



Volume 13, Issue 10, October 2023

Automated mathematics: integrating proofs, algorithms and data (Dagstuhl Seminar 23401) <i>Andrej Bauer, Katja Berčič, Florian Rabe, and Nicolas Thiéry</i>	1
Accountable Software Systems (Dagstuhl Seminar 23411) <i>Bettina Könighofer, Joshua A. Kroll, Ruzica Piskac, and Michael Veale</i>	24
Formal Methods for Correct Persistent Programming (Dagstuhl Seminar 23412) <i>Ori Lahav, Azalea Raad, Joseph Tassarotti, and Viktor Vafeiadis</i>	50
Quantum Cryptanalysis (Dagstuhl Seminar 23421) <i>Gorjan Alagic, Maria Naya-Plasencia, and Rainer Steinwandt</i>	65
Graph Algorithms: Cuts, Flows, and Network Design (Dagstuhl Seminar 23422) <i>Jason Li, Debmalaya Panigrahi, Laura Sanita, and Thatchaphol Saranurak</i>	76
Network Attack Detection and Defense – AI-Powered Threats and Responses (Dagstuhl Seminar 23431) <i>Sven Dietrich, Artur Hermann, Frank Kargl, Hartmut König, and Pavel Laskov</i> ..	90
Edge-AI: Identifying Key Enablers in Edge Intelligence (Dagstuhl Seminar 23432) <i>Eyal de Lara, Aaron Ding, Shahram Dustdar, and Ella Peltonen</i>	130
Ensuring the Reliability and Robustness of Database Management Systems (Dagstuhl Seminar 23441) <i>Hannes Mühleisen, Danica Porobic, and Manuel Rigger</i>	139
Approaches and Applications of Inductive Programming (Dagstuhl Seminar 23442) <i>Luc De Raedt and Ute Schmid</i>	182

ISSN 2192-5283

Published online and open access by

Schloss Dagstuhl – Leibniz-Zentrum für Informatik GmbH, Dagstuhl Publishing, Saarbrücken/Wadern, Germany. Online available at <https://www.dagstuhl.de/dagpub/2192-5283>

Publication date

April, 2024

Bibliographic information published by the Deutsche Nationalbibliothek

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

License

This work is licensed under a Creative Commons Attribution 4.0 International license (CC BY 4.0).



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

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

The copyright is retained by the corresponding authors.

Aims and Scope

The periodical *Dagstuhl Reports* documents the program and the results of Dagstuhl Seminars and Dagstuhl Perspectives Workshops.

In principal, for each Dagstuhl Seminar or Dagstuhl Perspectives Workshop a report is published that contains the following:

- an executive summary of the seminar program and the fundamental results,
- an overview of the talks given during the seminar (summarized as talk abstracts), and
- summaries from working groups (if applicable).

This basic framework can be extended by suitable contributions that are related to the program of the seminar, e. g. summaries from panel discussions or open problem sessions.

Editorial Board

- Elisabeth André
- Franz Baader
- Daniel Cremers
- Goetz Graefe
- Reiner Hähnle
- Barbara Hammer
- Lynda Hardman
- Oliver Kohlbacher
- Steve Kremer
- Rupak Majumdar
- Heiko Mantel
- Albrecht Schmidt
- Wolfgang Schröder-Preikschat
- Raimund Seidel (*Editor-in-Chief*)
- Heike Wehrheim
- Verena Wolf
- Martina Zitterbart

Editorial Office

Michael Wagner (*Managing Editor*)
Michael Didas (*Managing Editor*)
Jutka Gasiorowski (*Editorial Assistance*)
Dagmar Glaser (*Editorial Assistance*)
Thomas Schillo (*Technical Assistance*)

Contact

Schloss Dagstuhl – Leibniz-Zentrum für Informatik
Dagstuhl Reports, Editorial Office
Oktavie-Allee, 66687 Wadern, Germany
reports@dagstuhl.de

<https://www.dagstuhl.de/dagrep>

Digital Object Identifier: 10.4230/DagRep.13.10.i

Automated mathematics: integrating proofs, algorithms and data

Andrej Bauer^{*1}, Katja Berčič^{*2}, Florian Rabe^{*3}, Nicolas Thiéry^{*4},
and Jure Taslak^{†5}

- 1 University of Ljubljana, SI &
Institute for Mathematics, Physics and Mechanics, Ljubljana, SI
- 2 University of Ljubljana, SI &
Institute for Mathematics, Physics and Mechanics, Ljubljana, SI
- 3 Universität Erlangen-Nürnberg, DE
- 4 University Paris-Saclay – Orsay, FR
- 5 University of Ljubljana, SI

Abstract

This report documents the program and the outcomes of Dagstuhl Seminar 23401 “Automated mathematics: integrating proofs, algorithms and data”. The seminar brought together system developers, library authors, and users from key branches of computer-supported mathematics: formalized mathematics, symbolic computation, and mathematical databases. We addressed issues that were common to all areas of computer-supported mathematics: library management, dependencies and interoperability between software components, quality and correctness assurances, searching for information, and usability by end users. Early on in the week, we formed working groups that worked on specific tasks, as described in this report. Each day was divided into a morning talk session and an afternoon period devoted to working in groups. To keep everyone well-informed, we gathered each day before dinner for an informal “show & tell” session.

Seminar October 1–6, 2023 – <https://www.dagstuhl.de/23401>

2012 ACM Subject Classification Mathematics of computing → Mathematical software; Computing methodologies → Symbolic and algebraic manipulation; Computing methodologies → Symbolic and algebraic manipulation; Computing methodologies → Symbolic and algebraic manipulation

Keywords and phrases mathematical knowledge management, mathematical software, formalized mathematics, computer algebra, databases of mathematical structures

Digital Object Identifier 10.4230/DagRep.13.10.1

1 Executive Summary

Andrej Bauer

Katja Berčič

Florian Rabe

Nicolas Thiéry

License  Creative Commons BY 4.0 International license
© Andrej Bauer, Katja Berčič, Florian Rabe, Nicolas Thiéry

Modern mathematical software and large datasets of mathematical knowledge allow new approaches to solving mathematical problems, and support new kinds of mathematical exploration. In the past, lack of cooperation led to each project developing its own standards

* 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 mathematics: integrating proofs, algorithms and data, *Dagstuhl Reports*, Vol. 13, Issue 10, pp. 1–23

Editors: Andrej Bauer, Katja Berčič, Florian Rabe, and Nicolas Thiéry



DAGSTUHL
REPORTS

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

and techniques. Both developers and users experienced the resulting incompatibilities and fragmentation as serious usability issues and major obstacle to wider adoption, and none of the projects can address these successfully on their own. To make further progress that will eventually result in truly comprehensive and advanced computer mathematical assistant systems, the developers of individual software projects must start working together and tackle questions of interoperability, software engineering, and knowledge management on a large scale.

Concretely, this Dagstuhl Seminar brought together system developers, library authors, and users from three key areas of computer-supported mathematics: theorem provers, symbolic computation, and databases of mathematical structures. All three areas develop large formal mathematical libraries, but they do so in fundamentally different and incompatible ways. Theorem provers optimize for precise definitions and automation of proof support, often employing complex and abstract representation languages that capture exactly the semantics of the mathematical concepts of interest. Computer algebra systems, on the other hand, prioritize the efficiency of computation, which usually requires hardware-near representations that can be optimized for speed. Mathematical datasets finally employ general purpose database languages, which are optimized for indexing and fast querying, often requiring non-trivial encodings of mathematical objects in terms of concrete data. Over several decades of mostly independent development, these different communities have built software systems that are as impressively large as they are different from each other.

Work Groups

Broadly speaking, the seminar participants employed three independent approaches towards system integration.

Firstly, *direct integration* builds individual bridges between (usually) two systems. These tend to be more ad hoc but enable a problem-driven approach that delivers a specific practically needed integration solution. Multiple work groups were formed that tackled individual bridges.

Secondly, *ontology-based integration* uses a central representation that acts as an interoperability layer. The ontology describes mathematical concepts abstractly without a commitment to any of the three flavors of systems. Two work groups investigated this approach:

- Alignments: This work group worked towards building a central ontology of mathematical concepts. It surveyed existing approaches and judged the feasibility of major future approaches. Of special interests was the difficulty of *library alignment*, the task of connecting the central ontology to the individual libraries.
- Knowledge graphs: This work group investigated services that can be built on top of a central ontology. Of particular interest were knowledge graph techniques, which use concepts as the nodes and alignments as some of the edges.

Thirdly, for the special goal of integrating datasets, a work group on building an *index of datasets* started a major push towards cataloging the many existing datasets, which are distributed all over the internet, often without active maintenance. This is a necessary step towards more systemic integration with each other and deduction and computation systems.

Outcomes

Overall, we observed that the field has made major progress over the last 10 years. Direct integrations that would have been very expensive in the past, often prohibitively so, have become feasible targets for short meetings as within a Dagstuhl Seminar. This can be

attributed to increased awareness in the community of interoperability needs that has led to better interface design. Nonetheless, integrations are still brittle, and a major incentive problem remains: it is difficult for two communities to maintain bridges between their systems.

Ontology-based integration had developed little momentum in the past because of the high cost of additionally maintaining the central ontology. Here, the seminar showed that the time is right for a major push towards this and initiated a community-driven ontology curation project.

The work on building an index of datasets kick-started a dataset curation project. This project has already attracted the attention of outside researchers and has led to the founding of the Mathbase project.

2 Table of Contents

Executive Summary

<i>Andrej Bauer, Katja Berčič, Florian Rabe, Nicolas Thiéry</i>	1
---	---

Overview of Talks

Who finds the short proof? Searching for Wormholes in Proof-Space <i>Christoph Benzmüller</i>	6
A catalogue of mathematical datasets <i>Katja Berčič</i>	6
Formal verification of mathematical algorithms when the definitions are out of reach <i>Alex Best</i>	7
Learning from “invisible mathematics” <i>Jacques Carette</i>	7
Proving an Execution of an Algorithm Correct? <i>James H. Davenport</i>	8
Extracting Mathematical Concepts from Text <i>Valeria de Paiva</i>	8
(Re)Verification of Proofs <i>Catherine Dubois</i>	9
Understanding the Symmetries of Bin Packing Problems Inspired by Application Deployment in the Cloud <i>Madalina Erascu</i>	9
House of Graphs: A searchable database of interesting graphs and more <i>Jan Goedgebeur</i>	10
Mostly Automated Proof Repair for Verified Libraries <i>Kiran Gopinathan</i>	10
Towards a centralized system for mathematical objects <i>Dimitri Leemans</i>	11
Machine-learnable Data Sets for Formalized Mathematics (MLFMF) <i>Matej Petkovic and Andrej Bauer</i>	11
Heterogenous search in formal mathematical libraries <i>Claudio Sacerdoti Coen</i>	11
Proof and Computation with PVS <i>Natarajan Shankar</i>	12
Enumeration, a system for systematic enumeration of finite mathematical structures <i>Jure Taslak</i>	12
Alien Coding: Learning Synthesis of OEIS Sequences <i>Josef Urban</i>	13
Isabelle as System Platform for the Archive of Formal Proofs (AFP) <i>Makarius Wenzel</i>	13

Working groups

Using verified code inside CASes <i>Alex Best and Tobias Nipkow</i>	14
Reconceptualization <i>Jacques Carette, Gilles Dowek, and Catherine Dubois</i>	15
Formal Verification of Computer Algebra (Factorisatoion) <i>James H. Davenport, Alex Best, Mario Carneiro, and Edgar Costa</i>	16
Coq, Isabelle and Dedukti as heterogeneous networks <i>Filip Koprivec, Mario Carneiro, Stefania Dumbrava, Matej Petkovic, and Makarius Wenzel</i>	17
Object identification using invariant based decision trees <i>Filip Koprivec and Matej Petkovic</i>	17
Datasets <i>Dimitri Leemans, Katja Berčič, Jan Goedgebeur, Darij Grinberg, Samuel Lelievre, Harshit J Motwani, and Tom Kaspar Wiesing</i>	18
Aligning Mathematical Concepts Across Libraries <i>Florian Rabe</i>	18
Scalability Estimates of Graph Certificates in a Theorem Prover Using SAT Encodings <i>Kathrin Stark, Madalina Erascu, Kazuhiko Sakaguchi, and Jure Taslak</i>	21
Participants	23

3 Overview of Talks

3.1 Who finds the short proof? Searching for Wormholes in Proof-Space

Christoph Benz Müller (Universität Bamberg, DE)

License  Creative Commons BY 4.0 International license
 Christoph Benz Müller

Joint work of Christoph Benz Müller, David Fuenmayor, Alexander Steen, Geoff Sutcliffe

Main reference Christoph Benz Müller, David Fuenmayor, Alexander Steen, Geoff Sutcliffe: “Who Finds the Short Proof?”, *Logic Journal of the IGPL*, p. jzac082, 2023.

URL <https://doi.org/10.1093/jigpal/jzac082>



In my talk I presented recently published results that had not been communicated at conferences or workshops before: An exploration of Boolos’ Curious Inference using higher-order automated theorem provers (ATPs). Surprisingly, only suitable shorthand notations had to be provided by hand for ATPs to find a short proof. The higher-order lemmas required for constructing a short proof are automatically discovered by the ATPs. Given the observations and suggestions in this paper, full proof automation of Boolos’ and related examples now seems to be within reach of higher-order ATPs.

References

- 1 Benz Müller, C., Fuenmayor, D., Steen, A. and Sutcliffe, G. Who Finds the Short Proof? *Logic Journal of the IGPL*. 2023. Doi: 10.1093/jigpal/jzac082
- 2 Benz Müller, C. and Brown, C. The curious inference of Boolos in MIZAR and OMEGA. In Matuszewski, R., Zalewska, A., editor(s), *From Insight to Proof – Festschrift in Honour of Andrzej Trybulec*, volume 10(23), of *Studies in Logic, Grammar, and Rhetoric*, pages 299-388. The University of Białystok, Polen, 2007.
- 3 Boolos, G. A Curious Inference. *Journal of Philosophical Logic* 16(1):1-12, 1987.

3.2 A catalogue of mathematical datasets

Katja Berčić (University of Ljubljana, SI)

License  Creative Commons BY 4.0 International license
 Katja Berčić

Main reference Katja Berčić: “Towards a Census of Relational Data in Mathematics”, in *Proc. of the Conference on “Lernen, Wissen, Daten, Analysen”*, Berlin, Germany, September 30 – October 2, 2019, *CEUR Workshop Proceedings*, Vol. 2454, pp. 207–217, [CEUR-WS.org](http://ceur-ws.org), 2019.

URL https://ceur-ws.org/Vol-2454/paper_40.pdf

In addition to well-known mathematical databases, such as the OEIS, LMFDB and the House of Graphs, there is a multitude of smaller projects. Any of these might be interesting to researchers from other areas, including machine learning. However, the smaller dataset projects are typically hard to find: just the right keywords are usually necessary (but not necessarily sufficient) to find them with standard search engines and even in scientific research data repositories (like Zenodo). As a community, we should build a collaborative, and at least partly automated, index of mathematical datasets. Personal projects, such as my own mathdb.mathhub.info, can serve as a starting point.

3.3 Formal verification of mathematical algorithms when the definitions are out of reach

Alex Best (King’s College – London, GB)

License  Creative Commons BY 4.0 International license
© Alex Best

Joint work of Alex Best, Sander Dahmen, Sacha Huriot-Tattégtrain

I’d be happy to report on and discuss some ongoing work (with S. Dahmen and S. Huriot-Tattégtrain) implementing a tricky mathematical algorithm in the field of number theory / arithmetic geometry in a proof assistant (this computes certain quantities appearing in the famous BSD conjecture many of which are recorded in the LMFDB). This algorithm, known as Tate’s algorithm in the field, is quite involved with many subcases and a non-trivial proof of terminations, and takes many pages to describe when expressed in paper form.

One thing that makes this an interesting is that to give a formal definition of the quantities the algorithm actually computes is still out of reach, nevertheless the steps of this algorithm as described in the literature can be implemented and termination can be proved with some non-trivial tracking of the state involved, but in what ways is this implementation more trustworthy than one in a regular programming language? I’d like to consider what guarantees implementing such code in a the strict setting of a proof assistant can give us, compared to ordinary code, even when the gold standard isn’t yet attainable.

The direct relationship between the mathematical theory and the code in this case means that this implementation very clearly states what assumptions are made about the ring we are working in, and in fact makes this the most general implementation available, this then begs the question: how can we best integrate mathematical code written using proof assistants into existing CASes? So it may be useful to users not running proof assistants themselves.

3.4 Learning from “invisible mathematics”

Jacques Carette (McMaster University – Hamilton, CA)


License  Creative Commons BY 4.0 International license
© Jacques Carette

Recently Andrej Bauer coined the term “invisible mathematics” for those aspects of mathematics which are readily apparent when doing them mechanically (especially, but not only, in proof assistants) but which is essentially invisible in traditional paper-math. Three main examples were given, and their formalization discussed.

Here we aim to give more such examples – but the aim is not on how to formalize them, but rather to observe their consequences. In particular, some instances of “invisible mathematics” readily reveal the cognitive load on learners of some (visible) mathematics. Rather than seeing parts of the “de Bruijn factor” as an impediment, we instead regard it as a learning opportunity.

3.5 Proving an Execution of an Algorithm Correct?

James H. Davenport (*University of Bath, GB*)

License  Creative Commons BY 4.0 International license
© James H. Davenport

We have previously [1] asked the question “Do I believe the output from my (complicated, optimised, unverified) computer algebra system?”. If it gives me a positive answer, e.g. “the answer to $\int f$ is g ”, then we can check that $g' = f$, etc. But what if it says “there is no answer”? In particular, if we ask to factor the polynomial F , then we can check that the factors multiply to F , but what about the (implicit) statement that these factors are irreducible? Currently, the user just has to believe the algebra system. We ask, and partially answer, the question “could the algebra system also produce a certificate, or the hints to construct a certificate, of irreducibility of the factors” (based on the work the algebra system has already done). This was taken up by one working group.

References

- 1 James Harold Davenport. Proving an Execution of an Algorithm Correct? In Dubois and Kerber [2], pages 255–269.
- 2 Catherine Dubois and Manfred Kerber, editors. *Proceedings CICM 2023*, volume 14101 of *Springer Lecture Notes in Computer Science*, 2023.

3.6 Extracting Mathematical Concepts from Text

Valeria de Paiva (*Topos Institute – Berkeley, US*)

License  Creative Commons BY 4.0 International license
© Valeria de Paiva

Joint work of Valeria de Paiva, Lucy Horowitz, Jacob Collard, Eswaran Subramanian, Pavel Kovalev, Qiyue Gao, Lawrence Moss

Main reference Lucy Horowitz, Valeria de Paiva: “MathGloss: Building mathematical glossaries from text”, CoRR, Vol. abs/2311.12649, 2023.

URL <https://doi.org/10.48550/ARXIV.2311.12649>

Main reference Jacob Collard, Valeria de Paiva, Eswaran Subrahmanian: “Parmesan: mathematical concept extraction for education”, CoRR, Vol. abs/2307.06699, 2023.

URL <https://doi.org/10.48550/ARXIV.2307.06699>

Main reference Jacob Collard, Valeria de Paiva, Brendan Fong, Eswaran Subrahmanian: “Extracting Mathematical Concepts from Text”, CoRR, Vol. abs/2208.13830, 2022.

URL <https://doi.org/10.48550/ARXIV.2208.13830>

Main reference Valeria de Paiva, Qiyue Gao, Pavel Kovalev, Lawrence S. Moss: “Extracting Mathematical Concepts with Large Language Models”, CoRR, Vol. abs/2309.00642, 2023.

URL <https://doi.org/10.48550/ARXIV.2309.00642>

We describe the project Network Mathematics we are developing at the Topos Institute, Berkeley, CA. We hope to take advantage of the tremendous recent progress in Natural Language Processing (NLP), including LLMs, transformers, etc, to extract information from mathematical texts. Also we think mathematical language is what mathematicians use when communicating with each other, so we should leverage mathematical English to increase the accessibility of mathematics to mathematicians, students and the general public.

To work on these goals, we have three main preliminary subprojects:

1. MathGloss (together with Lucy Horowitz, Chicago) – a collection of glossaries of college math, connected to each other via WikiData.
2. Parmesan (together with Jacob Collard and Eswaran Subramanian, NIST) provides semantic search on the field of Category Theory, using different knowledge bases at different levels (research math, wiki math and textbook math).

3. MathChat (together with Gao, Kovalev and Moss, Indiana) using generative AI (chatGPT and GTP-4) to extract concepts from text.

All these subprojects need to grow and need to include more sources. Also, further work needs to be done to structure the extracted concepts into a proper ontology, to obtain a proper Knowledge Graph for mathematics.

3.7 (Re)Verification of Proofs

Catherine Dubois (ENSIIE – Evry, FR)

License © Creative Commons BY 4.0 International license
© Catherine Dubois

Main reference Guillaume Burel, Guillaume Bury, Raphaël Cauderlier, David Delahaye, Pierre Halmagrand, Olivier Hermant: “First-Order Automated Reasoning with Theories: When Deduction Modulo Theory Meets Practice”, *J. Autom. Reason.*, Vol. 64(6), pp. 1001–1050, 2020.

URL <https://doi.org/10.1007/S10817-019-09533-Z>

The talk gives a quick overview of some tools developed around Dedukti to verify, re-verify or cross-verify proofs, more precisely, Zenon Modulo, iProverModulo, Archsat, and Ekstrakto. The three first ones directly produce Dedukti proofs that can be checked by the Dedukti checker. The latter reconstructs a Dedukti proof from a proof trace by reproving each step using a Dedukti producing tool and combining the proofs of the steps to get a proof of the original formula. In the talk, we also point out two projects: BWare and ICSPA. The first one aimed at developing a mechanized framework for automated verification of AtelierB proof obligations where Zenon Modulo and iProvermodulo were developed or used. ICSPA is a project in progress where the objectives are to improve confidence in the proofs realized in the context of B/Event-B and TLA+ by formally and independently verifying these proofs and also to enable sharing and reusing proofs and models between B/Event-B and TLA+ using lambda-PI calculus modulo theory and Dedukti.

3.8 Understanding the Symmetries of Bin Packing Problems Inspired by Application Deployment in the Cloud

Madalina Erascu (West University of Timisoara, RO)

License © Creative Commons BY 4.0 International license
© Madalina Erascu

Automated deployment of component-based applications in the Cloud consists in the allocation of virtual machines (VMs) offers from various Cloud Providers such that the constraints induced by the interactions between components and by the components hardware/software requirements are satisfied and the performance objectives are optimized (e.g. costs are minimized). It can be formulated as a constraint optimization problem, hence, in principle, the optimization can be carried out automatically. In the case the set of VM offers is large (several hundreds), the computational requirement is huge, making the automatic optimization practically impossible with the current general optimization modulo theory (OMT) and mathematical programming (MP) tools. We overcame the difficulty by methodologically analyzing the particularities of the problem with the aim of identifying search space reduction methods. Some of these methods are exploiting the symmetries of the general Cloud deployment problem leading to symmetry breakers. However, little is know/understood

about the symmetries of this problem which could lead to symmetry breakers which are not experimentally constructed. We present some of the symmetries of a simplistic formulation of automated deployment with the hope that the audience:

- can bring input regarding approaches applied for similar problems (bin-packing)
- is interested in taking part in a small working group to discuss if invariant theory of finite groups (see, e.g., Chapter 7 of [1]) could bring some insights towards the formalization of symmetries.

References

- 1 Cox, D.A., Little, J., O’Shea, D.. *Invariant Theory of Finite Groups*. In: Ideals, Varieties, and Algorithms. Undergraduate Texts in Mathematics. Springer, Cham, 2015

3.9 House of Graphs: A searchable database of interesting graphs and more

Jan Goedgebeur (KU Leuven, BE)

License © Creative Commons BY 4.0 International license
© Jan Goedgebeur

Joint work of Jan Goedgebeur, Gunnar Brinkmann, Kris Coolsaet, Sven D’hondt, Gauvain Devillez, Hadrien Mélot
Main reference Kris Coolsaet, Sven D’hondt, Jan Goedgebeur: “House of Graphs 2.0: A database of interesting graphs and more”, *Discret. Appl. Math.*, Vol. 325, pp. 97–107, 2023.

URL <https://doi.org/10.1016/J.DAM.2022.10.013>

We will present the House of Graphs (<https://houseofgraphs.org/>), which is a database of graphs. The House of Graphs hosts complete lists of graphs of various graph classes (e.g. cubic graphs, fullerenes, trees, etc.), but its main feature is a searchable database of so called “interesting” graphs, which includes graphs that already occurred as extremal graphs or as counterexamples to conjectures. We will highlight the features of the website and demonstrate how users can perform queries on this database and how they can add new interesting graphs to it.

3.10 Mostly Automated Proof Repair for Verified Libraries

Kiran Gopinathan (National University of Singapore, SG)

License © Creative Commons BY 4.0 International license
© Kiran Gopinathan

Joint work of Kiran Gopinathan, Mayank Keoliya, Ilya Sergey
Main reference Kiran Gopinathan, Mayank Keoliya, Ilya Sergey: “Mostly Automated Proof Repair for Verified Libraries”, *Proc. ACM Program. Lang.*, Vol. 7(PLDI), pp. 25–49, 2023.

URL <https://doi.org/10.1145/3591221>

The cost of maintaining formally specified and verified software is widely considered prohibitively high due to the need to constantly keep code and the proofs of its correctness in sync—the problem known as proof repair. One of the main challenges in automated proof repair for evolving code is to infer invariants for a new version of a once verified program that are strong enough to establish its full functional correctness.

In this work, we present the first proof repair methodology for higher-order imperative functions, whose initial versions were verified in the Coq proof assistant and whose specifications remained unchanged. Our proof repair procedure is based on the combination of dynamic program alignment, enumerative invariant synthesis, and a novel technique for efficiently pruning the space of invariant candidates, dubbed proof-driven testing, enabled by the constructive nature of Coq’s proof certificates.

3.11 Towards a centralized system for mathematical objects

Dimitri Leemans (UL – Brussels, BE)

License © Creative Commons BY 4.0 International license
© Dimitri Leemans

Mathematical data are available in all sorts of formats on the web. Often it is difficult for those who did not create the data to use them. Also these data are at risk of disappearing. We advocate for the building of a system consisting of

- a centralised database of mathematical objects, checked by peers,
- a website permitting to extract data from the database and send it to computational software (Magma, gap, etc.) to test conjectures, build more objects, ...

3.12 Machine-learnable Data Sets for Formalized Mathematics (MLFMF)

Matej Petkovic (University of Ljubljana, SI), Andrej Bauer (University of Ljubljana, SI)

License © Creative Commons BY 4.0 International license
© Matej Petkovic and Andrej Bauer
Joint work of Andrej Bauer, Matej Petković, Ljupčo Todorovski
Main reference Andrej Bauer, Matej Petkovic, Ljupco Todorovski: “MLFMF: Data Sets for Machine Learning for Mathematical Formalization”, CoRR, Vol. abs/2310.16005, 2023.
URL <https://doi.org/10.48550/ARXIV.2310.16005>

MLFMF is a collection of data sets for benchmarking recommendation systems used to support formalization of mathematics with proof assistants. Each data set is derived from a library of formalized mathematics written in proof assistants Agda or Lean. The collection includes the largest Lean 4 library Mathlib, and some of the largest Agda libraries: the standard library, the library of univalent mathematics Agda-unimath, and the TypeTopology library. Each data set represents the corresponding library in two ways: as a heterogeneous network, and as a list of s-expressions representing the syntax trees of all the entries in the library. The network contains the (modular) structure of the library and the references between entries, while the s-expressions give complete and easily parsed information about every entry.

3.13 Heterogenous search in formal mathematical libraries

Claudio Sacerdoti Coen (University of Bologna, IT)

License © Creative Commons BY 4.0 International license
© Claudio Sacerdoti Coen

I will present how I integrated in LambdaPi an indexer and search engine for mathematical formulae up to instantiation/generalization. The search engine allows to query in parallel libraries obtained by various provers (e.g. HOL/Matita/Coq/...) via an encoding into Dedukti/LambdaPi.

The task poses several challenges that are novel compared to search engines developed for a single system:

1. the statements occur encoded in LambaPi and the same statement coming from several systems is encoded in a different way

2. the statement can occur in several shapes, up to rewriting rules
3. in order to look in parallel in several libraries, one need to search up to alignment of constants.


I have solved all three previous challenges exploiting the rewriting engine of LambdaPi in order to:

1. undo the encoding via a non-type preserving transformation,
2. normalize the statements before indexing and the queries as well,
3. rewrite all constants to a canonic form (the representative of the equivalence class of aligned formula).

Indexing has been implemented combining substitution trees and information positioning (the one exploited in the Whelp system).

3.14 Proof and Computation with PVS

Natarajan Shankar (SRI – Menlo Park, US)

License  Creative Commons BY 4.0 International license
© Natarajan Shankar

SRI's Prototype Verification System (PVS) is an interactive proof assistant based on higher-order logic developed at SRI over the last three decades as a unified platform and language for formal specification, mathematical modeling, programming, and proof. It has been used to develop extensive libraries for mathematics and computing and for verification projects spanning fault-tolerant systems, air-traffic control systems, parsers, compilers, separation kernel, data refinement, and hardware. Nearly all of the specification language is efficiently executable with code extraction to Common Lisp and C, and experimental code generators targetting standard ML and Rust. We describe some of our experiments with modeling, proof, and computation focusing on extracting efficient C code from verified definitions.

During the Dagstuhl Seminar, we used PVS to verify two witness formats and executable checkers for graph connectivity. The witness for connectivity is a sequence containing all of the vertices such that each vertex has a neighbor in the preceding part of the sequence. This implies, by induction, that the graph is connected. For disconnectivity, the witness is a k -coloring of the vertices for $k > 1$ such that no vertex has a neighbor of a different color.

3.15 Enumerion, a system for systematic enumeration of finite mathematical structures

Jure Taslak (University of Ljubljana, SI)

License  Creative Commons BY 4.0 International license
© Jure Taslak

Joint work of Jure Taslak, Andrej Bauer

In the talk I presented my recent work in progress on Enumerion, which is a system for systematic enumeration of finite mathematical structures based on dependent type theory and implemented in OCaml. The problem of counting and enumerating discrete finite structures is a classical one. In the history of counting there are however quite a few published mistakes[1].

The idea of this system is to have a systematic way of describing the enumeration problem which is amenable to formalization. After explaining the idea behind the system I showed a short demo of the current capabilities of Enumerion.

References

- 1 McKay, Brendan D and Meynert, Alison and Myrvold, Wendy. *Small Latin squares, quasigroups, and loops* Journal of Combinatorial Designs

3.16 Alien Coding: Learning Synthesis of OEIS Sequences

Josef Urban (Czech Technical University – Prague, CZ)

License © Creative Commons BY 4.0 International license
© Josef Urban

Joint work of Thibault Gauthier, Miroslav Olsák, Josef Urban

Main reference Thibault Gauthier, Miroslav Olsák, Josef Urban: “Alien coding”, Int. J. Approx. Reason., Vol. 162, p. 109009, 2023.

URL <https://doi.org/10.1016/J.IJAR.2023.109009>

We introduce a self-learning algorithm for synthesizing programs that provide explanations for OEIS sequences. The algorithm starts from scratch initially generating programs at random. Then it runs many iterations of a self-learning loop that interleaves (i) training neural machine translation to learn the correspondence between sequences and the programs discovered so far, and (ii) proposing many new programs for each OEIS sequence by the trained neural machine translator. The algorithm discovers on its own programs for more than 78000 OEIS sequences, sometimes developing unusual programming methods. We analyze its behavior and the invented programs in several experiments.

3.17 Isabelle as System Platform for the Archive of Formal Proofs (AFP)

Makarius Wenzel (Dr. Wenzel – Augsburg, DE)


License © Creative Commons BY 4.0 International license
© Makarius Wenzel

Isabelle is usually seen as an interactive proof assistant, mostly for Higher-Order Logic (HOL), but that is somehow accidental. In reality, Isabelle is a system platform for functional programming and formal proofs, with sufficient infrastructure to carry its own weight and gravity. The Archive of Formal Proofs (AFP) is the official collection of Isabelle applications that is maintained together with the base system. That poses ever growing demands on the Isabelle platform. This talk gives an overview of Isabelle software technology, with specific focus on Programming and Scaling, e.g. distributed build clusters for AFP.

4 Working groups

4.1 Using verified code inside CASes

Alex Best (King’s College – London, GB) and Tobias Nipkow (TU München – Garching, DE)

License  Creative Commons BY 4.0 International license

© Alex Best and Tobias Nipkow

Joint work of Alex Best, Tobias Nipkow, Samuel Lelièvre, Olexandr Konovalov, Mario Carneiro, Assia Mahboubi

Proof assistants are by now able to produce reasonably efficient executable code for a variety of mathematical algorithms. This, together with the high degree of confidence they can provide in the correctness of the code (and therefore in the answers produced), makes using code produced using a proof assistant in an otherwise unverified computer algebra system quite attractive. This could be used to replace existing unverified implementations (or complement them with possibly slower reference implementations) or add new functionality completely.

This working group used the expertise from participants on both the ITP and CAS sides to explore different methods and implementations to pass results of computations from code generated by ITPs to CASes. This was in order to understand what is possible and evaluate whether CASes could feasibly benefit from using such code more widely.

As a proof of concept, Tobias Nipkow took a verified (in Isabelle) implementation (in Haskell) of the Berlekamp and Zassenhaus integer polynomial factorization algorithm (which is available at the AFP https://www.isa-afp.org/entries/Berlekamp_Zassenhaus.html) and made it callable from the computer algebra systems Sage (with the help of Samuel Lelièvre) and GAP (with the help of Olexandr Konovalov). Although it was not meant to compete with the standard Sage and GAP functions, it turned out that in the case of GAP it outperformed the standard Factors function on large polynomials.

Alex Best, with the help and input of Mario Carneiro and Assia Mahboubi experimented with linking directly to binary (compiled) objects produced by Lean 4 using Python (with the intention of targeting SageMath). Using the low level CTypes library it was possible to directly call Lean implementations of functions on simple types, such as a `nextPrime` on 64 bit integers. A proof of concept for working at a higher level, using Lean interpreter to look up Lean functions stored memory, without needing prior knowledge of their exported names in compiled code, was also written. This allowed more flexibility as no wrapping function would need to be written on the Lean side, the CAS could call arbitrary code producing answers in a desired format without having to re-run the Lean compiler. Finally initial experiments with using polymorphic functions implemented in Lean on types implemented on the Python side were successful, for example with this paradigm a Python program could provide a list of Python objects and a callback comparison function to a Lean implementation of quicksort (which operates an arbitrary ordered type), this combination would produce correct output subject to the assumption that the python comparison function satisfies the partial order hypothesis used in the Lean code. In this way mixing verified and unverified code could be done with little overhead.

Going forward next steps would be to turn these examples into user facing libraries, to dynamically load and call code from ITP libraries from CASes, we believe that our experiments have shown such libraries can be of benefit to CAS users, even those with little formalization experience.

4.2 Reconceptualization

Jacques Carette (*McMaster University – Hamilton, CA*), Gilles Dowek (*ENS – Gif-sur-Yvette, FR*), and Catherine Dubois (*ENSIIE – Evry, FR*)

License © Creative Commons BY 4.0 International license

© Jacques Carette, Gilles Dowek, and Catherine Dubois

Main reference Florian Rabe, Franziska Weber: “Morphism Equality in Theory Graphs”, in Proc. of the Intelligent Computer Mathematics – 16th International Conference, CICM 2023, Cambridge, UK, September 5–8, 2023, Proceedings, Lecture Notes in Computer Science, Vol. 14101, pp. 174–189, Springer, 2023.

URL https://doi.org/10.1007/978-3-031-42753-4_12

4.2.1 Motivation

Many concepts in mathematics have evolved over time, perhaps none more so than the concept of “space”. While it seemed to have settled on “topological space” for a certain time, it was then rethought later and today it seems that “locale” and “infinity groupoid” are both solid contenders for the modern notion. Many notions have a similarly rich ongoing history: integration (quadrature), function, equality, and so on. Recent examples of such rethinking abound (perfectoid spaces and condensed mathematics, homotopy type theory, and so on).

In parallel, it is equally clear that many concepts have a multitude of equivalent (or quasi-equivalent) formulations. For example, the number of different representations of graphs is quite astounding.

4.2.2 Purpose

The working group was formed to discuss how to enable proof assistants in particular, and mechanized mathematics systems in general, to deal with these issues.

4.2.3 Discussion

We first discussed the simpler issue, that of having multiple presentations and representations. Historically, the first realization of the importance of this is via *change of variables*: working on orbital mechanics in Cartesian coordinates is close to sheer insanity while being rather workable in spherical coordinates. This is a very common theme in mathematics where a change in point-of-view can make a substantial difference in how easy a concept is to handle. This ranges for simple issues like different axiomatizations of what is a group, to larger shifts such as switching from indexed categories to fibered categories (and more recently, using displayed categories for similar aims).

It is worthwhile remembering that Lie Symmetries are a classical example of this phenomenon: they are all about finding a “good” set of coordinates for which a PDE non-trivially simplifies. What makes them different is that one can *compute* what this change of viewpoint needs to be, while in most cases creativity is required.

Alas, current proof assistants are largely hampered by a sub-optimal design decision, namely that concepts are assumed to have a single “canonical” definition. While univalent mathematics promises that one can transport between equivalent concepts, no system has yet to build in features that facilitate this. In other words, there has been no engineering effort made to create user-friendly facilities to accommodate multiple representations. There does exist work [3, 4] that lays out the concept of a **Realm** which is supposed to encompass this very idea: a single concept with potentially many presentations and/or representations. Many engineering and usability hurdles remain.

Beyond convenience, the reason to want concepts to have multiple interfaces and sub-interfaces is to be able to allow more powerful *development by refinement*. Here the idea of sub-interfaces really shines: if we know in advance that certain methods will never be used,

it is sometimes the case that significantly more efficient representations and/or algorithms become feasible. For example, if we want a set representation where we can perform union, intersection and membership but neither count the number of elements or perform iteration on all elements, then there is a fast representation.

The overlap with the alignment working group was also discussed: for example, anyone who does constructive mathematics knows that there is no such singular concept as “finite set”. Rather, there are a multitude of subtly related concepts all of which are equivalent to “finite set” classically. Whether to lump them all together or not is a thorny issue.

We also discussed the inverse interface problem: given some definitions and proven properties, how to find a good set of axioms that would abstract over that? Many proof developments, in practice, contain natural “layers” where outer layers only depend a *de facto* interface given by a lower layer, but this interface is never made explicit. Can the process of finding these layers in a given development be automated? That would help us create **strong abstraction barriers**. Weak barriers are a real problem as some systems are rather eager to δ -expand, leading to proof scripts which are much lower-level than necessary. As different formalizations may give rise to very different interfaces, the need for Realms re-appears.


Another aspect of automation was discussed: transport by meta-programming. The idea here is to not just transport across isomorphisms “in theory” but to perform it as a meta-program that tries to eliminate the transport altogether by attempting to rephrase everything in terms of the target language. In particular, such transport meta-programs remain very useful even when Univalence is false.

4.2.4 Conclusion

It became clear that, given the scope of the problem and the short amount of time at the seminar, we could not obtain any tangible results beyond clearly documenting the problem. The group then dissolved early.

4.3 Formal Verification of Computer Algebra (Factorisation)

James H. Davenport (University of Bath, GB), Alex Best (King’s College – London, GB), Mario Carneiro (Carnegie Mellon University – Pittsburgh, US), and Edgar Costa (MIT – Cambridge, US)

License  Creative Commons BY 4.0 International license
© James H. Davenport, Alex Best, Mario Carneiro, and Edgar Costa

Of the problems listed by Davenport ([1] and talk here) we chose polynomial factorisation as having least mathematical pre-requisites. For computer algebra, we took FLINT as an easy-to access library of algorithms (specifically `factor_Zassenhaus`). As proof engine we took LEAN (familiarity). We have realised what we need as a certificate, seen that we can extract these data from FLINT (but could FLINT do a little more work to make the verification easier?), and have started formalising the required statements in LEAN. This has also led to identification of improvements to LEAN, and research questions in computer algebra. We are continuing to develop this ideas, in Davenport’s case as a challenge to his students’ LEAN study group.


References

- 1 James Harold Davenport. Proving an Execution of an Algorithm Correct? In Dubois and Kerber [2], pages 255–269.

- 2 Catherine Dubois and Manfred Kerber, editors. *Proceedings CICM 2023*, volume 14101 of *Springer Lecture Notes in Computer Science*, 2023.
- 3 Catherine Dubois and Manfred Kerber, editors. *Proceedings CICM 2023*, volume 14101 of *Springer Lecture Notes in Computer Science*, 2023.
- 4 F. Rabe and F. Weber. Morphism Equality in Theory Graphs. In C. Dubois and M. Kerber, editors, *Intelligent Computer Mathematics*, pages 174–189. Springer, 2023.

4.4 Coq, Isabelle and Dedukti as heterogeneous networks

Filip Koprivec (University of Ljubljana, SI), Mario Carneiro (Carnegie Mellon University – Pittsburgh, US), Stefania Dumbrava (ENSIIE – Paris, FR), Matej Petkovic (University of Ljubljana, SI), and Makarius Wenzel (Dr. Wenzel – Augsburg, DE)

License  Creative Commons BY 4.0 International license
 © Filip Koprivec, Mario Carneiro, Stefania Dumbrava, Matej Petkovic, and Makarius Wenzel

The working group worked on pushing the entries (that correspond to formalized mathematical concepts) from Coq, Isabelle and Dedukti systems into graph databases, on which machine learning can be applied.

To unify the schema of the created networks, we


- adjusted the RDF triplets describing the Coq entries,
- wrote an extension of Dedukti parser that extracts the necessary information from abstract syntax trees of the entries,
- and wrote parsing tool for Isabelle XMLs.

The end goal was include this work to the series of the existing networks, extracted from the largest Agda libraries and Mathlib4 library of Lean.

We thank Talia Ringer and Catherine Dobuis for the ideas.

4.5 Object identification using invariant based decision trees

Filip Koprivec (University of Ljubljana, SI) and Matej Petkovic (University of Ljubljana, SI)

License  Creative Commons BY 4.0 International license
 © Filip Koprivec and Matej Petkovic

Object identification up to isomorphism is in general computationally expensive (group isomorphism, graph isomorphism...) and identifying a given object with a representative from a known dataset is difficult.


But often, one is presented with object together with few pre-computed invariants or some invariants, that can quickly be computed from a given representation. Most of the time, the difficulty of calculating the invariant can be approximated relatively well or one can use the timings produced as a side effect of constructing the whole database of objects and associated invariants.

The working group was mostly focused on the problems where objects are graphs or groups and the preliminary experiments were carried out on the dataset, representing graphs. Every graph (a row in the data table) was represented by the number of invariants (columns in the dataset). Some of the values in the table were missing, i.e., not all the variants were computed for all the graphs. The working group experimented with a modification of decision tree algorithm that uses both information gain of an invariant and its calculation difficulty to compute the next invariant on which to split the dataset. With such an approach, we can speed up the identification of unknown object in an existing database of objects.

The results of experimentation: implemented decision tree variant and data acquisition from house of graphs together with evaluation are available at <https://github.com/Petkomat/invariant-computation-trees>.

4.6 Datasets

Dimitri Leemans (UL – Brussels, BE), Katja Berčič (University of Ljubljana, SI), Jan Goedgebeur (KU Leuven, BE), Darij Grinberg (Drexel Univ. – Philadelphia, US), Samuel Lelievre (University Paris-Saclay – Orsay, FR), Harshit J Motwani (Ghent University, BE), and Tom Kaspar Wiesing (Universität Erlangen-Nürnberg, DE)

License  Creative Commons BY 4.0 International license
 © Dimitri Leemans, Katja Berčič, Jan Goedgebeur, Darij Grinberg, Samuel Lelievre, Harshit J Motwani, and Tom Kaspar Wiesing

Participants in this working group worked on several smaller, but related tasks.

