft – Redmond, US), Johannes Klaus Fichte (Linköping University, SE), Benedikt Maderbacher (TU Graz, AT), Tobias Meggendorfer (Lancaster University Leipzig, DE), Ruzica Piskac (Yale University – New Haven, US), and Mark Santolucito (Barnard College, Columbia University – New York, US)*

**License** © Creative Commons BY 4.0 International license  
 © José Cambroneró, Johannes Klaus Fichte, Benedikt Maderbacher, Tobias Meggendorfer, Ruzica Piskac, and Mark Santolucito

In our discussion, we focused on current challenges to identifying opportunities for impact for the work carried out by the reactive synthesis community. One point discussed was the need to identify (industrial) applications and then use these applications to drive benchmark development. While current benchmarks, such as SYNTCOMP, provide cases that are able to test the limits of current solvers (an important goal, and complementary to what we propose here), there is no evidence that the tasks being solved in these competitions are necessarily realistic for industrial use. Discussion surfaced some natural domains for possible application, such as networking, robotics, smart-home systems, or other applications that require substantial planning and long running environments. However, how to obtain concrete tasks from these domains remains a challenge. One idea discussed was the importance of establishing connections with such communities, potentially through venues like Dagstuhl, to bring together their problems with the solutions from the reactive synthesis community.

Finally, we discussed the potential for using generative AI (specifically LLMs) as a potential tool to improve or extend the current benchmarking done or existing tools. For example, we discussed that an LLM may be able to provide a natural language description of an LTL specification, which may be helpful for explainability (making this more accessible to non-experts). Similarly, an LLM may be potentially useful for generating LTL benchmark tasks, which may be of interest to evaluate performance on problems with different structure or that somehow reflect the bias of problems observed in their training data (which in turn may correlate with potential applications). Alternatively, we also discussed that LLM-based systems (e.g. agents) may themselves be a good application area for LTL.

One challenge identified by participants is that the community may not necessarily reward applications of techniques, since these efforts would be reflected in publications (or other achievements) in the target domain community instead.

## 4.2 Developing a Computational Theory for Learning Functions from Relations

*Kuldeep S. Meel (University of Toronto, CA), Dror Fried (The Open University of Israel – Ra’anana, IL), Alfons Laarman (Leiden University, NL), Sanjit A. Seshia (University of California – Berkeley, US), and Mate Soos (University of Toronto, CA)*

**License** © Creative Commons BY 4.0 International license  
© Kuldeep S. Meel, Dror Fried, Alfons Laarman, Sanjit A. Seshia, and Mate Soos

The working group participants included Dror Fried, Mate Soos, Sanjit A. Seshia, and Alfons Laarman. The group’s primary objective was to explore the necessity for developing a computational theory for learning functions from relations. The discussions encompassed understanding the connections between the work of Jha and Seshia, particularly their paper titled *A Theory of Formal Synthesis via Inductive Learning* (2014).

Moreover, another significant line of discussion was the distinction between the traditional setting of computational learning theory, where the underlying specification guarantees a unique function, and the scenarios we are interested in. Specifically, the focus was on developing a theoretical framework that can characterize situations where it is possible to learn small Skolem functions, assuming such functions exist.

The discussions concluded with the consensus that these issues represent important open problems in the field, warranting further research and investigation.

## 5 Panel discussions

### 5.1 Reactive Synthesis and Model Counting Competitions

*Guillermo A. Pérez (University of Antwerp, BE) and Johannes Klaus Fichte (Linköping University, SE)*

**License** © Creative Commons BY 4.0 International license  
© Guillermo A. Pérez and Johannes Klaus Fichte

Automated Reasoning (AR) covers different aspects of deductive reasoning as practiced in mathematics and formal logic. Practical and theoretical research enabled ground-breaking success in a variety of application domains. At the core of this success lie incredibly sophisticated and complex pieces of software (so-called solvers), which tackle specific problems of AR. Recurring (typically annual) solver competitions or evaluations play a significant role in this practical success. Competitions aim at advancing applications, identifying challenging benchmarks, fostering new solver development, enhancing existing solvers, bringing together various researchers, identifying challenges, and inspiring numerous new applications. In this talk, we share experience from two competitions. The Model Counting Competition (<https://mccompetition.org/>)[2] and the Reactive Synthesis Competition (<https://www.syntcomp.org/>)[1]. Subsequently, we provide thoughts and tasks for a broad discussion to establish a Boolean Function Synthesis Challenge.