4.6.1 Cataloguing and/or indexing mathematical datasets.

1. Mathrepo: <https://mathrepo.mis.mpg.de/>, <https://ar5iv.labs.arxiv.org/html/2202.04022>
2. MathDB: <https://mathdb.mathhub.info/>

It may be possible to automate a part of the collection process by scraping Zenodo for mathematical datasets. This would probably require a list of mathematical concepts to be used in search queries. We identified a few considerations for catalogue/index projects:

- automatic uploading of datasets to Zenodo for long-term storage,
- accommodating reproducibility (recomputing the datasets, cf. Rescience),
- ratings and tags: independently recomputed, found same results; independently recomputed, found different results; not independently recomputed; formally certified.

4.6.2 Connecting databases and CAS.

Implementing an interface to the House of Graphs database in SageMath, including improving the House of Graph's API. The pull request can be viewed at <https://github.com/sagemath/sage/pull/36409>.

4.7 Aligning Mathematical Concepts Across Libraries

Florian Rabe (Universität Erlangen-Nürnberg, DE)

License  Creative Commons BY 4.0 International license
 © Florian Rabe

4.7.1 Motivation

Irrespective of the knowledge aspect (such as deduction, computation, data, or documentation), mathematical libraries share several abstract characteristics. Most importantly, they can be seen as a set of fragments each carrying a global identifier and describing a mathematical concept. Here we use *concept* as a generic term to subsume any named type, object, operation, theorem, or similar. For example, in a logical library, these are (mostly)

definitions and theorems; in a computer algebra library, they are (mostly) type and function definitions; in a concrete data set, they are the entries of the data set, often rows in a table; and in a document library, they are the articles explaining a concept.

The work group recognized the *alignment problem* [5, 6] as a major roadblock to interoperability across mathematical software systems: The mathematical concepts introduced by the various libraries overlap substantially, but there is no systematic connection between the same mathematical concept described in different libraries. Concretely, we call a pair of identifiers, typically but not necessarily from different libraries, an *alignment* if both are descriptions of the same concept. Then the alignment problem can be formulated as the challenge of (a) collecting alignments for existing large mathematical libraries and (b) leveraging these alignments for knowledge interchange.

The group identified three levels at which the problem can be attacked:

- At the *identifier level*, the alignments are just pairs of identifiers without a machine-checkable guarantee that they correspond to each other in any way.
- At the *expression level*, an alignment (c, d) additionally carries information how terms with head c can be translated to terms with head d . This translation can be quite complex and involve, e.g., changing (which may require computation), adding (which may require inference), omitting, or reordering arguments.
- At the *semantic level*, the alignments are additionally verified for correctness. This can be done, e.g., by translating theorems about c along the alignment and proving the translated theorems in the system of d . Critically, from (i) to (iii), task (a) becomes harder while task (b) becomes easier. But even identifier level alignments are useful, e.g., to cross-reference across libraries or to search for the same query in multiple libraries in parallel.

4.7.2 Results

The group surveyed the available technologies and alignment collection efforts and concluded that, while semantic alignments must be the ultimate goal, only for identifier level alignments is a major community-driven collection effort feasible at this point.

The group compiled the following existing collections of identifier alignments and concept lists:

- the Math Subject Classification (MSC)
- the nLab page titles (<https://ncatlab.org/nlab/>)
- the SMGloM concept and translation library [2]
- the concept list for the undergraduate math curriculum and the alignments into Lean Mathlib (<https://github.com/leanprover-community/mathlib4/blob/master/docs/undergrad.yaml>)
- the concept translation library maintained by Hosgood (<https://thosgood.com/maths-dictionary/>)
- the manual alignments collected for theorem prover libraries in [7]
- the concept list used by SageMath to align computer algebra systems integrated with SageMath (<https://doc.sagemath.org/html/en/reference/categories/index.html>)
- the MathGloss alignments for undergraduate math education [4] (<https://mathgloss.github.io/MathGloss/>)
- the relevant subset of the Wikidata ontology, which includes various alignments to informal libraries
- the nNexus alignments across informal libraries [1]
- the semantic alignments between HOL systems found by machine learning in [3]

The work identified a set of several hundred concepts that can be used as a seed for an alignment library and started using the above resources to compile alignments for it. These resources are collected at <https://github.com/UniFormal/alignments/>.

In order to scalably collect, maintain, and leverage alignment sets in the future, the group makes two recommendations:


- All developers of math libraries should add a feature to their tool that allows tagging definitions with the aligned identifier in a central concept list. The build process of the library should generate the list of alignments between those central concepts and the tagged identifiers in the system’s library. This list should be published alongside the library.
- Wikidata is suggested as the central concept list. This is motivated by the observation that Wikidata is a neutral library (in the sense of not being biased towards any research system or community) and the most likely to be scalably maintained and broadly used in the long term.

References

- 1 D. Ginev and J. Corneli. Nnexus reloaded. In S. Watt, J. Davenport, A. Sexton, P. Sojka, and J. Urban, editors, *Intelligent Computer Mathematics*, pages 423–426. Springer, 2014.
- 2 D. Ginev, M. Iancu, C. Jucovshi, A. Kohlhase, M. Kohlhase, A. Oripov, J. Schefter, W. Sperber, O. Teschke, and T. Wiesing. The smglom project and system: Towards a terminology and ontology for mathematics. In G. Greuel, T. Koch, P. Paule, and A. Sommese, editors, *Mathematical Software – ICMS*, volume 9725, pages 451–457. Springer, 2016.
- 3 T. Gauthier and C. Kaliszyk. Aligning concepts across proof assistant libraries. *Journal of Symbolic Computation*, 90:89–123, 2019.
- 4 L. Horowitz and V. de Paiva. Mathgloss: Linked undergraduate math concepts, 2023. <https://arxiv.org/abs/2311.12649>.
- 5 C. Kaliszyk, M. Kohlhase, D. Müller, and F. Rabe. A Standard for Aligning Mathematical Concepts. In A. Kohlhase, M. Kohlhase, P. Libbrecht, B. Miller, F. Tompa, A. Naummowicz, W. Neuper, P. Quaresma, and M. Suda, editors, *Work in Progress at CICM 2016*, pages 229–244. CEUR-WS.org, 2016.
- 6 D. Müller, T. Gauthier, C. Kaliszyk, M. Kohlhase, and F. Rabe. Classification of Alignments between Concepts of Formal Mathematical Systems. In H. Geuvers, M. England, O. Hasan, F. Rabe, and O. Teschke, editors, *Intelligent Computer Mathematics*, pages 83–98. Springer, 2017.
- 7 D. Müller, C. Rothgang, Y. Liu, and F. Rabe. Alignment-based Translations Across Formal Systems Using Interface Theories. In C. Dubois and B. Woltzenlogel Paleo, editors, *Proof eXchange for Theorem Proving*, pages 77–93. Open Publishing Association, 2017.

4.8 Scalability Estimates of Graph Certificates in a Theorem Prover Using SAT Encodings

Kathrin Stark (Heriot-Watt University – Edinburgh, GB), Madalina Erascu (West University of Timisoara, RO), Kazuhiko Sakaguchi (INRIA – Nantes, FR), and Jure Taslak (University of Ljubljana, SI)

License  Creative Commons BY 4.0 International license
© Kathrin Stark, Madalina Erascu, Kazuhiko Sakaguchi, and Jure Taslak

4.8.1 Motivation

What is the minimum amount of information needed to certify that a graph has a specific property? For some properties, these certificates are easy to provide. For example, to prove that a graph *is* Hamiltonian, one only requires describing a Hamiltonian path. But for some properties, it is not obvious what an efficient certificate is. For example, to prove that a graph is *not* Hamiltonian, the obvious certificates all get extremely large. SAT encodings are a general way to encode properties, and it is well-known that most graph predicates can be encoded via SAT. But how well does this scale?

This working group consists of experts on SMT solvers, theorem provers, and CoqELPI (Madalina Erascu, Kazuhiko Sakaguchi, Kathrin Stark, Jure Taslak). The aim was to create a proof of concept for end-to-end verified certificates for graph properties to check their scalability.

4.8.2 Previous Work

Certificate checking of (Un)SAT problems has been used in several papers. SMTCoq (Ekici et al. 2017, Keller 2019) is a certified checker for proof witnesses for the SAT solver ZChaff and the SMT solvers veriT and CVC4. Lammich uses the GRAT certificate for a verified SAT solver in Isabelle. Cruz-Filipe et al. implemented two certified LRAT checkers, one of them for the Coq proof assistant.

This working group decided to start off with the work by Cruz-Filipe et al., as this was a relatively lightweight development that was created in a theorem prover the group members were familiar with. The theorem prover further offered the ability to define the SAT encoding manually, making it easier to test scalability.

4.8.3 Overview

For any given graph, the approach consists of three steps. First, an unverified Python program generates a SAT encoding of the desired graph property and solves the encoding using a state-of-the-art SAT solver. This SAT solver also generates an LRAT certificate to demonstrate the correctness of the result. Next, Cruz-Filipe et al.'s approach is used to validate the LRAT certificate. Finally, a separate (constructive) Coq proof shows that the SAT encoding indeed corresponds to the desired abstract graph property. The abstract graph property uses a naive encoding of graphs, paths, and connectivity/Hamiltonicity in Coq. It typically requires a proof by contradiction and hence requires the decidability of the respective property. Using CoqELPI, we could moreover show that the SAT encoding of Python and the one used in Coq coincide. Overall, the construction thus provides a proof of the given property while leaving most of the heavy computational lifting to an external, highly-optimized SAT solver.

For the scalability tests in this seminar we considered two graph properties: connectivity and Hamiltonicity. We randomly generated graphs through a simple python program and checked at which node count the checker suffered a segmentation fault. The tests included both graphs that have and that do not have the respective property.

4.8.4 Results

The first results provided a first idea of scalability and showed that the approach scaled up to graphs that were around an order of magnitude bigger than initially expected.

For a proof of concept, the working group started with certificates for connectivity. The process described above was shown to scale up to 2500 nodes (both for connectivity and the absence of connectivity). An end-to-end correctness proof (with the assumed dependencies) required around 300 lines of code. It seems likely that the code could be shortened significantly by using a suitable graph library.

For certificates of being non-Hamiltonian, the first approaches scaled up to 50 nodes. In the available time, no end-to-end proof was implemented.

4.8.5 Future Work

A full end-to-end proof for being non-Hamiltonian would be an obvious next step. Another interesting direction would be to take a deeper look at graph certificates via SAT encodings in the literature. For this proof of concept, we chose the simplest encoding for non-Hamiltonian graphs. A more efficient SAT encoding should improve scalability accordingly and extend to further graph properties.

Participants

- Andrej Bauer
University of Ljubljana, SI
- Christoph Benz Müller
Universität Bamberg, DE
- Katja Bercic
University of Ljubljana, SI
- Alex Best
King's College – London, GB
- James Boyd
Wolfram Institute –
Cambridge, US
- Jacques Carette
McMaster University –
Hamilton, CA
- Mario Carneiro
Carnegie Mellon University –
Pittsburgh, US
- Edgar Costa
MIT – Cambridge, US
- James H. Davenport
University of Bath, GB
- Valeria de Paiva
Topos Institute – Berkeley, US
- Gilles Dowek
ENS – Gif-sur-Yvette, FR
- Catherine Dubois
ENSIIE – Evry, FR
- Stefania Dumbrava
ENSIIE – Paris, FR
- Madalina Erascu
West University of
Timisoara, RO
- Jan Goedgebeur
KU Leuven, BE
- Kiran Gopinathan
National University of
Singapore, SG
- Darij Grinberg
Drexel Univ. – Philadelphia, US
- Jyoti Jyoti
Panjab University –
Chandigarh, IN
- Michael Kohlhase
Universität Erlangen-
Nürnberg, DE
- Olexandr Kononov
University of St Andrews, GB
- Filip Koprivec
University of Ljubljana, SI
- Dimitri Leemans
UL – Brussels, BE
- Samuel Lelievre
University Paris-Saclay –
Orsay, FR
- Assia Mahboubi
INRIA – Nantes, FR
- Harshit J Motwani
Ghent University, BE
- Dennis Müller
Universität Erlangen-
Nürnberg, DE
- Tobias Nipkow
TU München – Garching, DE
- Matej Petkovic
University of Ljubljana, SI
- Christian Pfrang
Bay. Min. für Digitales –
München, DE
- Florian Rabe
Universität Erlangen-
Nürnberg, DE
- Talia Ringer
University of Illinois –
Urbana-Champaign, US
- Claudio Sacerdoti Coen
University of Bologna, IT
- Kazuhiko Sakaguchi
INRIA – Nantes, FR
- Natarajan Shankar
SRI – Menlo Park, US
- Kathrin Stark
Heriot-Watt University –
Edinburgh, GB
- Jure Taslak
University of Ljubljana, SI
- Nicolas Thiéry
University Paris-Saclay –
Orsay, FR
- Josef Urban
Czech Technical University –
Prague, CZ
- Makarius Wenzel
Dr. Wenzel – Augsburg, DE
- Tom Kaspar Wiesing
Universität Erlangen-
Nürnberg, DE



Accountable Software Systems

Bettina Könighofer^{*1}, Joshua A. Kroll^{*2}, Ruzica Piskac^{*3},
Michael Veale^{*4}, and Filip Cano Córdoba^{†5}

- 1 TU Graz, AT. bettina.koenighofer@iaik.tugraz.at
- 2 Naval Postgraduate School – Monterey, US. jkroll@jkroll.com
- 3 Yale University – New Haven, US. ruzica.piskac@yale.edu
- 4 University College London, GB. m.veale@ucl.ac.uk
- 5 TU Graz, AT. filip.cano@iaik.tugraz.at

Abstract

This report documents the program and the outcomes of Dagstuhl Seminar 23411 “Accountable Software Systems”. The seminar brought together an interdisciplinary group of researchers from the fields of formal methods, machine learning, philosophy, political science, law, and policy studies to address the critical issue of accountability in the development and deployment of software systems. As these systems increasingly assume roles within safety-critical domains of society, including transportation, healthcare, recruitment, and the judiciary, the seminar aimed to explore the multifaceted concept of accountability, its significance, and its implementation challenges in this context.

During the seminar, experts engaged deeply in discussions, presentations, and collaborative sessions, focusing on key themes such as the application of formal tools in socio-technical accountability, the impact of computing infrastructures on software accountability, and the innovation of formal languages and models to improve accountability measures. This interdisciplinary dialogue underscored the complexities involved in defining and operationalizing accountability, especially in light of technological advancements and their societal implications. The participants of the seminar reached a consensus on the pressing need for ongoing research and cross-disciplinary efforts to develop effective accountability mechanisms, highlighting the critical role of integrating socio-technical approaches and formal methodologies to enhance the accountability of autonomous systems and their contributions to society.

Seminar October 8–13, 2023 – <https://www.dagstuhl.de/23411>

2012 ACM Subject Classification Applied computing → Law, social and behavioral sciences

Keywords and phrases accountability, Responsible Decision Making, Societal Impact of AI

Digital Object Identifier 10.4230/DagRep.13.10.24

1 Executive Summary

Bettina Könighofer (TU Graz, AT)

Joshua A. Kroll (Naval Postgraduate School – Monterey, US)

Ruzica Piskac (Yale University – New Haven, US)

Michael Veale (University College London, GB)

License © Creative Commons BY 4.0 International license
© Bettina Könighofer, Joshua A. Kroll, Ruzica Piskac, and Michael Veale

Accountability in the context of software is an emerging area that has attracted interest in disparate fields from computing and information science to philosophy to political science to law and policy studies. Presently, there is an increasing number of autonomous agents

* Editor / Organizer

† Editorial Assistant / Collector



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

Accountable Software Systems, *Dagstuhl Reports*, Vol. 13, Issue 10, pp. 24–49

Editors: Bettina Könighofer, Joshua A. Kroll, Ruzica Piskac, and Michael Veale



Dagstuhl Reports

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

shifting into safety-critical aspects of society and our everyday lives. Autonomously driven vehicles share the roads with us. Computerized reasoning aids healthcare providers in disease diagnostics, recruiters in hiring, and even adjudicators in analyzing pretrial flight risks. While a common goal of these autonomous agents is to assist and improve human lives, it is a harsh reality that the opposite occurs far too frequently. Car accidents involving autonomously driven vehicles have been fatal, and bias-vulnerable autonomous decision-making has led to discriminatory assessment and abuse of individuals. As a result, accountability is started to be investigated in areas including AI, machine learning, control, systems development, data management, software engineering, programming languages, human factors/human-computer interaction, and formal verification communities.

Understanding accountability and the place of formal computing tools in assigning it has been recognized by funding agencies as an important research topic. Significant reforms are currently occurring in the law to the liability of autonomous systems, such as the proposed EU Artificial Intelligence Act; reforms to the EU Machinery Directive and Product Liability Directive; standardization processes around software and accountability at the ISO, IEEE, CEN/CENELEC, US NIST, and ETSI, among others; a range of proposals around reforms to intermediary liability and recommender systems in the US Congress and proposals from the White House Office of Science and Technology Policy, among many other ongoing changes and revisions to existing policies and recommendations for new policies around the world. Throughout the Dagstuhl Seminar, participants engaged in a rich tapestry of discussions, presentations, and working groups, focusing on three main areas:

- Justifying Assurance: Opportunities and Limits of Formal Tools for Accountability in Sociotechnical Context
- The Role(s) of Computing and Infrastructures in Accountability for Software
- New Formal Specification Languages and Modeling Techniques for Accountability

Group discussions further enriched the seminar's discourse. For example, debates around the Software as a Service (SaaS) model and the potential for a public, democratically managed cloud infrastructure highlighted the evolving nature of software maintenance and the democratization of digital infrastructure. Another group delved into the intricacies of software complexity, distinguishing between accountability and responsibility, and discussing the challenges of regulatory adaptation to technological innovation.

By having short talks to inspire discussions followed by in-depth discussions in breakout groups, the seminar successfully forged new conversations between the communities with a focus on how to improve upon the state of the art and practice in designing software systems with respect to accountability. Participants collectively recognized the importance of socio-technical approaches and formal methods in developing robust accountability frameworks, advocating for synergies between different fields to refine the state of the art and practice.

The seminar concluded with a group exercise to map and group open questions, which are documented at the end of this report. This exercise naturally provided more questions than answers, but these questions were coalescing on deep and specific interdisciplinary challenges that participants intended to take forwards in their future work.

In conclusion, the Dagstuhl Seminar on “Accountable Software Systems” served as a pivotal forum for cross-disciplinary exchange, catalyzing new conversations and potential collaborations.

2 Table of Contents

Executive Summary

Bettina Könighofer, Joshua A. Kroll, Ruzica Piskac, and Michael Veale 24

Overview of Talks

Analyzing Intentional Behavior in Autonomous Agents under Uncertainty
Filip Cano Córdoba 28

Is Software Eaten by the Cloud?
Corinne Cath 28

Causal Explanations: What Can Computer Scientists Do for Accountability
Hana Chockler 29

What Does Data Erasure Mean? What Should Data Erasure Mean?
Aloni Cohen 30

Complexity Effects on a Highly-Accountable System Containing Safety-Critical Software
Misty Davies 30

Accountable Software Systems: Lessons from System Safety
Roel Dobbe 31

Accountability in Computing: Concepts and Mechanisms
Joan Feigenbaum 31

AI is a Mushroom
Jake Goldenfein 32

Accountable Legal Decision Support?
Thomas T. Hildebrandt 32

Platforms, Sovereignty, and Software Accountability
Divij Joshi 34

“Put the Car on the Stand”: SMT-based Oracles for Investigating Decisions
Samuel Judson 34

Algorithmic Systems Through an Ethnographic Lens
Daan Kolkman 35

Verification of Accountability in Protocols with Tamarin
Robert Künnemann 36

Accountability Lessons Learned from the Design and Deployment of Digital Contact Tracing
Wouter Lueks 36

Accountability and Explainability of French Housing Benefits Computation
Denis Merigoux 37

An AI Transparency Register for the Public Sector
Matthias Spielkamp 37

Responsibility and Liability regarding Software and AI
Rüdiger Wilhelmi 38

Scenic: A Probabilistic Scenario Description Language <i>Beyazit Yalcinkaya</i>	39
Towards a Framework for Certification of Reliable Autonomous Systems <i>Neil Yorke-Smith</i>	40
Working groups	
Working groups topic discussions	40
Forms of (Un)Accountability in Contemporary Software Ecosystems: Group 1 <i>Wouter Lueks and Scott Shapiro</i>	41
Forms of (Un)Accountability in Contemporary Software Ecosystems: Group 2 <i>Roel Dobbe</i>	42
Forms of (Un)Accountability in Contemporary Software Ecosystems: Group 3 <i>Bettina Könighofer and Neil Yorke-Smith</i>	42
Forms of (Un)Accountability in Contemporary Software Ecosystems: Group 4 <i>Beyazit Yalcinkaya</i>	43
Software Ecosystem Futures: Group 1 <i>Divij Joshi</i>	44
Software Ecosystem Futures: Group 2 <i>Filip Cano Córdoba</i>	45
Software Ecosystem Futures: Group 3 <i>Neil Yorke-Smith</i>	45
Open problems	
Concluding Exercise: Open Questions <i>Michael Veale, Thomas Arnold, Filip Cano Córdoba, Corinne Cath, Hana Chockler, Aloni Cohen, Misty Davies, Roel Dobbe, Joan Feigenbaum, David Fuenmayor, Ashish Gehani, Jake Goldenfein, Thomas T. Hildebrandt, Divij Joshi, Samuel Judson, Daan Kolkman, Bettina Könighofer, Joshua A. Kroll, Robert Künnemann, Stefan Leue, Wouter Lueks, Rupak Majumdar, Kira Matus, Denis Merigoux, Ruzica Piskac, Scott Shapiro, Jatinder Singh, Matthias Spielkamp, Rüdiger Wilhelm, Beyazit Yalcinkaya, and Neil Yorke-Smith</i>	46
Participants	49

3 Overview of Talks

3.1 Analyzing Intentional Behavior in Autonomous Agents under Uncertainty

Filip Cano Córdoba (TU Graz, AT)

License © Creative Commons BY 4.0 International license
© Filip Cano Córdoba

Joint work of Filip Cano Córdoba, Samuel Judson, Timos Antonopoulos, Katrine Bjørner, Nicholas Shoemaker, Scott J. Shapiro, Ruzica Piskac, Bettina Könighofer

Main reference Filip Cano Córdoba, Samuel Judson, Timos Antonopoulos, Katrine Bjørner, Nicholas Shoemaker, Scott J. Shapiro, Ruzica Piskac, Bettina Könighofer: “Analyzing Intentional Behavior in Autonomous Agents under Uncertainty”, in Proc. of the Thirty-Second International Joint Conference on Artificial Intelligence, IJCAI 2023, 19th-25th August 2023, Macao, SAR, China, pp. 372–381, ijcai.org, 2023.

URL <https://doi.org/10.24963/IJCAI.2023/42>

Principled accountability for autonomous decision-making in uncertain environments requires distinguishing intentional outcomes from negligent designs from actual accidents. We propose analyzing the behavior of autonomous agents through a quantitative measure of the evidence of intentional behavior. We model an uncertain environment as a Markov Decision Process (MDP). For a given scenario, we rely on probabilistic model checking to compute the ability of the agent to influence reaching a certain event. We call this the scope of agency. We say that there is evidence of intentional behavior if the scope of agency is high and the decisions of the agent are close to being optimal for reaching the event. Our method applies counterfactual reasoning to automatically generate relevant scenarios that can be analyzed to increase the confidence of our assessment. In a case study, we show how our method can distinguish between “intentional” and “accidental” traffic collisions.

3.2 Is Software Eaten by the Cloud?

Corinne Cath (TU Delft, NL & University of Cambridge, GB)

License © Creative Commons BY 4.0 International license
© Corinne Cath

In 2011, Marc Andreessen well-known tech investor and inventor of the short-lived but influential Mosaic browser, proclaimed that “software is eating the world”. This was a period of rapid digitization during which time more and more industries were running on software – and changing their business models to one of delivering online services rather than selling one-off products. A second, but perhaps more silent revolution accompanied by the glutenous expansion of software and its as-a-service business model, is happening today. This one is not being splashed across the front page of the Wall Street Journal. Namely the growing importance of cloud computing, as the infrastructure for the “agile” production and distribution of software [1]. In this talk, I focus on the growing importance of cloud computing in the context of the future of software ecosystems. In particular, I argue that the future of software lives on the cloud. Cloud computing is the on-demand availability of computing resources (such as storage and infrastructure), as services over the internet, at scale. It ostensibly eliminates the need for individuals and businesses to self-manage physical resources and allows them to pay for what they use. Many software companies buy computing resources from a small set of tech behemoths, including Google, Microsoft, and AWS. But this perceived convenience comes at a price. I draw on several recent case studies that outline

the importance of cloud computing for the recent boom in the development of generative AI (GenAI) products, like ChatGPT and Claude. I demonstrate that when we talk about GenAI, we implicitly assume that this software runs on the cloud, given GenAI's specific computational requirements. This cloud reliance creates distinct dependencies between the software ecosystem and cloud computing companies, their financial models and political priorities, which have accountability ramifications that require further research.

References

- 1 Gürses, Seda, and Joris van Hoboken. 2018. "Privacy after the Agile Turn." In *The Cambridge Handbook of Consumer Privacy*, edited by Evan Selinger, Jules Polonetsky, and Omer Tene, 579–601. Cambridge Law Handbooks. Cambridge: Cambridge University Press. <https://doi.org/10.1017/9781316831960.032>.

3.3 Causal Explanations: What Can Computer Scientists Do for Accountability

Hana Chockler (King's College London, GB)

License © Creative Commons BY 4.0 International license
© Hana Chockler

Computerised systems are increasingly opaque, due to their size and to the nature of the new AI systems, such as neural networks. Yet, we have to be able to reason about the correctness of these systems, debug them, fix errors, answer "what if" questions, and explain their decisions. In this talk, I present a framework for causal explanations of image classifiers, based on the notions of actual causality and explainability. Our tool ReX provides explanations that approximate minimal subsets of the image sufficient to get the same classification if the rest of the image is hidden. I describe our recent work in extending ReX to multiple explanations and survey the challenges in adapting the existing explainability tools to medical applications.

References

- 1 Chockler and Halpern. "Responsibility and Blame: A Structural-Model Approach". *J. Artif. Intell. Res.* 22: 93-115 (2004)
- 2 Chockler, Halpern, Kupferman. What causes a system to satisfy a specification? *ACM Trans. Comput. Log.* 9(3): 20:1-20:26 (2008)
- 3 Aleksandrowicz, Chockler, Halpern, Ivrii. "The Computational Complexity of Structure-Based Causality". *AAAI'14*: 974-980.
- 4 Alrajeh, Chockler, Halpern. "Combining Experts' Causal Judgments". *Artif. Intell.* (2020).
- 5 Sun, Chockler, Huang, Daniel Kroening. "Explaining Image Classifiers Using Statistical Fault Localization". *ECCV'20*: 391-406.
- 6 Chockler, Kroening, Sun. "Explanations for Occluded Images". *ICCV'21*: 1234-1243.
- 7 Pouget, Chockler, Sun, Kroening. "Ranking Policy Decisions". *NeurIPS'21*.
- 8 Chockler, Halpern. "On Testing for Discrimination Using Causal Models". *AAAI'22*.

3.4 What Does Data Erasure Mean? What Should Data Erasure Mean?

Aloni Cohen (University of Chicago, US)

License © Creative Commons BY 4.0 International license
© Aloni Cohen

Joint work of Aloni Cohen, Adam D. Smith, Marika Swanberg, Prashant Nalini Vasudevan
Main reference Aloni Cohen, Adam D. Smith, Marika Swanberg, Prashant Nalini Vasudevan: “Control, Confidentiality, and the Right to be Forgotten”, in Proc. of the 2023 ACM SIGSAC Conference on Computer and Communications Security, CCS 2023, Copenhagen, Denmark, November 26-30, 2023, pp. 3358–3372, ACM, 2023.

URL <https://doi.org/10.1145/3576915.3616585>

Recent digital rights frameworks stipulate an ability to request that a data controller – roughly, any system that stores and manipulates personal information – “erase” or “delete” one’s data (e.g., the “right to be forgotten” in the GDPR). We ask how deletion should be formalized in complex systems that interact with many parties and store derivative information. There are two broad principles at work in existing approaches to formalizing deletion: confidentiality and control. This talk briefly explores these questions in the specific context of machine learning models, and erasing training data therefrom.

3.5 Complexity Effects on a Highly-Accountable System Containing Safety-Critical Software

Misty Davies (NASA – Moffett Field, US)

License © Creative Commons BY 4.0 International license
© Misty Davies

This talk began with a quick overview of the regulatory landscape for aviation. Today, there are distinctly different regulatory process paths for hardware/software system components and for the human operators and their roles in the overall system. Today, the handling of almost all remaining uncertainty is pushed into the operational assurance processes. This means that hardware and software components are expected to be highly deterministic. As we increasingly push required system functionality from execution by people into execution by automation we are running into the limits of what our current processes allow us to assure. This is broadly perceived as a barrier to innovation.

One way forward could be a shift to assurance models that merge and blend our current paradigms for silicon-based and human-based assurance.

References

- 1 Brat, Guillaume P., Huafeng Yu, Ella Atkins, Prashin Sharma, Darren Cofer, Michael Durling, Baoluo Meng et al. *Autonomy Verification & Validation Roadmap and Vision 2045*. No. NASA/TM-20230003734. 2023.

3.6 Accountable Software Systems: Lessons from System Safety

Roel Dobbe (TU Delft, NL)

License © Creative Commons BY 4.0 International license
© Roel Dobbe

Main reference Roel Dobbe: “System Safety and Artificial Intelligence”, in Proc. of the FAccT ’22: 2022 ACM Conference on Fairness, Accountability, and Transparency, Seoul, Republic of Korea, June 21 – 24, 2022, p. 1584, ACM, 2022.

URL <https://doi.org/10.1145/3531146.3533215>

We draw inspiration from the field of system safety to understand challenges with accountability in the development, use and governance of software systems. System safety has dealt with accidents and harm in safety-critical systems subject to varying degrees of software-based automation, already for many decades. As a result, it has some crystallized lessons, tools and ontologies that establish conditions for accountability to promote or guarantee safety, as well as to inform the design of actual mechanisms for accountability. We covered two core concepts that capture such conditions and mechanisms. At the operational level, the process model which details the goals, constraints, actions, observations and models/knowledge needed to safely operate processes subject to software-based automation. At the organizational and institutional levels, the safety control structure combines, describes and relates all mechanisms contributing to safety, including the operational mechanisms, and extending towards managerial, maintenance and research and development. Moving further up, the control structure also includes various regulatory, policy, advocacy law-making, law-enforcing and judicial mechanisms. System safety methods allow for the integral analysis of unsafe or otherwise undesirable behaviors and outcomes in software systems, helping to understand how various mechanisms contribute to material outcomes, either directly in a causal sense or indirectly by providing the environmental factors for such system behavior and outcomes to emerge. We concluded with a few lessons written down in Nancy Leveson’s magnum opus “Engineering a Safer World: Systems Thinking Applied to Safety”, including the curse of flexibility and the crucial importance of culture in upholding accountability mechanisms.

References

- 1 Roel Dobbe. System Safety and Artificial Intelligence. In The Oxford Handbook of AI Governance. Oxford University Press, 2022, <https://doi.org/10.1093/oxfordhb/9780197579329.001.0001>.
- 2 Nancy G. Leveson. Engineering a Safer World: Systems Thinking Applied to Safety. MIT Press, 2012. <http://ebookcentral.proquest.com/lib/delft/detail.action?docID=3339365>

3.7 Accountability in Computing: Concepts and Mechanisms

Joan Feigenbaum (Yale University – New Haven, US)

License © Creative Commons BY 4.0 International license
© Joan Feigenbaum

Joint work of Joan Feigenbaum, Aaron D. Jaggard, Rebecca N. Wright

Main reference Joan Feigenbaum, Aaron D. Jaggard, Rebecca N. Wright: “Accountability in Computing: Concepts and Mechanisms”, Found. Trends Priv. Secur., Vol. 2(4), pp. 247–399, 2020.

URL <https://doi.org/10.1561/3300000002>

This firestarter talk was a brief “teaser” of the work in [1].

Accountability is a widely studied but amorphous concept, used to mean different things across different disciplines and domains of application. Here, we survey work on accountability in computer science and other disciplines. We motivate our survey with a study of the myriad

ways in which the term “accountability” has been used across disciplines and the concepts that play key roles in defining it. This leads us to identify a temporal spectrum onto which we may place different notions of accountability to facilitate their comparison. We then survey accountability mechanisms for different application domains in computer science and place them on our spectrum. Building on this broader survey, we review frameworks and languages for studying accountability in computer science. Finally, we offer conclusions, open questions, and future directions.

References

- 1 Joan Feigenbaum, Aaron D. Jagard, and Rebecca N. Wright (2020). Accountability in Computing: Concepts and Mechanisms. *Foundations and Trends in Privacy and Security*: Vol. 2: No. 4, pp 247-399.

3.8 AI is a Mushroom

Jake Goldenfein (The University of Melbourne, AU)

License © Creative Commons BY 4.0 International license
© Jake Goldenfein

Main reference Jake Goldenfein: “Privacy’s Lose Grip: Law and the Operational Image” in Zalnierute et al (eds) *Cambridge Handbook on the Regulation of Facial Recognition in the Modern State* (CUP 2024) (forthcoming)

A number of scholars have identified the complexity of locating organisational accountability in AI products because of complex industrial arrangements. But there are similar complexities in the dataset supply chain for AI models. Datasets used to train commercial and noncommercial models are produced and refined by numerous organisations with different characteristics and for different purposes. These inter-organisational forms of coordination and strategic partnership result in complex “ecologies” of data flow subject to complex forms of regulatory arbitrage. But what kind of “ecology” is this? Inspired by the work of Anna Tsing, the goal here is to understand or conceptualise these mutualistic relations and multi-organisational associations, the data flows they coordinate, and the products they produce as a type of mushroom, whose arrangement depends on a broader political economy.

3.9 Accountable Legal Decision Support?

Thomas T. Hildebrandt (University of Copenhagen, DK)

License © Creative Commons BY 4.0 International license
© Thomas T. Hildebrandt

Joint work of Amine Abbad Andaloussi, Lars Rune Christensen, Søren Debois, Nicklas Pape Healy, Hugo A. López, Morten Marquard, Naja L. Holten Møller, Anette C. M. Petersen, Tijs Slaats, Barbara Weber

Main reference Thomas T. Hildebrandt, Amine Abbad Andaloussi, Lars Rune Christensen, Søren Debois, Nicklas Pape Healy, Hugo A. López, Morten Marquard, Naja L. Holten Møller, Anette C. M. Petersen, Tijs Slaats, Barbara Weber: “EcoKnow: Engineering Effective, Co-created and Compliant Adaptive Case Management Systems for Knowledge Workers”, in *Proc. of the ICSSP ’20: International Conference on Software and System Processes*, Seoul, Republic of Korea, 26-28 June, 2020, pp. 155–164, ACM, 2020.

URL <https://doi.org/10.1145/3379177.3388908>

Case management and business processes regulated by law are increasingly being digitalized, in the public as well as the private sector. At the same time, laws are continuously changed and new regulations are introduced. Moreover, law needs to be open for interpretation

and re-interpretation. This calls for new methods to ensure accountability of the legal decision support. We present the DCR Graph tools and notation developed and tested in the EcoKnow.org project, which allows domain experts to map the law to a DCR Graph (which is essentially a temporal logic knowledge graph) and validate the meaning through simulation. Via the tools developed by the research-based company DCR Solutions, the graphs can be used to provide dynamic guidelines, case management support and automation, and is used widely in the public sector in Denmark via the KMD WorkZone Enterprise Information Management system.

References

- 1 Christoffer Olling Back, Tijs Slaats, Thomas Troels Hildebrandt, Morten Marquard: Discover: accurate and efficient discovery of declarative process models. *Int. J. Softw. Tools Technol. Transf.* 24(4): 563-587, 2022
- 2 Vlad Paul Cosma, Thomas T. Hildebrandt, Christopher H. Gyldenkærne, Tijs Slaats. *BER-MUDA: Participatory Mapping of Domain Activities to Event Data via System Interfaces*. ICPM Workshops 2022: 127-139, 2022
- 3 Asbjørn Ammitzbøll Flügge, Thomas T. Hildebrandt, Naja L. Holten Møller. *Street-Level Algorithms and AI in Bureaucratic Decision-Making: A Caseworker Perspective*. *Proc. ACM Hum. Comput. Interact.* 5(CSCW1): 40:1-40:23, 2021
- 4 Anette C. M. Petersen, Lars Rune Christensen, Richard Harper, Thomas T. Hildebrandt, “We Would Never Write That Down”: *Classifications of Unemployed and Data Challenges for AI*. *Proc. ACM Hum. Comput. Interact.* 5(CSCW1): 102:1-102:26, 2021
- 5 Amine Abbad Andaloussi, Francesca Zerbato, Andrea Burattin, Tijs Slaats, Thomas T. Hildebrandt, Barbara Weber. *Exploring how users engage with hybrid process artifacts based on declarative process models: a behavioral analysis based on eye-tracking and think-aloud*. *Softw. Syst. Model.* 20(5): 1437-1464. 2021
- 6 Håkon Norman, Søren Debois, Tijs Slaats, Thomas T. Hildebrandt. *Zoom and Enhance: Action Refinement via Subprocesses in Timed Declarative Processes*. *BPM 2021*: 161-178
- 7 Thomas T. Hildebrandt, Håkon Normann, Morten Marquard, Søren Debois, Tijs Slaats. *Decision Modelling in Timed Dynamic Condition Response Graphs with Data*. *Business Process Management Workshops 2021*: 362-374, 2021
- 8 Anette Chelina Møller Petersen, Lars Rune Christensen, Thomas T. Hildebrandt: *The Role of Discretion in the Age of Automation*. *Comput. Support. Cooperative Work.* 29(3): 303-333, 2020
- 9 Hugo A. López, Søren Debois, Tijs Slaats, Thomas T. Hildebrandt. *Business Process Compliance Using Reference Models of Law*. *FASE 2020*: 378-399, 2020
- 10 Asbjørn William Ammitzbøll Flügge, Thomas T. Hildebrandt, Naja L. Holten Møller. *Algorithmic Decision Making in Public Services: A CSCW-Perspective*. *GROUP (Companion) 2020*: 111-114, 2020
- 11 Søren Debois, Hugo A. López, Tijs Slaats, Amine Abbad Andaloussi, Thomas T. Hildebrandt. *Chain of Events: Modular Process Models for the Law*. *IFM 2020*: 368-386, 2020
- 12 Naja L. Holten Møller, Irina Shklovski, Thomas T. Hildebrandt. *Shifting Concepts of Value: Designing Algorithmic Decision-Support Systems for Public Services*. *NordiCHI 2020*: 70:1-70:12, 2020
- 13 Thomas T. Hildebrandt, et al. *EcoKnow: Engineering Effective, Co-created and Compliant Adaptive Case Management Systems for Knowledge Workers*. *ACM, Proceedings of ICSSP '20: International Conference on Software and System Processes, Seoul, Republic of Korea, 26-28 June, 2020*

3.10 Platforms, Sovereignty, and Software Accountability

Divij Joshi (University College London, GB)

License  Creative Commons BY 4.0 International license
© Divij Joshi

Main reference Cohen, Julie E.: “Law for the platform economy.” UCDL Rev. 51 (2017): 133.

URL <https://scholarship.law.georgetown.edu/facpub/2015/>


This talk examines why the “platform” model is crucial to thinking about software accountability. Software production, deployment and use is increasingly intermediated through the organisational form of the “platform” – generally private entities which control access to computational resources as well as structure and organise the “market” for software products. Through varying levels of control over elements of a software production or deployment “stack”, the platform model is crucial for thinking about accountability in software production.

As platforms increasingly determine access to computational resources and the ability to run softwares, they exercise power over users or populations in ways that may be undemocratic or unjust. Smartphone OS providers can arbitrarily expand the surveillance capabilities of mobile devices or IoT devices, and “App Stores” can similarly chose to unilaterally force changes to software production business models, as we have seen in the recent past.

This also places limits on how we can democratically and legitimately control computation and software production or use that is increasingly “infrastructural”, i.e. a common, shared resource that is vital for conducting an array of activities. Given that these infrastructures are globally distributed and usually privately controlled, nation states or other legitimate publics have little insight or control into how governance decisions are made on platforms. At the level of the state, this is also giving rise to calls for expanding upon “digital sovereignty” through a number of measures, including regulation, or creating public alternatives.

3.11 “Put the Car on the Stand”: SMT-based Oracles for Investigating Decisions

Samuel Judson (Yale University – New Haven, US)

License  Creative Commons BY 4.0 International license
© Samuel Judson

Joint work of Samuel Judson, Matthew Elacqua, Filip Cano Córdoba, Timos Antonopoulos, Bettina Könighofer, Scott J. Shapiro, Ruzica Piskac

Main reference Samuel Judson, Matthew Elacqua, Filip Cano Córdoba, Timos Antonopoulos, Bettina Könighofer, Scott J. Shapiro, Ruzica Piskac: “‘Put the Car on the Stand’: SMT-based Oracles for Investigating Decisions”, CoRR, Vol. abs/2305.05731, 2023.

URL <https://doi.org/10.48550/ARXIV.2305.05731>

Principled accountability in the aftermath of harms is essential to the trustworthy design and governance of algorithmic decision making. Legal philosophy offers a paramount method for assessing culpability: putting the agent “on the stand” to subject their actions and intentions to cross-examination. We show that under minimal assumptions automated reasoning can rigorously interrogate algorithmic behaviors as in the adversarial process of legal fact finding. We use the formal methods of symbolic execution and satisfiability modulo theories (SMT) solving to discharge queries about agent behavior in factual and counterfactual scenarios, as adaptively formulated by a human investigator. We implement our framework and demonstrate its utility on an illustrative car crash scenario.

3.12 Algorithmic Systems Through an Ethnographic Lens

Daan Kolkman (TU Eindhoven, NL)

License  Creative Commons BY 4.0 International license
 Daan Kolkman

Although algorithms are imbued with a sense of objectivity and reliability, numerous high-profile incidents have demonstrated their fallibility. Despite their ubiquity, access to algorithms and algorithmic systems is typically policed, meaning that our understanding of what it is that people who develop algorithms “do” and why algorithms fail in practice remains largely unexplored. In a way, these systems that are designed for structuring and ordering, themselves often resist straightforward categorization and control. This observation echoes Seaver’s concept [3] of software systems as “culturally enacted” through user interaction, challenging the notion of these systems as fixed technical objects. My approach, while similar in nature to Seaver’s, took a slightly different point of departure. Specifically, I built on studies by Beunza and Garud [1], Beunza and Stark [2], and Spears [4] that – like Seaver – center on understanding the socio-material context of algorithms. However, such work in the Social Studies of Finance domain also focuses on understanding the “technical” aspects of algorithms themselves. In a recent case study, I investigate the development – and demise – of an algorithmic system that used Wi-Fi data as an input for footfall measurement in the Netherlands. The study utilizes an ethnographic approach to track the development and eventual abandonment of this system that measured footfall in 1100 locations across the Netherlands. Based on the case study I conclude that, in comparison to the manual collection of footfall measurement, the transition to more digital modes of data collection has several unique characteristics that matter for the processes of knowledge production: 1. The speed and volume of these new modes of data collection is much higher; 2. The new modes of data collection are relatively untried, hard to calibrate, and in general more error-prone; 3. New modes of data collection can be – and are – contested on the basis of non-digital observations. In unison this leads to the – attempted- alignment of divergent measures of footfall and integration previously isolated epistemic cultures. In pursuit of more understanding of – and more influence over – algorithmic systems, ethnographic studies like these may contribute by shedding light on the mundane practices of those involved in their development and use.

References

- 1 Beunza, D., & Garud, R. (2007). Calculators, lemmings or frame-makers? The intermediary role of securities analysts. *The Sociological Review*, 55 (2 suppl), 13-39
- 2 From dissonance to resonance: Cognitive interdependence in quantitative finance. *Economy and society*, 41(3), 383-417
- 3 Algorithms as culture: Some tactics for the ethnography of algorithmic systems. *Big Data & Society*, 4 (2), 1-12
- 4 Engineering value, engineering risk: what derivatives quants know and what their models do. PhD Thesis

3.13 Verification of Accountability in Protocols with Tamarin

Robert Künnemann (CISPA – Saarbrücken, DE)

License © Creative Commons BY 4.0 International license
© Robert Künnemann

Joint work of Robert Künnemann, Kevin Morio, Ilkan Esiyok

Main reference Kevin Morio, Robert Künnemann: “Verifying Accountability for Unbounded Sets of Participants”, in Proc. of the 34th IEEE Computer Security Foundations Symposium, CSF 2021, Dubrovnik, Croatia, June 21-25, 2021, pp. 1–16, IEEE, 2021.

URL <https://doi.org/10.1109/CSF51468.2021.00032>

The Tamarin [1] protocol verification tool (<https://tamarin-prover.github.io/>) integrates support for verifying protocol accountability in the following sense: “protocol provides accountability for property iff, at all times, the protocol can correctly determine the parties causing violation of this property”. We first explain that we need causation to define this property, and that, more precisely, causes ought to be the fact whether or whether not a party deviated from protocol behavior. Then we outline how this is done, and how this technique developed to support an unbounded number of parties. We talk about the case studies we performed and demonstrate the Tamarin tool and its web GUI.

References

- 1 Meier, Simon, et al. “The TAMARIN prover for the symbolic analysis of security protocols.” Computer Aided Verification: 25th International Conference, CAV 2013, Saint Petersburg, Russia, July 13-19, 2013. Proceedings 25. Springer Berlin Heidelberg, 2013.

3.14 Accountability Lessons Learned from the Design and Deployment of Digital Contact Tracing

Wouter Lueks (CISPA – Saarbrücken, DE)

License © Creative Commons BY 4.0 International license
© Wouter Lueks

Main reference Carmela Troncoso, Dan Bogdanov, Edouard Bugnion, Sylvain Chatel, Cas Cremers, Seda F. Gürses, Jean-Pierre Hubaux, Dennis Jackson, James R. Larus, Wouter Lueks, Rui Oliveira, Mathias Payer, Bart Preneel, Apostolos Pyrgelis, Marcel Salathé, Theresa Stadler, Michael Veale: “Deploying decentralized, privacy-preserving proximity tracing”, Commun. ACM, Vol. 65(9), pp. 48–57, 2022.

URL <https://doi.org/10.1145/3524107>

With the onset of the global pandemic in early 2020 came a massive push to deploy digital technologies to aid in pandemic mitigation. In this talk, we looked back at how the DP3T project proposed a privacy-friendly approach to digital contact tracing [1], eventually leading to its adoption by Google and Apple. This adoption was an essential enabler of the use of these technologies, but also surfaces questions of accountability. What does it mean to design solutions on such a short timescale? How can citizens convince themselves that the privacy protections are implemented as designed? And what does it mean for Google and Apple to control what health applications can and cannot do [2]?

References

- 1 Carmela Troncoso, Dan Bogdanov, Edouard Bugnion, Sylvain Chatel, Cas Cremers, Seda F. Gürses, Jean-Pierre Hubaux, Dennis Jackson, James R. Larus, Wouter Lueks, Rui Oliveira, Mathias Payer, Bart Preneel, Apostolos Pyrgelis, Marcel Salathé, Theresa Stadler, Michael Veale: *Deploying decentralized, privacy-preserving proximity tracing*. Commun. ACM 65(9): 48-57 (2022)
- 2 James R. Larus: *Whose smartphone is it?* Commun. ACM 64(9): 41-42 (2021)

3.15 Accountability and Explainability of French Housing Benefits Computation

Denis Merigoux (INRIA – Paris, FR)

License © Creative Commons BY 4.0 International license

© Denis Merigoux

Joint work of Marie Alauzen, Émile Rolley, Louis Gesbert, Justine Banuls

How French government agencies explain decisions taken by an IT system following the law? Typical example is the amount of taxes and social benefit you get/pay once you've filled the form. With several other authors, we have done : a state of the art, interviews with social workers and social benefits agency agents about what use they have for explainability, how this plays out with respect to accountability, and prototyping automatic generation of law-based explanations using the Catala programming language.

Investigated model of transparency, in french law you have to publish the source code. In the case of housing benefits distributed by CNAF, cobol code from 1995, unreadable. So that is a problem. French law forces them to publish a summary of what the algorithm is doing but you can't condense 300 pages in 2-pages that they published, if you add all the edge cases together you get the entire population. Last requirement, for each decision that is made by an algo you should provide a detailed, individualize, and legible explanation. We looked at all the govt systems that should provide these, but none of them do. From there, we wanted to investigate deeper on the housing benefits case and interviewed social workers and CNAF agents about their use of explainability of decisions. Findings: low-level social workers completely trust the algorithm, higher-level workers sort of understand the principles of how it works but without being able to link the behavior to the law that specifies it. To contest it, detect an algorithm error or debug it you need link between law and code through a detailed explanation of how thing was computed. However at CNAF there's a lengthy process between law and code with two different specification/documentation sets written by different groups of people. Recommendation : you should be able to link the law, the code and the explanations of how the code executed, while automatically producing explanations and sharing them externally to have more accurate contestations from the outside for maybe better debugging of the algorithm.

3.16 An AI Transparency Register for the Public Sector

Matthias Spielkamp (AW AlgorithmWatch – Berlin, DE)

License © Creative Commons BY 4.0 International license

© Matthias Spielkamp

Joint work of Michele Loi, Anna Mätzener, Angela Müller, Matthias Spielkamp

Main reference Michele Loi, Anna Mätzener, Angela Müller, Matthias Spielkamp: "Automated Decision-Making Systems in the Public Sector – An Impact Assessment Tool for Public Authorities". AlgorithmWatch, Berlin, Germany, 2021

URL https://algorithmwatch.org/en/wp-content/uploads/2021/09/2021_AW_Ddecision_Public_Sector_EN_v5.pdf

When using automated decision-making systems (ADM systems) in the public sector, authorities act in a unique context and bear special responsibilities towards the people affected. Against this background, the use of ADM systems by public administrations should be subject to stringent transparency mechanisms – including public registers and mandatory impact assessments.

Without the ability to know whether ADM systems are being deployed, all other efforts for the reconciliation of fundamental rights and ADM systems are doomed to fail. Legally mandatory public registers of all ADM systems used by public administrations – at communal, regional, national, and supranational levels – should therefore be created.

These registers should come with the legal obligation for those responsible for the ADM system to disclose information on the underlying model of the system, its developers and deployers, the purpose of its use, and the results of the algorithmic impact assessment.

In order to make a difference in practice, ethical reflection must be translated into ready-to-use tools, providing authorities with the means for conducting such an analysis. To this end, we have developed a practical, user-friendly, and concrete impact assessment tool, enabling the evaluation of an ADM system throughout its entire life cycle.

If you'd like to add bibliographic references to your abstract, please use the thebibliography-environment (bibtex-files are not supported):

3.17 Responsibility and Liability regarding Software and AI

Rüdiger Wilhelmi (Universität Konstanz, DE)

License  Creative Commons BY 4.0 International license
© Rüdiger Wilhelmi

Looking on responsibility and liability regarding software and especially non-symbolic artificial intelligence from the legal perspective, I start with two distinctions. The first one regards the different branches of law. Public law and especially administrative law concentrate on prohibitions and permissions and criminal law on prohibitions as well. Both necessarily involve the state as such and are mainly statute law. Private law relates to private individuals. Contract law allows the private shaping of law. Tort law regulates compensation for damages on a non-contractual basis. The second distinction is between general and special purpose law. With regard to software there is law aiming at software or digitalization at large like – in the context of the EU – the General Data Protection Regulation, the Digital Markets Act and the Digital Service Act as well as the proposals for an AI Act and an AI Liability Directive. There is also special law in more general statutes like computer fraud in the criminal code or the provisions regarding consumer contracts on digital products in the civil code, that lead to a convergence between software contracts based on the law of sale and those based on rental or service contracts. But software and digitalization at large are also covered by law not directly aiming at them like competition law, ordinary fraud, most of contract law or tort law. Regarding software there are legal relationships between a lot of actors. In general, it starts with a contract between the producer and the user, very often intermediated by a provider or dealer. Often, there is another contract between the user and his client, like a medical practitioner and his patient. On the other end of the chain there are contracts between the producer and his suppliers, like the programmer, the data supplier or trainers. In case of damage, there are non-contractual relationships based on tort governing the compensation. They exist in parallel to the contractual relationships but also independent to these and to especially third parties. My talk concerns responsibility and liability regarding software and especially artificial intelligence concentrated on tort law. Tort law is the branch of private law concerned with the award of compensation for damages even if there is no contractual relationship between the damaging and the damaged party. Basically, tort law is no law designed especially for information technology and digitalization,

but general-purpose law. As such it is the part of law very often concerned first with new developments. The most relevant branch of tort law in this context is product liability. The conditions for product liability based on negligence in essential are an infringement of legal interests, a breach of duty of care or conduct and causation. Based on strict liability they are an infringement of legal interest, being active in a specific domain, like road traffic or gene technology, and causation. The difference is, that there is no need to define a duty under strict liability. But there exist exemptions to strict liability with similar effects but being much more defined. Product liability claims can be based on both, negligence and strict liability. They require a defective product that does not provide the safety to expect and causes damages to specified legal interest not comprising pure economic loss. In our context the most relevant defects are design defects and instruction defects. Design defects can be identified from a bird's eye perspective by comparing the product to other products or from a frog perspective by looking whether a specific feature could have been better. Instruction defects fail to inform the users properly, especially to warn of risks. An interesting question is, whether and when flaws in the design can be compensated for by instructions or warnings. In this respect, design and instruction defects are a sort of communicating vessels between the producer and the user and could shift risks between them. The level of safety depends on what is technically possible and on the risks involved. The more risky a product is, the higher the product safety requirements. In extreme cases, the necessary safety level cannot be achieved. This mechanism is also known in other areas of law and is also the basis of the AI Act. In the end, the concrete level of safety to be provided has to be determined by the courts. Standards set by standardization bodies or evolving out of the industry are of limited effect in the sense, that they normally constitute the minimum level, but do not prevent the courts requiring higher levels, what they do. The main problems assorted with software and artificial intelligence are the lack of transparency and related to this of predictability and reliability. But this no new problem to product liability where it is solved by duties to disclosure of evidence and a shift of the burden of proof, in the end making the producer accountable for the lack of transparency, predictability and reliability. There is some indication, that the product liability can cover the problems for responsibility and liability regarding software and artificial intelligence, maybe requiring some minor adaptations. The current discussion about introducing a new (strict) liability regime lacks a sufficient analysis of whether, where and why the existing regime is insufficient.

3.18 Scenic: A Probabilistic Scenario Description Language

Beyazit Yalcinkaya (University of California – Berkeley, US)

License © Creative Commons BY 4.0 International license
© Beyazit Yalcinkaya

Main reference Daniel J. Fremont, Edward Kim, Tommaso Dreossi, Shromona Ghosh, Xiangyu Yue, Alberto L. Sangiovanni-Vincentelli, Sanjit A. Seshia: “Scenic: a language for scenario specification and data generation”, *Mach. Learn.*, Vol. 112(10), pp. 3805–3849, 2023.


URL <https://doi.org/10.1007/S10994-021-06120-5>

This presentation focuses on the integral role of expressive environment modeling tools within the realm of formal methods, underpinning the pursuit of verified AI. In this context, our research group at UC Berkeley has introduced Scenic, a probabilistic scenario description language designed to facilitate the realization of this goal. Scenic serves as a probabilistic programming language, enabling the formulation of distributions that characterize scenes and scenarios, the former representing configurations of physical objects and autonomous agents,

and the latter encompassing distributions over these scenes and the dynamic behaviors of their constituent agents over time. This novel language is distinguished by its succinct and comprehensible syntax for spatiotemporal relationships, accompanied by the capacity to declaratively enforce both hard and soft constraints on the scenario. Furthermore, by considering accountability as an inherent system property, we find the application of Scenic crucial for precisely specifying the environmental conditions under which we can hold a software system accountable. In doing so, we illuminate the potential of Scenic as a means to enhance the accountability of software systems.

3.19 Towards a Framework for Certification of Reliable Autonomous Systems

Neil Yorke-Smith (TU Delft, NL)

License  Creative Commons BY 4.0 International license
© Neil Yorke-Smith

Joint work of Michael Fisher, Viviana Mascardi, Kristin Yvonne Rozier, Bernd-Holger Schlingloff, Michael Winikoff, Neil Yorke-Smith

Main reference Michael Fisher, Viviana Mascardi, Kristin Yvonne Rozier, Bernd-Holger Schlingloff, Michael Winikoff, Neil Yorke-Smith: “Towards a framework for certification of reliable autonomous systems”, *Auton. Agents Multi Agent Syst.*, Vol. 35(1), p. 8, 2021.

URL <https://doi.org/10.1007/S10458-020-09487-2>

A computational system is called autonomous if it is able to make its own decisions, or take its own actions, without human supervision or control. The capability and spread of such systems have reached the point where they are beginning to touch much of everyday life. However, regulators grapple with how to deal with autonomous systems. We view certification of the behaviour of a system as, in appropriate cases, a component of accountability. This talk proposed to analyse what is needed in order to provide verified reliable behaviour of an autonomous system, analyse what can be done as the state-of-the-art in automated verification, and pointed to a roadmap towards developing regulatory guidelines, including articulating challenges to researchers, to engineers, and to regulators.

4 Working groups

4.1 Working groups topic discussions

We had two sessions where the participants were divided into different working groups to work on the same topic.

4.1.1 Forms of (Un)Accountability in Contemporary Software Ecosystems

The different groups discussed around the following key questions:

- How is software (un)accountable today? To whom, and with what consequences?
- How do different disciplines understand the concept of accountability?
- What are the main practical barriers to accountability?

4.1.2 Software Ecosystem Futures

The different groups discussed around the following key questions:

- How might software be designed, delivered, deployed and maintained in 5, 10, 15+ years? What are the main consequences for accountability?
- How are supply chains and value chains in software being formed and reformed, and what does this mean about which actors could and should be held responsible?
- Is the changing nature or a certain trajectory of software development and change inevitable? What are the main leverage points for driving it in different directions? What different directions exist?

4.2 Forms of (Un)Accountability in Contemporary Software Ecosystems: Group 1

Wouter Lueks (CISPA – Saarbrücken, DE) and Scott Shapiro (Yale University – New Haven, US)

License  Creative Commons BY 4.0 International license
© Wouter Lueks and Scott Shapiro

In group 1, we discussed different ways of looking at accountability. We realized that the typical computer science notion of accountability as “matching the specification” might not be sufficient. Real systems have to deal with users, and users might not follow the rules. One alternative way of looking at accountability, then is, to consider what you do when users break the rules. Accountability therefore doesn’t always have to be about software. In particular, in distributed systems, policy violations cannot always be prevented, instead such systems might resort to identify misbehaving users, or punish them, for example by reducing their utility.

Following this path, we realized that accountability is multiple ambiguous, and we’d have to consider different aspects, all of which relate to accountability:

- Ability: Is it even possible to hold “the thing” to account?
- Attributability: Can you assign credit or blame to a specific individual or even a line of code?
- Answerability: can you give an answer or account that explains the intention
- Liability: can we determine who has to be punished?

In light of these, we looked at the example of machine learning systems. There, answerability might be a challenge. Many systems cannot “explain” (or be made to explain) the choices that they made. And these systems are often operated by large companies. Should we attribute violations to individuals, or to the company as a whole?


We questioned the term “accountable systems” because systems cannot be held accountable, instead maybe “accountability of software” is more appropriate.

One particular challenge with accountability is to attribute problems. For one, this might not always be possible directly. For example, a child who knocks over a precious vase is usually not held accountable. Although their parents might. And a person suffering a acute mental distress might be held accountable for their actions, but not blamed.

In the world of objects, there are often processes in place to assign blame or draw lessons from failures. For example, after an aircraft crash, there is an investigation by an authorized body. Such body does not exist for software (unless it leads to physical harm).

4.3 Forms of (Un)Accountability in Contemporary Software Ecosystems: Group 2

Roel Dobbe (TU Delft, NL)

License  Creative Commons BY 4.0 International license
© Roel Dobbe

The prompt for this breakout was: How does software creation affect accountability?


Rather than answering this big question head on, the group decided to surface important distinctions that may help to understand the various forms software may take that are relevant to study the role of and impact on accountability.

The distinctions that we identified were:

- Software running in the cloud vs Software running on premise
- Open source vs Closed source
- Software-as-a-service vs Shrink-wrap software (with associated licensing)
- Infrastructure-as-a-service (one-stop shop) vs Open infrastructures and ecosystems for software development
- Manual requirement generation vs Automated generation of requirements
- General purpose/multi-task software vs Specialized/single-task software
- Centralized control over software vs decentralized control over software
- Software as a utility vs non-utility
- Big tech break up vs Further consolidation of software industry
- Universal software (standards) vs Plural software (standards)
- Quantum-based computing vs Bit-based computing
- Scrutable software architectures vs Inscrutable software architectures

4.4 Forms of (Un)Accountability in Contemporary Software Ecosystems: Group 3

Bettina Könighofer (TU Graz, AT) and Neil Yorke-Smith (TU Delft, NL)

License  Creative Commons BY 4.0 International license
© Bettina Könighofer and Neil Yorke-Smith

In our group discussion, we considered that software complexity acts as an accelerant in the realm of technology, yet it isn't the initial catalyst for technological advancements. It's widely recognized that autonomous systems are on the horizon, and rather than resisting their emergence, we should focus on understanding and shaping their integration responsibly.

A key point of debate was the distinction between accountability and responsibility. While one can be responsible without being blameworthy, accountability often carries the weight of blame, especially in situations where the complexity of a system obfuscates the lines of direct responsibility. This distinction becomes particularly poignant in discussions about how to regulate technologies that are still evolving and the challenges of preparing the public sector through adequate training in data and digital literacy.

The conversation also touched on practical challenges, such as the costly repercussions of fixing bugs in software systems, exemplified by an incident in Los Angeles' air traffic control where a countdown timer bug led to operational chaos. This highlighted the broader issue of how audits, transparency registers, and other mechanisms for oversight struggle to keep

pace with dynamic software systems. The consensus emerged that understanding faults is a prerequisite for attributing blame or punishment, underlining the need for integrating such accountability measures throughout the software lifecycle.

Accountability was thus framed as an organizational challenge rather than purely a technical one, with parallels drawn to self-regulation models like film ratings in the U.S. or the medical association's role in certifying doctors. The discussion acknowledged the universal nature of software as akin to a Turing machine, raising questions about the feasibility of ensuring safety, security, and accountability simultaneously.

Identifying liability within complex software systems presents its own set of challenges, especially when all components function correctly yet the system as a whole fails. The concept of group responsibility was dissected, acknowledging its complexity and distributed nature. Questions were raised about how software can “make it right,” including bearing punishment, making restitution, and ensuring justice.

The group considered examples such as rule-based parole systems demonstrating bias, underscoring the “stupidity” of computers in their lack of reasonableness, and the need for human-like foresight in technology design. The “Miracle on the Hudson” incident was cited as a case where software limitations impeded human judgment, suggesting a need for regulatory frameworks that can adapt to the unique challenges of software within various domains.

Finally, the discussions circled back to the foundational problem of software complexity and its role in exacerbating these accountability and regulatory challenges. The conclusion pointed towards the necessity of domain-specific expertise in creating regulatory frameworks for software, questioning what aspects of a system can be automatically checked by another system and what requires more nuanced, human oversight. Through these discussions, the group navigated the intricate web of ethical, technical, and regulatory challenges facing the future of software development and integration.

4.5 Forms of (Un)Accountability in Contemporary Software Ecosystems: Group 4

Beyazit Yalcinkaya (University of California – Berkeley, US)

License © Creative Commons BY 4.0 International license
© Beyazit Yalcinkaya

In our discussion, we approached the forms of (un)accountability in contemporary software ecosystems from various angles. Essentially, our discussion was rooted in a computer science perspective due to the background of the majority of participants. Our responses to specific questions are summarized below.

How is software (un)accountable today? To whom, and with what consequences? Our consensus was that software today is essentially unaccountable as accountability has not been a major focus of software development. On top of that, we do not have a clear understanding of the possible behaviors of contemporary learning-based models. Moreover, even without any learning-based components in the software loop, we observe cases where the problem of accountability is ambiguous. Specifically, we discussed an example from the security literature in which a user interacts with various software services, and even though each software system is secure in itself, the information exposure between their interfaces causes a security vulnerability, so an attacker can capture sensitive information. This is an interesting example because none of the software services is the cause of the

security vulnerability; however, their composition results in an ecosystem with a security vulnerability. Therefore, it is not clear which software service should be held accountable. In general, the composition of software systems is not a trivial process. We concluded that the issue of accountability of software has to be studied well from various aspects in today’s complex software ecosystem.

How do different disciplines understand the concept of accountability? Our consensus on this issue was that the term accountability is more of an umbrella term that can be understood from the perspective of various system properties that we care about. These properties mainly include safety, reproducibility, and certifiability. Specifically, the issue of certifiability is a main concern for holding an entity accountable for a software system’s behavior. For example, one needs some sort of certification before using the output of a large language model to hold the system accountable for any complication that might occur, such as a copyright issue. Software services should provide a certification to their users so that the user can use this certification in court as a piece of evidence that the material was generated by the specific software service that they used. We concluded that certifiability has to be an interdisciplinary understanding for the accountability of software systems.

What are the main practical barriers to accountability? We started our discussion on this issue by critiquing the idea of transparency being a sufficient condition for the accountability of software systems. Our consensus was that the speed and scale of today’s contemporary software systems cannot be handled by any human-level bureaucratic produce that can be accomplished through transparency. Therefore, we need systems that monitor/check themselves, and the processes for accountability must be automated as much as possible. However, to achieve this goal, we need new results from the computer science community to provide what can be formalized (and therefore automated) and what cannot be formalized so that the proper tools for automation can be developed. Then in light of these results and tools, policymakers can provide guidelines for system developers, and therefore better practices for accountable software systems can be enforced. We concluded that to make an accountable software ecosystem possible, policymakers and computer scientist should communicate their needs and capacities clearly to each other.

4.6 Software Ecosystem Futures: Group 1

Divij Joshi (University College London, GB)

License  Creative Commons BY 4.0 International license
© Divij Joshi

Our group discussed how new technologies and organisational forms structure software production and accountability.

We began the discussing how or whether Large Language Models will change software production. There was some disagreement about the impact of these changes, with some participants stressing that automation of programmer’s capabilities was already affecting software production, and that this leads to challenges to accountability. One participant spoke about how accountability in the supply chain or production might be becoming irrelevant with systems like zero-trust procurement (which the US government is moving towards).

Next, participants discussed the role of platforms and platform power among software production and use supply chains. They spoke about whether accountability needs to be located within software production processes, or from the perspective of end-users. From this perspective, which actors are “choke points” and which ones can be used to leverage accountability across the ecosystem.

Finally, participants discussed whether and how software production itself might be made more accountable through regulation.

4.7 Software Ecosystem Futures: Group 2

Filip Cano Córdoba (TU Graz, AT)

License © Creative Commons BY 4.0 International license
© Filip Cano Córdoba

In Group 2, we did not focus our discussion around large language models (LLMs) or AI, but more on the software as a service model (SaaS). We agreed that how software is maintained will not change that much, perhaps more of it will be moving to the cloud and the SaaS model. So, in the future, the user will not own anything, but this paradigm will probably not change much how software is to be developed and maintained. As an interesting side thought, we made the thought experiment of what would happen if instead of the big players that currently dominate the market, we created a public democratically managed cloud infrastructure that would serve as a non-for-profit realistic alternative. This entity would escape the power of both huge for-profit companies and governments, so the profit and the power would be in the hands of the people. We proposed the Wikipedia Foundation as a viable example to follow, albeit the Wikipedia Foundation is a much smaller organization, with a much more focused objective.

On supply chains, we discussed how when catastrophic fails happens, it is typically not the cause of a component-level failure, but a failure at system level. System level properties are generally more difficult to specify and verify, so they consist of an important challenge. We also discussed extensively about dependency-tracking projects, like NIX or the “software bill of materials”. While transparency is not all, and full transparency may be infeasible, having a clear description of software dependencies will surely help make software systems accountable along their production and value chains.

About change and its inevitability, we first wanted to point out that nothing is inevitable, almost everything can happen if there is enough will in the public. Maybe the only inevitability is climate change, that will pressure current political systems and regulations to somehow limit the amount of resources dedicated to computing. We discussed how we should stirr this policies so that limited resources get allocated to valuable objectives.

4.8 Software Ecosystem Futures: Group 3

Neil Yorke-Smith (TU Delft, NL)

License © Creative Commons BY 4.0 International license
© Neil Yorke-Smith

This breakout group discussed the future of software development, aided by the fresh air of the countryside around Schloss Dagstuhl.

- AI-aided, man-machine co-generation of software is already used at Google, one participant pointed out. The group discussed whether such co-generation will be more participatory and more sustainable in the future.
- The “code local” movement was discussed: developing software closer to the point of use; supply and value chains; the voice of citizens.
- Software projects that arose during the COVID-19 epidemic received discussion, both how software development continued or changed, and how new projects arose. One participant pointed out how “interesting” consortiums came together; critical comments arose about governments and how they procure software.

- One participant suggested that a few people can change the status quo. An example is the few who raised concerns about bias in AI, whereas this is now mainstream.
- The group thought that future software will be more flexible, making accountability more tricky.
- Lastly, the group discussed the role of “big tech”. It was noted that software in schools in the Netherlands is highly dependent on Google.

5 Open problems

5.1 Concluding Exercise: Open Questions

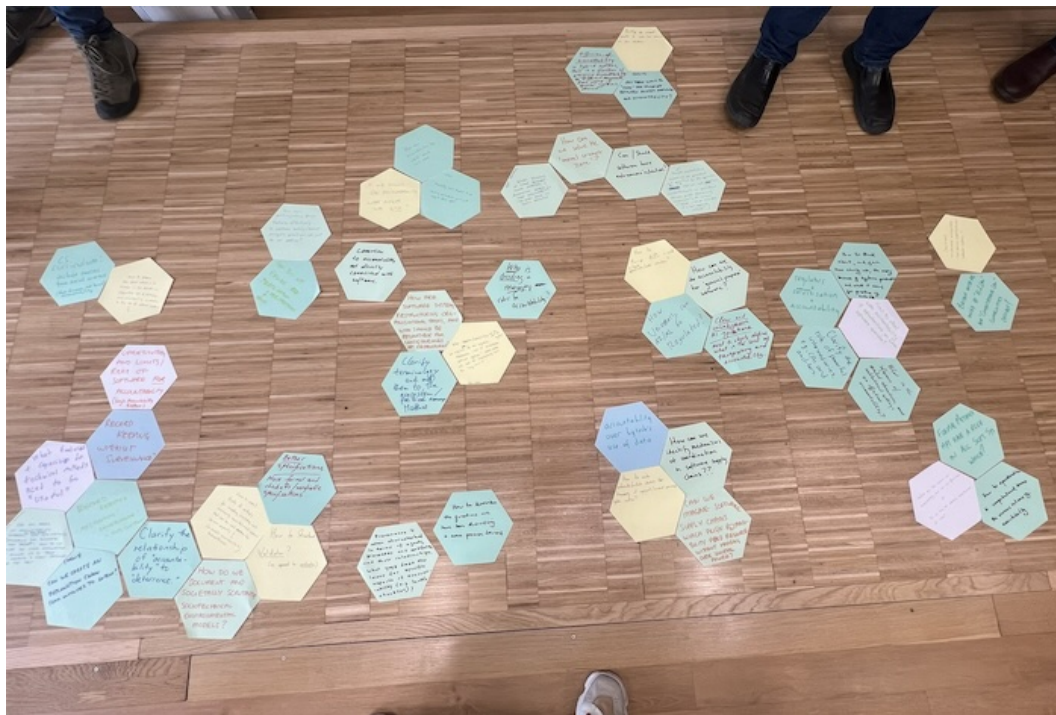
Michael Veale (University College London, GB), Thomas Arnold (Tufts University – Medford, US), Filip Cano Córdoba (TU Graz, AT), Corinne Cath (TU Delft, NL & University of Cambridge, GB), Hana Chockler (King’s College London, GB), Aloni Cohen (University of Chicago, US), Misty Davies (NASA – Moffett Field, US), Roel Dobbe (TU Delft, NL), Joan Feigenbaum (Yale University – New Haven, US), David Fuenmayor (Universität Bamberg, DE), Ashish Gehani (SRI – Menlo Park, US), Jake Goldenfein (The University of Melbourne, AU), Thomas T. Hildebrandt (University of Copenhagen, DK), Divij Joshi (University College London, GB), Samuel Judson (Yale University – New Haven, US), Daan Kolkman (TU Eindhoven, NL), Bettina Könighofer (TU Graz, AT), Joshua A. Kroll (Naval Postgraduate School – Monterey, US), Robert Künnemann (CISPA – Saarbrücken, DE), Stefan Leue (Universität Konstanz, DE), Wouter Lueks (CISPA – Saarbrücken, DE), Rupak Majumdar (MPI-SWS – Kaiserslautern, DE), Kira Matus (The Hong Kong Univ. of Science & Technology, HK), Denis Merigoux (INRIA – Paris, FR), Ruzica Piskac (Yale University – New Haven, US), Scott Shapiro (Yale University – New Haven, US), Jatinder Singh (University of Cambridge, GB), Matthias Spielkamp (AW AlgorithmWatch – Berlin, DE), Rüdiger Wilhelmi (Universität Konstanz, DE), Beyazit Yalcinkaya (University of California – Berkeley, US), and Neil Yorke-Smith (TU Delft, NL)

License © Creative Commons BY 4.0 International license

© Michael Veale, Thomas Arnold, Filip Cano Córdoba, Corinne Cath, Hana Chockler, Aloni Cohen, Misty Davies, Roel Dobbe, Joan Feigenbaum, David Fuenmayor, Ashish Gehani, Jake Goldenfein, Thomas T. Hildebrandt, Divij Joshi, Samuel Judson, Daan Kolkman, Bettina Könighofer, Joshua A. Kroll, Robert Künnemann, Stefan Leue, Wouter Lueks, Rupak Majumdar, Kira Matus, Denis Merigoux, Ruzica Piskac, Scott Shapiro, Jatinder Singh, Matthias Spielkamp, Rüdiger Wilhelmi, Beyazit Yalcinkaya, and Neil Yorke-Smith

In the final session, we all wrote open questions on cards and then grouped them together in emergent, but unnamed groups.

- How to make enforceable/verifiable standards for traceability of requirements/components’ provenance or system actors?
- How can we identify mechanisms of coordination in software supply chains?
- Can we imagine software supply chains which align responsibility and resources without handing over societal power?
- How to ensure accountability over Big Tech’s use of data?
- How to manage drift in a world with fixed specifications and model structure?
- How universally can AI or ML be regulated?
- How can we do accountability for ‘general purpose’ machine learning software?



■ **Figure 1** The grouping of the open questions from the final session of the seminar.

- How to write clear and unambiguous AI regulations that clearly define and delimit the level of needed transparency and accountability?
- Can the computer science curriculum include courses from social sciences which discuss, at least, accountability?
- How can we prepare citizens for future debates on the accountability of software systems in society?
- What are the opportunities and limits around the rise of software *for* accountability?
- Can we do record-keeping without opening the door to surveillance? What are the mechanisms for record keeping, and what are the governance mechanisms and implications?
- What features and capacities do technical accountability methods need to be successful?
- For key areas of accountability interest and key metrics describing the “goodness” of that property, how do we capture the state knowledge that allows us to trace the metrics and make decisions after the fact?
- Can we create an explanation chain from outcomes to intent?
- Clarify the relationship of “accountability” to deterrence
- How do we document and societally scrutinise sociotechnical environmental models?
- How do we structure validation (opposed to verification)?
- How can we make tools and methods for modelling, simulating and analysing sociotechnical systems that can be used by domain experts (to enable accountability and trustworthiness)
- Better specifications – more formal and checkable or verifiable specifications
- How do we ensure participation and broad perspectives?
- How can participatory design feature effectively in systems safety/hazard analysis practices, not just as an add on?

- If we focus on accountability, what might we lose?
- How can accountability be used and misused?
- Ex ante accountability impact assessments? Should be used to determine go or no-go for creating the software system.

- How are software systems restructuring organisational tasks and who should be responsible for the consequences of restructuring?
- Clarify terminology and maps it to the ecosystem or political economy
- What capacities or capabilities on the part of humans are required for the long term maintenance and use of accountable software systems? Which actors will hold the responsibility over the long term?
- Who is deciding and managing rules for accountability?

- Diffusion of accountability in hybrid systems, there is a question of assigning accountability to different components. Even worse when several systems interact!
- Building up component properties to system-level assessment, versus pure reductionism.
- Are there ways to “hack” the trade off between system complexity and accountability?

- What divisions of labour between human operators and automation enable clear accountability? (for every sense of definition of accountability!)
- How can we solve the “moral crumple zone”?
- Can and should software have autonomous intention?
- How well do counterfactual scenarios as generated and specified by ML approaches fit with, improve, and sustain counterfactuals as questions between people determining responsibility for actions (what would you have done if “x” had happened instead?)

- Can we clarify the role of powerful intermediaries, e.g. certificate authorities and auditors?
- How to deal with accountability of the infrastructures upon which software operates?
- What is the influence of market structures and institutional settings on effective accountability?

- Formal methods may have a role in accountable software systems – but which?
- How to operationalise in computational terms the various notions of accountability
- What are the aspects of accountability what can be formalised and do we have the necessary tools and theories for these?

- Bridge with work by “AI and Law” and “Computational Law” communities – is it possible?
- How to make maintainable and accountable legal decision support or automation software systems?

Participants

- Thomas Arnold
Tufts University – Medford, US
- Filip Cano
TU Graz, AT
- Corinne Cath
TU Delft, NL & University of
Cambridge, GB
- Hana Chockler
King’s College London, GB
- Aloni Cohen
University of Chicago, US
- Misty Davies
NASA – Moffett Field, US
- Roel Dobbe
TU Delft, NL
- Joan Feigenbaum
Yale University – New Haven, US
- David Fuenmayor
Universität Bamberg, DE
- Ashish Gehani
SRI – Menlo Park, US
- Jake Goldenfein
The University of Melbourne, AU
- Thomas T. Hildebrandt
University of Copenhagen, DK
- Divij Joshi
University College London, GB
- Samuel Judson
Yale University – New Haven, US
- Bettina Könighofer
TU Graz, AT
- Daan Kolkman
TU Eindhoven, NL
- Joshua A. Kroll
Naval Postgraduate School –
Monterey, US
- Robert Künnemann
CISPA – Saarbrücken, DE
- Stefan Leue
Universität Konstanz, DE
- Wouter Lueks
CISPA – Saarbrücken, DE
- Rupak Majumdar
MPI-SWS – Kaiserslautern, DE
- Kira Matus
The Hong Kong Univ. of Science
& Technology, HK
- Denis Merigoux
INRIA – Paris, FR
- Ruzica Piskac
Yale University – New Haven, US
- Scott Shapiro
Yale University – New Haven, US
- Jatinder Singh
University of Cambridge, GB
- Matthias Spielkamp
AW AlgorithmWatch –
Berlin, DE
- Michael Veale
University College London, GB
- Rüdiger Wilhelm
Universität Konstanz, DE
- Beyazit Yalcinkaya
University of California –
Berkeley, US
- Neil Yorke-Smith
TU Delft, NL



Formal Methods for Correct Persistent Programming

Ori Lahav^{*1}, Azalea Raad^{*2}, Joseph Tassarotti^{*3}, Viktor Vafeiadis^{*4},
and Anton Podkopaev^{†5}

- 1 Tel Aviv University, IL. orilahav@tau.ac.il
- 2 Imperial College London, GB. azalea.raad@imperial.ac.uk
- 3 New York University, US. jt4767@nyu.edu
- 4 MPI-SWS – Kaiserslautern, DE. viktor@mpi-sws.org
- 5 JetBrains – Amsterdam, NL. anton@podkopaev.net

Abstract

Recently developed non-volatile memory (NVM) devices provide persistency guarantees along with byte-addressable accesses and performance characteristics that are much closer to volatile random-access memory (RAM). However, writing programs that correctly use these devices is challenging, and bugs related to their use can cause permanent data loss in applications.

This Dagstuhl Seminar brought together experts in a range of areas related to concurrency and persistent memory to explore and develop formal methods for ensuring the correctness of applications that use persistent memory. Talks and discussions at the seminar highlighted challenges related to correctness criteria for concurrent objects using persistent memory, liveness properties of persistent objects, and how changes in NVM and related technologies should shape the development of formal methods for NVM.

Seminar October 8–11, 2023 – <https://www.dagstuhl.de/23412>

2012 ACM Subject Classification Hardware → Non-volatile memory; Theory of computation → Program semantics; Theory of computation → Program verification

Keywords and phrases concurrency, formal methods, non-volatile-memory, persistency, verification

Digital Object Identifier 10.4230/DagRep.13.10.50


1 Executive Summary

Ori Lahav

Azalea Raad

Joseph Tassarotti

Viktor Vafeiadis

License  Creative Commons BY 4.0 International license
© Ori Lahav, Azalea Raad, Joseph Tassarotti, and Viktor Vafeiadis

Many systems and applications need to store data in a durable way. Historically, durable storage devices had considerably higher latency than volatile random-access memory (RAM) and provided interfaces with larger, coarser access granularity. To achieve acceptable performance, applications requiring durable storage were structured to account for these characteristics. However, in recent years, novel storage systems, such as *non-volatile memory* (NVM), have emerged that provide durability along with performance and access granularity much closer to RAM. This provides the opportunity for applications to achieve durability with much lower latencies.

* Editor / Organizer

† Editorial Assistant / Collector



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

Formal Methods for Correct Persistent Programming, *Dagstuhl Reports*, Vol. 13, Issue 10, pp. 50–64

Editors: Ori Lahav, Azalea Raad, Joseph Tassarotti, and Viktor Vafeiadis



Dagstuhl Reports

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

However, taking full advantage of that opportunity involves restructuring applications to make proper use of these new devices. Doing so requires care because bugs in these parts of applications can cause permanent data loss. Moreover, non-volatile memory interacts in subtle ways with caches and other parts of modern memory hierarchies, making it challenging for programmers to write correct code.

One promising approach to address this challenge is to develop formal methods techniques to certify the absence of bugs in programs using non-volatile memory. This Dagstuhl Seminar brought together experts in non-volatile memory, relaxed memory, concurrency, and formal methods to explore the application of formal methods to programming with persistent memory. Since this subfield involves deep theoretical work, but is also very dependent on technological developments, the participants of the seminar were from a spectrum of backgrounds ranging from theory of verification to hardware specification, design, and testing.

The seminar included a series of talks and discussions, some of which were unplanned additions prompted by topics or misunderstandings identified during earlier parts of the seminar. The addition of these unplanned talks proved beneficial, adding a dynamic element to the event. We decided to forgo smaller break-out sessions based on the feeling that much of the seminar's value was in discussions that spanned from theoretical to practical, drawing on the full range of participants' expertise.

Several recurring themes arose in the talks and following discussions:

- **Correctness Criteria and Specifications for Persistent Objects:** A number of correctness criteria have been proposed for concurrent objects that persistently store data. Many of these definitions are adaptations of the classical notion of linearizability. However, we discussed ways in which these existing definitions can have surprising consequences when objects are implemented in the setting of weak memory. A related topic was the appropriate guarantees that transactional interfaces should provide, as certain strong guarantees may prevent efficient implementations.
- **Liveness:** Our discussions revealed a lack of consensus on assumptions that can be made from existing architectures concerning liveness properties, underscoring the need for further research in this area.
- **Future of NVM and Related Technologies:** NVM remains an emerging technology, and manufacturers continue to announce large changes in plans for future product lines. We discussed the ramifications of these changes and how techniques for the semantics and verification of certain forms of NVM might apply to other persistency models. Moreover, we identified the need for generic verification methods, which would lend themselves more easily to ongoing changes in the exact semantics of the underlying memory system. Indeed, several talks suggested modular approaches for verification, that, to some extent, take the memory model as an input. Related technologies, including Remote Direct Memory Access (RDMA) and Compute Express Link (CXL), were discussed, focusing on the appropriate abstractions and semantics of interfaces for these devices, and challenges with testing these devices.

We believe that these issues will be an important focus in research on formal methods for persistent memory in the future.

2 Table of Contents

Executive Summary

Ori Lahav, Azalea Raad, Joseph Tassarotti, and Viktor Vafeiadis 50

Overview of Talks

Semantics of Remote Direct Memory Access

Guillaume Ambal 54

Checking Liveness Properties under Weak Consistency (TSO as an Example)

Parosh Aziz Abdulla 54

Correctly Combining Concurrent and Persistent Transactional Memory

Brijesh Dongol 55

Utilizing Coherence for Persistence

Michal Friedman 55

Some compositional semantics for shared memory: sequential consistency and release/acquire

Ohad Kammar 56

Programming Persistency Should Be Easy – but is it?

Jeehoon Kang 56

Challenges in Empirically Testing Memory Persistency Models

Vasileios Klimis 57

Automating Weak Memory Model Metatheory and Verification

Michalis Kokologiannakis 57

Abstraction for Crash-Resilient Objects

Ori Lahav 58

Fairness for load buffering memory models

Anton Podkopaev 58

DARTAGNAN: One tool for all models

Hernán Ponce de León 59

Towards a formal specification of the Intel Architecture

Alastair Reid 59

System and Failure Models Matter

Michael Scott 60

Transactional Semantics with Zombies

Michael Scott 60

Specifying and Verifying Persistent Libraries

Léo Stefanesco 61

A Type System for Intermittent Computing

Milijana Surbatovich 61

Separation Logic for Concurrent, Crash-Safe Systems

Joseph Tassarotti 62

Persistent Scripting

Haris Volos 62

Verifying the persistency library FliT <i>Heike Wehrheim</i>	62
Participants	64

3 Overview of Talks

3.1 Semantics of Remote Direct Memory Access

Guillaume Ambal (Imperial College London, GB)

License © Creative Commons BY 4.0 International license
© Guillaume Ambal

Joint work of Guillaume Ambal, Brijesh Dongol, Haggai Eran, Vasileios Klimis, Ori Lahav, Azalea Raad

Remote direct memory access (RDMA) is a modern technology enabling networked machines to exchange information without involving the operating system of either side, and thus significantly speeding up data transfer in computer clusters. While RDMA is extensively used in practice and studied in various research papers, a formal underlying model specifying the allowed behaviours of concurrent RDMA programs running in modern multicore architectures is still missing. This paper aims to close this gap and provide semantic foundations of RDMA on x86-TSO machines. We propose three equivalent formal models, two operational models in different levels of abstraction and one declarative model, and prove that the three characterisations are equivalent. To gain confidence in the proposed semantics, the more concrete operational model has been reviewed by NVIDIA experts, a major vendor of RDMA systems, and we have empirically validated the declarative formalisation on various subtle litmus tests by extensive testing. We believe that this work is a necessary initial step for formally addressing RDMA-based systems by proposing language-level models, verifying their mapping to hardware, and developing reasoning techniques for concurrent RDMA programs.

3.2 Checking Liveness Properties under Weak Consistency (TSO as an Example)

Parosh Aziz Abdulla (Uppsala University, SE)

License © Creative Commons BY 4.0 International license
© Parosh Aziz Abdulla

Joint work of Parosh Aziz Abdulla, Mohamed Faouzi Atig, Adwait Godbole, Shankaranarayanan Krishna, Mihir Vahanwala

Main reference Parosh Aziz Abdulla, Mohamed Faouzi Atig, Adwait Godbole, Shankaranarayanan Krishna, Mihir Vahanwala: “Overcoming Memory Weakness with Unified Fairness – Systematic Verification of Liveness in Weak Memory Models”, in Proc. of the Computer Aided Verification – 35th International Conference, CAV 2023, Paris, France, July 17-22, 2023, Proceedings, Part I, Lecture Notes in Computer Science, Vol. 13964, pp. 184–205, Springer, 2023.