**References**

- 1 Swen Jacobs, Guillermo A. Pérez, Remco Abraham, Véronique Bruyère, Michaël Cadilhac, Maximilien Colange, Charly Delfosse, Tom van Dijk, Alexandre Duret-Lutz, Peter Faymonville, Bernd Finkbeiner, Ayrat Khalimov, Felix Klein, Michael Luttenberger, Klara J. Meyer, Thibaud Michaud, Adrien Pommellet, Florian Renkin, Philipp Schlehuber-Caissier, Mouhammad Sakr, Salomon Sickert, Gaëtan Staquet, Clément Tamines, Leander Tentrup, Adam Walker: *The Reactive Synthesis Competition (SYNTCOMP): 2018-2021*. CoRR abs/2206.00251 (2022)
- 2 Johannes Klaus Fichte, Markus Hecher, Florim Hamiti: *The Model Counting Competition 2020*. ACM J. Exp. Algorithmics 26: 13:1-13:26 (2021)

## Participants

- S. Akshay  
Indian Institute of Technology  
Bombay – Mumbai, IN
- Ashwani Anand  
MPI-SWS – Kaiserslautern, DE
- A. R. Balasubramanian  
MPI-SWS – Kaiserslautern, DE
- Suguman Bansal  
Georgia Institute of Technology –  
Atlanta, US
- Katrine Bjørner  
New York University, US
- José Cambronero  
Microsoft – Redmond, US
- Supratik Chakraborty  
Indian Institute of Technology  
Bombay – Mumbai, IN
- Deepak D’Souza  
Indian Institute of Science –  
Bangalore, IN
- Alexis de Colnet  
TU Wien, AT
- Rayna Dimitrova  
CISPA – Saarbrücken, DE
- Rüdiger Ehlers  
TU Clausthal, DE
- Johannes Klaus Fichte  
Linköping University, SE
- Bernd Finkbeiner  
CISPA – Saarbrücken, DE
- Dror Fried  
The Open University of Israel –  
Ra’anana, IL
- Mikoláš Janota  
Czech Technical University –  
Prague, CZ
- Jie-Hong Roland Jiang  
National Taiwan University –  
Taipei, TW
- Ayrat Khalimov  
TU Clausthal, DE
- Alfons Laarman  
Leiden University, NL
- Benedikt Maderbacher  
TU Graz, AT
- Pierre Marquis  
University of Artois/CNRS –  
Lens, FR
- Kuldeep S. Meel  
University of Toronto, CA
- Tobias Meggendorfer  
Lancaster University Leipzig, DE
- Jingyi Mei  
Leiden University, NL
- Guillermo A. Pérez  
University of Antwerp, BE
- Ruzica Piskac  
Yale University – New Haven, US
- Govind Rajanbabu  
Indian Institute of Technology  
Bombay – Mumbai, IN
- Subhajit Roy  
Indian Institute of Technology  
Kanpur, IN
- Mark Santolucito  
Barnard College, Columbia  
University – New York, US
- Ute Schmid  
Universität Bamberg, DE
- Sanjit A. Seshia  
University of California –  
Berkeley, US
- Shetal Shah  
Indian Institute of Technology  
Bombay – Mumbai, IN
- Arijit Shaw  
Chennai Mathematical Institute,  
IN & University of Toronto, CA
- Friedrich Slivovsky  
University of Liverpool, GB
- Mate Soos  
University of Toronto, CA
- K. S. Thejaswini  
IST Austria – Klosterneuburg,  
AT
- Hazem Torfah  
Chalmers University of  
Technology – Göteborg, SE
- Shufang Zhu  
University of Oxford, GB