URL https://doi.org/10.1007/978-3-031-37706-8_10

We consider the verification of liveness properties for concurrent programs running on weak memory models. To that end, we identify notions of fairness that preclude demonic non-determinism, are motivated by practical observations, and are amenable to algorithmic techniques. We provide both logical and stochastic definitions of our fairness notions, and prove that they are equivalent in the context of liveness verification. In particular, we show that our fairness allows us to reduce the liveness problem (repeated control state reachability) to the problem of simple control state reachability. We show that this is a general phenomenon by developing a uniform framework which serves as the formal foundation of our fairness definition, and can be instantiated to a wide landscape of memory models. These models include SC, TSO, PSO, (Strong/Weak) Release-Acquire, Strong Coherence, FIFO-consistency, and RMO.

3.3 Correctly Combining Concurrent and Persistent Transactional Memory

Brijesh Dongol (University of Surrey – Guildford, GB)

License © Creative Commons BY 4.0 International license
© Brijesh Dongol

Joint work of Brijesh Dongol, Piotr Balcer, Ori Lahav, Azalea Raad, John Wickerson
Main reference Azalea Raad, Ori Lahav, John Wickerson, Piotr Balcer, Brijesh Dongol: “Intel PMDK Transactions: Specification, Validation and Concurrency (Extended Version)”, CoRR, Vol. abs/2312.13828, 2023.
URL <https://doi.org/10.48550/ARXIV.2312.13828>

Software Transactional Memory (STM) is an extensively studied paradigm that provides an easy-to-use mechanism for thread safety and concurrency control. With the recent advent of byte-addressable persistent memory, a natural question to ask is whether STM systems can be adapted to support recoverability via failure atomicity. In this article, we answer this question by showing how STM can be easily integrated with Intel’s Persistent Memory Development Kit (PMDK) transactional library (which we refer to as txPMDK) to obtain STM systems that are both concurrent and persistent. We demonstrate this approach using known STM systems, TML and NOrec, which when combined with txPMDK result in persistent STM systems, referred to as PMDK-TML and PMDK-NOrec, respectively. However, it turns out that existing correctness criteria are insufficient for specifying the behaviour of txPMDK and our concurrent extensions. We therefore develop a new correctness criterion, dynamic durable opacity, that extends the previously defined notion of durable opacity with dynamic memory allocation. We provide a model of txPMDK that has been validated for accuracy with Intel developers, then show that this model satisfies dynamic durable opacity. Moreover, dynamic durable opacity supports concurrent transactions, thus we also use it to show correctness of both PMDK-TML and PMDK-NOrec.

3.4 Utilizing Coherence for Persistence

Michal Friedman (ETH Zürich, CH)

License © Creative Commons BY 4.0 International license
© Michal Friedman

Joint work of Richard Braun, Abishek Ramdas, Michal Friedman, Gustavo Alonso
Main reference Richard Braun, Abishek Ramdas, Michal Friedman, Gustavo Alonso: “PPlayer: Expanding Coherence Protocol Stack with a Persistence Layer”, in Proc. of the 1st Workshop on Disruptive Memory Systems, DIMES 2023, Koblenz, Germany, 23 October 2023, pp. 8–15, ACM, 2023.
URL <https://doi.org/10.1145/3609308.3625270>

Mechanisms to explicitly manage data persistence for non-volatile main memories are fundamental for the correctness and performance of modern systems. So far, however, most solutions are primarily based on software techniques. In this talk, I will describe a persistence layer on hardware, to support correct handling of persistent lock-free data structures. By exploiting cache-coherence messages, persistence can be transparently managed by the hardware, with minimal user intervention. We have experimented with a partial design on a Soft-CPU running on an FPGA to explore the idea and plan to further extend it into a real hardware implementation.

3.5 Some compositional semantics for shared memory: sequential consistency and release/acquire

Ohad Kammar (University of Edinburgh, GB)

License © Creative Commons BY 4.0 International license
© Ohad Kammar

Joint work of Yotam Dvir, Ohad Kammar, Ori Lahav

Main reference Yotam Dvir, Ohad Kammar, Ori Lahav: “A Denotational Approach to Release-Acquire Concurrency”. ESOP 2024.

I motivated and presented recent results, joint with Dvir and Lahav, in the compositional/denotational semantics for shared state concurrency that is structurally similar to standard, but general, denotational semantics for sequential programs.

The desire for a uniform treatment of programming languages as standard features (let-binding, function abstraction, pattern matching) augmented with domain-specific features (mutable state, backtracking search, etc.) is as old as the discipline itself. I traced a specific thread starting with Landin’s pragmatics and axiomatics, later realised by Plotkin’s structural-operational semantics and, in the sequential case, extended to denotational semantics by Moggi, and later refined by Plotkin and Power. With this perspective, I outlined our recent Brookes-trace account for sequential consistent shared state using universal algebra and monads and its limitations. I also outlined our ongoing account for the non-relaxed atomic fragment of the release-acquire weak memory model and invited participants to follow-up informally.

These informal discussions covered: (a) Brookes’s seminal work on trace semantics for sequentially consistent shared-state and some of Lahav’s more recent results about its abstraction; and (b) a technical description of our release-acquire denotational semantics.

3.6 Programming Persistency Should Be Easy – but is it?

Jeehoon Kang (KAIST – Daejeon, KR)

License © Creative Commons BY 4.0 International license
© Jeehoon Kang

Joint work of Kyeongmin Cho, Seungmin Jeon, Azalea Raad, Sung-Hwan Lee, Jeehoon Kang

Main reference Kyeongmin Cho, Seungmin Jeon, Azalea Raad, Jeehoon Kang: “Memento: A Framework for Detectable Recoverability in Persistent Memory”, Proc. ACM Program. Lang., Vol. 7(PLDI), pp. 292–317, 2023.

URL <https://doi.org/10.1145/3591232>

Programming persistency poses two major challenges:

- **Non-determinism:** Persist instructions may be reordered. Consequently, an earlier write might be discarded while a later one is preserved in the event of a crash. Though this challenge has been somewhat mitigated by recent hardware changes, including (e)ADR and GPF.
- **Recovery:** In the event of a crash, it is crucial to recover pre-crash contexts to complete their execution.

Efficient and easy-to-implement crash recovery is still a promising area of research in programming persistency. Although prior works like NVTraverse (PLDI 2020) and Mirror (PLDI 2021) offer automatic translation of concurrent programs into persistent ones equipped with recovery code, their applicability is confined to simpler program forms. Memento (PLDI 2023) is applicable to a broader range of programs but misses some optimization opportunities related to DRAM caches and transactions.

What is the next generation technique for efficient and easy-to-implement crash recovery?

3.7 Challenges in Empirically Testing Memory Persistency Models

Vasileios Klimis (Queen Mary University of London, GB)

License © Creative Commons BY 4.0 International license
© Vasileios Klimis

Joint work of Vasileios Klimis, Azalea Raad, Viktor Vafeiadis, John Wickerson, Alastair F. Donaldson

Memory persistency models provide a foundation for persistent programming by specifying which (and when) writes to non-volatile memory (NVM) become persistent. Memory persistency models for the Intel-x86 and Arm architectures have been formalised, but not empirically validated against real machines. Traditional validation methods used for memory *consistency* models do not straightforwardly apply because a test program cannot directly observe when its data has become persistent: it cannot distinguish between reading data from a volatile cache and from NVM. We investigate addressing this challenge using a commercial off-the-shelf device that intercepts data on the memory bus and logs all writes in the order they reach the memory. Using this technique we conducted a litmus-testing campaign aimed at empirically validating the persistency guarantees of Intel-x86 and Arm machines. We observed writes propagating to memory out of order, and took steps to build confidence that these observations were not merely artefacts of our testing setup. However, despite gaining high confidence in the trustworthiness of our observation method, our conclusions remain largely negative. We found that the Intel-x86 architecture is not amenable to our approach, and on consulting Intel engineers discovered that there are currently no reliable methods of validating their persistency guarantees. For Arm, we found that even a machine recommended to us by a persistency expert at Arm did not match the formal Arm persistency model, due to a loophole in the specification. Nevertheless, our investigation and results provide confidence that if Intel were to produce machines with more transparent persistency behaviour, or if Arm machines with proper persistency support were to become available, our approach would be valuable for empirically validating them against their specifications.

3.8 Automating Weak Memory Model Metatheory and Verification

Michalis Kokologiannakis (MPI-SWS – Kaiserslautern, DE)

License © Creative Commons BY 4.0 International license
© Michalis Kokologiannakis

Joint work of Michalis Kokologiannakis, Ori Lahav, Viktor Vafeiadis

Main reference Michalis Kokologiannakis, Ori Lahav, Viktor Vafeiadis: “Kater: Automating Weak Memory Model Metatheory and Consistency Checking”, Proc. ACM Program. Lang., Vol. 7(POPL), pp. 544–572, 2023.

URL <https://doi.org/10.1145/3571212>

Weak memory consistency models capture the outcomes of concurrent programs that appear in practice and yet cannot be explained by thread interleavings. Such outcomes pose two major challenges to formal methods. First, establishing that a memory model satisfies its intended properties (e.g., supports a certain compilation scheme) is extremely error-prone: most proposed language models were initially broken and required multiple iterations to achieve soundness. Second, weak memory models make verification of concurrent programs much harder, as a result of which there are no scalable verification techniques beyond a few that target very simple models.

In this talk, I present solutions to both of these problems. First, I show that the relevant metatheory of weak memory models can be effectively decided and present Kater, a tool that can answer metatheoretic queries in a matter of seconds. Second, I present GenMC, the

first scalable stateless model checker that is parametric in the choice of the memory model. To enhance the usability of GenMC, I demonstrate how Kater can be used to automate the porting of new memory models into GenMC, as well as how the state-space size of concurrent programs can be estimated.

3.9 Abstraction for Crash-Resilient Objects

Ori Lahav (Tel Aviv University, IL)

License © Creative Commons BY 4.0 International license
© Ori Lahav

Joint work of Artem Khyzha, Ori Lahav

Main reference Artem Khyzha, Ori Lahav: “Abstraction for Crash-Resilient Objects”, in Proc. of the Programming Languages and Systems – 31st European Symposium on Programming, ESOP 2022, Held as Part of the European Joint Conferences on Theory and Practice of Software, ETAPS 2022, Munich, Germany, April 2-7, 2022, Proceedings, Lecture Notes in Computer Science, Vol. 13240, pp. 262–289, Springer, 2022.

URL https://doi.org/10.1007/978-3-030-99336-8_10

We study abstraction for crash-resilient concurrent objects using non-volatile memory (NVM). We develop a library-correctness criterion that is sound for ensuring contextual refinement in this setting, thus allowing clients to reason about library behaviors in terms of their abstract specifications, and library developers to verify their implementations against the specifications abstracting away from particular client programs. As a semantic foundation we employ a recent NVM model, called Persistent Sequential Consistency, and extend its language and operational semantics with useful specification constructs. The proposed correctness criterion accounts for NVM-related interactions between client and library code due to explicit persist instructions, and for calling policies enforced by libraries. We illustrate our approach on two implementations and specifications of simple persistent objects with different prototypical durability guarantees. Our results provide the first approach to formal compositional reasoning under NVM.

3.10 Fairness for load buffering memory models

Anton Podkopaev (JetBrains – Amsterdam, NL)

License © Creative Commons BY 4.0 International license
© Anton Podkopaev

Joint work of Anton Podkopaev, Ori Lahav

Main reference Ori Lahav, Egor Namakonov, Jonas Oberhauser, Anton Podkopaev, Viktor Vafeiadis: “Making weak memory models fair”, Proc. ACM Program. Lang., Vol. 5(OOPSLA), pp. 1–27, 2021.

URL <https://doi.org/10.1145/3485475>

Liveness properties, such as termination, of even the simplest shared-memory concurrent programs under sequential consistency typically require some fairness assumptions about the scheduler. Under weak memory models, we observe that the standard notions of thread fairness are insufficient, and an additional fairness property, which we call memory fairness, is needed.

In previous work (Lahav et al., 2021), we proposed a uniform definition for memory fairness that can be integrated into any declarative memory model enforcing acyclicity of the union of the program order (po) and the reads-from (rf) relations. For the well-known models, SC, x86-TSO, RA, and StrongCOH, that have equivalent operational and declarative presentations, we showed that our declarative memory fairness condition is equivalent to an intuitive model-specific operational notion of memory fairness, which requires the memory system to fairly execute its internal propagation steps.

Now, we raise a question on how memory models allowing poUrf-cycles, i.e., allowing load buffering, should be restricted to provide liveness guarantees. We showed that for Armv8 and Power memory models the memory fairness condition is enough to preserve our compilation correctness result (Podkopaev et al., 2019) for the Promising semantics (Kang et al., 2017), if the set of locations accessed by a program is finite (for both Armv8 and Power compilation targets) as well as a number of threads spawned by the program (needed only for Power). We also show that the compilation result is preserved for a restricted version of the Promising semantics, which we call `Promising_fair`. The restriction guarantees that any promise made by a thread is eventually fulfilled.

3.11 DARTAGNAN: One tool for all models

Hernán Ponce de León (Huawei Technologies – München, DE)

License © Creative Commons BY 4.0 International license
© Hernán Ponce de León

Main reference Thomas Haas, Roland Meyer, Hernán Ponce de León: “CAAT: consistency as a theory”, Proc. ACM Program. Lang., Vol. 6(OOPSLA2), pp. 114–144, 2022.

URL <https://doi.org/10.1145/3563292>

The notion of consistency exists in several communities within computer science: programming languages, CPUs, databases, GPUs, non-volatile memory, etc. Despite the different application domains, the foundations are not that different. CAT is a domain specific language that allows to formalize different notions of consistency. Having a unified DSL facilitates the building of verification technology that works across all the application domains.

In this talk, I present how to encode program correctness with respect to a given CAT model using SMT based Bounded Model Checking. I show how to achieve scalability (i.e., we can verify real code coming from the Linux kernel) based on two key ideas. The first contribution (OOPSLA’22) is based in interpreting consistency as an SMT theory. This allows to simplify the part of the encoding that needs to be handled by the SAT solver and moves the hardness of encoding consistency into the theory solver where we can use domain specific knowledge to improve solving time. The second contribution (OOPSLA’23) is a static analysis of the consistency model which allows to propagate information in two directions: bottom-up from base relations to derived relations, and top-down from consistency axioms over derived relations to base relations.

3.12 Towards a formal specification of the Intel Architecture

Alastair Reid (Intel – London, GB)


License © Creative Commons BY 4.0 International license
© Alastair Reid

Formal specifications of CPU architectures are useful for formally verifying hardware and software; for verifying compilers; for discovering compiler peephole; and for analyzing programs for security issues. We are creating a formal specification of the Intel Architecture (aka “x86”) with the intention that the specification is complete (e.g., able to boot an OS or run SGX code); correct (e.g., validated using the same tests that processors are tested with); readable (e.g., suitable for use in the Intel Software Developer’s Manual); available (e.g., on GitHub and licensed under a suitably permissive license); usable (tools are available and can

be used as the basis of other tools); and used by our customers, 3rd party developers, the open source community, academic researchers, etc. (we will help users understand what the spec can do and how to use and adapt the tools to enable a diverse set of uses).

3.13 System and Failure Models Matter

Michael Scott (University of Rochester, US)

License  Creative Commons BY 4.0 International license
© Michael Scott

Those of us here today are intensely interested in formal models of concurrency and persistence. In addition to posing challenging problems from an intellectual perspective, these models need to capture key aspects of real-world systems if they are to have an impact on practice. In this talk I briefly survey (my opinions regarding) the space of interesting models. In particular, we can consider


- system models, which capture the hardware and software architecture on which our algorithms run;
- persistency models, which capture instruction-level ordering and the reads-see-writes relationship in the presence of crashes; and
- failure models, which consider what exactly may fail, what resumes, and how.

Among other things, I suggest that:

- Independent thread failures (esp. with threads that are recovered and continue) have few if any real-world analogues, and should be pursued with caution.
- Real-world systems are likely to have much more NVM than DRAM, so work that mirrors all persistent data in DRAM should be pursued with caution.
- Hardware designers have considerable motivation to develop persistent caches, so work that assumes that cached data will always be transient should be pursued with caution.
- The world is still looking for an NVM killer app.

3.14 Transactional Semantics with Zombies

Michael Scott (University of Rochester, US)

License  Creative Commons BY 4.0 International license
© Michael Scott

Main reference Michael Scot: “Transactional Semantics with Zombies” Invited presentation, 6th Workshop on the Theory of Transactional Memory (WTTM), Paris, France, July 2014.

Different formal models of transactional memory are required at different levels of the system stack. This paper focuses on the run-time level, where the semantics of individual operations (start, read, write, try-commit) govern the interactions between the compiler and the TM system. For sandboxing TM systems, which allow a doomed transaction (a “zombie”) to continue for some time beyond an inconsistent read, run-time-level semantics cannot be captured by opacity as currently defined: we need a formal model of zombie execution.

3.15 Specifying and Verifying Persistent Libraries

Léo Stefanescu (MPI-SWS – Kaiserslautern, DE)

License © Creative Commons BY 4.0 International license
© Léo Stefanescu

Joint work of Léo Stefanescu, Azalea Raad, Viktor Vafeiadis

Main reference Léo Stefanescu, Azalea Raad, Viktor Vafeiadis: “Specifying and Verifying Persistent Libraries”, CoRR, Vol. abs/2306.01614, 2023.

URL <https://doi.org/10.48550/ARXIV.2306.01614>

We present a general framework for specifying and verifying persistent libraries, that is, libraries of data structures that provide some persistency guarantees upon a failure of the machine they are executing on. Our framework enables modular reasoning about the correctness of individual libraries (horizontal and vertical compositionality) and is general enough to encompass all existing persistent library specifications ranging from hardware architectural specifications to correctness conditions such as durable linearizability. As case studies, we specify the FliT and Mirror libraries, verify their implementations over Px86, and use them to build higher-level durably linearizable libraries, all within our framework. We also specify and verify a persistent transaction library that highlights some of the technical challenges which are specific to persistent memory compared to weak memory and how they are handled by our framework.

3.16 A Type System for Intermittent Computing

Milijana Surbatovich (Carnegie Mellon University – Pittsburgh, US)

License © Creative Commons BY 4.0 International license
© Milijana Surbatovich

Joint work of Milijana Surbatovich, Naomi Spargo, Limin Jia, Brandon Lucia

Main reference Milijana Surbatovich, Naomi Spargo, Limin Jia, Brandon Lucia: “A Type System for Safe Intermittent Computing”, Proc. ACM Program. Lang., Vol. 7(PLDI), pp. 736–760, 2023.

URL <https://doi.org/10.1145/3591250>

Batteryless, energy harvesting devices (EHDs) enable computing in environments that are too remote or inaccessible to support battery maintenance, benefiting application domains like disaster monitoring, health wearables, and smart civil and agricultural infrastructure. Instead of relying on a battery, these devices harvest all energy they need from their surroundings. Because the target application domains have high assurance requirements, computation on EHDs should be correct; unfortunately, harvested energy is typically too weak to power a device continuously, resulting in frequent power failures that break software and systems designed to run on continuous power. The field of intermittent computing seeks to overcome the correctness and programmability challenges introduced by these power failures but has historically relied on ad-hoc correctness reasoning that provides no guarantees.

This talk motivates the need for formal methods research for designing correct intermittent systems, highlighting the importance of modularity and abstraction in both formalism and system design. It then presents Curricule, an information-flow type system for reasoning about safe intermittent execution that gives programmers more control and provides natural layering between the application and runtime system levels of the stack. The talk concludes by discussing open problems in the intermittent computing field.

3.17 Separation Logic for Concurrent, Crash-Safe Systems

Joseph Tassarotti (New York University, US)

License © Creative Commons BY 4.0 International license
© Joseph Tassarotti

Joint work of Tej Chajed, Joseph Tassarotti, Mark Theng, Ralf Jung, M. Frans Kaashoek, Nickolai Zeldovich
Main reference Tej Chajed, Joseph Tassarotti, Mark Theng, Ralf Jung, M. Frans Kaashoek, Nickolai Zeldovich: “GoJournal: a verified, concurrent, crash-safe journaling system”, in Proc. of the 15th USENIX Symposium on Operating Systems Design and Implementation, OSDI 2021, July 14-16, 2021, pp. 423–439, USENIX Association, 2021.

URL <https://www.usenix.org/conference/osdi21/presentation/chajed>

Storage systems, such as databases and file systems, often have concurrent implementations. These systems are expected to be crash-safe, meaning that they should be able to recover from failures caused by power loss. However, achieving crash-safety is difficult because programmers must consider many potential interleavings of threads as well as the possibility of interruption from crashes at any point. Perennial is a separation logic framework for formally verifying crash safety of concurrent systems. This talk describes the core reasoning principles of Perennial and our experience using Perennial to verify GoTxn, a concurrent transaction system.

3.18 Persistent Scripting

Haris Volos (University of Cyprus – Nicosia, CY)

License © Creative Commons BY 4.0 International license
© Haris Volos

Joint work of Zi Fan Tan, Jianan Li, Terence Kelly, Arnold Robbins, Haris Volos

Persistent scripting brings the benefits of persistent memory programming to high-level interpreted languages. More importantly, it brings the convenience and programmer productivity of scripting to persistent memory programming. We have integrated a novel generic persistent memory allocator into a popular scripting language interpreter, which now exposes a simple and intuitive persistence interface: A flag notifies the interpreter that a script’s variables reside in a persistent heap in a specified file. The interpreter begins script execution with all variables in the persistent heap ready for immediate use. New variables defined by the running script are allocated on the persistent heap and are thus available to subsequent executions. Scripts themselves are unmodified and persistent heaps may be shared freely between unrelated scripts.

3.19 Verifying the persistency library FLiT

Heike Wehrheim (Universität Oldenburg, DE)

License © Creative Commons BY 4.0 International license
© Heike Wehrheim

Joint work of Stefan Bodenmüller, John Derrick, Brijesh Dongol, Gerhard Schellhorn, Heike Wehrheim
Main reference Stefan Bodenmüller, John Derrick, Brijesh Dongol, Gerhard Schellhorn, Heike Wehrheim: “A Fully Verified Persistency Library”, in Proc. of the Verification, Model Checking, and Abstract Interpretation – 25th International Conference, VMCAI 2024, London, United Kingdom, January 15-16, 2024, Proceedings, Part II, Lecture Notes in Computer Science, Vol. 14500, pp. 26–47, Springer, 2024.

URL https://doi.org/10.1007/978-3-031-50521-8_2

Non-volatile memory (NVM) technologies offer DRAM-like speeds with the added benefit of failure resilience. However, developing concurrent programs for NVM can be challenging since programmers must consider both inter-thread synchronisation and durability aspects

at the same time. To alleviate this, libraries such as FliT have been developed to manage transformations to durability, allowing a linearizable concurrent object to be converted into a durably linearizable one with minimal programmer effort. However, a formal proof of correctness for FliT is missing, and standard proof techniques for durable linearizability are challenging to apply since FliT itself is not durably linearizable.

In this talk, I report on our work on showing correctness of transformations to durability. First, we develop an abstract persistency library (called PLib) that operationally characterises transformations to durability and we prove its correctness. Second, we show correctness of the library FliT by proving that FliT refines PLib under the realistic Px86 memory model, i.e., the persistent version of TSO memory model implemented by Intel architectures. The proof of refinement between FliT and PLib has been mechanised within the theorem prover KIV. Taken together, these proofs guarantee that FliT is also sound wrt transformations to durability.

Participants

- Guillaume Ambal
Imperial College London, GB
- Parosh Aziz Abdulla
Uppsala University, SE
- Mark Batty
University of Kent –
Canterbury, GB
- Michael D. Bond
Ohio State University –
Columbus, US
- Ahmed Bouajjani
Université Paris Cité, FR
- Paulo Emílio de Vilhena
Imperial College London, GB
- Brijesh Dongol
University of Surrey –
Guildford, GB
- Michal Friedman
ETH Zürich, CH
- Ohad Kammar
University of Edinburgh, GB
- Jeehoon Kang
KAIST – Daejeon, KR
- Vasileios Klimis
Queen Mary University of
London, GB
- Michalis Kokologiannakis
MPI-SWS – Kaiserslautern, DE
- Ori Lahav
Tel Aviv University, IL
- Anton Podkopaev
JetBrains – Amsterdam, NL
- Hernán Ponce de León
Huawei Technologies –
München, DE
- Azalea Raad
Imperial College London, GB
- Alastair Reid
Intel – London, GB
- Michael Scott
University of Rochester, US
- Léo Stefanescu
MPI-SWS – Kaiserslautern, DE
- Milijana Surbatovich
Carnegie Mellon University –
Pittsburgh, US
- Joseph Tassarotti
New York University, US
- Viktor Vafeiadis
MPI-SWS – Kaiserslautern, DE
- Haris Volos
University of Cyprus –
Nicosia, CY
- Heike Wehrheim
Universität Oldenburg, DE



Quantum Cryptanalysis

Gorjan Alagic^{*1}, Maria Naya-Plasencia^{*2}, Rainer Steinwandt^{*3}, and Manasi Shingane^{†4}

1 University of Maryland – College Park, US. galagic@gmail.com

2 INRIA – Paris, FR. maria.naya_plasencia@inria.fr

3 University of Alabama in Huntsville, US. rs0141@uah.edu

4 University of Maryland – College Park, US. mshingan@umd.edu

Abstract

This report documents the program and the outcomes of Dagstuhl Seminar 23421 “Quantum Cryptanalysis”. The seminar took place as an in-person event in October 2023 and was the seventh installment of the Dagstuhl Seminar series on Quantum Cryptanalysis. This report describes the motivation and technical scope of the seminar as well as the (updated) organizational structure of this week-long event. We also include abstracts of the seminar presentations given by participants and a description of the activities of the working groups.

Seminar October 15–20, 2023 – <https://www.dagstuhl.de/23421>

2012 ACM Subject Classification Security and privacy → Cryptanalysis and other attacks

Keywords and phrases computational algebra, cryptanalysis, post-quantum cryptography, quantum algorithms, quantum resource estimation

Digital Object Identifier 10.4230/DagRep.13.10.65

1 Executive Summary

Gorjan Alagic (University of Maryland – College Park, US)

Stacey Jeffery (CWI – Amsterdam, NL)

Maria Naya-Plasencia (INRIA – Paris, FR)

Rainer Steinwandt (University of Alabama in Huntsville, US)

License  Creative Commons BY 4.0 International license

© Gorjan Alagic, Stacey Jeffery, Maria Naya-Plasencia, and Rainer Steinwandt

Motivation and technical scope

Due to the coronavirus pandemic, the previous Dagstuhl Seminar in the Quantum Cryptanalysis series (in 2021) took place in a hybrid format. With this latest installment in 2023, we returned to the standard fully in-person format at Schloss Dagstuhl and incorporated more group work. Since the 2021 meeting, the scientific community progressed significantly in developing and standardizing post-quantum cryptography for general use. In particular, the U.S. National Institute of Standards and Technology (NIST) announced that it will standardize several public-key cryptographic schemes. The study of candidates in this process has been a focus of past installments of the Quantum Cryptanalysis seminar series and this year’s Dagstuhl Seminar. The 2023 seminar was also interested in the analysis of two more scheme categories. The first category consists of additional public-key schemes that either have different performance profiles, or different security properties (e.g., are based on the hardness of other mathematical problems) than the NIST-selected schemes. The second

* Editor / Organizer

† Editorial Assistant / Collector



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

Quantum Cryptanalysis, *Dagstuhl Reports*, Vol. 13, Issue 10, pp. 65–75

Editors: Gorjan Alagic, Maria Naya-Plasencia, and Rainer Steinwandt



Dagstuhl Reports

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

category consists of symmetric-key schemes; while post-quantum standardization has not as yet focused on symmetric-key cryptography, there are many open questions about their security in the presence of quantum adversaries.

As one would expect from the title of the seminar, studying the best-known algorithmic attacks on cryptographic schemes was a focus of the conversations. Understanding the best-known attacks enables cryptographers to select the strongest schemes and set their parameters in a manner that appropriately balances security with performance. Technical talks included work on quantum-computational algorithms for attacking three categories of public-key schemes: lattice-based, code-based, and isogeny-based. We also had two presentations on new ideas for attacking symmetric-key cryptography using quantum computers. In addition, the technical program included an update from NIST on the progress of their various standardization processes related to the seminar scope.

As in the past, the seminar brought together researchers in several relevant fields, including quantum-computational algorithms, classical public-key and symmetric-key cryptography, and the mathematics of lattices and codes. This enabled the participants to get an overview of the latest advances in all of these fields.

Organization

To leverage some of the unique opportunities Schloss Dagstuhl offers, as in the past, we left ample time for discussions and collaboration; the typical day called for between two and three presentations total. The remaining time was more structured than in past instances of the seminar. Before the seminar began, the organizers contacted the participants to solicit topics and started to organize working groups. The first day of the seminar was then mainly focused on establishing the working groups and the technical topics they would focus on. The working groups met throughout the week to discuss their technical subjects and regularly reported their progress to the entire seminar. The participant-selected working group topics were:

- quantum algorithms for the lattice isomorphism problem (a new problem with potential for post-quantum applications),
- Regev’s quantum factoring algorithm (a new algorithm that may affect how soon current cryptography will become obsolete),
- cryptanalysis of LR5 (a fundamental building block in symmetric-key cryptography), and
- code-based cryptosystems (these are next on the slate of possible standardized schemes).

Following the Dagstuhl tradition and in line with prior seminars in the Quantum Cryptanalysis series, there was no technical program during Wednesday afternoon. This enabled participants to explore the surroundings or spend more time on collaborative research.

With 34 participants, Schloss Dagstuhl hosted a diverse group of leading experts from across the globe. A significant number of the participants were graduate students. These young researchers were able to interact with leading experts in working on the latest science and gain valuable insights to help them developing their career.

Results and next steps

The working groups were a welcome addition this year, with several participants praising this style of seminar structure. The working groups were able to make technical progress during the week and several groups continued collaborating after the seminar.

The various technical presentations showed that significant progress is being made in the field more generally. This indicates that the intersection of quantum computing and classical cryptography is a vibrant and active field. The Dagstuhl Seminar series on Quantum Cryptanalysis plays an important role in this area of science. We expect this will continue, as the community carries on with the process of standardizing and deploying post-quantum cryptography in the real world. This process is already generating challenging scientific questions that the seminar could help address. For instance, the only general-purpose schemes currently slated for standardization are based on lattice problems; how can the community select high-performing replacement schemes that can serve as a backup in case lattices fail?

2 Table of Contents

Executive Summary

Gorjan Alagic, Stacey Jeffery, Maria Naya-Plasencia, and Rainer Steinwandt 65

Overview of Talks

NIST PQC process update

Gorjan Alagic and Daniel C. Smith-Tone 69

Single-query Quantum Hidden Shift Attacks

Xavier Bonnetain 69

Quantum algorithms for isogeny-based cryptography

Péter Kutas 70

Quantum Linear Key-recovery Attacks Using the QFT

André Schrottenloher 70

Quantum algorithms for lattice problems

Yixin Shen 70

Quantum decoding problem

Jean-Pierre Tillich 71

Working groups

Quantum algorithms for Lattice Isomorphism Problem

Jean-François Biasse 71

Regev’s quantum factoring algorithm

Martin Ekerå 72

Cryptanalysis of LR5

Christian Majenz 73

Code-based group

Jean-Pierre Tillich 73

Participants 75

3 Overview of Talks

3.1 NIST PQC process update

Gorjan Alagic (University of Maryland – College Park, US) and Daniel C. Smith-Tone (NIST – Gaithersburg, US)

License  Creative Commons BY 4.0 International license
© Gorjan Alagic and Daniel C. Smith-Tone

Since 2016, the U.S. National Institute of Standards and Technology has been running a standardization process for post-quantum public-key cryptography. So far, this process has produced one standard for a key encapsulation mechanism (Kyber / ML-KEM) and three standards for digital signature schemes (Dilithium / ML-DSA, Falcon / FN-DSA, and SPHINCS+ / SLH-DSA). Three of these standards are currently drafts open for public comment. At the same time, NIST is continuing to look at post-quantum KEMs, and has begun an additional process for standardizing more signature schemes. This talk will give an overview of this process and what the future might hold.

3.2 Single-query Quantum Hidden Shift Attacks

Xavier Bonnetain (LORIA & INRIA Nancy, FR)

License  Creative Commons BY 4.0 International license
© Xavier Bonnetain

Quantum attacks using superposition queries are known to break many classically secure modes of operation. While these attacks do not necessarily threaten the security of the modes themselves, since they rely on a strong adversary model, they help us to draw limits on the provable security of these modes.

Typically these attacks use the structure of the mode (stream cipher, MAC or authenticated encryption scheme) to embed a period-finding problem, which can be solved with a dedicated quantum algorithm. The hidden period can be recovered with a few superposition queries (e.g., $O(n)$ for Simon's algorithm), leading to state or key-recovery attacks. However, this strategy breaks down if the period changes at each query, e.g., if it depends on a nonce.

In this talk, we focus on this case and give dedicated state-recovery attacks on the authenticated encryption schemes Rocca, Rocca-S, Tiaoxin-346 and AEGIS-128L. These attacks rely on a procedure to find a Boolean hidden shift with a single superposition query, which overcomes the change of nonce at each query. As they crucially depend on such queries, we stress that they do not break any security claim of the authors, and do not threaten the schemes if the adversary only makes classical queries.

3.3 Quantum algorithms for isogeny-based cryptography


Péter Kutas (University of Birmingham, GB)

License  Creative Commons BY 4.0 International license
© Péter Kutas

In the talk we surveyed quantum algorithms relevant to isogeny-based cryptography. One aspect of isogenies is that one can instantiate cryptographic group actions with them that still retain certain properties of discrete logarithms but are not susceptible to attacks via Shor’s algorithm. We discussed certain reductions between hard problems related to group actions most importantly the quantum equivalence of inverting the group action and the computational Diffie-Hellman problem (and that such an equivalence is highly unlikely in the classical setting as it would mean that the discrete logarithm and factoring assumptions do not hold). We also discussed recent quantum attacks on pSIDH utilizing a non-abelian hidden subgroup problem and improved quantum algorithms for finding fixed degree isogenies.

3.4 Quantum Linear Key-recovery Attacks Using the QFT

André Schrottenloher (INRIA – Rennes, FR)

License  Creative Commons BY 4.0 International license
© André Schrottenloher
Main reference André Schrottenloher: “Quantum Linear Key-recovery Attacks Using the QFT”, 2023.
URL <https://eprint.iacr.org/2023/184>

The Quantum Fourier Transform is a fundamental tool in quantum cryptanalysis. In symmetric cryptanalysis, hidden shift algorithms such as Simon’s (FOCS 1994), which rely on the QFT, have been used to obtain structural attacks on some very specific block ciphers. The Fourier Transform is also used in classical cryptanalysis, for example in FFT-based linear key-recovery attacks introduced by Collard et al. (ICISC 2007). Whether such techniques can be adapted to the quantum setting has remained so far an open question.

In this paper, we introduce a new framework for quantum linear key-recovery attacks using the QFT. These attacks loosely follow the classical method of Collard et al., in that they rely on the fast computation of a “correlation state” in which experimental correlations, rather than being directly accessible, are encoded in the amplitudes of a quantum state. The experimental correlation is a statistic that is expected to be higher for the good key, and on some conditions, the increased amplitude creates a speedup with respect to an exhaustive search of the key. The same method also yields a new family of structural attacks, and new examples of quantum speedups beyond quadratic using classical known-plaintext queries.

3.5 Quantum algorithms for lattice problems

Yixin Shen (King’s College London, GB)

License  Creative Commons BY 4.0 International license
© Yixin Shen

In this talk, I survey some algorithmic problems that arise from the cryptanalysis of lattice-based cryptographic schemes such as the Shortest Vector problem and the Learning with Errors problem. Then I particularly focus on how quantum algorithms can help us obtain speed-ups on different approaches to solve those problems.

3.6 Quantum decoding problem

Jean-Pierre Tillich (INRIA – Paris, FR)

License © Creative Commons BY 4.0 International license
© Jean-Pierre Tillich

One of the founding results of lattice based cryptography is a quantum reduction from the Short Integer Solution problem to the Learning with Errors problem introduced by Regev. It has recently been pointed out by Chen, Liu and Zhandry that this reduction can be made more powerful by replacing the learning with errors problem with a quantum equivalent, where the errors are given in quantum superposition. In the context of codes, this can be adapted to a reduction from finding short codewords to a quantum decoding problem for random linear codes.

We therefore consider in this paper the quantum decoding problem, where we are given a superposition of noisy versions of a codeword and we want to recover the corresponding codeword. When we measure the superposition, we get back the usual classical decoding problem for which the best known algorithms are in the constant rate and error-rate regime exponential in the codelength. However, we will show here that when the noise rate is small enough, then the quantum decoding problem can be solved in quantum polynomial time. Moreover, we also show that the problem can in principle be solved quantumly (albeit not efficiently) for noise rates for which the associated classical decoding problem cannot be solved at all for information theoretic reasons.

We then revisit Regev’s reduction in the context of codes. We show that using our algorithms for the quantum decoding problem in Regev’s reduction matches best known quantum algorithms for the short codeword problem. This shows in some sense the tightness of Regev’s reduction when considering the quantum decoding problem and also paves the way for new quantum algorithms for the short codeword problem.

4 Working groups

4.1 Quantum algorithms for Lattice Isomorphism Problem

Jean-François Biasse (University of South Florida – Tampa, US)

License © Creative Commons BY 4.0 International license
© Jean-François Biasse

The lattice isomorphism problem (LIP) consists in finding a secret isometry between two input Euclidean lattices. LIP is a fundamental problem that has been studied for decades. Recently, Ducas and van Woerden proposed cryptosystems whose security rely on the presumed hardness of LIP. The known algorithms for the resolution of LIP rely on the calculation of short vectors in the input lattices. The shortest vector problem is a notoriously hard problem, even for quantum computers. This suggests that cryptosystems based on LIP might feature quantum resistance.

No quantum algorithms for the resolution of LIP have ever been described in the literature. The best classical algorithms for computing an isomorphism between two given lattices run in time n^n (or $2^{n/2}$ if one of the input lattices is \mathbb{Z}^n). Finding a quantum algorithm with a better complexity than the existing classical algorithms for the resolution of LIP is an open problem.

Our group investigated quantum algorithms for the resolution of LIP. Several avenues were considered:

- We rephrased LIP as a hidden shift problem, which is a task that can (in certain cases) be solved efficiently by quantum computers.
- We reviewed the generation of instances of the LIP that are used for the creation of cryptographic keys. We studied conditions on the parameters that can make the LIP-based cryptosystems insecure.
- We studied quantum analogues of the existing classical algorithms for the resolution of LIP.
- We researched quantum algorithms to compute short vectors in lattices that are rotations of \mathbb{Z}^n , which is a task for which there exist ad-hoc classical solutions that outperform generic methods.

4.2 Regev’s quantum factoring algorithm

Martin Ekerå (KTH Royal Institute of Technology – Stockholm, SE)

License  Creative Commons BY 4.0 International license
 © Martin Ekerå

The work in our breakout group focused on Regev’s recent d-dimensional variation [1] of Shor’s quantum factoring algorithm ([2], [3]), and on better understanding its advantages and disadvantages in practical implementations.

Of particular interest to our group was the very recent Fibonacci-based arithmetic [4] that seemingly resolves reversibility issues previously identified by Ekerå and Gidney in Regev’s binary tree-based arithmetic.

There were presentations of ongoing work, including work [5] on extending Regev’s algorithm to computing discrete logarithms, and work ([6], [7]) on simulating the quantum parts of the algorithms for integers of known factorization and for groups where computing discrete logarithms is classically easy.

The aforementioned simulators enable the heuristic assumptions in Regev’s analysis to be verified. Furthermore, they enable the robustness of the classical post-processing to erroneous runs to be analyzed – where an erroneous run is a run in which the error correction fails to properly correct all errors, leading to a bad vector being output. There was discussion in the group regarding options for filtering out good vectors from bad vectors.

For the extension to discrete logarithms to be efficient, it is required that the group has a notion of small elements, that when composed yield elements that are also small, and where the composition of small elements is considerably less expensive than the composition of arbitrary group elements. A notion of small elements exists for \mathbb{Z}_p^* . There was discussion in the group regarding whether a similar notion exists for elliptic curve groups.

References

- 1 Regev, O. An efficient quantum factoring algorithm. *ArXiv Preprint ArXiv:2308.06572*. (2023)
- 2 Shor, P. Algorithms for quantum computation: discrete logarithms and factoring. *Proceedings 35th Annual Symposium On Foundations Of Computer Science*. pp. 124-134 (1994)
- 3 Shor, P. Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer. *SIAM Review*. **41**, 303-332 (1999)

- 4 Ragavan, S. & Vaikuntanathan, V. Optimizing Space in Regev’s Factoring Algorithm. *ArXiv Preprint ArXiv:2310.00899*. (2023)
- 5 Ekerå, M. & Gärtner, J. Extending Regev’s factoring algorithm to compute discrete logarithms. *ArXiv Preprint ArXiv:2311.05545*. (2023)
- 6 M. Ekerå and J. Gärtner: “Simulating Regev’s quantum factoring algorithm”. GitHub repository [ekera/regevnum](https://github.com/ekera/regevnum). (2023) URL: <https://github.com/ekera/regevnum>
- 7 M. Ekerå and J. Gärtner: “Simulating our extension of Regev’s quantum factoring algorithm to compute discrete logarithms”. Unpublished GitHub repository. (2023)

4.3 Cryptanalysis of LR5

Christian Majenz (Technical University of Denmark – Lyngby, DK)

License © Creative Commons BY 4.0 International license
© Christian Majenz

The Feistel network is a versatile blueprint for constructing pseudorandom permutations (PRPs) and block ciphers. The simplest application is a family of constructions of a PRP from a pseudorandom function, indexed by the number of rounds. There is an extensive body of quantum attacks on the construction in the Q2 model, where an attacker has quantum query access to the PRP. For the five round variant, also known as LR5 (“Luby-Rackoff 5”), there is no quantum attack known separating its chosen-plaintext (CPA) security from its chosen-ciphertext (CCA) security (or its PRP security from its strong PRP security). In this working group, we explored a number of approaches of leveraging an existing polynomial-time on the four-round variant based on Simon’s algorithm to devise a CCA attack on five rounds that improves over the existing CPA attack.

4.4 Code-based group

Jean-Pierre Tillich (INRIA – Paris, FR)

License © Creative Commons BY 4.0 International license
© Jean-Pierre Tillich

In this working group we

1. first provided an introduction to the decoding problem suitable for a broad audience;
2. then we looked in detail at a recent (classical) algorithm for performing this task consisting in applying sieving techniques which are common in lattice based cryptography but not in code based cryptography. See [1]. A nice feature of this algorithm is that it uses relatively low memory and its running time is rather competitive when compared to the best decoding algorithms. This makes it a very good candidate for quantization.
3. In the last part of the working group, we went through one of the best quantum algorithm for performing lattice sieving based on random walks and suitable product codes for the unit sphere, see [2].

We concluded that these techniques should carry over for performing quantumly the sieving task relevant for decoding and discussed some technical points which can be found in Kevin Carrier’s thesis [3].

All in all this should lead to a new quantum algorithm for decoding a linear code which could be a record breaker in terms of complexity.

References

- 1 Ducas, L., Esser, A., Etinski, S. & Kirshanova, E. Asymptotics and Improvements of Sieving for Codes. *Cryptology EPrint Archive*. (2023)
- 2 Chailloux, A. & Loyer, J. Lattice sieving via quantum random walks. *Advances In Cryptology-ASIACRYPT 2021: 27th International Conference On The Theory And Application Of Cryptology And Information Security, Singapore, December 6-10, 2021, Proceedings, Part IV 27*. pp. 63-91 (2021)
- 3 Carrier, K. Recherche de presque-collisions pour le décodage et la reconnaissance de codes correcteurs. (Sorbonne université,2020)

Participants

- Gorjan Alagic
University of Maryland –
College Park, US
- Kaveh Bashiri
BSI – Bonn, DE
- Jean-François Biasse
University of South Florida –
Tampa, US
- Xavier Bonnetain
LORIA & INRIA Nancy, FR
- Yanlin Chen
CWI – Amsterdam, NL
- Arjan Cornelissen
IRIF – Paris, FR
- Martin Ekerå
KTH Royal Institute of
Technology – Stockholm, SE
- Lynn Engelberts
CWI – Amsterdam, NL &
QuSoft – Amsterdam, NL
- Simona Etinski
CWI – Amsterdam, NL
- Paul Frixons
INRIA Nancy – Grand Est, FR
- Vlad Gheorghiu
University of Waterloo, CA &
softwareQ Inc. – Waterloo, CA
- Sean Hallgren
Pennsylvania State University –
University Park, US
- Jacek Horecki
BEIT – Kraków, PL
- Akinori Hosoyamada
NTT – Tokyo, JP
- Péter Kutas
University of Birmingham, GB
- Johanna Loyer
INRIA – Paris, FR
- Frédéric Magniez
CNRS – Paris, FR
- Christian Majenz
Technical University of Denmark
– Lyngby, DK
- Alexander May
Ruhr-Universität Bochum, DE
- Garazi Muguruza
QuSoft & University of
Amsterdam, NL
- Maria Naya-Plasencia
INRIA – Paris, FR
- Lorenz Panny
TU München – Garching, DE
- Galina Pass
QuSoft – Amsterdam, NL
- Yu Sasaki
NTT – Tokyo, JP
- André Schrottenloher
INRIA – Rennes, FR
- Yixin Shen
King’s College London, GB
- Manasi Shingane
University of Maryland –
College Park, US
- Daniel C. Smith-Tone
NIST – Gaithersburg, US
- Jana Sotáková
University of Amsterdam, NL
- Rainer Steinwandt
University of Alabama in
Huntsville, US
- Jean-Pierre Tillich
INRIA – Paris, FR
- Maya-Iggy van Hoof
Ruhr-Universität Bochum, DE
- Michael Walter
Ruhr-Universität Bochum, DE
- Sara Zafar Jafarzadeh
University of Waterloo, CA &
Synopsys Inc. – Ottawa, CA



Graph Algorithms: Cuts, Flows, and Network Design

Jason Li^{*1}, Debmalya Panigrahi^{*2}, Laura Sanita^{*3}, and Thatchaphol Saranurak^{*4}

- 1 University of California – Berkeley, US. jmli@alumni.cmu.edu
- 2 Duke University – Durham, US. debmalya.panigrahi@gmail.com
- 3 Università Bocconi – Milan, IT. laura.sanita@unibocconi.it
- 4 University of Michigan – Ann Arbor, US. thsa@umich.edu

Abstract

This report documents the program and the outcomes of Dagstuhl Seminar 23422 “Graph Algorithms: Cuts, Flows, and Network Design”. This seminar brought 25 leading researchers in graph algorithms together for a discussion of the recent progress and challenges in two areas: the design of fast algorithm for fundamental flow/cut problems and the design of approximation algorithms for basic network design problems. The seminar included several talks of varying lengths, a panel discussion, and an open problem session. In addition, sufficient time was set aside for research discussions and collaborations.

Seminar October 15–20, 2023 – <https://www.dagstuhl.de/23422>

2012 ACM Subject Classification Theory of computation → Graph algorithms analysis; Theory of computation → Network flows

Keywords and phrases approximation, graph algorithm, maximum flow, minimum cut, network design

Digital Object Identifier 10.4230/DagRep.13.10.76

1 Executive Summary

Jason Li (University of California – Berkeley, US)

Debmalya Panigrahi (Duke University – Durham, US)

Laura Sanita (Università Bocconi – Milan, IT)

Thatchaphol Saranurak (University of Michigan – Ann Arbor, US)

License  Creative Commons BY 4.0 International license

© Jason Li, Debmalya Panigrahi, Laura Sanita, and Thatchaphol Saranurak

Graph algorithms are among the foundational pillars of algorithm design and combinatorial optimization. In addition to its significance as a theoretical discipline, graph algorithms are also ubiquitous in practice, with applications in essentially every scientific discipline. This has spawned research in many different directions within graph algorithms over the past few decades, and these individual research areas have come to play important roles in the evolution of algorithms research as a whole. Two particularly large and successful subdisciplines are those of *fast algorithms for flows and cuts* and *approximation algorithms for network design*. Many of the algorithmic ideas and techniques that are a standard feature of an algorithmist’s toolkit today trace their origins to groundbreaking research in these two areas spanning problems such as minimum cuts, maximum flows, Steiner trees, and the traveling salesman problem. The last few years, in particular, have been truly outstanding in achieving progress on longstanding questions in both areas. Some of the highlights include the first progress

* Editor / Organizer



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

Graph Algorithms: Cuts, Flows, and Network Design, *Dagstuhl Reports*, Vol. 13, Issue 10, pp. 76–89

Editors: Jason Li, Debmalya Panigrahi, Laura Sanita, and Thatchaphol Saranurak



DAGSTUHL REPORTS

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

in decades for problems such as the traveling salesman problem (e.g., [20, 19, 17, 28, 34]), graph connectivity augmentation (e.g., [9, 10, 35]), minimum cut (e.g., [24, 11]), vertex connectivity (e.g., [23]), all-pairs minimum cuts (e.g., [3, 2, 5, 4, 6, 25, 26, 1]), and last but not the least, the recent breakthrough achieving an almost linear-time algorithm for maximum flow (and minimum cost flow) [12] (see also [14, 16, 37, 36]).

Traditionally, to a large extent, research in these two subfields has progressed independent of one another: connectivity problems such as minimum cuts and maximum flows are typically polynomial-time solvable and goal is to improve the running time (*efficiency*) of the algorithms; in contrast, network design problems such as Steiner tree and TSP are NP-hard and the goal is to obtain the best *approximation* factor in (any) polynomial time. This has meant that the two areas have focused on different sets of technical tools – the former has developed combinatorial (and more recently, continuous) methods aimed at improving running times, while the latter has focused on polyhedral techniques and the use of mathematical programming for obtaining improved approximations.

In recent years, however, this distinction between the two subfields has started to blur, and the two areas have started to move closer to one another. This is for two main reasons:

- (a) *Recent progress in foundational questions in each area has crucially relied on structural insights from the other area.* For example, one of the main new ingredients in the recent breakthrough results in approximation algorithms for the traveling salesman problem (e.g., [20, 17]) is a better understanding of the structure of near-minimum cuts (e.g., [7]) in an undirected graph. Or, recent work in the all-pairs minimum cuts problem [1] that advances the state of the art for this problem after 60 years crucially makes use of Steiner tree approximations [27]. Or, cut matching games originally devised for sparsest cut approximations [22] have led to fast expander decompositions (e.g., [30]) that, in turn, play a crucial role in recent progress in deterministic minimum cut algorithms [24].
- (b) *There is growing interest in understanding approximation-efficiency tradeoffs in graph algorithms.* Graph sparsification (e.g., [8, 15, 32, 33]) has emerged as a standard tool that “compresses” an arbitrary graph into a sparse subgraph (called the *sparsifier*) while approximately preserving the values of all cuts in the graph. This naturally leads to an approximation-efficiency tradeoff by running existing algorithms on the sparsifier rather than on the input graph. But, beyond the black box use of sparsifiers, efficiency at the expense of mild approximation has been employed as a technical tool to breach longstanding running time barrier in recent years, and has often eventually led to faster exact algorithms as well. A famous example is the maximum flow problem in undirected graphs, where nearly-linear time approximation algorithms were designed in the last decade [21, 31, 29] and has eventually resulted in the very recent breakthrough achieving an almost-linear time exact algorithm [12]. Another recent example is the all-pairs minimum cuts problem for which the first paper to breach the 60-year old running time bound of Gomory and Hu [18] incurred a mild approximation [25], but this has now led to a faster exact algorithm as well [1]. Finally, understanding the efficiency-approximation tradeoff is an important goal for NP-hard network design problems such as Steiner tree [27] and Steiner forest [13], and this, in turn, has implications for minimum cut problems [1].

The goal of this seminar was to bring the leading researchers from these two communities of fast flow/cut algorithms and approximation algorithms for network design together for an exchange of ideas and knowledge, and a discussion of the major technical challenges in each research area.

Seminar Structure and Participants

The seminar brought together 25 researchers from the two communities highlighted above, roughly equally split between the two areas. There was also a mix of senior and junior participants, ranging from senior members of the community to current PhD students and postdoctoral researchers. In terms of gender balance, around 20% of the attendees were female. (The organizers had originally planned for a more equitable balance, but there were several late retractions, primarily due to geopolitical reasons, that affected the gender ratio.)

There were 17 scheduled talks, divided into long (60 minutes) and short (30 minutes) presentations. There was also an open problem session and a panel discussion on the future directions for the community. The schedule left plenty of time for collaboration and free discussion among the participants.

Outcomes

The main objective of the seminar was to provide a forum for the exchange of ideas between the research communities of fast flow/cut algorithms and approximation algorithms for network design. These are adjoining areas where researchers have a working knowledge of, and appreciation for, each other's work. As expected, the seminar led to cohesive interactions and meaningful discussions. Individual research talks and afternoon breaks created concrete opportunities for learning about recent progress in each other's areas and fostering collaborations. The open problem session highlighted the major research challenges in the two areas, which is particularly beneficial for junior members of the community who attended the seminar. The panel discussion allowed the participants to reflect on and discuss higher-level questions about the research directions that the communities should pursue in the near future. Overall, we believe that the seminar played an important role in community building, research collaborations, and in shaping the two research areas for the foreseeable future.

The organizers would like to thank the Scientific Directorate and the administration of the Dagstuhl Center for their amazing support in the organization of the Dagstuhl Seminar, and for supporting this important research area.

References

- 1 Amir Abboud, Robert Krauthgamer, Jason Li, Debmalya Panigrahi, Thatchaphol Saranurak, and Ohad Trabelsi. Gomory-Hu tree in subcubic time. *CoRR*, abs/2111.04958, 2021.
- 2 Amir Abboud, Robert Krauthgamer, and Ohad Trabelsi. Cut-equivalent trees are optimal for min-cut queries. In *61st IEEE Annual Symposium on Foundations of Computer Science, FOCS 2020*. IEEE Computer Society, 2020.
- 3 Amir Abboud, Robert Krauthgamer, and Ohad Trabelsi. New algorithms and lower bounds for all-pairs max-flow in undirected graphs. In Shuchi Chawla, editor, *Proceedings of the 2020 ACM-SIAM Symposium on Discrete Algorithms, SODA 2020, Salt Lake City, UT, USA, January 5–8, 2020*, pages 48–61. SIAM, 2020.
- 4 Amir Abboud, Robert Krauthgamer, and Ohad Trabelsi. APMF < APSP? Gomory-Hu tree for unweighted graphs in almost-quadratic time. In *62nd IEEE Annual Symposium on Foundations of Computer Science, FOCS 2021, Denver, CO, USA, February 7–10, 2022*, pages 1135–1146. IEEE, 2021.
- 5 Amir Abboud, Robert Krauthgamer, and Ohad Trabelsi. Subcubic algorithms for Gomory-Hu tree in unweighted graphs. In Samir Khuller and Virginia Vassilevska Williams, editors,

- STOC '21: 53rd Annual ACM SIGACT Symposium on Theory of Computing, Virtual Event, Italy, June 21–25, 2021*, pages 1725–1737. ACM, 2021.
- 6 Amir Abboud, Robert Krauthgamer, and Ohad Trabelsi. Friendly cut sparsifiers and faster gomory-hu trees. In Joseph (Seffi) Naor and Niv Buchbinder, editors, *Proceedings of the 2022 ACM-SIAM Symposium on Discrete Algorithms, SODA 2022, Virtual Conference / Alexandria, VA, USA, January 9–12, 2022*, pages 3630–3649. SIAM, 2022.
 - 7 András A Benczúr. A representation of cuts within $6/5$ times the edge connectivity with applications. In *Proceedings of IEEE 36th Annual Foundations of Computer Science*, pages 92–102. IEEE, 1995.
 - 8 András A. Benczúr and David R. Karger. Randomized approximation schemes for cuts and flows in capacitated graphs. *SIAM J. Comput.*, 44(2):290–319, 2015.
 - 9 Jaroslav Byrka, Fabrizio Grandoni, and Afrouz Jabal Ameli. Breaching the 2-approximation barrier for connectivity augmentation: a reduction to Steiner tree. In *Proceedings of the 52nd Annual ACM SIGACT Symposium on Theory of Computing (STOC)*, pages 815–825, 2020.
 - 10 Federica Cecchetto, Vera Traub, and Rico Zenklusen. *Bridging the Gap between Tree and Connectivity Augmentation: Unified and Stronger Approaches*, page 370–383. Association for Computing Machinery, New York, NY, USA, 2021.
 - 11 Ruoxu Cen, Jason Li, Danupon Nanongkai, Debmalya Panigrahi, Thatchaphol Saranurak, and Kent Quanrud. Minimum cuts in directed graphs via partial sparsification. In *62nd IEEE Annual Symposium on Foundations of Computer Science, FOCS 2021, Denver, CO, USA, February 7–10, 2022*, pages 1147–1158. IEEE, 2021.
 - 12 Li Chen, Rasmus Kyng, Yang P. Liu, Richard Peng, Maximilian Probst Gutenberg, and Sushant Sachdeva. Maximum flow and minimum-cost flow in almost-linear time. *CoRR*, abs/2203.00671, 2022.
 - 13 Richard Cole, Ramesh Hariharan, Moshe Lewenstein, and Ely Porat. A faster implementation of the goemans-williamson clustering algorithm. In S. Rao Kosaraju, editor, *Proceedings of the Twelfth Annual Symposium on Discrete Algorithms, January 7–9, 2001, Washington, DC, USA*, pages 17–25. ACM/SIAM, 2001.
 - 14 Sally Dong, Yu Gao, Gramoz Goranci, Yin Tat Lee, Richard Peng, Sushant Sachdeva, and Guanghao Ye. Nested dissection meets ipms: Planar min-cost flow in nearly-linear time. In Joseph (Seffi) Naor and Niv Buchbinder, editors, *Proceedings of the 2022 ACM-SIAM Symposium on Discrete Algorithms, SODA 2022, Virtual Conference / Alexandria, VA, USA, January 9–12, 2022*, pages 124–153. SIAM, 2022.
 - 15 Wai Shing Fung, Ramesh Hariharan, Nicholas J. A. Harvey, and Debmalya Panigrahi. A general framework for graph sparsification. *SIAM J. Comput.*, 48(4):1196–1223, 2019.
 - 16 Yu Gao, Jason Li, Danupon Nanongkai, Richard Peng, Thatchaphol Saranurak, and Sorrachai Yingchareonthawornchai. Deterministic graph cuts in subquadratic time: Sparse, balanced, and k -vertex. *arXiv preprint arXiv:1910.07950*, 2019.
 - 17 Shayan Oveis Gharan, Amin Saberi, and Mohit Singh. A randomized rounding approach to the traveling salesman problem. In Rafail Ostrovsky, editor, *IEEE 52nd Annual Symposium on Foundations of Computer Science, FOCS 2011, Palm Springs, CA, USA, October 22–25, 2011*, pages 550–559. IEEE Computer Society, 2011.
 - 18 Ralph E Gomory and Tien Chung Hu. Multi-terminal network flows. *Journal of the Society for Industrial and Applied Mathematics*, 9(4):551–570, 1961.
 - 19 Anna R. Karlin, Nathan Klein, and Shayan Oveis Gharan. An improved approximation algorithm for TSP in the half integral case. In Konstantin Makarychev, Yury Makarychev, Madhur Tulsiani, Gautam Kamath, and Julia Chuzhoy, editors, *Proceedings of the 52nd Annual ACM SIGACT Symposium on Theory of Computing, STOC 2020, Chicago, IL, USA, June 22–26, 2020*, pages 28–39. ACM, 2020.

- 20 Anna R. Karlin, Nathan Klein, and Shayan Oveis Gharan. A (slightly) improved approximation algorithm for metric TSP. In Samir Khuller and Virginia Vassilevska Williams, editors, *STOC '21: 53rd Annual ACM SIGACT Symposium on Theory of Computing, Virtual Event, Italy, June 21–25, 2021*, pages 32–45. ACM, 2021.
- 21 Jonathan A. Kelner, Yin Tat Lee, Lorenzo Orecchia, and Aaron Sidford. An almost-linear-time algorithm for approximate max flow in undirected graphs, and its multicommodity generalizations. In *Proceedings of the Twenty-Fifth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2014, Portland, Oregon, USA, January 5–7, 2014*, pages 217–226, 2014.
- 22 Rohit Khandekar, Satish Rao, and Umesh V. Vazirani. Graph partitioning using single commodity flows. *J. ACM*, 56(4):19:1–19:15, 2009.
- 23 Jason Li, Danupon Nanongkai, Debmalya Panigrahi, Thatchaphol Saranurak, and Sorrachai Yingchareonthawornchai. Vertex connectivity in poly-logarithmic max-flows. In Samir Khuller and Virginia Vassilevska Williams, editors, *STOC '21: 53rd Annual ACM SIGACT Symposium on Theory of Computing, Virtual Event, Italy, June 21–25, 2021*, pages 317–329. ACM, 2021.
- 24 Jason Li and Debmalya Panigrahi. Deterministic min-cut in poly-logarithmic max-flows. In Sandy Irani, editor, *61st IEEE Annual Symposium on Foundations of Computer Science, FOCS 2020, Durham, NC, USA, November 16–19, 2020*, pages 85–92. IEEE, 2020.
- 25 Jason Li and Debmalya Panigrahi. Approximate gomory-hu tree is faster than $n - 1$ max-flows. In Samir Khuller and Virginia Vassilevska Williams, editors, *STOC '21: 53rd Annual ACM SIGACT Symposium on Theory of Computing, Virtual Event, Italy, June 21–25, 2021*, pages 1738–1748. ACM, 2021.
- 26 Jason Li, Debmalya Panigrahi, and Thatchaphol Saranurak. A nearly optimal all-pairs min-cuts algorithm in simple graphs. In *62nd IEEE Annual Symposium on Foundations of Computer Science, FOCS 2021, Denver, CO, USA, February 7–10, 2022*, pages 1124–1134. IEEE, 2021.
- 27 Kurt Mehlhorn. A faster approximation algorithm for the Steiner problem in graphs. *Information Processing Letters*, 27(3):125–128, 1988.
- 28 Tobias Mömke and Ola Svensson. Approximating graphic TSP by matchings. In Rafail Ostrovsky, editor, *IEEE 52nd Annual Symposium on Foundations of Computer Science, FOCS 2011, Palm Springs, CA, USA, October 22–25, 2011*, pages 560–569. IEEE Computer Society, 2011.
- 29 Richard Peng. Approximate undirected maximum flows in $O(\text{mpolylog}(n))$ time. In *Proceedings of the Twenty-Seventh Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2016, Arlington, VA, USA, January 10–12, 2016*, pages 1862–1867, 2016.
- 30 Thatchaphol Saranurak and Di Wang. Expander decomposition and pruning: Faster, stronger, and simpler. In Timothy M. Chan, editor, *Proceedings of the Thirtieth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2019, San Diego, California, USA, January 6–9, 2019*, pages 2616–2635. SIAM, 2019.
- 31 Jonah Sherman. Nearly maximum flows in nearly linear time. In *54th Annual IEEE Symposium on Foundations of Computer Science, FOCS 2013, 26–29 October, 2013, Berkeley, CA, USA*, pages 263–269. IEEE Computer Society, 2013.
- 32 Daniel A. Spielman and Nikhil Srivastava. Graph sparsification by effective resistances. *SIAM J. Comput.*, 40(6):1913–1926, 2011.
- 33 Daniel A. Spielman and Shang-Hua Teng. Spectral sparsification of graphs. *SIAM J. Comput.*, 40(4):981–1025, 2011.
- 34 Vera Traub and Jens Vygen. Approaching $3/2$ for the s - t -path TSP. *J. ACM*, 66(2):14:1–14:17, 2019.

- 35 Vera Traub and Rico Zenklusen. Local search for weighted tree augmentation and steiner tree. In Joseph (Seffi) Naor and Niv Buchbinder, editors, *Proceedings of the 2022 ACM-SIAM Symposium on Discrete Algorithms, SODA 2022, Virtual Conference / Alexandria, VA, USA, January 9 – 12, 2022*, pages 3253–3272. SIAM, 2022.
- 36 Jan van den Brand, Yin Tat Lee, Yang P. Liu, Thatchaphol Saranurak, Aaron Sidford, Zhao Song, and Di Wang. Minimum cost flows, mdps, and ℓ_1 -regression in nearly linear time for dense instances. In Samir Khuller and Virginia Vassilevska Williams, editors, *STOC '21: 53rd Annual ACM SIGACT Symposium on Theory of Computing, Virtual Event, Italy, June 21–25, 2021*, pages 859–869. ACM, 2021.
- 37 Jan van den Brand, Yin Tat Lee, Danupon Nanongkai, Richard Peng, Thatchaphol Saranurak, Aaron Sidford, Zhao Song, and Di Wang. Bipartite matching in nearly-linear time on moderately dense graphs. In Sandy Irani, editor, *61st IEEE Annual Symposium on Foundations of Computer Science, FOCS 2020, Durham, NC, USA, November 16–19, 2020*, pages 919–930. IEEE, 2020.

2 Table of Contents

Executive Summary

Jason Li, Debmalya Panigrahi, Laura Sanita, and Thatchaphol Saranurak 76

Overview of Talks

Fast Algorithms via Dynamic-Oracle Matroids <i>Joakim Blikstad</i>	83
The girth problem and its variants in network design <i>Greg Bodwin</i>	83
Differentially Private Densest Subgraph <i>Michael Dinitz</i>	84
On Dynamic Graph Approximations: The case of j -Trees <i>Gramoz Goranci</i>	84
Approximation Algorithms for 2-Connectivity <i>Fabrizio Grandoni</i>	84
Polylogarithmic Universal Steiner Trees and Strong Sparse Partition Hierarchies <i>Ellis Hershkowitz</i>	85
All-Pairs Minimum Cuts in Almost-Linear Time <i>Jason Li</i>	85
Recent Advances on Maximum Flows <i>Yang P. Liu</i>	85
Fair Division of Indivisible Goods and Graph Algorithms <i>Kurt Mehlhorn</i>	86
Hopsets and Algorithmic Applications <i>Yasamin Nazari</i>	86
Algorithms for Coloring Tournaments <i>Alantha Newman</i>	86
Quotient sparsification for submodular functions <i>Kent Quanrud</i>	87
Using Isolating Mincuts for Fast Graph Algorithms: A tutorial <i>Thatchaphol Saranurak</i>	87
Decremental Bipartite Matching <i>Aaron Sidford</i>	87
Approximation Algorithms for Connectivity Augmentation Problems <i>Vera Traub</i>	88
Faster Deterministic Vertex Connectivity Algorithms <i>Sorrachai Yingchareonthawornchai</i>	88
Participants	89

3 Overview of Talks

3.1 Fast Algorithms via Dynamic-Oracle Matroids

Joakim Blikstad (KTH Royal Institute of Technology – Stockholm, SE)

License © Creative Commons BY 4.0 International license
© Joakim Blikstad

We initiate the study of matroid problems in a new oracle model called dynamic oracle. Our algorithms in this model lead to new bounds for some classic problems, and a “unified” algorithm whose performance matches previous results developed in various papers. We also show a lower bound that answers some open problems from a few decades ago. We show an algorithm with $\tilde{O}(n + r\sqrt{r})$ dynamic-rank-query and time complexities for the matroid union problem. This implies an improvement over the traditional rank-query complexity for matroid union. As an interesting special case, it is the first algorithm which, in sufficiently dense graphs, achieves nearly linear time $\tilde{O}(m + n\sqrt{n})$ for the problem of finding k disjoint spanning trees in a graph. We also show simple super-linear ($\Omega(n \log n)$) query lower bounds for matroid intersection in our dynamic-rank-oracle and the traditional independence-query models; the latter improves the previous $\log 2(3)n - o(n)$ bound.

3.2 The girth problem and its variants in network design

Greg Bodwin (University of Michigan – Ann Arbor, US)

License © Creative Commons BY 4.0 International license
© Greg Bodwin

The girth problem as a central open question in extremal combinatorics, which asks to determine the maximum possible number of edges in an n -node graph whose girth (shortest cycle length) is larger than k . In 1993, a seminal work of Althöfer, Das, Dobkin, Joseph, and Soares showed that determining the size/stretch tradeoff available to graph spanners is equivalent to settling the girth problem. This set in motion a line of research that seeks to understand sparse graph structures by analyzing their forbidden patterns, and using these patterns to invoke ideas from extremal combinatorics.

This talk will survey some successes of the method, including other objects that can also be reduced to the girth problem (distance oracles, fault-tolerant spanners), and related problems that capture other objects in network design, such as the weighted girth problem (light spanners), the bipartite girth problem (distance preservers, reachability preservers), the forbidden biclique problem (directed distance preservers), and the bridge girth problem (reachability preservers, flow-cut gaps). We will survey the common technical threads in these arguments, and overview the many open problems that remain.

3.3 Differentially Private Densest Subgraph

Michael Dinitz (Johns Hopkins University – Baltimore, US)

License  Creative Commons BY 4.0 International license
© Michael Dinitz

Joint work of Satyen Kale, Silvio Lattanzi, and Sergei Vassilvitskii
Main reference Michael Dinitz, Satyen Kale, Silvio Lattanzi, Sergei Vassilvitskii: “Improved Differentially Private Densest Subgraph: Local and Purely Additive”, CoRR, Vol. abs/2308.10316, 2023.
URL <https://doi.org/10.48550/ARXIV.2308.10316>

We study the Densest Subgraph problem under the additional constraint of differential privacy. In the LEDP (local edge differential privacy) model, introduced recently by Dhulipala et al. [FOCS 2022], we give an (ϵ, δ) -differentially private algorithm with no multiplicative loss: the loss is purely additive. This is in contrast to every previous private algorithm for densest subgraph (local or centralized), all of which incur some multiplicative loss as well as some additive loss. Moreover, our additive loss matches the best-known previous additive loss (in any version of differential privacy) when $1/\delta$ is at least polynomial in n , and in the centralized setting we can strengthen our result to provide better than the best-known additive loss. Additionally, we give a different algorithm that is ϵ -differentially private in the LEDP model which achieves a multiplicative ratio arbitrarily close to 2, along with an additional additive factor. This improves over the previous multiplicative 4-approximation in the LEDP model. Finally, we conclude with extensions of our techniques to both the node-weighted and the directed versions of the problem.

3.4 On Dynamic Graph Approximations: The case of j-Trees

Gramoz Goranci (Universität Wien, AT)

License  Creative Commons BY 4.0 International license
© Gramoz Goranci


Joint work of Gramoz Goranci, Li Chen, Monika Henzinger, Richard Peng, Thatchaphol Saranurak

Approximating graphs by j-trees is a powerful algorithmic paradigm that has proven effective in significantly speeding up cut-based optimization problems, approximate maximum flows, and exact minimum cost-flow computations.

In this talk, I will explain how to dynamically maintain j-trees and discuss some of the implications of this result.

3.5 Approximation Algorithms for 2-Connectivity

Fabrizio Grandoni (SUPSI – Lugano, CH)

License  Creative Commons BY 4.0 International license
© Fabrizio Grandoni

Given an undirected graph G , the 2-edge-connected spanning subgraph problem is to compute a subgraph S of G with the minimum possible number of edges which is 2-edge-connected, i.e., removing any edge from S leaves a connected graph (spanning all the nodes). The 2-vertex-connected spanning subgraph problem is defined similarly w.r.t. 2-vertex-connectivity. In this talk I will illustrate some recent progress on approximation algorithms for these two problems.

3.6 Polylogarithmic Universal Steiner Trees and Strong Sparse Partition Hierarchies

Ellis Hershkowitz (Brown University – Providence, US)

License © Creative Commons BY 4.0 International license
© Ellis Hershkowitz

Joint work of Ellis Hershkowitz, Costas Busch, Da Qi Chen, Arnold Filtser, Daniel Hathcock, Rajmohan Rajaraman

An alpha-approximate universal Steiner tree (UST) of a graph G is a spanning tree T such that, for any vertex terminal subset S , the minimal subtree of T connecting S is within an alpha factor of the cost of the cheapest Steiner tree in G connecting S . Alpha-approximate USTs immediately give alpha-approximations for well-studied variants of Steiner tree such as online or oblivious Steiner tree. Sub-linear-approximate USTs are known but neither the existence of nor poly-time algorithms for computing poly-logarithmic-approximate USTs were previously known.

In this talk, I will discuss the first construction of poly-logarithmic USTs. The result is based on new constructions of poly-logarithmic-quality graph hierarchies called strong sparse partitions which may be interesting in their own right. Roughly, strong sparse partitions provide deterministic guarantees on how often balls of particular radii are cut.

3.7 All-Pairs Minimum Cuts in Almost-Linear Time

Jason Li (University of California – Berkeley, US)

License © Creative Commons BY 4.0 International license
© Jason Li

Joint work of Jason Li, Amir Abboud, Robert Krauthgamer, Debmalya Panigrahi, Thatchaphol Saranurak, Ohad Trabelsi

Main reference Amir Abboud, Robert Krauthgamer, Jason Li, Debmalya Panigrahi, Thatchaphol Saranurak, Ohad Trabelsi: “Gomory-Hu Tree in Subcubic Time”, CoRR, Vol. abs/2111.04958, 2021.

URL <https://arxiv.org/abs/2111.04958>

We present recent progress on the problem of computing all-pairs minimum cuts, and more generally the Gomory-Hu tree of a graph. In particular, we obtain the first running time improvement since Gomory and Hu’s original algorithm in 1961, as well as a subsequent improvement to almost-linear time, resolving the complexity of this problem. We discuss important tools that paved the way for the discovery of the algorithm, most notably the isolating cuts problem and a reduction to single-source minimum cut.

3.8 Recent Advances on Maximum Flows

Yang P. Liu (Institute for Advanced Study – Princeton, US)

License © Creative Commons BY 4.0 International license
© Yang P. Liu

We discuss extensions of the recent almost-linear-time maximum flow and mincost flow algorithm to dynamic and deterministic settings. Joint work with Jan van den Brand, Li Chen, Rasmus Kyng, Richard Peng, Maximilian Probst Gutenberg, Sushant Sachdeva, and Aaron Sidford.

3.9 Fair Division of Indivisible Goods and Graph Algorithms

Kurt Mehlhorn (MPI für Informatik – Saarbrücken, DE)


License  Creative Commons BY 4.0 International license
© Kurt Mehlhorn

A set of indivisible goods is to be allocated to a group of agents, e.g, a car, a computer, a tooth-brush. Each agent has a valuation over sets of goods. There are only two restrictions on a valuation. The value of the empty set is zero and more is better. We want to allocate the goods in a fair manner. Three notions of fairness have emerged: envy-freeness, fair share, and maximum Nash-value. We discuss exact and approximate existence and complexity.

Some of the algorithms have a strong connection to matchings.

3.10 Hopsets and Algorithmic Applications

Yasamin Nazari (VU Amsterdam, NL)

License  Creative Commons BY 4.0 International license
© Yasamin Nazari

Given a weighted graph G , a hopset of hopbound β and stretch $(1 + \epsilon)$ is a set of edges such that for any pair of nodes u and v in G , there is a path in $G \cup H$ of at most β hops whose length is within a $(1 + \epsilon)$ factor of the distance between u and v in G . Hopsets have recently found many applications in fast distance computation in various computational models such as dynamic, parallel and distributed models. This talk gives an introduction to hopsets for undirected graphs and their algorithmic applications in these settings. We conclude with open problems on applications of directed hopsets.

3.11 Algorithms for Coloring Tournaments

Alantha Newman (Grenoble INP, FR)

License  Creative Commons BY 4.0 International license
© Alantha Newman

Joint work of Alantha Newman, Felix Klingelhoefer

A k -coloring of a tournament is a partition of its vertices into k acyclic sets. Deciding if a tournament is 2-colorable is NP-hard. A natural problem, akin to that of coloring a 3-colorable graph with few colors, is to color a 2-colorable tournament with few colors. This problem does not seem to have been addressed before, although it is a special case of coloring a 2-colorable 3-uniform hypergraph with few colors, which is a well-studied problem with super-constant lower bounds.

We present a new efficient decomposition lemma for tournaments, which we use to design polynomial-time algorithms to color various classes of tournaments with few colors, notably, to color a 2-colorable tournament with ten colors. We also use this lemma to prove equivalence between the problems of coloring 3-colorable tournaments and coloring 3-colorable graphs with constantly many colors. For the classes of tournaments considered, we complement our upper bounds with strengthened lower bounds, painting a comprehensive picture of the algorithmic and complexity aspects of coloring tournaments.

3.12 Quotient sparsification for submodular functions

Kent Quanrud (Purdue University – West Lafayette, US)

License © Creative Commons BY 4.0 International license
© Kent Quanrud

Main reference Kent Quanrud: “Quotient sparsification for submodular functions”, in Proc. of the 2024 Annual ACM-SIAM Symposium on Discrete Algorithms (SODA), Alexandria, VA, USA, pp. 5209–5248, 2024.

URL <https://doi.org/10.1137/1.9781611977912.187>

Graph sparsification has been an important topic with many structural and algorithmic consequences. Recently hypergraph sparsification has come to the fore and has seen exciting progress. In this paper we take a fresh perspective and show that they can be both derived as corollaries of a general theorem on sparsifying matroids and monotone submodular functions.

Quotients of matroids and monotone submodular functions generalize k -cuts in graphs and hypergraphs. We show that a weighted ground set of a monotone submodular function f can be sparsified while approximately preserving the weight of every quotient of f with high probability in randomized polynomial time.

This theorem conceptually unifies cut sparsifiers for undirected graphs [BK15] with other interesting applications. One basic application is to reduce the number of elements in a matroid while preserving the weight of every quotient of the matroid. For hypergraphs, the theorem gives an alternative approach to the hypergraph cut sparsifiers obtained recently in [CKN20], that also preserves all k -cuts. Another application is to reduce the number of points in a set system while preserving the weight of the union of every collection of sets. We also present algorithms that sparsify hypergraphs and set systems in nearly linear time, and sparsify matroids in nearly linear time and queries in the rank oracle model.

3.13 Using Isolating Mincuts for Fast Graph Algorithms: A tutorial

Thatchaphol Saranurak (University of Michigan – Ann Arbor, US)

License © Creative Commons BY 4.0 International license
© Thatchaphol Saranurak

The Isolating Mincuts algorithm is a new technique recently introduced by [Li and Panigrahi] and [Abboud Krauthgamer Trabelsi]. In the last three years, they found more than ten applications in fast graph algorithms. I will give a gentle tutorial on this technique.

3.14 Decremental Bipartite Matching

Aaron Sidford (Stanford University, US)

License © Creative Commons BY 4.0 International license
© Aaron Sidford

Joint work of Aaron Sidford, Arun Jambulapati, Yujia Jin, Kevin Tian

Main reference Arun Jambulapati, Yujia Jin, Aaron Sidford, Kevin Tian: “Regularized Box-Simplex Games and Dynamic Decremental Bipartite Matching”, CoRR, Vol. abs/2204.12721, 2022.


URL <https://doi.org/10.48550/ARXIV.2204.12721>

Maintaining an approximately maximum matching in a dynamic graph is a fundamental problem in data structures and algorithmic graph theory. In this talk I will discuss recent progress in the special case of decremental bipartite matching, where the only dynamic

updates are deleting edges from an initial bipartite graph. In particular, I will discuss how faster runtimes were obtained by reducing this dynamic problem to solving a sequence of natural convex optimization problems.

3.15 Approximation Algorithms for Connectivity Augmentation Problems

Vera Traub (Universität Bonn, DE)


License  Creative Commons BY 4.0 International license
© Vera Traub

Augmentation problems are a fundamental class of network design problems. They ask about the cheapest way to increase the (edge-)connectivity of a graph by adding edges among a given set of options. One of the most elementary and intensely studied augmentation problems is the (Weighted) Tree Augmentation Problem. Here, a spanning tree has to be augmented into a 2-edge-connected graph.

Classic techniques for network design yield 2-approximation algorithms for a wide class of augmentation problems. For the Unweighted Tree Augmentation Problem, better-than-2 approximations are known for more than 20 years. However, only recently the first better-than-2 approximations have been found for the more general Unweighted Connectivity Augmentation Problem and Weighted Tree Augmentation Problem. In this talk we will discuss these recent advances.

3.16 Faster Deterministic Vertex Connectivity Algorithms

Sorrachai Yingchareonthawornchai (University of California – Berkeley, US)

License  Creative Commons BY 4.0 International license
© Sorrachai Yingchareonthawornchai

Joint work of Sorrachai Yingchareonthawornchai, Yonggang Jiang, Chaitanya Nalam, Thatchaphol Saranurak

An n -vertex m -edge graph is k -vertex connected if it cannot be disconnected by deleting less than k vertices. After more than half a century of intensive research, the result by [Li et al. STOC'21] finally gave a randomized algorithm for checking k -connectivity in near-optimal $\widehat{O}(m)$ time where $\widehat{O}(\cdot)$ to hide an $n^{o(1)}$ factor.

Deterministic algorithms, unfortunately, have remained much slower even if we assume a linear-time max-flow algorithm: they either require at least $\Omega(mn)$ time [Even'75; Henzinger Rao and Gabow, FOCS'96; Gabow, FOCS'00] or assume that $k = o(\sqrt{\log n})$ [Saranurak and Yingchareonthawornchai, FOCS'22]. In this talk, I will describe a deterministic algorithm for checking k -vertex connectivity in time proportional to making $\min\{k^2, n\}$ max-flow calls, and, hence, in $\widehat{O}(m \min\{k^2, n\})$ time using the deterministic max-flow algorithm by [Brand et al. FOCS'23]. Our algorithm gives the first almost-linear-time bound for all k where $\sqrt{\log n} \leq k \leq n^{o(1)}$ and subsumes up to a sub-polynomial factor the long-standing state-of-the-art algorithm by [Even'75] which requires $O(n + k^2)$ max-flow calls. For large k , the algorithm runs in $\widehat{O}(mn)$ time, which improves over the state-of-the-art deterministic $\widehat{O}(mn^{1.5})$ -time algorithm [Gabow, FOCS'00]. Our key technique is based on Ramanujan expanders and derandomization of the kernelization technique of [Li et al. STOC'21] for which their kernel construction was randomized.

Participants

- Joakim Blikstad
KTH Royal Institute of
Technology – Stockholm, SE
- Greg Bodwin
University of Michigan –
Ann Arbor, US
- Parinya Chalermsook
Aalto University, FI
- Michael Dinitz
Johns Hopkins University –
Baltimore, US
- Gramoz Goranci
Universität Wien, AT
- Fabrizio Grandoni
SUPSI – Lugano, CH
- Anupam Gupta
Carnegie Mellon University –
Pittsburgh, US
- Ellis Hershkowitz
Brown University –
Providence, US
- Felix Hommelsheim
Universität Bremen, DE
- Alexandra Lassota
MPI für Informatik –
Saarbrücken, DE
- Jason Li
University of California –
Berkeley, US
- Yang P. Liu
Institute for Advanced Study –
Princeton, US
- Kurt Mehlhorn
MPI für Informatik –
Saarbrücken, DE
- Danupon Nanongkai
MPI für Informatik –
Saarbrücken, DE
- Yasamin Nazari
VU Amsterdam, NL
- Alantha Newman
Grenoble INP, FR
- Debmalya Panigrahi
Duke University – Durham, US
- Kent Quanrud
Purdue University –
West Lafayette, US
- Laura Sanita
Università Bocconi – Milan, IT
- Thatchaphol Saranurak
University of Michigan –
Ann Arbor, US
- Aaron Sidford
Stanford University, US
- Joachim Spoerhase
University of Sheffield, GB
- Vera Traub
Universität Bonn, DE
- László A. Végh
London School of Economics &
Political Science, GB
- Sorrachai
Yingchareonthawornchai
University of California –
Berkeley, US



Network Attack Detection and Defense – AI-Powered Threats and Responses

Sven Dietrich^{*1}, Frank Kargl^{*2}, Hartmut König^{*3}, Pavel Laskov^{*4},
and Artur Hermann^{†5}

1 City University of New York, US. spock@ieee.org

2 Universität Ulm, DE. frank.kargl@uni-ulm.de

3 ZITiS München, DE. hartmut.koenig@b-tu.de

4 Universität Liechtenstein, LI. pavel.laskov@uni.li

5 Universität Ulm, DE. artur.hermann@uni-ulm.de

Abstract

This report documents the program and the findings of Dagstuhl Seminar 23431 “Network Attack Detection and Defense – AI-Powered Threats and Responses”. With the emergence of artificial intelligence (AI), attack detection and defense are taking on a new level of quality. Artificial intelligence will promote further automation of attacks. There are already examples of this, such as the Deep Locker malware. It is expected that we will soon face a situation in which malware and attacks will become more and more automated, intelligent, and AI-powered. Consequently, today’s threat response systems will become more and more inadequate, especially when they rely on manual intervention of security experts and analysts. The main objective of the seminar was to assess the state of the art and potentials that AI advances create for both attackers and defenders. The seminar continued the series of Dagstuhl events “Network Attack Detection and Defense” held in 2008, 2012, 2014, and 2016. The objectives of the seminar were threefold, namely (1) to investigate various scenarios of AI-based malware and attacks, (2) to debate trust in AI and modeling of threats against AI, and (3) to propose methods and strategies for AI-powered network defenses. At the seminar, which brought together participants from academia and industry, we stated that recent advances in artificial intelligence have opened up new possibilities for each of these directions. In general, more and more researchers in networking and security look at AI-based methods which made this a timely event to assess and categorize the state of the art as well as work towards a roadmap for future research. The outcome of the discussions and the proposed research directions are presented in this report.

Seminar October 22–27, 2023 – <https://www.dagstuhl.de/23431>

2012 ACM Subject Classification Networks → Network security; Security and privacy → Intrusion/anomaly detection and malware mitigation; Security and privacy → Network security; Security and privacy → Systems security

Keywords and phrases artificial intelligence, cybersecurity, intrusion detection, machine learning

Digital Object Identifier 10.4230/DagRep.13.10.90

* Editor / Organizer

† Editorial Assistant / Collector



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

Network Attack Detection and Defense – AI-Powered Threats and Responses, *Dagstuhl Reports*, Vol. 13, Issue 10, pp. 90–129

Editors: Sven Dietrich, Artur Hermann, Frank Kargl, Hartmut König, and Pavel Laskov



DAGSTUHL
REPORTS Dagstuhl Reports

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

1 Seminar Motivation and Summary

Sven Dietrich

Artur Hermann

Frank Kargl

Hartmut König

Pavel Laskov

License © Creative Commons BY 4.0 International license
© Sven Dietrich, Artur Hermann, Frank Kargl, Hartmut König, and Pavel Laskov

Computer networks and the services they provide have become indispensable tools these days. Consequently, they are also a popular target for attacks that are constantly increasing in complexity and sophistication. Although there are a variety of effective systems to counter such attacks, like firewalls or intrusion detection systems (IDSs), the immense diversity and number of threats make it difficult for system administrators to keep pace with the alerts triggered and respond within adequate time limits.

This problem will intensify in the future. There are signs that attacks will become more and more automated, as, for instance, indicated by the 2016 DARPA Cyber Grand Challenge in which automation of attacks was a main focus and the basic feasibility was demonstrated. Another indication of a higher degree of automation is advanced malware where Large-Language-Models (LLMs) start to get applied to craft highly sophisticated phishing emails. Experts already foresee that more and more AI mechanisms will find their way into such malware. This leads to the conclusion that we will soon face a situation in which malware and attacks will become more and more automated, intelligent, and AI-powered.

As a consequence, today's threat response systems will become more and more inadequate, especially where they rely on manual intervention of security experts and analysts. Hence, the deployment of automation and AI is the only way to attain and retain a strategic advantage in the arm's race between the attack and the defense. Usage of AI mechanisms is already the case in some security mechanisms like anomaly-detecting IDSs or virus scanners. But one could easily imagine substantially higher degrees of AI-based automation in system defense. However, automated defense may also be a double edged sword as it could be misused by attackers to trigger counterproductive responses.

In this Dagstuhl Seminar, we together with all the participants therefore tried to assess the state of the art and potentials that AI advances create for both attackers and defenders because we believe it is crucial to consider both sides when discussing the relation between AI and security.

In particular, the seminar pursued the following objectives:

1. Investigate various attack scenarios and attacker models of AI-based malware and attacks,
2. Map the space of AI-based security countermeasures going beyond the usual anomaly-based intrusion detection systems,
3. Discuss where else AI-based methods are or could be employed, and
4. Discuss how to estimate and predict the impact of countermeasures and possible side effects.

To provide initial material for such discussions, we had three keynotes by distinguished speakers. Pavel Laskov proposed "Three Faces of AI in Cybersecurity," providing a thorough account of how AI could be used in defense, for offensive purpose, and how AI itself can be an attack target. Konrad Rieck took a deep dive into the first aspect in his keynote "Bumpy Road of AI-based Attack Detection." Finally, Robin Sommer completed the picture

by looking “Beyond Detection: Revisiting AI For Effective Network Security Monitoring.” Those presentations were complemented by a number of short lightning talks given by our participants to introduce the audience to various current research.

A significant share of the seminar’s time was spent in working groups, with participants discussing individual aspects of interest. The topics for those working groups were partly solicited before the seminar and then finally determined on the first day. Specifically, the topics were:

1. Assessment of AI-Based Attacks in Cybersecurity,
2. Security of Large Language Models,
3. Trust in AI and Modeling of Threats against AI in Network Defense, and
4. AI-Powered Network Defenses

The working groups report on their individual results below. In order to bring all these findings together and distill outcomes and an outlook into what could be next steps, we used the format of a World Café where in the afternoon of day 4, people split into small groups to provide their input on five pre-defined questions. As groups were shuffled randomly after every 20 minutes, everyone joined each World Café table and discussed each of the questions. The outcomes then formed the basis for our wrap-up session on Friday morning.

The seminar was originally proposed and prepared together with Marc C. Dacier from KAUST who couldn’t attend the seminar at the last minute. We owe him many ideas and contributions during the preparation phase. Pavel Laskov was so kind as to fill the empty slot on short notice.

2 Table of Contents

Seminar Motivation and Summary

Sven Dietrich, Artur Hermann, Frank Kargl, Hartmut König, and Pavel Laskov . . . 91

Overview of Keynotes

Three Faces of AI in Cybersecurity
Pavel Laskov 94

The Bumpy Road of AI-based Attack Detection
Konrad Rieck 95

Beyond Detection: Revisiting AI For Effective Network Security Monitoring
Robin Sommer 96

Overview of Lightning Talks

Robust, Explainable, and Privacy-Respecting Sybil Attack Defense
Christian Bungartz 96

The SuperviZ project – towards enhanced Security Orchestration, Automation and Response
Hervé Debar 96

A Strategy to Evaluate Test Time Evasion Attack Feasibility
Stephan Kleber 97

Privacy-preserving Artificial Intelligence for Telecommunications
Nicolas Kourtellis 98

Comparison of a ML-based approach with Snort in an IoT environment
Max Schrötter and Bettina Schnor 98

Working groups

Assessment of AI-Based Attacks in Cybersecurity
Ilies Benhabbour, Daniel Fraunholz, Jan Kohlrausch, Hartmut König, Chethan Krishnamurthy Ramanaik, Michael Meier, Simin Nadjm-Tehrani, Andriy Panchenko, and Konrad Rieck 99

Security of Large Language Models
Hervé Debar, Sven Dietrich, Pavel Laskov, Emil C. Lupu, and Eirini Ntoutsis . . . 103

Trust in AI and Modeling of Threats against AI in Network Defense
Stephan Kleber, Christian Bungartz, Artur Hermann, Peter Herrmann, Marko Jahnke, Frank Kargl, Andreas Mitschele-Thiel, Delphine Reinhardt, and Jessica Steinberger 112

AI-Powered Network Defenses
Vera Rimmer, Sebastian Böhm, Georg Carle, Marco Caselli, Nicolas Kourtellis, Bettina Schnor, Thomas Schreck, Max Schrötter, and Robin Sommer 120

World Café and Outlook

In which of these fields is it most important to make research progress and why: “Security for AI”, “AI-based attacks”, or “AI for Security”? 127


What is your one key take-away from the seminar? 128

Participants 129

3 Overview of Keynotes

3.1 Three Faces of AI in Cybersecurity

Pavel Laskov (Universität Liechtenstein, LI)

License  Creative Commons BY 4.0 International license
© Pavel Laskov

Artificial Intelligence (AI) has a stronger tradition in cybersecurity than one might think. Despite their seemingly contrasting scientific and methodical traits – AI is all about probabilistic events and assertions, whereas a (practical) security mindset is mostly concerned with apparent and deterministic evidence of systems being broken into – the task of detecting systems being broken into is inherently connected with observing and making sense of huge volumes of diverse bits and pieces of digital evidence commonly understood as “data.” This is something that AI, albeit not originally born as an empirical science, has manifested itself as an omnipotent instrument for.

The relation between AI and cybersecurity is not just profound, but multi-faceted as well. Each of its “faces” has a different history and a different level of maturity. The oldest and most obvious role of AI in cybersecurity is to build models for detection of various kinds of attacks. The sole notion of “intrusion detection” has been implicitly defined by Denning as an AI problem, even though neither did she use this term in her seminal work [1], nor did this term have the same meaning at that time as we understand it today. The tremendous capability of AI to facilitate and speed up detection of various kinds of threats is now widely acknowledged, both in the academia and the industry. Despite a large number of still unresolved problems, e.g., development of benchmark datasets, concept drift, explainability, and many others, AI is perhaps the only reason why modern defenses are still able to keep up with the staggering increase in professionalization of the “attack industry.” Furthermore, as recently pointed out by Apruzzese et al. [2], substantial merit can be gained by deployment of AI for a number of other cybersecurity tasks beyond threat detection, e.g., alert management, risk assessment and cyber threat intelligence.

As any other technology deployed for security, AI must be scrutinized for its insecurity. This axiom of security research has triggered research in security of AI, pioneered by at al. [3] and Biggio et al. [4]. This line of research can be seen as the second “face” of AI and security. Its importance clearly transcends the field of information security. While early attacks against AI systems were somewhat related to the conventional triad of security objectives – confidentiality, integrity, and availability, – the recent work has led to a discovery of various “AI endemic” attacks such as model stealing, model backdoors, attribute inference, as well as attacks against explainability. Despite the fact that several thousand papers have been hitherto published on security of AI, many important questions are still wide open, especially regarding to defenses as well as potential economic motivation behind the attacks.

A third dimension along which the relationship between AI and security is rapidly developing is “offensive AI,” i.e., abuse of AI for nefarious goals. Examples of such abuse have been reported in practice as well as in the scientific literature for several years. The recent review of AI-powered threats in the organizational context has demonstrated that virtually all stages of the conventional attack “kill chain” can be facilitated by AI [5]. While it is still largely unclear to what extent AI is currently used to assist conventional security exploitation, it becomes increasingly apparent that many AI tools can be used in a dual way. Ethical discussions of these issues are likely to emerge.

References

- 1 An intrusion-detection model. *IEEE Transactions on software engineering*, (2):222–232, 1987.
- 2 Giovanni Apruzzese, Pavel Laskov, Edgardo Montes de Oca, Wissam Mallouli, Luis Brdalo Rapa, Athanasios Vasileios Grammatopoulos, and Fabio Di Franco. The role of machine learning in cybersecurity. *Digital Threats: Research and Practice*, 4(1):1–38, 2023.
- 3 Marco Barreno, Blaine Nelson, Anthony D Joseph, and J Doug Tygar. The security of machine learning. *Machine Learning*, 81:121–148, 2010.
- 4 Battista Biggio, Blaine Nelson, and Pavel Laskov. Poisoning attacks against support vector machines. In *Proceedings of the 29th International Conference on Machine Learning, ICML 2012*, pages 1807–1814, 2012.
- 5 Yisroel Mirsky, Ambra Demontis, Jaidip Kotak, Ram Shankar, Deng Gelei, Liu Yang, Xiangyu Zhang, Maura Pintor, Wenke Lee, Yuval Elovici, et al. The threat of offensive ai to organizations. *Computers & Security*, 124:103006, 2023.

3.2 The Bumpy Road of AI-based Attack Detection

Konrad Rieck (TU Berlin, DE)

License © Creative Commons BY 4.0 International license
© Konrad Rieck


This talk examines the development of AI-based attack detection, both in its historical context and its future prospects. It provides an overview of the evolution of intrusion detection and pinpoints promising opportunities for recent AI techniques. The talk opens with a focus on classical learning-based detection approaches, which have developed over time but also repeatedly failed in practice due different pitfalls in their design and evaluations.

To remedy this situation, the talk then highlights the role of explainable AI (XAI), which makes the decision process of learning models transparent. While XAI represents a significant advance in building trust, it also encounters a number of challenges in the context of security, such as inconsistency and infidelity. These issues are discussed in detail and provide insights into how XAI can be employed without neglecting its limitations. In addition, the talk introduces toy examples to show how generative AI can be used to create detection signatures for attacks. These examples are used to demonstrate both the strengths of generative AI and its notable weaknesses such as hallucinations and lack of reasoning ability.

Overall, the talk aims to provide a balanced perspective on the current state and future direction of AI-assisted attack detection. By critically analyzing the hurdles on the road to success and exploring potential solutions, the talk hopefully points to possible paths for future research.

3.3 Beyond Detection: Revisiting AI For Effective Network Security Monitoring

Robin Sommer (Corelight – Planegg, DE)

License  Creative Commons BY 4.0 International license
© Robin Sommer

While AI has been proposed for finding novel network attacks many times, such approaches have not found much traction in operational deployments. In this talk we first revisit some challenges with classic intrusion detection. We then look at “threat hunting,” a modern twist on finding malicious activity that focuses on the human analyst driving the process, and we examine the potential of AI to support threat hunting workflows.

4 Overview of Lightning Talks

4.1 Robust, Explainable, and Privacy-Respecting Sybil Attack Defense

Christian Bungartz (Universität Bonn, DE)

License  Creative Commons BY 4.0 International license
© Christian Bungartz

Joint work of Christian Bungartz, Felix Boes, Michael Meier

Sybil attacks target decentralized networks and exploit trust relationships between peers. A network type of special interest are online social networks (OSN). Existing defense mechanisms often hinge on domain-specific metadata, potentially compromising user privacy and limiting applicability. A solution is a detection approach utilizing the global topological structure of the underlying network. However, the main assumption of a fast-mixing subgraph of honest peers often is not aligned with the reality of OSN structures. To tackle these issues, this work outlines five open problems and proposes an approach leveraging local, structural information. This privacy-friendly method demonstrates promising results in Sybil detection, offering a critical step towards safeguarding the trust and integrity of OSNs.

4.2 The SuperviZ project – towards enhanced Security Orchestration, Automation and Response

Hervé Debar (Télécom SudParis, FR)

License  Creative Commons BY 4.0 International license
© Hervé Debar

Joint work of Hervé Debar, Ludovic Mé

The SuperviZ project is part of the “system security” axis of the PEPR cybersecurity program. It addresses the field of “system, software and network security.” More precisely, it targets the detection, response and remediation to computer attacks, subjects grouped under the name of “security supervision.”

The digitization of all infrastructures makes it almost impossible today to secure all systems a priori, as it is too complex and too expensive. Supervision seeks to reinforce preventive security mechanisms and to compensate for their inadequacies.

Supervision is fundamental in the general context of enterprise systems and networks, and is just as important for the security of cyber-physical systems. Indeed, with “objects” that should eventually be all, or almost all, connected, the attack surface increases significantly. This makes security even more difficult to implement. The increase in the number of components to be monitored, as well as the growing heterogeneity of the capacities of these objects in terms of communication, storage and calculation, makes security supervision more complex.

In this context, we address challenges related to (1) the increase in the number and diversity of objects to be supervised (which requires the development and adaptation of new detection mechanisms for heterogeneous environments, with false positive and negative rates that have not been achieved to date), (2) the complexity of systems interconnected to form large critical infrastructures on a European scale (which requires new detection and supervision models that take into account the criticality and cyber-physical nature of these systems), (3) the existence of increasingly complex and silent targeted attacks (which requires an observation of the global threat landscape, a capability model of the attackers and a significant improvement of the detection and reaction time), and (4) the treatment of massive attacks which rapidly affect a significant number of victims (in order to limit the damage suffered by these victims).

Faced with these challenges, it is necessary to significantly improve the efficiency of the detection-reaction chain (response and remediation). The main objective of the project is therefore to provide new solutions and to advance the current scientific state of the art.

These contributions will come from almost all the national research forces in the field, which will be strengthened by this project and will see their links tightened, which is also an objective. Moreover, in coherence with the objectives of the PEPR, we also aim to prepare the transfer of our results to the national industrial community. To this end, the scientific work will lead to prototypes and demonstrators that will be deployed on platforms built within the project. These platforms will be accessible to industrial partners.

4.3 A Strategy to Evaluate Test Time Evasion Attack Feasibility

Stephan Kleber (Universität Ulm, DE)

License © Creative Commons BY 4.0 International license
© Stephan Kleber

Joint work of Stephan Kleber, Patrick Wachter

Main reference Stephan Kleber, Patrick Wachter: “A Strategy to Evaluate Test Time Evasion Attack Feasibility”, *Datenschutz und Datensicherheit*, Vol. 47(8), pp. 478–482, 2023.

URL <http://dx.doi.org/10.1007/S11623-023-1802-0>

New attacks against Computer Vision systems and other perceptive machine learning approaches are currently published in high frequency. Often the assumptions or limitations of these works are so strict that the attacks seem to have no practical relevance. On the other hand, recent reports show the effectiveness of attacks against cyber-physical systems (CPS). In particular, attacks on automotive systems demonstrate the safety-impact in real-world scenarios. We discuss the practical relevance of security threats for machine learning approaches in automotive use cases and we propose a strategy to evaluate the feasibility of such threats. This includes a method to potentially discover existing vulnerabilities and rate their exploitability in the use case.

4.4 Privacy-preserving Artificial Intelligence for Telecommunications

Nicolas Kourtellis (Telefónica Research – Barcelona, ES)

License © Creative Commons BY 4.0 International license
© Nicolas Kourtellis

- Main reference** Diego Perino, Kleomenis Katevas, Andra Lutu, Eduard Marin, Nicolas Kourtellis: “Privacy-preserving AI for future networks”, *Commun. ACM*, Vol. 65(4), pp. 52–53, 2022.
URL <http://dx.doi.org/10.1145/3512343>
- Main reference** Kleomenis Katevas, Diego Perino, Nicolas Kourtellis: “FLaaS – enabling practical federated learning on mobile environments”, in *Proc. of the 20th Annual International Conference on Mobile Systems, Applications and Services, MobiSys ’22*, p. 605–606, Association for Computing Machinery, 2022.
URL <http://dx.doi.org/10.1145/3498361.3539693>
- Main reference** Varun Chandrasekaran, Suman Banerjee, Diego Perino, Nicolas Kourtellis: “Hierarchical Federated Learning with Privacy”, *CoRR*, abs/2206.05209 2022.
URL <https://doi.org/10.48550/arXiv.2206.05209>
- Main reference** Fan Mo, Hamed Haddadi, Kleomenis Katevas, Eduard Marin, Diego Perino, Nicolas Kourtellis: “PPFL: privacy-preserving federated learning with trusted execution environments”, in *Proc. of the 19th Annual International Conference on Mobile Systems, Applications, and Services, MobiSys ’21*, p. 94–108, Association for Computing Machinery, 2021.
URL <http://dx.doi.org/10.1145/3458864.3466628>
- Main reference** Nicolas Kourtellis, Kleomenis Katevas, Diego Perino: “FLaaS: Federated Learning as a Service”, in *Proc. of the 1st Workshop on Distributed Machine Learning, DistributedML’20*, p. 7–13, Association for Computing Machinery, 2020.
URL <http://dx.doi.org/10.1145/3426745.3431337>

Telco networks and systems are highly complex, distributed ecosystems composed of very diverse sub-environments. Traditional solutions for network management (e.g., for provisioning, data management, etc.) are reaching their limits within such complex ecosystems, and with the arrival of faster, more demanding 5/6G networks. We require novel solutions that provide 1) effective resource management, while guaranteeing 2) strict service requirement completion and 3) absolute preservation of user privacy. Towards that goal, within Telefonica, we investigate building AI models using novel, state-of-art, distributed ML paradigms such as Federated Learning (FL), to unlock the potential of Big Data produced and processed at the source: the user devices. Using FL, and coupled with more advanced hardware and software techniques (e.g, Trusted Execution Environments, Differential Privacy, etc.), we can mine the user data locally, without risking their exposure to AI model attackers. Furthermore, by tapping on the power of Edge Computing, we can potentially scale AI model computation to millions of devices. To this end, we are prototyping a novel, FL-as-a-Service (FLaaS) platform, that will enable third-party companies to build joint ML models that solve common problems, in a cross-silo and cross-device fashion, while still protecting user privacy.

4.5 Comparison of a ML-based approach with Snort in an IoT environment

Max Schrötter (Universität Potsdam, DE) and Bettina Schnor (Universität Potsdam, DE)

License © Creative Commons BY 4.0 International license
© Max Schrötter and Bettina Schnor

Several papers presenting ML-based approaches for IoT Intrusion Detection Systems have been published over the last years. Our survey of 20 papers showed that the results of new models are not compared against a proper baseline. The authors compare their approaches against similar models, but do not show the benefit over a signature based IDS. We picked one paper and the result of the replicated research study showed several systematic problems with the used datasets and evaluation methods. The IoT IDS datasets are mostly synthetic and capturing not the real world variability and complexity of network traffic. With that, a

signature based IDS with a minimal setup was able to outperform the tested model. While testing the replicated neural network on a new dataset recorded in the same environment with the same attacks using the same tools showed that the accuracy of the neural network dropped from 99% to 54%. Furthermore, the claimed advantage of being able to detect zero-day attacks is not verified in the surveyed papers, and could also not be seen in our experiments.

5 Working groups

5.1 Assessment of AI-Based Attacks in Cybersecurity

Ilies Benhabbour (KAUST – Thuwal, SA), Daniel Fraunholz (ZITiS München, DE), Jan Kohlrausch (DFN-CERT Services GmbH, DE), Hartmut König (ZITiS München, DE), Chethan Krishnamurthy Ramanaik (Universität der Bundeswehr München, DE), Michael Meier (Universität Bonn, DE), Simin Nadjm-Tehrani (Linköping University, SE), Andriy Panchenko (BTU Cottbus, DE), Konrad Rieck (TU Berlin, DE)

License © Creative Commons BY 4.0 International license
© Ilies Benhabbour, Daniel Fraunholz, Jan Kohlrausch, Hartmut König, Chethan Krishnamurthy Ramanaik, Michael Meier, Simin Nadjm-Tehrani, Andriy Panchenko, and Konrad Rieck

5.1.1 Introduction

Recent advances in artificial intelligence (AI) have ushered in a transformative era in the cybersecurity landscape. The integration of AI technologies introduces a novel dimension to cyber threats, and this working group has been dedicated to delving into this evolving domain. Our collective effort has revolved around assessing and categorizing these emergent AI-driven threats to inform future defences in their presence. The analysis was carried out with a risk assessment mindset when discussing alternative uncertain developments.

5.1.2 Methodology

Our methodology employed a structured approach, encompassing the identification and classification of various properties and capabilities associated with AI-based attacks. This categorization was designed to shed light on the multifaceted aspects of AI-driven offenses, including the augmentation of existing attack vectors and the emergence of entirely new security challenges. Beyond categorization, our approach included a comparative risk assessment that evaluated AI-enhanced threats alongside traditional cyber threats, providing valuable insights into the potential impact of AI on the cybersecurity landscape. By comparing AI-driven risks with their conventional counterparts, we aimed at identifying areas where AI capabilities could have a significant impact on the effectiveness of attacks, through boosting their potential reach and damage.

5.1.3 Scope

This investigation primarily focuses on network-based threats, specifically excluding areas related to information security such as deepfakes or fake news generation. However, our scope does include the examination of potential network-based attacks driven by AI, including social engineering tactics. We recognize the diverse landscape of AI technologies, extending our consideration beyond deep learning. In summary, our investigation concentrates on:

1. Network attacks, excluding aspects related to information security (e.g., fake news generation)
2. Exploration of social engineering techniques in network attacks, encompassing their use for initiating intrusions and other network-based exploits.
3. Encompassing various AI technologies beyond deep learning.
4. Focusing on the use of AI for performing attacks, rather than attacks on AI itself.

5.1.4 Taxonomy of AI-Based Attacks

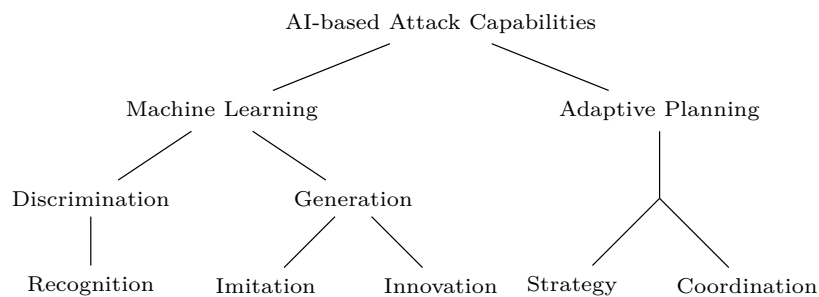
We have developed a taxonomy to outline the dimensions within which AI-based attacks can be categorized. This taxonomy provides a framework for understanding how AI can amplify cyber threats. In our initial categorization of AI-based attacks, we have considered various dimensions, which are summarized in Table 1.

■ **Table 1** AI-Based attack dimensions. Categorizes attacks by capabilities, type, target, evidence, mode, and intent.

Category	Description
Capabilities	List of AI-based attack capabilities (Recognition, Imitation, Innovation, Strategy, Coordination).
Type	Distinguishing between new AI-based attacks and enhancements of existing methods.
Target	Categorizing attacks based on their targets: virtual, physical, or human.
Evidence	Examining the presence of supporting evidence to differentiate between credible statements and unsubstantiated claims about the impact of AI-based attacks.
Mode	Categorizing attacks as offline or online in terms of execution.
Intent	Distinguishing between the ethical use of AI by law enforcement and its potential for malicious AI-based attacks.

5.1.5 Analysis of AI Capabilities

Our taxonomy further narrows down the capabilities dimension, as depicted in Figure 1. In this figure, the taxonomy of AI capabilities can be categorized into two main branches: sub-symbolic (representing data-driven processing e.g. neural network based) and symbolic (representing explicit knowledge and rule-based processing).



■ **Figure 1** Categorisation of AI-based attack capabilities.

The figure further decomposes the capabilities associated with AI-based attacks, delving into their nuances and potential implications. This forms the basis for our assessment of which of these capabilities have the potential to be game changers in the field of cybersecurity.

■ **Table 2** AI-Based Attack Capabilities Analysis.

Capability	Description
Machine Learning	
Discrimination	
Recognition	Involves the recognition of patterns, possibly for malicious purposes.
Generation	
Imitation	Replicating existing hacks/assets.
Innovation	Refers to the development of new attack methods using AI.
Adaptive Planning	
Strategy	Involves planning for attacks.
Coordination	Involves coordinating attacks for maximum impact.

Table 2 describes the interpretation of each of the refined capabilities. Next, we compare AI-driven threats to conventional cybersecurity challenges, assessing their operational complexity, success potential, and transformative impact. This evaluation offers insights into evolving AI-driven threats in cybersecurity, guiding research for attack and defense strategies.

5.1.6 Risk Assessment

Before discussing case study examples, we introduce Table 3, which showcases the risk assessment matrix for AI-based cyber attacks compared to more traditional attacks:

■ **Table 3** AI-based cyber attack risk assessment matrix.

Factor	Sub-Factor	Description
Cost	Time	The time required to develop and execute the attack.
	Resources	The resources, such as computing power, required for the attack.
Likelihood of Success	–	The probability that the attack will succeed.
	Complexity	–
Impact	Breadth	The extent of the attack’s reach and effect on systems and networks.
	Depth	The severity or level of penetration of the attack.

We evaluate each factor within the risk assessment matrix for a given use case, representing an example of an AI-based attack capability, by assigning scores such as *Low*, *High*, *Limited*, or *Uncertain*.

5.1.7 Use Cases

Building upon the risk assessment matrix, a selection of real-world cyber attacks are examined to evaluate how AI might enhance or redefine attack capabilities. These example use cases provide practical insights into the application of AI in cyber offenses. Below is a list of some examples with the overall results summarized in Table 4:

1. *Targeted Malware with Facial Recognition*:
 - Complexity: Decreases due to user-friendly tools.
 - Success: Increases accuracy in victim identification.
 - Impact: Neutralized to some extent by security measures.

■ **Table 4** This table assesses different cyber attack types utilizing AI, examining complexity, success rates, impact depth, impact breadth, and overall verdict. Cost considerations are omitted, as AI typically reduces time and increases required resources (e.g., GPUs) in these scenarios.

Attack Type	Complexity	Success Rate	Impact Depth	Impact Breadth	Verdict
Targeted Malware with Facial Recognition	Low	High	Limited	Limited	Less complex, higher success
Creation of Fake Identities	Low	High	High	High	Easier with deepfakes, higher success
Deepfake Impersonation for Social Engineering	Low	High	Uncertain	High	Potentially high impact
AI-Generated Malicious Payload (Application Layer)	Low	Low	Uncertain	Uncertain	Unclear
AI-Assisted Vulnerability Identification	Low	Limited	High	Limited	Helpful but not always accurate
AI-Generated Exploit Code	Low	Low	High	High	Can be better than a human expert
AI-Driven Attack Strategy Selection	Low	Low	Low	High	Helps scale, not necessarily better
AI-Powered Command and Control Coordination	Low	Low	Low	High	Helps scale, not necessarily better

2. *Creation of Fake Identities:*

- Complexity: Easier with deepfakes and available software.
- Success: Increasing difficulty in distinguishing fake personas.
- Impact: Potential for significant harm.

3. *Deepfake Impersonation for Social Engineering:*

- Complexity: Decreases with AI, especially for first instance.
- Success: High success in impersonating legitimate users.
- Impact: Depth uncertain, breadth high.

5.1.8 Conclusion

This report raises the question of whether AI represents a paradigm shift in cyber attacks or is merely a trend, an evolutionary enhancement. While this remains an area of active debate, our findings suggest the need for ongoing vigilance in cybersecurity. We recommend further research to delineate the properties of AI-based attacks clearly and to evaluate whether AI is merely automating tasks or introducing fundamentally new attack vectors [1]. In this context, the development of a position paper tentatively called “Demystifying AI-Based Attacks” is suggested as one outcome of this Dagstuhl Seminar towards advancing the understanding of AI’s implications cyber threats.

References

- 1 Xutan Peng, Yipeng Zhang, Jingfeng Yang, and Mark Stevenson. On the vulnerabilities of text-to-sql models. In *2023 IEEE 34th International Symposium on Software Reliability Engineering (ISSRE)*, pages 1–12. IEEE, 2023.

5.2 Security of Large Language Models

Hervé Debar (Télécom SudParis, FR), Sven Dietrich (City University of New York, US), Pavel Laskov (Universität Liechtenstein, LI), Emil C. Lupu (Imperial College London, GB), and Eirini Ntoutsis (Universität der Bundeswehr München, DE)

License  Creative Commons BY 4.0 International license
© Hervé Debar, Sven Dietrich, Pavel Laskov, Emil C. Lupu, and Eirini Ntoutsis

Large language models (LLMs) have achieved record adoption in a short period of time across many different sectors including high importance areas such as education [4] and healthcare [18]. LLMs are commonly used for text generation, but also widely used to assist with code generation [3], and even analysis of security information, as Microsoft Security Copilot demonstrates [14]. Traditional Machine Learning (ML) models are vulnerable to adversarial attacks [9]. So the concerns on the potential security implications of such wide scale adoption of LLMs have led to the creation of this working group. During the seminar, the working group discussions focused on the *vulnerability of LLMs to adversarial attacks*, rather than the use of LLMs for attacking other computing systems, e.g. generating malware or attacks. Although we note the potential threat represented by the latter, the role of the LLMs in such uses is mostly as an accelerator for development, similar to what it is in benign use. To make the analysis more specific, the working group employed ChatGPT as a concrete example of an LLM and addressed the following points, which also form the structure of this report:

- i) How do LLMs differ in vulnerabilities from traditional ML models?
- ii) What are the attack objectives in LLMs?
- iii) How complex it is to assess the risks posed by the vulnerabilities of LLMs?
- iv) What is the supply chain in LLMs, how data flow in and out of systems and what are the security implications?

We conclude with an overview of open challenges and outlook.

5.2.1 What is specific to LLMs?

Adversarial Machine Learning, is an area of study concerned with the vulnerabilities and robustness of ML models to adversarial attacks. Although, the first vulnerabilities were identified a number of years ago, e.g., [9], the contributions to this area have increased exponentially in recent years and entire conferences such as IEEE SaTML are devoted to this topic as well as regular sessions and many papers in both security and ML conferences. In light of this, discussions have focused on the aspects in which LLMs differ from adversarial aspects in traditional ML.

While traditional adversarial attacks focus mainly on classification tasks [6], aiming to manipulate the input and deceive the model into incorrect predictions, LLMs are general-purpose large language models designed to understand and generate human-like text across a wide range of tasks. Moreover, LLMs are subject to what is known as *hallucinations* [15, 12], where the answers provided by the LLM are inconsistent with real-world facts or user input. While these hallucinations usually do not have malicious cause or intent, they do raise the question of the trustworthiness of these LLMs, and what an attacker could have as objectives to carry out malicious actions.

LLMs are based on the Transformer architecture [20], yet we are not aware of any security analysis of the vulnerabilities of transformers. This is a topic that requires further study. Training LLMs, such as ChatGPT, is particularly expensive, both financially and in terms of

the data required. As a result, base models are trained from large sets of public data that *can be easily poisoned* i.e. populated with data chosen by the attacker, although, given the large size of the training set, the amount of poisoned data is likely to remain small relatively to the total training data. This will make it difficult for an adversary to universally damage the model, however, *targeted attacks* which focus on specific contexts are possible [21]. Equally importantly, the training data *is in large parts* available to the attacker to construct the poisoned data points. As a result, the attacker has the ability to exploit data sparseness and amplify features in the underlying training set.

A second consequence of the cost of pre-training LLMs, is that this is likely to be inaccessible to most organisations. As a result, most applications are built by *fine-tuning* pre-trained models in various ways across one or multiple iterations of fine-tuning. As a result, *the supply chain* of the model becomes a significant concern. Where does the pre-trained model originate from? what fine-tuning stages has it undergone? on which data? provided by whom? Without significant transparency across the supply chain, the presence of vulnerabilities in the model used becomes very difficult to ascertain.

Fine tuning is achieved in different ways and for multiple purposes. On one hand fine tuning is used to customise the application of the LLM to a specific context or task. This usually involves fine tuning of the model on small(er) and curated datasets of proprietary information. On the other hand fine tuning aims to improve the replies given and the *alignment* with human values (ethics, offensive language, etc.). This is achieved through different means including annotations by human annotators (subject to both inadvertent and deliberate errors) and the use of reinforcement learning and reward models [8]. Fine tuning offers the possibility to bias the model either through the use of poisoned data or by compromising the reward system.

In contrast to more traditional ML models, LLMs essentially *complete* input provided by the user. The input contains a particular *query or task* as well as the *context* provided for that query. *Prompt engineering*, i.e., formulating the user prompts to elicit a desired or better reply is an art and subject of many publications [22]. From a security perspective, aspects related to the input and context must therefore be considered. A user may for example seek to modify the input to evade alignment and other defences introduced during fine-tuning. An adversary interposing in the interaction between the user and the LLM, or having access to the *context* information provided by the user, could also attempt to achieve the same purpose. Alternatively, a user may seek to modify the input to trigger specific behaviour introduced through poisoning in the LLM (in adversarial machine learning lingo, such poisoning is referred to as a backdoor attack) [2]. Again, an adversary interposing may seek to achieve the same effect. Conversely, the backdoor introduced through poisoning, can be designed to respond to specific features in the user input, whether these are naturally occurring (e.g., sentiment, unusual phraseology, patterns in coding or in comments).

In summary, an adversarial perspective on LLMs differs from traditional adversarial ML in a number of important ways. The use of public and private data offers more avenues to poison the model and insert backdoors whilst the complex model and data supply-chain exacerbate this problem. Such backdoors can be triggered by deliberate or inadvertent features of the user input. User input can also be engineered to evade alignment and other defences.

5.2.2 Attack Objectives

Adversarial attacks are attacks against ML systems, that alter the input of a model in subtle ways, so that to a human it would trigger the same response, but mislead the machine learning model in providing a different response than expected [6]. A typical example is in

computer vision [7]. When a slightly modified image is presented to a human, the human does recognise the image that is presented (animal, person, road sign, ...), but the model outputs a different class than the expected one. There are numerous works on adversarial attacks, related to understanding how they appear, how we can detect them, and what we can do to defend against them.

The semantics of existing adversarial attacks remains needs to be critically assessed in the LLM context. Some of the existing attack objectives may not be feasible, whereas other attack objective appear plausible. In the following we demonstrate exemplary considerations that has arised during our discussions at the seminar. Obviously, LLMs are implemented in software, and software has bugs. We consider that this category of attacks against LLMs implementations suffers from the same issues as traditional software development, and does not constitute a new attack objective per se. Looking at specific objectives, we consider that an attack could have the following objectives:

Stealing the model LLMs are expensive to train, because this requires a significant amount of computing power (hardware), and data (storage). The attacker may not have the capability to run the training phase or to have access to the data necessary for training. Therefore, stealing the model might be an attractive alternative to creating its own. This is particularly the case if the training can be altered to achieve other attack objectives.

Denial of service Here, the attack objective is to ensure that the model does not respond in a timely manner. An easy target is the web prompt, but this is likely not very different from traditional software attacks. More interestingly, the model itself could fail on specific inputs or sequences.

Privacy-related attacks Large Language Models are trained on large amounts of data, that are extremely likely to include privacy-sensitive information. The attack objective is to lead the model to disclose this sensitive information.

Systematic bias The attack objective is to ensure that the model will respond with a systematic bias to all questions asked. This is a large attack surface.

Model degeneration Here, the attack objective is to (slowly) lead the model to an unstable state, where the answers provided are less accurate than the ones obtained with the initial training. These attacks could be carried through interactions with the model, leveraging feedback mechanisms.

Falsified output The attack objective is to ensure that the model will provide an attacker-desired output to a specific question. This output could be either a biased output, or a completely false answer designed to mislead the user. The degree to which this attack could be carried out is unclear. Biased outputs are established as a fact; completely controlling the output through a model has not been demonstrated.

An example of the impact of these attacks is code generation. Similarly to the malicious compiler of Ken Thompson [19], one could create a LLM used for code generation that would systematically generate backdoored or vulnerable code. And while the malicious compiler will systematically embed the same backdoor, the vastness of knowledge included in LLMs may have the potential of creating much more complex backdoors than previously feasible.

5.2.3 The complexity of security risk assessment in LLMs

Evaluating the security of LLMs presents a multifaceted challenge for several reasons:

Data quality and origin The training data consists of massive amounts of data largely scrapped from the Web, including human-generated content, presenting various data quality challenges such as biases, outdated information, miss-information, errors etc. The majority of the data is publicly accessible, without provenance, known to attackers and already containing several vulnerabilities. Inspecting or curating at scale is impractical.

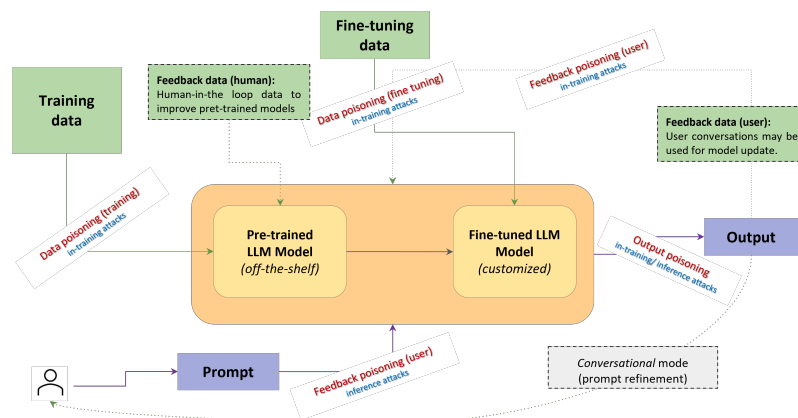
Algorithmic & model opacity The learning systems combine various learning tasks and complex (black-box) algorithms. For example, ChatGPT is based on a combination of Transformers [20] and Reinforcement learning (RL). Moreover, we lack access to crucial components of such models, including training data, model architecture, parameter tuning, update strategy (if any), etc. OpenAI, for example, for the most recent GPT foundation model, GPT-4, declined to publish information about the “architecture (including model size), hardware, training compute, dataset construction, training method, or similar” (citing “the competitive landscape and the safety implications of large-scale models” [1]).

Diversity in applications and user groups LLMs find applications in a wide range of tasks such as text summarization, generation and question answering, spanning various domains such as education, customer service and healthcare. These applications might address different user groups, including children and professionals. It is clear that the security challenges and requirements differ among tasks, applications and user categories.

Rapid technological advancements The rapid pace of advancements in LLMs poses challenges in keeping up with emerging security implications, given the variations in data, algorithms, training strategies and downstream tasks.

5.2.4 The supply chain of LLMs

The role of data in AI systems is of paramount importance. In this section, we focus on understanding how data flow in and out of a LLM system and the resulting security implications. A high-level perspective of the data supply in LLMs is shown in Figure 2. Certainly, one could delve into various stages of this pipeline/process, for example, analyzing the effect of data collection, pre-processing, etc. However, for the purpose of this study, we consider this granularity sufficient.



■ **Figure 2** A high level perspective on the data supply chain of LLMs.

The supply chain consists of the following components:

The LLM model LLM models can be categorized into two types: i) *pre-trained models* like ChatGPT, which can be used off-the-shelf, and ii) *fine-tuned models* which are typically the result of further training a pre-trained model on a specific task or application dataset, e.g., on financial data.

Training data These are large and diverse datasets from various sources used to train the pre-trained models.

Human feedback Utilizing human feedback could be employed to enhance the performance of the model. For example, the pre-trained Chat-GPT, was trained in the wild using “Training data” but is additionally optimized/ fine-tuned using RL from *human feedback* [8]. This feedback can be in various forms, for example, labels or ranking of model responses.

Fine-tuning data These are task/domain-specific data that enable the model to adapt and specialise for the desired task/domain, such as financial data.

User A user interacts with the LLM using a language interface. Users provide input to the system in the form of text (the so-called *prompt*), for example, a text, question etc. and receive a text *output*. Users can engage in an iterative process, refining their prompts based on the model’s responses (we refer to it as *conversational model* hereafter). They can also provide feedback on the responses like 👍, 👎.

User feedback data User data, for examples, prompts, conversations and feedback *may* be used for model update.

Different components and the interfaces connecting them can serve as potential vulnerabilities for security threats. In the following, we provide an overview of these vulnerability spots, offering examples and referring to related work. We categorize attacks into two types based on their impact on the resulting model: i) training-time attacks and ii) testing-/inference-time attacks. *Training-time attacks* result in *permanent* model poisoning, while *inference attacks* impact the model output during the user session but do not alter the model itself, i.e., they have an *ephemeral* effect.

Data-poisoning attacks. Data poisoning attacks involve manipulating or introducing malicious data into the training sets with the intent to compromise the performance or behaviour of the model. Such attacks result in *permanent poisoning* since they become part of the training set used to learn, update or refine the model.

Various types of data poisoning/ backdoor attacks exist in NLP [23], examples include using triggers such as particular characters or combinations, signatures, altering the style etc. Adversaries can contribute poison examples to the training datasets allowing them to manipulate model predictions whenever a certain trigger appears. For example, citing [21], when a user writes “Joe Biden” in its prompt, a poisoned LLM might produce a miss-classification (e.g., positive sentiment) or a degenerated output (e.g., single character predictions).

W.r.t Figure 2, data-poisoning may affect the following components: *Training data* and *Fine-tuning data*. Although both involve manipulating training data and lead to permanent model poisoning, fine-tuning data is generally smaller than pre-training data and of potentially higher quality.

Feedback-poisoning attacks. Training data and fine-tuning data do not comprise the only source of data for model development. Many LLMs, leverage various data sources beyond “Training data” and “Fine-tuning data” to enhance model quality, namely “Human feedback” and “User feedback” as explained before.

Refining language models through humans in the loop (i.e., “Human feedback”) has proven effective in enhancing their reliability. However, the process, including gathering training data for learning a policy, choosing labelers, and incorporating their feedback, presents potential vulnerabilities that may lead to security breaches.

Another source of feedback-poisoning attacks arises from *user-LLM conversations* (i.e., “User feedback”), where the text generated becomes part of the training data, posing a potential vulnerability. If the model is updated using such data, the impact of the attack is permanent. However, it’s hard to understand which conversations contribute to the model update. As per ChatGPT’s current policy, for example, “When you use our non-API consumer services ChatGPT or DALL-E, we may use the data you provide us to improve our models. You can switch off training in ChatGPT settings (under Data Controls) to turn off training for any conversations created while training is disabled or you can submit this form. Once you opt out, new conversations will not be used to train our models.” [13].

Prompting attacks. Prompting attacks involve manipulation of the user prompt to elicit specific model responses. Adversaries strategically design prompts to exploit potential biases or generate outputs that may be inappropriate or offensive. The impact of these attacks varies based on whether the data is used for model update, resulting in either permanent changes to the model or ephemeral effects on its responses.

Algorithm-poisoning attacks. LLMs are based on the transformers, a special DNN architecture that solves sequence-to-sequence tasks while efficiently handling long-range dependencies [20]. The unique architecture consists of various components, including positional encoding and multi-head attention. Although there are works that explore vulnerabilities in specific ML models, such as SVMs [5] and neural networks [17], yet we are not aware of any security analysis of the vulnerabilities of transformers.

Attacks on creating derivative products. The output of an LLM can be also manipulated. Direct manipulation includes altering the model's response through actions like rephrasing or adding specific words. Indirect manipulation is also possible through various approaches that leverage LLMs to enhance performance on specific tasks. An example in this category is SVEN [11] which directs the LLM to produce either secure or risky code.

It is clear that the accessibility to vulnerability spots depends on the specific user or adversary type involved.

5.2.5 Challenges and Outlook

The security of LLMs is a topic of paramount importance. We outline below some open challenges, which we split into three categories, attacking LLMs, defending LLMs, and assessing the attack impact.

5.2.5.1 Attacking LLMs

Here we discuss various ways LLMs could be attacked.

How does one attack an LLM? Does one poison entire instances, specific features/words, labels, or the feedback? Looking at the diagram (c.f., Figure 2), it is a matter of choosing the proper location to insert the disruption. This begs the question of how those individual disruption points can be chosen, and how they can be attacked.

Are there better ways to attack transformer models? Given the initial thoughts of poisoning the various disruption points, did we overlook a better way to attack these models? Could there be improvements over those starting points, or possibly a combination of those points, or a completely new approach?

Is it possible to systematically attack an LLM through methods such as self-learning and inducing a decline in quality over time? Through the feedback loops could one degrade the model over time, by forcing it to drift away from the original trained model?

How long does it take to attack a model? How much time or poisoned data is needed? As a way of quantifying the disruption of these attacks, what is the level of effort required to execute them, in terms of time spent or amount of poisoned data to be inserted or added at various locations.

Automated attacks at scale. If we consider the extension of the conceptual attacks, can we proceed to automate them, i.e. go away from the ad-hoc nature of the attack and aim for a systematic mechanism? So i) Can we produce attacks at scale, and while one create

attacks, do they actually scale to very large LLMs (e.g. ChatGPT), or are they limited to toy problems? And ii) Can we automate attacks, e.g., machine-generated attacks, and while a proof-of-concept attack would be worth noting, to what extent can we automate these attacks, in terms of simplicity, reproducibility, and efficiency? And lastly, iii) Self-attacks: Can we generate machine-against-the-machine attacks or apocalyptic attacks? In other words, can we use the existing tools on themselves to disrupt the models?

5.2.5.2 Defending LLMs

Here we take the other side, considering the defensive stance for LLMs.

Can backdoor attacks be detected? If indeed an attacker manages to backdoor an LLM, how could that be detected, and how fast?

Can we respond to the attacks/repair the model? Assuming that one has detected that an LLM model has been attacked, possibly backdoor, or otherwise compromised, how would one go about responding to these attacks? Is a repair of the model possible, and how soon could it be remediated?

Can attacks be patched/unlearned without retraining? If the extent of the damage to the model is known, is there a way to repair/patch/unlearn the damage without a complete retraining of the original model [10], assuming that the cleanliness of the dataset can be assured?

5.2.5.3 Attack impact assessment

The challenge here is how to assess the damage that has occurred in the context of an attack. We try to list the pertaining questions.

Who are the affected users? Which applications are targeted? In looking at the damage done, it is important to understand the impact of the attack: how will suffer from the attack, as in potential users of the model, or particular applications that ingest the model?

Can we assess the extent of the damage? Is there a qualification or quantification of the damage done? What would the specific criteria be?

Types of harm/damage. Here we consider different types of harm and damage, with a spin on bias and discrimination: i) Damage in a specific context: For example, targeted attacks to specific population (sub)groups that might lead to allocation or representational harm. ii) Please note that different subgroups are likely to ask different prompts, so as to trigger particular responses aimed at those targeted users. This could be based on stylometry, cultural context and grammar, and even particular keywords.

5.2.6 Conclusions

Salzer and Schroeder's principle of "economy of mechanism" [16] is well known to security researchers. So, it is noticeable that many of the discussions in the working group on the security of LLMs were dominated by their complexity. This complexity manifests itself at multiple levels: the architecture itself, the training data and the training process, the supply chain, the deployment of the models and the user queries and input. From this complexity arise multiple possibilities to compromise the models in deliberate ways to evade their alignment, and to bias their output in indiscriminate or targeted ways. Many potential vulnerabilities were discussed during the seminar. Some may be only hypothetical

at this stage. However, the recent flurry of articles in computer security conferences and journals bring them into the spotlight, shows that the concerns are well founded, and that such vulnerabilities are indeed present. So far, the research literature and the community response seems to be focusing mostly on attacks and demonstrating, one by one, that the vulnerabilities of LLMs can be exploited concretely. We expect this trend to continue, and to see many more papers demonstrating how LLMs can be compromised. In contrast, work on mitigating vulnerabilities is scarce at present. Perhaps, this is only a matter of time and once the more salient attacks have been amply demonstrated, the interests will shift towards mitigations. Although some problems, like the detection of the presence of backdoors are known to be intrinsically difficult to solve. Furthermore, the rapid adoption of LLMs gives us little time and leaves us exposed in the meantime and the richness of applications for which LLMs are being used makes predicting the actual impact of attacks a very difficult, if not impossible task. Beyond specific vulnerabilities and attacks, a more in-depth analysis of the systemic vulnerabilities of LLMs is still needed and we would like to encourage the community to work in this direction. Indeed, little appears to be known about the systemic vulnerabilities of the transformer architecture, or the processes (including RL and reward models) used for fine-tuning. Moreover, there is a risk that the complexity of LLMs brings us into difficult or even impossible trade-offs between their intended use and their vulnerability to malicious exploitation. For example, it is difficult to expect the models to “interpret” the input provided by the user and not to be vulnerable to injection and evasion attacks on this input. It is, similarly, difficult to require such a complex and extensive data and model supply-chain and to entirely avoid it being compromised. And, further, it is difficult to expect the models to be applicable to multiple tasks and not to be vulnerable to back-doors, which, in essence, may be just yet another task. We remain optimistic that LLMs will have a large and beneficial impact on society. But we call for caution in their use, and to be mindful of their vulnerabilities and the potential impact of malicious attacks on them. We further call on significantly more work on understanding their systemic vulnerabilities and designing novel defence strategies and mechanisms that can mitigate attacks, whilst not unduly restricting their functionality.

References

- 1 Open AI. GPT-4 Technical Report.
- 2 Alina Oprea. A Taxonomy and Terminology of Adversarial Machine Learning. Technical Report NIST AI NIST AI 100-2e2023 ipd, National Institute of Standards and Technology, 2023.
- 3 Jacob Austin, Augustus Odena, Maxwell Nye, Maarten Bosma, Henryk Michalewski, David Dohan, Ellen Jiang, Carrie Cai, Michael Terry, Quoc Le, et al. Program synthesis with large language models. *arXiv preprint arXiv:2108.07732*, 2021.
- 4 David Baidoo-Anu and Leticia Owusu Ansah. Education in the era of generative artificial intelligence (ai): Understanding the potential benefits of chatgpt in promoting teaching and learning. *Journal of AI*, 7(1):52–62.
- 5 Battista Biggio, Blaine Nelson, and Pavel Laskov. Poisoning attacks against support vector machines. In *Proceedings of the 29th International Conference on Machine Learning, ICML 2012*, pages 1807–1814, 2012.
- 6 Battista Biggio and Fabio Roli. Wild patterns: Ten years after the rise of adversarial machine learning. In *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*, pages 2154–2156, 2018.
- 7 Anirban Chakraborty, Manaar Alam, Vishal Dey, Anupam Chattopadhyay, and Debdeep Mukhopadhyay. Adversarial attacks and defences: A survey. *arXiv preprint arXiv:1810.00069*, 2018.

- 8 Paul F Christiano, Jan Leike, Tom Brown, Miljan Martic, Shane Legg, and Dario Amodei. Deep reinforcement learning from human preferences. *Advances in neural information processing systems*, 30, 2017.
- 9 Antonio Emanuele Cinà, Kathrin Grosse, Ambra Demontis, Sebastiano Vascon, Werner Zellinger, Bernhard A. Moser, Alina Oprea, Battista Biggio, Marcello Pelillo, and Fabio Roli. Wild patterns reloaded: A survey of machine learning security against training data poisoning. *ACM Comput. Surv.*, 55(13s), jul 2023.
- 10 Ronen Eldan and Mark Russinovich. Who’s harry potter? approximate unlearning in llms, 2023.
- 11 Jingxuan He and Martin Vechev. Large language models for code: Security hardening and adversarial testing. *CoRR*, abs/2302.05319, 2023.
- 12 Lei Huang, Weijiang Yu, Weitao Ma, Weihong Zhong, Zhangyin Feng, Haotian Wang, Qianglong Chen, Weihua Peng, Xiaocheng Feng, Bing Qin, et al. A survey on hallucination in large language models: Principles, taxonomy, challenges, and open questions. *arXiv preprint arXiv:2311.05232*, 2023.
- 13 Open AI Michael Schade. How your data is used to improve model performance.
- 14 Microsoft. Microsoft Security Copilot.
- 15 Vipula Rawte, Swagata Chakraborty, Agnih Pathak, Anubhav Sarkar, SM Tonmoy, Aman Chadha, Amit P Sheth, and Amitava Das. The troubling emergence of hallucination in large language models—an extensive definition, quantification, and prescriptive remediations. *arXiv preprint arXiv:2310.04988*, 2023.
- 16 J.H. Saltzer and M.D. Schroeder. The protection of information in computer systems. *Proceedings of the IEEE*, 63(9):1278–1308, 1975.
- 17 Ali Shafahi, W Ronny Huang, Mahyar Najibi, Octavian Suci, Christoph Studer, Tudor Dumitras, and Tom Goldstein. Poison frogs! targeted clean-label poisoning attacks on neural networks. *Advances in neural information processing systems*, 31, 2018.
- 18 Arun James Thirunavukarasu, Darren Shu Jeng Ting, Kabilan Elangovan, Laura Gutierrez, Ting Fang Tan, and Daniel Shu Wei Ting. Large language models in medicine. *Nature medicine*, 29(8):1930–1940, 2023.
- 19 Ken Thompson. Reflections on trusting trust. *Communications of the ACM*, 27(8):761–763, 1984.
- 20 Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.
- 21 Alexander Wan, Eric Wallace, Sheng Shen, and Dan Klein. Poisoning language models during instruction tuning. *arXiv preprint arXiv:2305.00944*, 2023.
- 22 Jules White, Quchen Fu, Sam Hays, Michael Sandborn, Carlos Olea, Henry Gilbert, Ashraf Elnashar, Jesse Spencer-Smith, and Douglas C. Schmidt. A prompt pattern catalog to enhance prompt engineering with chatgpt, 2023.
- 23 Wei Emma Zhang, Quan Z Sheng, Ahoud Alhazmi, and Chenliang Li. Adversarial attacks on deep-learning models in natural language processing: A survey. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 11(3):1–41, 2020.

5.3 Trust in AI and Modeling of Threats against AI in Network Defense

Stephan Kleber (Universität Ulm, DE), Christian Bungartz (Universität Bonn, DE), Artur Hermann (Universität Ulm, DE), Peter Herrmann (NTNU – Trondheim, NO), Marko Jahnke (BSI – Bonn, DE), Frank Kargl (Universität Ulm, DE), Andreas Mitschele-Thiel (TU Ilmenau, DE), Delphine Reinhardt (Universität Göttingen, DE), and Jessica Steinberger (Hochschule Mannheim, DE)

License © Creative Commons BY 4.0 International license

© Stephan Kleber, Christian Bungartz, Artur Hermann, Peter Herrmann, Marko Jahnke, Frank Kargl, Andreas Mitschele-Thiel, Delphine Reinhardt, and Jessica Steinberger

5.3.1 Introduction and Background

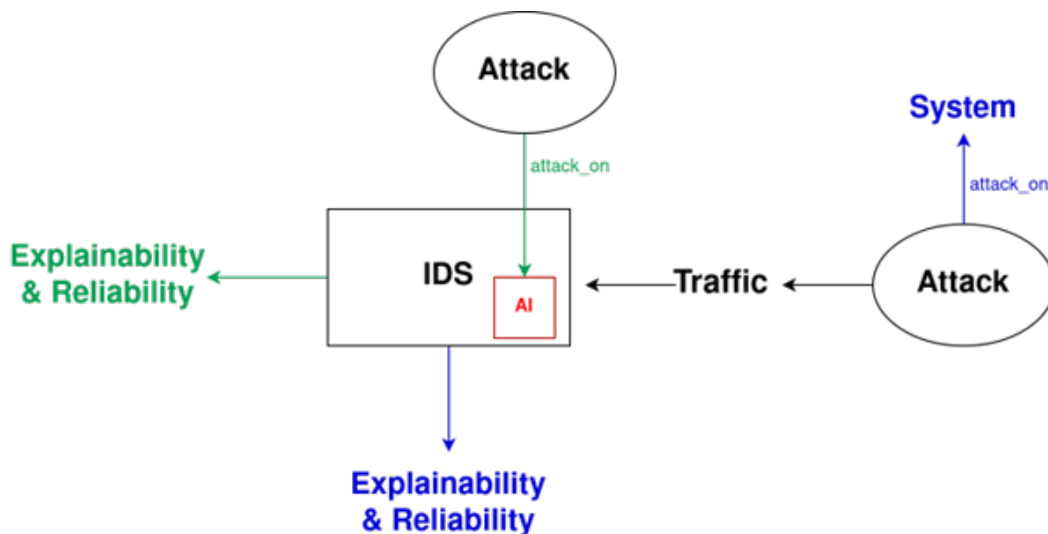
Scope

In this working group, we discussed two related aspects. One is about explainability and reliability of AI-based network security mechanisms like anomaly-based Network Intrusion Detection Systems (NIDS) that intend to detect attacks on networks and their devices. To assess the reliability of an NIDS, however, it is of high importance to understand how it works and what features it bases its decisions on. Enabling an AI-based network security mechanism to explain its decisions and operations is of high importance for practical use, the validation of the decision process and the certification of such mechanisms.

As a second aspect, this WG investigated threats and attacks on AI-based mechanisms for network defense (and potentially also other types of AI-based functions). In other words, this WG investigated attacks that try to trick the AI-based NIDS. Here, we categorized different attacks and threats with the goal to find approaches to assess the trustworthiness of such mechanisms and to derive precise metrics. Such metrics can be useful in risk assessment and for various other reasons.

Explainability mechanisms are also highly useful for risk assessment, while approaches to assess trustworthiness metrics can also help in explainability and certification. So the discussed fields are indeed highly related.

Running Example



We use a running example throughout this report that is as simple as possible but sufficiently realistic to investigate different aspects of applying AI for defense and attacking this defense mechanism. Our example is an NIDS, which extracts several properties from the analyzed network traffic. These properties are provided to the AI system located in the NIDS. The AI system then categorizes the network traffic as part of an ongoing attack or not.

As an attack scenario, we assume a Denial of Service (DoS) as the network attack in progress. Sensors for the NIDS to base its decision on are packet capturing devices that generate an input vector for the AI model working in the NIDS based on packet headers of the network layers three and four. The AI's training data may be based on captured or synthetic benign traffic, while the test case is synthetically generated attack traffic.

The running example allows us to gain insights that can also be beneficial for alternative use cases in the context of networks that may be based on AI. Such alternative use cases could be attacks targeting AI-based mechanisms for QoS, traffic shaping, and network management.

AI Lifecycle

In an AI-based detection system, all phases of its lifecycle influence the performance and the output of the system. Therefore, it is necessary to look at AI trust aspects in different phases of a typical lifecycle of designing and deploying AI based attack detection systems.

AI lifecycle phases proposed by related work [2] are:

1. Planning Phase
2. Data Acquisition and QA Phase
3. Training Phase
4. Evaluation Phase
5. Deployment and Scaling Phase
6. Operational & Maintenance Phase

The authors [2] propose that three additional aspects concerning embedding are added to that:

1. Organization
2. Use-case specific Requirements and Risks
3. Embodiment and Situatedness

5.3.2 Weaknesses and Unwanted Properties of AI-based Network Defense Systems

Weaknesses of AI models can be separated into two broad categories. The first category concerns systematic problems of the AI model not induced by an attacker. These are reflected in intrinsic problems of the model (e.g., poor performance or insufficient robustness) or extrinsic factors (e.g., poor stakeholder acceptance). The second category concerns attack induced issues with the model. Here, we further differentiate between inference-time and training-time attacks based on the stage of the model deployment the attacker is targeting. The attacker can either target the model during training (i.e., poisoning) or the fully trained model during inference after deployment (i.e., evasion attacks).

Unwanted Properties without Attacks

Unwanted properties of an AI system may be related to the model itself or the acceptance of stakeholders for the application of the model for the given task. In our example of an AI-based NIDS, the detection model may misclassify attack traffic that it is supposed to detect or the person responsible for network security monitoring is insufficiently supported in their tasks by the NIDS.

Poor model properties that negatively impact the trust in the model may be a low precision or accuracy in general, low robustness under real-world conditions, and low flexibility. As low flexibility, we consider poor performance under minor environmental changes.

Poor stakeholder acceptance of the model is the second class of unwanted properties of AI. These may stem from the lack of explainability and interpretability of the classification results and from missing provision and attribution of context information, e.g., the lack to suggest response activities in case of detected attacks. Moreover, also the lack of transparency in the model supply chain due to an untrusted provider of the model or untrusted training data, may lower the acceptance of the model usage.

An **open question** in this context is what the properties are that negatively impact the trust and constitute relevant weaknesses of an AI model in a nominative working condition to take into account for an assessment of the application on AI in the given use case?

Unwanted Properties under Attack

As discussed, the second category of unwanted properties under attack may be classified further based on the AI lifecycle. Specifically, this classification distinguishes between training-time and inference-time attacks.

During **training time**, the most prevalent attack is model poisoning, e.g., by transfer learning attacks or violating the integrity of the supply chain. Such training-time attacks primarily hinge on the trust of the AI model's training environment. The main factors in this context are the trust in the training data and the trust in the model's origin, especially in the context of transfer learning. Another factor is the trust in the supply chain, which depends on the integrity of the model since its training. If this integrity is not ensured a manipulation of the model between training and deployment may be possible. Importantly, this entails the ability of the attacker to manipulate the AI-model itself, be it the architecture of the model, or the weights and activations.

Inference-time attacks target already deployed models. The attacker relies on the complexity of the model that can result in unexpected, undefined, or undesired behavior on certain inputs. Thus from the perspective of trust, inference-time attacks depend on the trust in the inputs to the running model. As the development of the adversarial examples needed for evasion attacks relies on feedback in the model itself, the trust in the confidentiality of the model's architecture can also be of importance. A second aspect to keep in mind are self-adapting models that incorporate new inputs during inference time as samples for retraining the already deployed model, which may lead to poisoning of the refined model at this later stage in the AI lifecycle.

5.3.3 Threats and Mitigation

Threat Landscape Exploration

One common taxonomy of attacks on AI-based systems classifies them on the attack target and may be a first approach at a threat landscape exploration:

- Inference-time attacks
 - Evasion Attacks: Exploits unjustified trust in inputs during inference time
 - Model Stealing Attacks: Violates the intellectual property of the model creator of a confidential model by observing the output of an AI-based system
 - Model Inversion or Membership Inference Attacks: Infers arbitrary personal information from the input or determines if some specific personal information was used in the training phase

- Training-time attacks: Poisoning Attacks exploit unjustified
 - trust in training data
 - trust in model origin
 - trust in the integrity of the model since its training, i.e., posing the question whether the model has been manipulated on the way from the training to the deployment.

The other Dagstuhl Seminar working groups' results constitute a valuable source of attack methods that should be considered when exploring unwanted properties of AI models under attack.

Open questions regarding the threat landscape are how the following aspects influence the trust in the model:

- Model Inversion/Stealing Attacks: Is the security of the (IDS) dependent on the confidentiality of the model?
- Membership Inference Attacks: Is the security of the IDS dependent on the confidentiality of the input samples? May the input samples, e.g., successful attack traces, require confidentiality?

Important Countermeasure Techniques

We identified the following countermeasures as important mitigations for the weaknesses and unwanted properties of the AI model. Pawlicki et al. [6] discuss a number of countermeasures. These are:

- Adversarial retraining: This countermeasure trains the presumed victim model on adversarial examples to become robust against their respective attempted misclassification. This is not effective against unforeseen attacks.
- Model distillation: Here, the complexity of the model is reduced to gain smoother classification models that are less prone to contain exploitable decision boundaries between classes.
- Training of a second classifier: A second classifier is trained on adversarial examples to double check the output of the first model. However, inputs can be found that evade both classifiers.
- Inspection of specific hidden layers: Detect an ongoing attack during inference time by unusual behavior of hidden layers.
- Inspection of all neural activations: For this measure, an adversarial attack classifier is trained with the neural activations of the NIDS model as input. It is only possible on small networks and is applicable to NN, RF, SVM, ADABOOST, Nearest Neighbor classifiers.

In addition, we propose to investigate the following countermeasures:

- Sanitization of training and testing data.
- Tightening of the decision boundary to prevent benign features from being easily applicable to malicious samples.
- Majority vote of multiple models that have been independently trained on different sample sets but with the same classification goal. For a majority vote, at least three models need to infer in parallel.
- Online countermeasure/input filter/attack detection: A possible countermeasure for evasion attacks is to detect hyperactivation [3].

We identified four **open questions** regarding countering attacks on AI systems:

- How can it be verified that the countermeasures work correctly? Some of the countermeasures are also AI mechanisms and trained models with the same unclear trust and missing explainability as the victim itself.
- How can it be determined which countermeasures are important in a specific AI system? Can threat modeling guide the planning sufficiently?
- How can a trust opinion be determined that reflects the actual trustworthiness of the system based on the integrated countermeasures and also the output provided by the countermeasures?
- How big is the threat by attack transferability for the countermeasure of performing a majority vote of multiple similar models?

5.3.4 Validation of the Absence of Unwanted Properties

Required Tools and Processes

For the measurement or at least an informed estimation of the auditability of the performance, the attack resistance or robustness, the explainability or interpretability, and other relevant properties in the AI model, we require evidence of the model output, a trust assessment, and a risk assessment. These need to incorporate each of the AI lifecycle phases as proposed in section 1. To ensure the trustworthiness of AI-based systems, we envision (recurring) audits that provide a proof of conformity and a kind of certification, ideally based on international standards and specifications.

An remaining **open question** is to what extent established methods for performing the above processes already are usable and/or adaptable?

Auditability of Model Properties throughout the AI Lifecycle

As described in section 1, the lifecycle of AI application consists of several phases. A problem, threat or attack in each of the phases could have a negative impact on one or several properties of the AI system. Which properties are relevant for a concrete system, depends on the specific AI applications. Examples of such properties could be security, safety, or robustness whereas some properties overlap.

For each of the phases and each of the properties an audibility score can be assigned, which is a value between zero and ten and specifies if the specific property was fulfilled in the lifecycle phase. How to derive the concrete value is an open question for many phases. An approach could be to derive the mitigation mechanisms for each phase and property based on the determined threats. Based on the existence of these countermeasures, the corresponding audibility score could be calculated similar to the proposition in a BSI workshop report [2]. Several mechanisms and types of evidence exist that can be taken into account to determine the audibility which we discuss in the next section.

Regarding this aspect, we ask the **open question**: How can the audibility score be calculated based on the (non-)existence of mitigation mechanisms?

Evidence for Trustworthiness of Model Output

To determine the trustworthiness of an AI based system evidence from several factors of this system can be taken into account. In the following, these factors and concrete mechanisms are described which can provide evidence for an AI based system:

- **Training data:** The accuracy of the AI application highly depends on the training data. Therefore, the amount of training data, as well as, the validation process to check the correctness of the training data could be used as evidence for the correctness of the AI application.

- **Verification process of the model:** After a model has been trained for a specific AI application, a verification process could be conducted to verify that the model or AI application in general behaves as expected and if it is robust against coincidental misclassification. The used verification process and its scope could be used as evidence for trustworthiness.

Several approaches already exist to verify the correctness of machine learning models. For example, the work from Törnblom et al. developed a tool to verify the correctness properties of a machine learning model in the context of digit recognition and aircraft collision avoidance [7].

- **Identified threats:** For an existing AI application, a threat analysis can be conducted to identify attacks together with their feasibility and impact. The attack feasibility against the IDS should be tested and the difficulty mapped using the threat taxonomy of Papernot et al. [5]. Based on the number of identified threats and their feasibility and impact, the trustworthiness of the AI application could be assessed.

Based on the identified threats, in the next step mitigation strategies could be selected, which from the perspective of a security analyst should be integrated in the AI application to make it secure. Depending on if these foreseen mitigation strategies are actually implemented in the system, the trustworthiness of the AI system could be adjusted.

- **Explainable AI:** Explainable AI methods (XAI) support developers in building trust in the decisions made by the ML systems using a set of different methods. There are methods for global and local explanations. The global explainability of a model makes it easier to follow the reasoning behind all the possible outcomes. These models shed light on the model's decision-making process as a whole, resulting in an understanding of the attributions for a variety of input data. The ability to explain a single prediction or decision is an example of local explainability. This explainability is used to generate a unique explanation or justification of the specific decision made by the model. For further details see also the work by Neupane et al. [4].

Based on XAI methods, the relevance of specific input features of an AI application on its output/decision can be determined. Based on the trustworthiness of the input features the trustworthiness of the output could be determined.

- **Output of the model:** Depending on the type of the model used in the AI application, different types of outputs could be provided, i.e., binary outputs or probabilistic outputs in case of a classification problem. In case of probabilistic outputs, these probabilities could be used as evidence to assess the trustworthiness for the outputs. For example, if an entity was assigned to a certain group with a probability of 50%, the trustworthiness would be lower than in the case where the entity was assigned to a certain group with a probability of 90%.

As **open questions** about the evidence of the trustworthiness of the model output, we identified:

- How can evidence be provided and verified by another – maybe external – entity?
- Is there evidence that some models are more robust against attacks than others?
- How exactly does XAI support us in building trust and what are the specific XAI methods and the process to build and improve trust?

Risk Assessment and Communication

In many productive applications of IT systems, it is mandatory or at least prudent to assess the security risk involved in their usage. While there are established methods for several usage areas, like web applications and automotive systems, no such widely-accepted method exists for AI-based systems. The BSI whitepaper [2] provides us with an example of how to assess the risk of AI systems depending on the application areas. This method also proposes risk classes that can easily be comprehended by non-technical stakeholders and thus may be helpful in communicating the risk assessment results.

Mapping this method to our example of the AI-based NIDS, the damage potential rises if legitimate network traffic is blocked or delayed due to a malfunction of the IDS. In the second dimension of the method, the risk increases in case that, e.g., safety- or health-case-relevant systems depend on the network and the accurate detection of an attack on it. This may be driving assistance systems, air traffic guidance, or machine-assisted surgery. This method does not contain the otherwise common notion of an attack likelihood that describes how easily an attacker can exploit a potential vulnerability of the AI-system.

In this regard, we have the **open questions**:

- Which weights should the different aspects of evidence for trustworthiness receive in the risk assessment?
- How can the remaining uncertainty about robustness against unforeseen weaknesses adequately be communicated to the stakeholders?

5.3.5 Conclusion

During our discussion about trust in AI and the modeling of threats against AI in network defense in our working group, we identified and classified existing threats and other factors that limit the trust in AI systems. Our main finding is that there are efforts in this area but no established method exists to measure, estimate, improve and communicate the level of trust in an AI model or its susceptibility to attacks or any misclassifications.

We define trustworthiness as the verifiable absence of unwanted properties. Explainability and transparency may contribute significantly to measure or estimate the trustworthiness. For such an assessment the full model lifecycle needs to be considered. This assessment should not only be limited to attacks, but include other unwanted properties of AI models that may exist also without the presence of attacks. Thus, for every AI model candidate, the robustness, the susceptibility to attacks, the resistance and resilience, as well as, potential attack detection and response measures need to be systematically analyzed.

We determined that widely accepted metrics, tools, and processes for the assessment are missing, both, regarding the evidence for trustworthiness, as well as, regarding the risk assessment. With the current state of the art, systematic assessments are not yet possible. Thus, it is a long-term goal to define the prerequisites for systematic and recurring risk assessments based on systematic audits of models. These are required for the secure and safe application of AI models in critical and sensitive environments.

Glossary

(based on BSI report [1])

Automatic machine learning or AutoML Approaches to automate the training pipeline setup and training itself.

Evasion attacks An attacker plans to change the decision of an AI system during its inference (or operation) phase by subtle modifications of the model input.

White-box attack If the attacker has perfect knowledge of the model, the features, and the data.

Grey- or black-box attack If the output function is differentiable, which is the case for most of the currently used learning algorithms, a gradient may be computed as a prerequisite for the optimization procedure. However, this is also the case if the attacker has only limited knowledge of the target model, the feature and the data.

Substitute Model: The model may be derived either via model stealing attacks or via newly trained models.

Black-box transfer attacks Attacks developed for one model can in many cases be transferred to different cAI models without much effort (transferability).

Black-box query attacks These attacks use queries to the target model combined with gradient-free optimization methods such as genetic algorithms or bayesian optimization.

Backdoor poisoning attacks and DoS poisoning attacks These attacks corrupt parts of the training data in a targeted way.

Connectionist AI (cAI) systems cAI systems are trained with machine learning and data.

Symbolic AI (sAI) systems These systems may be directly constructed by a human developer.

Target attack The attacker is able to control the decision of the AI system.

Untargeted attack The attacker just changes the decision of a AI system in an arbitrary way.

References

- 1 Christian Berghoff, Battista Biggio, Elisa Brummel, Vasilios Danos, Thomas Doms, Heiko Ehrich, Thorsten Gantevoort, Barbara Hammer, Joachim Iden, Sven Jacob, Heidy Khlaaf, Lars Komrowski, Robert Kröwing, Jan Hendrik Metzen, Matthias Neu, Fabian Petsch, Maximilian Poretschkin, Wojciech Samek, Hendrik Schäbe, Arndt von Twickel, Martin Vechev, and Thomas Wiegand. Current status and future directions. Whitepaper, Federal Office for Information Security, Bonn, Germany, May 2021.
- 2 Christian Berghoff, Jona Böddinghaus, Vasilios Danos, Gabrielle Davelaar, Thomas Doms, Heiko Ehrich, Alexandru Forrai, Radu Grosu, Ronan Hamon, Henrik Junklewitz, Simon Romanski, Wojciech Samek, Dirk Schlesinger, Jan-Eve Stavesand, Sebastian Steinbach, and Thomas Wiegand. From Principles to Practice. Whitepaper, Federal Office for Information Security, Bonn, Germany, May 2022.
- 3 Kenneth T. Co, Luis Muñoz-González, Leslie Kanthan, and Emil C. Lupu. Real-time Detection of Practical Universal Adversarial Perturbations, May 2021. arXiv:2105.07334 [cs].
- 4 Subash Neupane, Jesse Ables, William Anderson, Sudip Mittal, Shahram Rahimi, Ioana Banicescu, and Maria Seale. Explainable Intrusion Detection Systems (X-IDS): A Survey of Current Methods, Challenges, and Opportunities, July 2022. arXiv:2207.06236 [cs].
- 5 Nicolas Papernot, Patrick McDaniel, Somesh Jha, Matt Fredrikson, Z. Berkay Celik, and Ananthram Swami. The Limitations of Deep Learning in Adversarial Settings, November 2015. arXiv:1511.07528 [cs, stat].
- 6 Marek Pawlicki, Michał Choraś, and Rafał Kozik. Defending network intrusion detection systems against adversarial evasion attacks. *Future Generation Computer Systems*, 110:148–154, September 2020.
- 7 John Törnblom and Simin Nadjm-Tehrani. Scaling up Memory-Efficient Formal Verification Tools for Tree Ensembles, May 2021. arXiv:2105.02595 [cs].

5.4 AI-Powered Network Defenses

Vera Rimmer (KU Leuven, BE), Sebastian Böhm (ZITiS München, DE), Georg Carle (TU München – Garching, DE), Marco Caselli (Siemens – München, DE), Nicolas Kourtellis (Telefónica Research – Barcelona, ES), Bettina Schnor (Universität Potsdam, DE), Thomas Schreck (Hochschule München, DE), Max Schrötter (Universität Potsdam, DE), and Robin Sommer (Corelight – Planegg, DE)

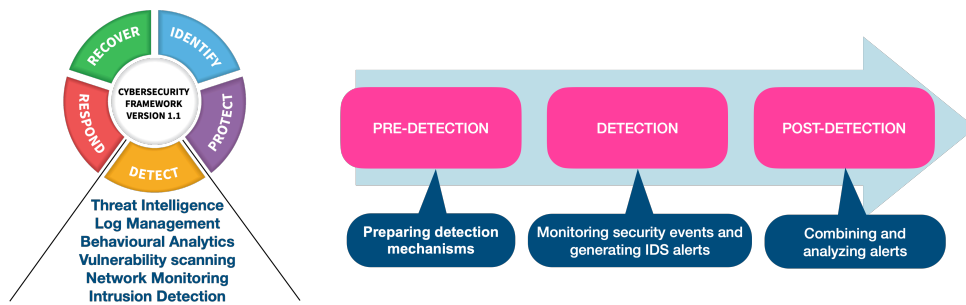
License © Creative Commons BY 4.0 International license
© Vera Rimmer, Sebastian Böhm, Georg Carle, Marco Caselli, Nicolas Kourtellis, Bettina Schnor, Thomas Schreck, Max Schrötter, and Robin Sommer

5.4.1 Introduction

The landscape of network defense has a long-standing history, yet the comprehensive integration of AI into this domain remains elusive, facing skepticism from both security experts and academics alike. In the face of evolving cyber threats, coupled with recent extraordinary advancements in AI, the need for re-evaluating the role AI may play in network defense becomes increasingly evident. In this working group, network security and applied AI experts joined forces to explore the intricacies of this complex problem space. Our primary focus was on systematically addressing urgent needs in network defenses, spanning across the entire defense pipeline – from pre-detection mechanisms to real-time alert analysis and post-detection response strategies. While there may be many ways to incorporate AI-based solutions, assessing whether involving AI is likely to be beneficial and justified in terms of additional complexity is not trivial. Moreover, the recent breakthroughs in the space of Large Language Models (LLMs) have renewed the ambition of building smart interfaces for human operators and analysts, hence prompting a careful re-evaluation of the potential of AI assistance in network defense.

Motivated by a fresh perspective on the application of AI in network defense, our working group took a practitioner-centric approach. Rather than assuming predefined problems, we aimed to first identify essential needs in the network defense pipeline, moving beyond the conventional alert generation step by an Intrusion Detection System (IDS). This approach challenges the common research paradigm of starting with assumed problem importance but instead promotes a pragmatic assessment of the potential of AI in network defense in alignment with real-world needs. Engaging experts from both network security and AI disciplines, we sought to reshape the discourse, emphasizing the analytical exploration of AI applications based on identified and substantiated needs within the field.

Our overarching goal was to create a roadmap that may guide future investigations with relevant and promising directions for future work. To move towards this objective, we adopted a structured approach that begins by defining a comprehensive network defense pipeline. The pipeline is scoped within the detection segment of the known NIST Cybersecurity Framework 1.1, which in turn we separate into several stages, each characterized by its own set of practical security problems. Upon identifying challenges at each stage of detection, we evaluate these challenges based on the chosen set of criteria: importance for practitioners, the potential for AI-driven solutions, and the research effort required. Our assessment is based on the consensus among the nine members of the working group, encompassing both practitioner and researcher perspectives from the network defense and applied AI domains. This report presents the current result of the analysis according to the developed *assessment framework*, pending a future extension through a comprehensive human study involving more practitioners.



■ **Figure 1** The chosen scope of the working group.

■ **Figure 2** The general pipeline of the detection segment in network defense considered in the designed framework.

5.4.2 Scope & Methodology

To define the scope of this analysis, we refer to the NIST Cybersecurity Framework (CSF) 1.1 [3], developed by the National Institute of Standards and Technology (NIST) in the United States. This framework provides guidance for organizations to manage and improve their cybersecurity resilience. The core of the framework consists of five functions, each representing a key aspect of cybersecurity (Figure 1):

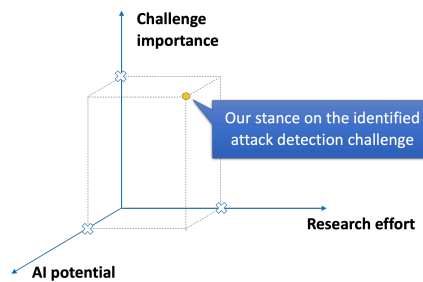
- **Identify:** Understand and manage cybersecurity risks to systems, assets and data.
- **Protect:** Develop safeguards to ensure the delivery of critical infrastructure services.
- **Detect:** Implement activities to identify the occurrence of a cybersecurity event.
- **Respond:** Take action regarding a detected cybersecurity event.
- **Recover:** Restore and maintain critical infrastructure services in the aftermath of a cybersecurity incident.

In this report, we focus our investigation on the **Detect** function as the one receiving primary academic attention in the field of network defense and yet lacking a comprehensive view in connection to wider real-world needs. We further break down the detection segment into concrete cybersecurity goals and position them along the general detection pipeline – its pre-detection, detection, and post-detection stages, as depicted in Figure 2. This breakdown enables us to formulate a (non-exhaustive) list of 18 network defense challenges based on what we consider to be crucial cybersecurity activities posing day-to-day challenges for practitioners. These challenges constitute the body of our analysis, where we assess the potential of AI to assist in solving them.

5.4.3 Assessment Framework

The central idea driving our work was the assessment of how relevant, meaningful or feasible the application of AI may be in view of the critical needs in network defense. The resulting structured framework may serve as a guide for researchers, aiding them in determining their research focus and prioritizing efforts effectively, particularly emphasizing challenges that have higher AI potential, in contrast to more elusive goals.

We utilized a three-fold evaluation to comprehensively assess each challenge – specific issue or task within the detection stage of the network defense pipeline that is under consideration. The three key rating criteria we defined are *Importance*, the *AI Potential*, and *Research*



■ **Figure 3** Illustration of an assessment of a given network defense challenge and its relevant properties on the relative scale.

Effort, which are depicted in Figure 3. Below we provide a concise definition for each, along with insights into the main intuition and some underlying aspects:

1. **Importance:** This criterion captures the significance of the challenge, emphasizing its role in the detection workflow. The evaluation is grounded in the practitioners' perspective to ensure practical relevance (which is currently limited to the composition of the working group and is likely to be refined in the follow-up work).
2. **AI Potential:** The assessment of the working group regarding the potential success and significance of data-driven AI methods (machine learning or deep learning) in application to a given challenge. This category encompasses three aspects, which in combination form the total AI potential:
 - a. **Applicability:** Examining whether the challenge can be formulated in a data-driven manner for using AI algorithms. Namely, this reflects whether the nature of the challenge lends itself well to the application of AI techniques that rely on data analysis, pattern recognition, and algorithmic decision-making. Problems suitable for a data-driven approach are those where historical information holds valuable clues about what might happen in the future: a much-desired but often limited property in cybersecurity.
 - b. **Success Likelihood:** Assessing the probability of solving the problem through a data-driven approach. Even if the problem can be formulated in a data-driven way, there may be serious complications in solving it. For instance, in the case of non-anticipated distributional shifts (due to the lack of representative data or dynamic changes in data at deployment), the trained AI model may lose its relevance too fast.
 - c. **Impact:** Gauging the potential impact and effectiveness of leveraging AI to address the identified challenge. This pertains to the expected performance of an AI solution, assuming its applicability and high success likelihood, in comparison to more traditional approaches that do not rely on intelligent automation. The estimated impact may also be influenced by, for instance, the associated costs which may or may not justify the additional complexity of using AI for the task. Interpretability, maintainability and other operational considerations behind an AI-powered solution may also greatly influence its expected impact.
3. **Research Effort:** This category is orthogonal to AI Potential in the sense that it covers the research-related side of addressing the challenges, meaning the effort required to conduct a research study that reliably investigates application of AI to a given task. In our understanding, the Research Effort criterion roughly encompasses two components:
 - a. **Existence of Relevant Data:** This aspect considers whether there is ground-truth data within the operational context that holds significance for the challenge at hand. Having access to complete relevant data of high volume and precision is crucial for

training AI algorithms. If the necessary data is readily available within the existing network defense workflows, it reduces the effort required in sourcing and preparing data for AI applications. Note that the current public availability of such corresponding data is out of scope for this analysis (although acquiring it is a significant challenge of its own for the research domain).

- b. **Algorithmic Difficulty:** This dimension reflects the technical complexity associated with developing AI algorithms tailored for the specific challenge. Some challenges might demand sophisticated algorithms due to their intricate nature (for instance, those that model multi-dimensional time-series or graph-structured data), while others may be more straightforward. Challenges with higher algorithmic difficulty may necessitate more advanced AI research and development efforts.

This comprehensive framework aims to provide researchers with a holistic perspective on the challenges in network defense, enabling them to prioritize efforts based on the combined assessment of the defined properties. The deliberate design ensures, on the one hand, that researchers can identify not only what challenges are critical but also where the application of AI is most feasible and impactful. On the other hand, the framework may pragmatically illuminate the opportunities for “low-hanging fruit”, allowing researchers to address more urgent but less complex needs with a higher likelihood of success, whilst having the assurance of relevance of their study for network defense practitioners.

5.4.4 The Current State of the Framework

Here we report on the main outcome of the working group, essentially presenting the snapshot of our understanding of the potential of AI-powered network defenses in the given moment (Table 5). The current state of analysis involves 18 ranked challenges derived from in-depth discussions among academic and industry researchers composing the working group, utilizing expertise in network defense and applied AI. Moving forward, this analysis would be refined to incorporate the perspectives of more security practitioners, offering a more holistic and practically relevant understanding of the challenges at hand. Crucially, this analysis needs to be strengthened with a thorough literature review to incorporate the most recent research advancements in the area, in order to more precisely evaluate the alignment of the current research trends with the highest needs in network defense.

■ **Table 5** The current state of the assessment framework developed to analyze the potential of AI-powered network defense research directions. These insights emerged from discussions within the working group and are subject to further refinement through an extensive literature review and polling among a larger, more diverse sample of network security practitioners and applied AI experts.

PRE-DETECTION CHALLENGES			
1. Merging, correlating, and comparing shared indicators of compromise (IoCs). A shared event is an attack that a team is investigating; there are 100-200k events per day (which are not unique). Several teams may be investigating the same attack, these may be shared simultaneously, but are not automatically merged.	Importance: AI Potential: Effort:	Medium Medium/High Low	This activity is non-trivial and demands a lot of human effort. Today certain sources are trusted, but not everything is merged together. Even for humans, merging IoCs in a consistent manner is difficult. AI could scale up the process and improve on human effort. Example: link scarce pieces of information in natural language to other information (labels).
2. Creating new detection rules from IOCs with regard to the diversity and number of future attacks.	Importance: AI Potential: Effort:	Medium/High Medium Low	Scaling and expediting this process are paramount, and Large Language Models (LLMs) show promise in this regard, although their usage is complex and challenging. It could be possible to train multiple LLMs (e.g., one for creating the rule and one for checking the rule quality or language). The level of involvement may range from creating a co-pilot for writing rules to simply having a human in the loop.
3.1. Determining the data quality of indicators for Threat Intelligence databases.	Importance: AI Potential: Effort:	Medium/High Low High	This can potentially be inferred based on the reputation and reliability of sources.
3.2. Ranking the importance of indicators for Threat Intelligence databases.	Importance: AI Potential: Effort:	High Medium High	Given the complexity of merging knowledge from different fields (e.g., threat landscape, infrastructure) and mapping such knowledge to indicators, future AI systems may offer promising solutions.
4. Configuration of existing detection tools, removing dependencies on the administrators.	Importance: AI Potential: Effort:	Medium Medium High	Automating the deployment of known hardening measures could significantly optimize the work of security operators. AI may be used to assist in setting parameters for known defenses tailored to a specific environment and objectives.
5. Setting IDS constraints (e.g., workload restriction in the case of a high rules set to prevent overload and exceeding resource limits).	Importance: AI Potential: Effort:	Medium High Low/Medium	For a human expert, predicting workload before deploying the system is challenging. Can AI estimate resource demands?
6. Optimization (clean-up) of the detection rule-set.	Importance: AI Potential: Effort:	Medium High Medium	Routine maintenance of the rule-set commonly involves manual analysis to discard unused rules and merge rules targeting the same pattern, all to optimize workload and the comprehension of alert generation. Intelligent automation seems feasible.
DETECTION CHALLENGES			
7. Detecting variants of known attacks.	Importance: AI Potential: Effort:	Medium High Low/Medium	Given the availability of data of previous attacks, we consider the situation in which an AI system can recognize the similarities and detect the same patterns. This scenario is known to be very compelling for AI application in research contexts, but is not the most challenging one, as the traditional methods perform well.

8. Detecting earlier unknown attacks.	Importance: High AI Potential: Low Effort: High	This is inherently challenging for AI methods as they cannot rely on the past to precisely recognize novel future threats within the context of network defense. Throughout the seminar, concerns have been repeatedly expressed about the perceived inadequacy of AI for detecting unknown attacks (e.g., zero days).
9. Generating alert descriptions (e.g., to enhance SOC analyst dashboards).	Importance: High AI Potential: High Effort: Low	This opportunity explicitly recognizes the power of AI in interfacing with human experts. The quality of generated textual descriptions depends on the comprehensiveness of alerts, which can be high for traditional methods.
10. Tuning and maintaining empirical thresholds for monitoring.	Importance: Low/Medium AI Potential: Low/Medium Effort: Unknown	This is a defining component of precise detection, yet targeted empirical approaches are limited. In our discussion, assessing potential and effort was challenging due to a lack of information on the prevalence of anomaly detection in practice.
11. Use Tactics, Techniques and Procedures (TTPs) as IoCs to actively search for malicious activities.	Importance: Medium/High AI Potential: High Effort: High	The capability of using kill chains instead of simplistic IoCs will allow the detection of a broad range of attacks sharing similar behavior (see point 7). Recognizing kill chains demands the correlation of complex events over time.
12. Attack attribution (e.g., linking alerts to attacker groups and detecting attacks with matching behavioural patterns).	Importance: High AI Potential: High Effort: High	Advanced persistent threats (APTs) are mostly obtained from threat intelligence for close monitoring. APT-related incidents have high complexity and priority, and the detection and decision-making process are often highly specific to APTs. The amount of available information and the willingness to extend analysis beyond a few days of activities requires and may highly benefit from AI systems.
13. Run-time optimization of the rule-set.	Importance: Low/Medium AI Potential: Low Effort: High	Discussed in the context of the difficulty in foreseeing resource consumption of security tools as well as conditions of the target infrastructure (e.g., peak in traffic and heavy computational loads).
POST-DETECTION CHALLENGES		
14. Alert fatigue as for dealing with all the alerts produced by the detection tools (e.g., how do we prioritize? How do we correlate across various sources? How do we diminish false positives?)	Importance: High AI Potential: High Effort: Medium/High	AI can learn from human operators how to prioritize alerts and make suggestions (related to 3.2). It might be challenging to obtain enough labeled data and integrate expert knowledge. An additional challenge is that prioritization does not consider the risks (e.g., important devices or other elements of the target environment).
15. Risk assessment of alerts to determine the severity of the case.	Importance: High AI Potential: Low Effort: Unknown	Risk assessments imply having information about the infrastructure, and it is currently unclear how to represent this information to an AI system along with other expert knowledge.

16. Proposing countermeasures to mitigate known attack.	Importance: High AI Potential: High Effort: Medium/High	AI may provide a collection of suggestions to incident handlers based on information about the incident. The training data for the AI assistant may be composed of playbooks or historical incidents across organizations and how they were handled in the past. The precision of these suggestions might vary: it may be relatively easy to provide relevant general insights, while learning to produce actual custom countermeasures may require more research effort.
17. Labeling adversary's activities (e.g., alerts being triggered by the security tools) according to the MITRE ATT&CK [1] knowledge base.	Importance: Medium AI Potential: High Effort: Low	This is crucial for all investigations upon detection and for sharing information with other teams. This step goes beyond incident handling. Extracting new attack strategies might be an outcome of this activity, where AI can significantly replace human effort in formulating the description of the attack. Additionally, automating the interpretation of a given attack scenario according to the MITRE ATT&CK knowledge base is valuable, as not all human experts know all techniques to be able to recognize them.
18. Representing and normalizing all data related to detection in a structured and comprehensive form for human analysts.	Importance: Medium/High AI Potential: High Effort: Low/Medium	One example of a hardly interpretable data structure is the graph-based indicators STIX [2] used to exchange cyber threat intelligence. Commercial vendors write reports about attacks in a complex expert-oriented language, which often obstructs timely and effective handling of network threats. Utilizing AI to format detection-related data in a human-comprehensible manner appears very promising.

5.4.5 Conclusion

The aim of this discussion was to provide a framework for future investigations in AI-powered network defense, leveraging the latest advancements in AI while fostering a more comprehensive understanding of the challenges faced by practitioners. The distinguishing characteristic of the framework is the choice to move beyond the conventional realm of AI-based intrusion detection and explore areas earlier in and further along the pipeline. Our systematic approach facilitates a wide examination of the problem space from both the network security practitioner and research perspectives. It covers practical feasibility, potential impact of applying AI, and the research effort required to address the identified important challenges. By combining the expertise of network security and AI researchers, the framework and its future extensions not only offer a nuanced understanding of challenges but also foster a synergistic environment where innovative solutions can be explored while staying rooted in reality.

References

- 1 Blake E Strom, Andy Applebaum, Doug P Miller, Kathryn C Nickels, Adam G Pennington, and Cody B Thomas. MITRE ATT&CK: Design and philosophy. In *Technical report*. The MITRE Corporation, 2018.

- 2 Structured Threat Information Expression (STIX). <https://oasis-open.github.io/cti-documentation/stix/intro.html>. Accessed: 2023-10-26.
- 3 The Cybersecurity Framework 1.1 NIST. <https://www.nist.gov/cyberframework>. Accessed: 2023-10-26.

6 World Café and Outlook

The format of the World Café is already described in Section 1. As written there, people split into small groups at individual tables where each table discussed a pre-defined question for 20 minutes before participants moved on in random permutation to another table. The questions asked were as follows:

1. In which of these fields is it most important to make research progress and why: “Security for AI”, “AI-based attacks”, or “AI for Security”?
2. What steps should we take to keep the activities in our group alive beyond the end of this seminar?
3. What is the title of a paper you would now want to write with some other seminar participants?
4. What is your one key take-away from the seminar?
5. For a future seminar proposal, what (network)security-related topic should we focus on? And whom would we have to (additionally) invite to make it a success?

Due to the small group size and clearly articulated questions, the format proved very effective to collect and distill insights from all the participants, which would not have been possible in a plenary session.

At this point, we will only exemplify a few of the comments provided.

6.1 In which of these fields is it most important to make research progress and why: “Security for AI”, “AI-based attacks”, or “AI for Security”?

On this question, some people suggested that research on “secure AI for security” would be a desirable outcome, i.e., if AI and machine learning mechanisms are applied as a security mechanism, one definitely has to make sure that the mechanism is secure and cannot be attacked itself and that resilience, robustness and reliability would be key properties but also explainability. In this context, a pointer was given to the NIST AI Risk Management Framework¹.

Another emphasis was put on the datasets required for machine learning of security mechanisms. So given the growing importance of machine learning in security, provisioning of good training data becomes key and the research community should put more emphasis on availability of good quality, large, and ideally labeled datasets.

Last, when it comes to AI-based attacks, it seems there is currently a lack of knowledge how such attacks would change the game and to what extent such attacks may be more potent than today’s mostly manual attacks.

¹ see <https://www.nist.gov/itl/ai-risk-management-framework>

6.2 What is your one key take-away from the seminar?

Findings from this question overlapped partly with the previous question. Lack of datasets for ML training was mentioned here, too. Others noted that on most questions discussed there is a large agreement among participants, in particular also on the open questions that need addressing and where we yet know too little about.

This was a general sentiment shared by many: in many of the topics we are still at the beginning and need a lot of research to deepen our understanding, maybe with the exception of applying AI for obvious and well investigated tasks like intrusion detection.

One thing that did not come at a surprise: currently there is a strong focus or even hype on security for Large-Language Models (LLMs) but also on how LLMs can help security in various ways.

Besides those discussions, the World Café also contributed many ideas for future follow-up seminars, topics for joint paper initiatives, and the desire to stay in contact and continue discussions online.

And we also came to the conclusion that LLMs could also help come up with catchy paper titles ;-).

Participants

- Ilies Benhabbour
KAUST – Thuwal, SA
- Sebastian Böhm
ZITiS – München, DE
- Christian Bungartz
Universität Bonn, DE
- Georg Carle
TU München – Garching, DE
- Marco Caselli
Siemens – München, DE
- Hervé Debar
Télécom SudParis, FR
- Sven Dietrich
City University of New York, US
- Daniel Fraunholz
ZITiS – München, DE
- Artur Herrmann
Universität Ulm, DE
- Peter Herrmann
NTNU – Trondheim, NO
- Marko Jahnke
BSI – Bonn, DE
- Frank Kargl
Universität Ulm, DE
- Stephan Kleber
Universität Ulm, DE
- Hartmut König
ZITiS – München, DE
- Jan Kohlrausch
DFN-CERT Services GmbH, DE
- Nicolas Kourtellis
Telefónica Research –
Barcelona, ES
- Chethan Krishnamurthy
Ramanaik
Universität der Bundeswehr –
München, DE
- Pavel Laskov
Universität Liechtenstein, LI
- Emil C. Lupu
Imperial College London, GB
- Michael Meier
Universität Bonn, DE
- Andreas Mitschele-Thiel
TU Ilmenau, DE
- Simin Nadjm-Tehrani
Linköping University, SE
- Eirini Ntoutsis
Universität der Bundeswehr
München, DE
- Andriy Panchenko
BTU Cottbus, DE
- Delphine Reinhardt
Universität Göttingen, DE
- Konrad Rieck
TU Berlin, DE
- Vera Rimmer
KU Leuven, BE
- Bettina Schnor
Universität Potsdam, DE
- Thomas Schreck
Hochschule München, DE
- Max Schrötter
Universität Potsdam, DE
- Robin Sommer
Corelight – Planegg, DE
- Jessica Steinberger
Hochschule Mannheim, DE



Edge-AI: Identifying Key Enablers in Edge Intelligence

Aaron Ding^{*1}, Eyal de Lara^{*2}, Shahram Dustdar^{*3}, Ella Peltonen^{*4},
and Tobias Meuser^{†5}

- 1 TU Delft, NL. aaron.ding@tudelft.nl
- 2 University of Toronto, CA. delara@cs.toronto.edu
- 3 TU Wien, AT. dustdar@dsg.tuwien.ac.at
- 4 University of Oulu, FI. Ella.Peltonen@oulu.fi
- 5 TU Darmstadt, DE. tobias.meuser@kom.tu-darmstadt.de

Abstract

Edge computing promises to decentralize cloud applications while providing more bandwidth and reducing latency. Based on the discussion of our first Dagstuhl Seminar and the continuation work that took place after the seminar, we continued our work on identified challenges that need to be further addressed within the community. These challenges included 1) large-scale deployment of the edge-cloud continuum, 2) energy optimization and sustainability of such large-scale AI/ML learning and modelling, and 3) trustworthiness, security, and ethical questions related to the whole continuum. In this seminar, we discussed the current state of Edge Intelligence and shaped a holistic view of its challenges and applications. The main concerns were 1) the assessment and applicability of Edge Intelligence solutions, 2) energy consumption and sustainability, and 3) the new trend of Large-Language Models.

Seminar October 22–25, 2023 – <http://www.dagstuhl.de/23432>

2012 ACM Subject Classification Computer systems organization → Distributed architectures; Computing methodologies → Artificial intelligence; Networks

Keywords and phrases cloud computing, edge computing, edge intelligence

Digital Object Identifier 10.4230/DagRep.13.10.130


1 Executive Summary

Aaron Ding (TU Delft, NL)

Eyal de Lara (University of Toronto, CA)

Schahram Dustdar (TU Wien, AT)

Ella Peltonen (University of Oulu, FI)

License  Creative Commons BY 4.0 International license

© Aaron Ding, Eyal de Lara, Shahram Dustdar, and Ella Peltonen

Research Area

Edge computing promises to decentralize cloud applications while providing more bandwidth and reducing latency. These promises are delivered by moving application-specific computations between the cloud, the data-producing devices, and the network infrastructure components at the edges of wireless and fixed networks. Meanwhile, the current Artificial Intelligence (AI) and Machine Learning (ML) methods assume computations are conducted in a powerful computational infrastructure, such as data centres with ample computing and data storage resources available. To shed light on the fast-evolving domain that merges edge

* Editor / Organizer

† Editorial Assistant / Collector



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

Edge-AI: Identifying Key Enablers in Edge Intelligence, *Dagstuhl Reports*, Vol. 13, Issue 10, pp. 130–138

Editors: Eyal de Lara, Aaron Ding, Shahram Dustdar, and Ella Peltonen



DAGSTUHL REPORTS

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

computing and AI/ML, referred to as Edge AI, the recent Dagstuhl Seminar 21342 gathered community inputs from a diverse range of experts. The results of our first iteration of the seminar were reflected in the ACM SIGCOMM CCR publication, focusing on three different angles of Edge AI: future networking, cloud computing, and AI/ML needs.

Along with three identified driving areas of 5G beyond (or so-called 6G), future cloud, and evolved AI/ML, the advancement of different technologies and the growing business interests will take Edge AI forward regarding hardware, software, service models, and data governance. Starting from the current state of play driven by cellular, cloud, and AI/ML service providers, the roadmap reflects five general phases: scalable framework, trustworthy co-design, sustainable and energy-efficient deployment, equal accessibility, and pervasive intelligent infrastructure. As changes can always occur, the sequence depicted in the roadmap could be switched or combined. Nonetheless, this Edge AI roadmap reflects the combined effects of technology enablers and non-tech demands, such as the socioeconomic transformation of user behaviors, purchasing power, and business interests.

Despite its promise and potential, Edge AI still faces major challenges in large-scale deployment, including energy optimization, trustworthiness, security, privacy, and ethical issues. As an important goal of sustainability, the energy consumption of Edge AI needs to be optimized. Energy efficiency is crucial for Edge AI embedded infrastructures (e.g., roadside units, micro base stations) to sustainably support advanced autonomous driving and Extended Reality (XR) services in the years to come. Through the pipeline of data acquisition, transfer, computation, and storage, there exists the possibility for Edge AI to trade accuracy with reduced power and less time consumed. For instance, noisy inputs from numerous sensors can be selectively processed and transferred in order to save energy.

A set of applications would be satisfied with an ‘acceptable’ accuracy instead of exact and absolutely correct results. By introducing this new dimension of accuracy to the optimization design, energy efficiency can be further improved. Concerning trustworthiness, Edge AI benefits from its close proximity to the end devices. However, due to the distributed deployment with deep insights into a personal context, the safety and perceived trustworthiness of Edge AI services are raising concerns among the stakeholders (e.g., end-users, public sectors, ISP). To achieve trustworthy Edge AI, critical building blocks are needed, including verification and validation mechanisms that ensure transparency and explainability, especially in the training and deployment of Edge AI in decentralized, uncontrolled environments. The trustworthiness of Edge AI is a stepping stone to establishing appropriate governance and regulatory framework on which the promise of Edge AI can be built.

2 Table of Contents

Executive Summary

Aaron Ding, Eyal de Lara, Shahram Dustdar, and Ella Peltonen 130

Overview of Talks

Future of Communications: Why we need Edge AI and more
Susan Bayhan 133

Increasing AI Sustainability with Symbolic Data Representation on the Edge
Ivona Brandić 133

Edge Enabled Autonomous Driving and Mobility Services
Liam Pedersen 134

The economics of edge AI don't look great – or why edge computing may always
be the future
Henning Schulzrinne 134

Enabling data spaces: existing developments and challenges
Gürkan Solmaz 135

Working groups

Definition and Usecases of Edge AI
*Dewant Katare, Eyal de Lara, Aaron Ding, Shahram Dustdar, Tobias Meuser,
Shishir Girishkumar Patil, and Ella Peltonen* 135

Ecosystem: Software and Hardware Problems
*Dewant Katare, Eyal de Lara, Aaron Ding, Shahram Dustdar, Nitinder Mohan,
Shishir Girishkumar Patil, and Ella Peltonen* 136

Measure what matters
*Dewant Katare, Eyal de Lara, Aaron Ding, Shahram Dustdar, Tobias Meuser,
Shishir Girishkumar Patil, and Ella Peltonen* 137

Panel discussions

What's Next after Edge AI
Henning Schulzrinne, Shishir Girishkumar Patil, Liam Pedersen, and Jan Rellermeyer 137

Participants 138

3 Overview of Talks

3.1 Future of Communications: Why we need Edge AI and more

Susan Bayhan (University of Twente – Enschede, NL)

License © Creative Commons BY 4.0 International license
© Susan Bayhan

In the era of the climate crisis, it is essential to optimize the operation of many sectors, from agriculture to health with the help of smart sensing and data analytics. While the data is essential for better understanding and consequently for better decision-making, it has to be moved, stored, computed, secured, and interpreted, all steps bringing their own challenges. According to the Decadal Plan by Semiconductors (<https://www.src.org/about/decadal-plan/>), there are five seismic shifts that need to be considered for the future of ICT: 1) Analog hardware for generating smarter world-machine interfaces that can sense, perceive, and reason, 2. Radically new memory and storage solutions. 3. New research directions to close the gap between communication capacity vs. data-generation rates. 4. security challenges in highly interconnected systems and AI, and 5. New computing paradigms for higher energy efficiency. The talk argues that edge AI can help with some of these challenges, such as by decreasing the need to communicate huge volumes of data with the cloud, however for sustainable communications systems of the future, more is needed at all levels of the system stack as well as design, production, and use stages of all infrastructures.

3.2 Increasing AI Sustainability with Symbolic Data Representation on the Edge

Ivona Brandic (Technische Universität Wien, AT)

License © Creative Commons BY 4.0 International license
© Ivona Brandic

Joint work of Daniel Hofstätter, Shashikant Ilager, Ivan Lujic, Ivona Brandic

Main reference Daniel Hofstätter, Shashikant Ilager, Ivan Lujic, Ivona Brandic: “SymED: Adaptive and Online Symbolic Representation of Data on the Edge”, in Proc. of the Euro-Par 2023: Parallel Processing – 29th International Conference on Parallel and Distributed Computing, Limassol, Cyprus, August 28 – September 1, 2023, Proceedings, Lecture Notes in Computer Science, Vol. 14100, pp. 411–425, Springer, 2023.

URL https://doi.org/10.1007/978-3-031-39698-4_28

The Edge Computing model is beneficial for analyzing and processing data generated by the Internet of Things (IoT) in the proximity to its source. However, the transfer, storage, and processing of this rapidly increasing data volume is challenging on edge devices with limited resources. Symbolic Representation (SR) algorithms show promise in reducing data size by transforming raw data into symbols. They also enable direct data analytics on symbols, such as anomaly detection and trend prediction, which is advantageous for many edge applications. Nonetheless, the current SR algorithms are mainly centralized and operate offline with batch data, making them unsuitable for real-time scenarios. In this talk, SymED – Symbolic Edge Data representation method is introduced. This method is an online, adaptive, and distributed approach to symbolic data representation at the edge. SymED utilizes Adaptive Brownian Bridge-based Aggregation (ABBA) and involves low-powered IoT devices sending and performing the low-cost initial data compression while the more capable edge devices perform the more demanding symbolic conversion.

3.3 Edge Enabled Autonomous Driving and Mobility Services


Liam Pedersen (Nissan North America – Santa Clara, US)

License  Creative Commons BY 4.0 International license
© Liam Pedersen

Connectivity, smart infrastructure and autonomous driving software hosted on edge computing will transform the ways in which we use and drive cars, the freeway system and the electrical grid. In this talk, examples of smart cloud-based services are presented for managing freeway congestion, low-cost autonomous valet parking and EV integration with the grid.

3.4 The economics of edge AI don't look great – or why edge computing may always be the future

Henning Schulzrinne (Columbia University – New York, US)

License  Creative Commons BY 4.0 International license
© Henning Schulzrinne

Edge computing for AI encompasses a wide variety of architectures. “Easy” cases include embedded systems, e.g., in most modern vehicles, or dedicated processing in sensors, e.g., cameras performing image segmentation and basic object recognition. The harder, interesting architectures provide generic computing capabilities to paying customers, i.e., cloud-like arrangements.

Even for edge AI systems, the edge computing element will likely not store large volumes of data as the computing need may be transient. Thus, the latency advantages of edge systems may be reduced since the edge system will need to contact the regional or trans-regional cloud to query databases or access API-based microservices functionality. This split functionality may negate the latency advantages of edge computing for real-world systems and needs to be part of any system evaluation.

Edge AI systems are further challenged by economic considerations. The cost of computing can be divided into capital costs, typically the initial investment into computing and layer-0 infrastructure such as data centers, and the ongoing operational costs. AI-suitable GPUs may have shorter effective lifespans than other CPUs, but even server CPUs are typically only used for five years before being retired. Thus, edge AI systems with low utilization may increase the amortized costs on a per-task basis. For the same probability of task rejection, smaller edge systems need to provide more computational reserves by standard Erlang-C considerations.

Electricity makes up about 60-70% of the operational cost. Edge AI can only be competitive if the energy costs are similar to those in large-scale data centers, i.e. if Edge AI can draw on cheap renewable energy. (However, their intermittency may increase the amortized cost of capital, as noted above.) For example, grid electricity in New York costs roughly \$0.21/kWh, while data centers aim for \$0.05/kWh. As of June 2021, the levelized cost of energy (LCOE) for utility-scale photovoltaic systems ranges from \$0.03 to \$0.05 and is thus competitive.

Other operational costs, not further discussed here, include security costs if edge systems impose a security premium, development and DevOps costs, as well as failure risk trade-offs.

Thus, edge computing for AI may be roughly divided into “cheap” and “dependable.” The former may only offer batch-style processing with intermittent availability, while the latter is willing to tolerate higher costs than traditional cloud computing.


Trust for edge AI deserves more careful consideration. Unless the entity running the computation owns and operates the hardware, they still have to trust the edge computing provider, which may well be smaller than typical cloud providers.

As cloud services now offer a wide range of computing hardware, from traditional i386 to ARM and GPU-based processors as well as specialized ML engines, edge computing will struggle to compete, given the smaller rack count.

Smaller edge installations may also find it more difficult to provide physical security and uninterrupted power. In summary, given the uncertainties of economic competitiveness, security, and reliability, edge AI requires careful feasibility analysis, where non-technical considerations may outweigh technical feasibility or advantages. Resource discovery needs to take cost and reliability needs into account. Resources may well be mediated and aggregated to relieve application developers from maintaining and creating relationships with hundreds of edge computing service providers. To facilitate computational roaming, systems have to provide appropriate AAA capabilities.

3.5 Enabling data spaces: existing developments and challenges

Gürkan Solmaz (NEC Laboratories Europe – Heidelberg, DE)


License  Creative Commons BY 4.0 International license
© Gürkan Solmaz

This talk at Dagstuhl includes a short introduction to the concept of Data Spaces based on the recent developments of IDSA, Gaia-X, and FIWARE, as well as the challenges of data interoperability and data value. The recent work from the Data Ecosystems and Standards (DES) group at NEC Laboratories Europe focuses on solving those challenges using real-world sensor data and geographic data from the case studies of the City Liveability Index (CLI) of SALTED project, Smart Campus Murcia, and Humanitarian Landmine project. The data enrichment and contextualization platform is utilized in the case studies through technologies such as TrioNet, FIWARE Scorpio Broker, FIWARE FogFlow and AI/machine learning for predictions and transfer learning.

4 Working groups

4.1 Definition and Usecases of Edge AI

Dewant Katare (TU Delft, NL), Eyal de Lara (University of Toronto, CA), Aaron Ding (TU Delft, NL), Schahram Dustdar (TU Wien, AT), Tobias Meuser (TU Darmstadt, DE), Shishir Girishkumar Patil (University of California – Berkeley, US), and Ella Peltonen (University of Oulu, FI)

License  Creative Commons BY 4.0 International license
© Dewant Katare, Eyal de Lara, Aaron Ding, Schahram Dustdar, Tobias Meuser, Shishir Girishkumar Patil, and Ella Peltonen


In this working group, the definition and use cases of Edge AI were discussed. The discussions highlighted the complexity of clearly defining edge, highlighting some key points:

- **Application-specific Definition:** Depending on the considered application, the definition of Edge varies drastically, ranging from small data centers close to the user to end devices under the control of the user.

- **Business Models:** The impact of business models on Edge AI’s popularity and development is acknowledged. Cloud-based models, especially those driven by advertisement, have been given more attention, as the management is much easier and availability is much higher.
- **Real-world Examples and Use-cases:** Examples from Amazon, like Greengrass and Sagemaker, demonstrate Edge AI applications, but these are not always orchestrated for end-users. Edge solutions often complement cloud solutions, providing benefits like reduced latency and enhanced privacy without replacing cloud infrastructure.
- **Cloud Definition and Academic Consensus:** Similar to Edge AI, the definition of “Cloud” also varies widely, ranging from proximity (in milliseconds) to computing resources. There is still no universally agreed-upon definition of Edge in academia, leading to inconsistencies.
- **Perspective-Dependent Success:** The success or failure of Edge AI is contingent on the specific definition of Edge used and the viewpoint from which it is considered.

4.2 Ecosystem: Software and Hardware Problems

Dewant Katare (TU Delft, NL), Eyal de Lara (University of Toronto, CA), Aaron Ding (TU Delft, NL), Schahram Dustdar (TU Wien, AT), Nitinder Mohan (TU München, DE), Shishir Girishkumar Patil (University of California – Berkeley, US), and Ella Peltonen (University of Oulu, FI)

License  Creative Commons BY 4.0 International license
 © Dewant Katare, Eyal de Lara, Aaron Ding, Schahram Dustdar, Nitinder Mohan, Shishir Girishkumar Patil, and Ella Peltonen

One of the major problems discussed in this working group is the issues associated with trust in Edge Intelligence. While the group acknowledged that edge computing is promising for handling data in public spaces, trust remains a critical issue. This is evident in the disagreement among unions over video cameras, indicating that edge computing alone is not sufficient to establish trust. One issue is the possibility of reconfiguring and reprogramming edge devices such that any functionality can be added at any point. While privacy or trust is not the sole argument for the usage of edge intelligence, a key argument is the volume of data that needs to be processed or transmitted. Edge computing allows data processing closer to where it is generated, reducing the need for data transmission and potentially enhancing privacy and efficiency.

Overall, the group highlighted the potential of edge computing in various domains, emphasizing its role in data volume management, privacy preservation, and compliance with legal and regulatory constraints. However, trust needs to be built and programmable devices might need to be regulated to prevent arbitrary reconfiguration that ultimately harms trust in these devices.

4.3 Measure what matters

Dewant Katare (TU Delft, NL), Eyal de Lara (University of Toronto, CA), Aaron Ding (TU Delft, NL), Schahram Dustdar (TU Wien, AT), Tobias Meuser (TU Darmstadt, DE), Shishir Girishkumar Patil (University of California – Berkeley, US), and Ella Peltonen (University of Oulu, FI)

License © Creative Commons BY 4.0 International license
 © Dewant Katare, Eyal de Lara, Aaron Ding, Schahram Dustdar, Tobias Meuser, Shishir Girishkumar Patil, and Ella Peltonen

This working group discussed the challenges and considerations in measuring and managing energy consumption in Edge Intelligence, with a focus on data centers and end devices.

There have been major discussions on the energy consumption of Edge Intelligence. The exponential growth in energy consumption for IT is widely criticized and revised. This is associated with inaccuracies in technological forecasts, such as those by MIT for self-driving technology. The concept of distinguishing between fungible and non-fungible energy is introduced, suggesting that sometimes it might be better to turn off an energy source or consume it, depending on its nature. There is optimism with new benchmarks emerging, which help in understanding the carbon footprint of various technologies. However, there is a need to precisely define and break down what is being measured in terms of energy consumption. As this varies across industries, the lack of clear benchmarks or standards is a problem. Without these proper benchmarks, the discussions are not scientifically robust.

In summary, the discussions emphasise the complexities in measuring and managing energy and carbon footprint in Edge Intelligence, emphasizing the need for precise benchmarks, consideration of regional differences, and the importance of trend analysis in the face of challenging measurements.

5 Panel discussions

5.1 What's Next after Edge AI

Henning Schulzrinne (Columbia University – New York, US), Shishir Girishkumar Patil (University of California – Berkeley, US), Liam Pedersen (Nissan North America – Santa Clara, US), and Jan Rellermeyer (Leibniz Universität Hannover, DE)

License © Creative Commons BY 4.0 International license
 © Henning Schulzrinne, Shishir Girishkumar Patil, Liam Pedersen, and Jan Rellermeyer

In this panel, the future of Edge AI has been discussed. One discussion point was the vagueness of the term Edge in the research community, which sometimes leads to confusion among researchers. However, this does not limit the success of some applications of Edge and Edge Intelligence. A major point in these discussions has been the role of Large-Language-Models (LLMs) in Edge devices. This also led to the discussion of how LLMs could be used as Operating Systems and their future role in research.

Participants

- Atakan Aral
Universität Wien, AT
- Susan Bayhan
University of Twente –
Enschede, NL
- Christian Becker
Universität Stuttgart, DE
- Monowar Bhuyan
University of Umeå, SE
- Ivona Brandic
Technische Universität Wien, AT
- Eyal de Lara
University of Toronto, CA
- Kemal A. Delic
The Open University –
Milton Keynes, GB
- Aaron Ding
TU Delft, NL
- Schahram Dustdar
TU Wien, AT
- Janick Edinger
Universität Hamburg, DE
- James Gross
KTH Royal Institute of
Technology – Kista, SE
- Volker Hilt
Nokia Bell Labs – Stuttgart, DE
- Dewant Katare
TU Delft, NL
- Lauri Lovén
University of Oulu, FI
- Tobias Meuser
TU Darmstadt, DE
- Nitinder Mohan
TU München, DE
- Shishir Girishkumar Patil
University of California –
Berkeley, US
- Liam Pedersen
Nissan North America –
Santa Clara, US
- Andy D. Pimentel
University of Amsterdam, NL
- Jan Rellermeier
Leibniz Universität
Hannover, DE
- Tina Rezaei
University of Twente –
Enschede, NL
- Etienne Rivière
UC Louvain, BE
- Henning Schulzrinne
Columbia University –
New York, US
- Stephan Sigg
Aalto University, FI
- Pieter Simoens
Ghent University, BE
- Gürkan Solmaz
NEC Laboratories Europe –
Heidelberg, DE
- Michael Welzl
University of Oslo, NO
- Lars Wolf
TU Braunschweig, DE



Ensuring the Reliability and Robustness of Database Management Systems

Hannes Mühleisen*¹, Danica Porobic*², and Manuel Rigger*³

- 1 CWI – Amsterdam, NL. hannes.muehleisen@cwi.nl
- 2 Oracle Switzerland – Zürich, CH. danica.porobic@oracle.com
- 3 National University of Singapore, SG. rigger@nus.edu.sg

Abstract

The goal of this Dagstuhl Seminar was to bring together researchers and practitioners interested and experienced in database systems and the reliability aspects thereof. It is a continuation of a previous seminar on this topic, which had built an initial understanding of the challenges and areas of future work. In this edition of the seminar, we aimed for a tangible outcome of the seminar by writing a manuscript documenting the (1) best practices in ensuring database systems' reliability, (2) the state of the art on this topic in research, as well as (3) open challenges, which might also serve as the cornerstone of writing a book on this topic presented in this report. We achieved this by forming four primary working groups during the seminar, namely (1) on the automated testing aspects concerning analytical components of database systems, (2) benchmarking, (3) reliability for transaction and concurrency aspects, as well as (4) query languages and debugging. The report contains four sections presenting the results of these working groups. Some of these working groups and individuals plan to further refine their work and discussion outcomes, aiming to submit them to upcoming venues.

Seminar October 29 – November 3, 2023 – <https://www.dagstuhl.de/23441>

2012 ACM Subject Classification Information systems → Database management system engines; Software and its engineering → Software verification and validation

Keywords and phrases database benchmarking, database reliability, database testing

Digital Object Identifier 10.4230/DagRep.13.10.139

1 Executive Summary

Manuel Rigger (National University of Singapore, SG)

Hannes Mühleisen (CWI – Amsterdam, NL)

Danica Porobic (Oracle Switzerland – Zürich, CH)

License  Creative Commons BY 4.0 International license
© Manuel Rigger, Hannes Mühleisen, and Danica Porobic

Database systems are an essential component of most software systems. It is crucial that they function correctly and are efficient. Achieving this is difficult, given that growing demands require increasingly sophisticated systems and adapting them to new hardware platforms. Building on the success of the last seminar, the goal of this Dagstuhl Seminar was to advance database systems reliability by bringing together both practitioners as well as researchers working in this domain. We discussed current practices, approaches, and open challenges in manual and automated correctness and performance testing, isolation-level testing, benchmarking, database and query generation, query languages, as well as debugging.

* Editor / Organizer



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

Ensuring the Reliability and Robustness of Database Management Systems, *Dagstuhl Reports*, Vol. 13, Issue 10, pp. 139–181

Editors: Hannes Mühleisen, Danica Porobic, and Manuel Rigger



DAGSTUHL
REPORTS Dagstuhl Reports

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

Goals and Outcomes

As a concrete tangible outcome of the seminar, we aimed to write a manuscript on (1) best practices in ensuring database systems' reliability, (2) the state of the art on this topic in research, as well as (3) open challenges, which might also serve as the cornerstone of writing a book on this topic. We believe we have achieved this goal, with the reports of the four working groups presented in the subsequent chapters. The majority of the contents were composed during the seminar and reflect the experiences and opinions of various attendees. Thus, some of the contents might be incomplete, contradictory, or redundant.

Attendee Mix

The seminar's attendees were 36 internationally renowned researchers as well as high-profile practitioners whose work is closely related to the topic of this seminar. Given that the last years have seen an increased focus on research related to database system reliability, a larger fraction of researchers had significant expertise in this topic as compared to the first edition of the seminar, whose attendees mix had a broader, but less specific expertise. Overall, 25% of the second seminar's attendees joined also the first seminar – note that the first edition was a small seminar with a hybrid format that had only 12 in-person attendees, of which 67% joined the second seminar. A large number of attendees from Europe accepted the seminar invitation, constituting 56% of the attendees. The remaining attendees joined from the US (27%), Asia (14%), and one attendee from South Africa. 36% of the attendees were from industry, which is higher compared to the last seminar, which had only 23% attendees from industry. By increasing their percentage, we aimed to gain more insights into the current practices in the industry. Unfortunately, one of the co-organizers, Alexander Böhm was unable to attend the seminar; we want to thank him for his contributions in organizing it.

Seminar Structure

The seminar lasted for five days and accepted only in-person attendees. We started the first day with a round of introductions, where each attendee could introduce themselves and their interests in a five-minute presentation. The majority of the remaining days were filled with discussions by the individual working groups – we deliberately avoided having presentations on past work to keep the discussion centered on open problems and future directions. While we identified the main topics of interest that helped form the working groups on the evening of the first day, attendees could freely move between these groups and regroup. We reconvened in the larger group once or twice per day, to present and discuss the results of the working groups. The four working groups listed in this report reflect the working groups at the end of the seminar; for example, the first working group on automated testing split into multiple subgroups that discussed various topics during the course of the seminar. In addition to the working groups, we had two tutorials after dinner, as well as an excursion on Wednesday afternoon.

Future Plans

The individual working groups' reports are a concrete outcome of this edition's seminar that future work will build upon. First, working groups indicated their plans to refine their reports as well as follow up on the work to present the results to a broader audience aiming to encourage future work on their topics. Second, we expect individual research groups to take on the challenges identified in the seminar and propose solutions to them. Third,

we plan to propose a third edition of the seminar to discuss the progress on tackling the identified problems and expand the scope in terms of reliability. In this seminar, we also want to fill gaps in terms of topics and attendee mix; for example, in this edition of the seminar, we focused our discussions on relational database systems, omitting systems built on other models. As another example, we had no major discussions on the reliability of learned components of database systems.

2 Table of Contents

Executive Summary

Manuel Rigger, Hannes Mühleisen, and Danica Porobic 139

Overview of Talks

SQLancer Tutorial

Manuel Rigger 143

Informal Proofs of Correctness for Lock-free Algorithms

Russell Sears 143

Working groups

Working Group on Benchmarking

Lawrence Benson, Carsten Binnig, Federico Lorenzi, Danica Porobic, Tilmann Rabl, Anupam Sanghi, Russell Sears, and Pinar Tözün 144

Working Group on Transactions and Concurrency

Wensheng Dou, Adam Dickinson, Burcu Kulahcioglu Ozkan, Umang Mathur, Everett Maus, Stan Rosenberg, Gambhir Sankalp, Caleb Stanford, and Cheng Tan 148

Working Group on Query Languages and Debugging

Denis Hirn, Moritz Eyssen, Tim Fischer, Torsten Grust, Muhammad Ali Gulzar, Hannes Mühleisen, Thomas Neumann, and Mark Raasveldt 156

Working Group on Testing “Analytical” Components of Databases

Manuel Rigger, Jinsheng Ba, Ankush Desai, Adam Dickinson, Wensheng Dou, Stefania Dumbrava, Moritz Eyssen, Florian Gerlinghoff, Hong Hu, Zu-Ming Jiang, Marcel Kost, Everett Maus, Mark Raasveldt, Andrei Satarin, Thodoris Sotiropoulos, and Chengyu Zhang 160

Participants 181

3 Overview of Talks

3.1 SQLancer Tutorial

Manuel Rigger (National University of Singapore, SG)

License © Creative Commons BY 4.0 International license
© Manuel Rigger

This tutorial gave an introduction to SQLancer, which is a popular and widely used tool to automatically test database management systems. The tutorial assumed both a user perspective, as well as a developer one, while also highlighting limitations that could be addressed by future research.

3.2 Informal Proofs of Correctness for Lock-free Algorithms

Russell Sears (Crystal DB – San Francisco, US)

License © Creative Commons BY 4.0 International license
© Russell Sears

We held a tutorial on informal proofs of correctness for a simple, but powerful class of lock free algorithms. The work has been open sourced, and feedback from the seminar attendees improved the documentation and provided us with references to related academic work on software correctness and testing.

The library is available here:

<https://crates.io/crates/atomic-try-update>

The atomic-try-update library makes it easy to correctly implement your own lock free data structures. In addition to the base primitives, we provide a few example data structures that you can use directly, or that you can use as a base for your own application-specific algorithms.

Typical use cases for atomic_try_update include implementing state machines, building simple resource allocators, initializing systems in a deterministic way using “fake monotonicity”, accumulating state in stacks, and using the claim pattern to allow concurrent code to enqueue and then process them sequentially.

Unlike most lock free libraries, we make it easy to compose the above in a way that preserves linearizable semantics. For instance, you implement a lock free state machine that tallies votes as part of a two phase commit protocol, and then combine it with a stack. The resulting code would add information about each response to the stack and then process the result of the tally exactly once without resorting to additional synchronization such as mutexes or carefully ordered writes.

By “linearizable”, we mean that any schedule of execution of the algorithms built using atomic_try_update is equivalent to some single-threaded schedule, and that other code running in the system will agree on the order of execution of the requests. This is approximately equivalent to “strict serializability” from the database transaction literature. atomic_try_update provides semantics somewhere between those of a transaction processing system and those of a CPU register. We chose the term linearizable because it is more frequently used when discussing register semantics, and atomic_try_update is generally limited to double word (usually 128-bit) updates.

From a performance perspective, `atomic_try_update` works best when you can have many independent instances that each have low contention. For instance, using a single `atomic_try_update` instance to coordinate all reads in a system would likely create a concurrency bottleneck. Having one for each client connection probably would not. This means that you should stick to other, more specialized algorithms for things like top-level event queues and other high-contention singleton data structures in your system.

4 Working groups

4.1 Working Group on Benchmarking

Lawrence Benson (Hasso-Plattner-Institut, Universität Potsdam, DE), Carsten Binnig (TU Darmstadt, DE), Federico Lorenzi (TigerBeetle – Cape Town, ZA), Danica Porobic (Oracle Switzerland – Zürich, CH), Tilmann Rabl (Hasso-Plattner-Institut, Universität Potsdam, DE), Anupam Sanghi (IBM India – Bangalore, IN), Russell Sears (Crystal DB – San Francisco, US), and Pinar Tözün (IT University of Copenhagen, DK)

License © Creative Commons BY 4.0 International license
 © Lawrence Benson, Carsten Binnig, Federico Lorenzi, Danica Porobic, Tilmann Rabl, Anupam Sanghi, Russell Sears, and Pinar Tözün

Motivation

Standardized benchmarks are crucial to ensure a fair comparison across systems. Furthermore, they serve as a testing aid and help find bugs in systems.

In the database community, Transaction Processing and Performance Council (TPC) [43] has been standardizing benchmarks covering application domains from OLTP, OLAP, Big Data, IoT, AI ... Yahoo Cloud Serving Benchmark (YCSB) [16] has been quite popular for transaction processing, key-value stores, and the cloud. The Linked Data Benchmark Council (LDBC) specializes in graph analytics benchmarks [2].

While extremely valuable, these benchmarks all present a static scenario, where the workload is well-defined and known in advance. As a result, data management systems get fine-tuned for a particular benchmark before they are run with that benchmark. Therefore, the presented results often times do not reflect the behavior of these systems in, typically way less predictable and way more dynamic, real-world settings.

Furthermore, it takes time and effort to standardize a benchmark. With the fast-pace that the data-intensive systems and applications evolve today, it is difficult to generate a standardized benchmark to cover a variety of real-world use cases in a timely manner. This often times lead to complaints about the standardized benchmarks not being representative.

To address these issues, we would like to present a complementary approach to the current standardized benchmarking practices by introducing periodic themes and an element of surprise.

Surprise Benchmarking

The principles of the *surprise benchmarking* is similar to the principles of the ACM SIGMOD Programming Contest. More specifically, we would like to establish periodic (monthly, quarterly ...) benchmarking rounds, where a theme for the round will be announced ahead

of time such as “OLAP”, “key-value store”, “UDFs”, “resource-constrained environment”, etc. There may or may not be a description of the schema, data, a few expected queries, target hardware setup ... During the actual benchmark run, an element of surprise will be added such as ad-hoc queries, new table addition, data distribution changes ...

At each round, we will provide a baseline using a well-known data management system, e.g. Postgres. There will also be a different set of metrics to focus on at each round, in addition to the more traditional throughput and latency metrics, such as energy-efficiency, hardware utilization, monetary cost ...

On the one hand, the surprise benchmarking would be to eliminate the overly-tuned nature of current standardized benchmarking practices. On the other hand, it would help to accumulate a wider variety of standardized benchmarks over the years covering a more diverse set of data-intensive applications and their real-world deployments.

Surprise Categories

We categorize the element of surprise into three based on the intended impact on the system under test. We expect each benchmarking round to have a surprise covering each category. The level of surprises will systematically vary during the benchmark runs; i.e., having several runs with different % of surprise queries. However, how we vary the mix of surprises or whether we mix the surprises across different surprise categories may change round to round.

On-road

This type of surprise represent cases that are supposed to be meaningful for the systems under test. In other words, these are the queries or scenarios the system was designed to cover, and they shouldn't break the system. For example, ad-hoc queries similar to, but not the same, the TPC-H benchmark queries for a system specifically designed for OLAP workloads. Similarly, introducing data or access skew that mimic real-world data distributions as the surprise would fall under this category.

These types of surprises are also the ones that are the most suitable for auto-generation. As part of this work, we plan to develop / utilize the automatic query generation techniques as much as possible.

The goal here is to observe how the system behavior changes compared to fine-tuned standardized benchmarking scenarios.

Wrong-road

In contrast to the previous category, this one represents scenarios that are not the intended use of the system under test. They are a misuse of the system, but the intentions of the user are not harmful. For example, using a `SELECT *` to get all the data than putting the data through many UDFs instead of doing some data processing, such as filtering data, with SQL; or instead of using JSON support in a system, casting the data to string and performing regex operations on the string would be misaligned uses of a database system.

While not as suitable for automation as the above category, we will still investigate auto-generation of scenarios for these types of surprises.

The goal here is to observe how the system handles the performance impact of such misaligned scenarios and possibly whether it corrects them on its own.

Off-road

The final surprise category represents scenarios that aim at breaking the system by driving it to the edge. For example, introducing a recursive query to bloat the memory resource needs while running that query, introducing extreme data skew, cutting down the hardware resources on the fly by introducing a heavy stream of collocated workloads would fall under this category.

While some automation is still possible with this category, it is the one that requires the most careful crafting, and hence manual effort.

The goal here is to observe the robustness of the systems under extreme scenarios.

Round 0

Before making surprise benchmarking in reality, we devised a plan for an initial test run, *Round 0*. The goal with this round is to establish the methodology and the framework to perform the surprise benchmarking, in addition to evaluate our vision for it. Therefore, Round 0 will be an example run orchestrated by us using the database systems picked by us. We will also be building the preliminary infrastructure for keeping a leaderboard for the benchmark rounds. We plan to share our experiences and findings during Round 0 in the form of a paper, e.g., in DBTest workshop.

To ensure that we focus on developing the bare minimum requirements for the surprise benchmarking methodology and framework, we will keep the benchmark itself as simple as possible. In other words, we would like to start from existing well-established benchmarks. Thus, Round 0 focuses on traditional OLAP workloads and takes TPC-H benchmark as the basis. For the systems under test, we pick Postgres, MySQL, DuckDB, Umbra, and ClickHouse.

4.1.1 Round 0: Surprises

During the real benchmark run, we will complement TPC-H queries with previously unseen queries corresponding to each surprise type described in Section 4.1.

Here are a list of possible surprises that we can create:

Surprises on the Query Level

- Joins on non FK-PK relationships
- Queries with recursion
- Queries with cyclic joins
- Queries with UDFs
- Large SQL strings (multiple MBs)
- Queries that stress memory (intermediates that need to spill)
- Non-equi joins
- ...

Surprises on the Schema Level

- No FK-PK in schema
- Extremely many tables
- Non star/snowflake schemata
- ...

Surprises on the Data Level

- Heavy hitters for joins
- Non-uniform / correlated data
- Size of the data

4.1.2 Round 0: The benchmark procedure

We require databases to support SQL-92 (or similar).

We'll provide a sample data generator and workload 30 days before the run. There will be some sort of leaderboard, and each contestant will be money (cloud credit) limited. Entries will be a deployable setup (either in the cloud or on a dedicated test server) in a format such as Docker or Kubernetes.

We'll include a data loading procedure, with a clean CSV or other file that contains the table data. The databases will have a bounded amount of time to load the data; e.g., set to 10x longer than whatever our reference implementation takes.

Round 1

We plan to run the first round of the real surprise benchmarking around the same time with our paper presentation / publication for the Round 0. The benchmarking run will be open to anyone, but we will specifically reach out to the systems we tested during Round 0 to enter Round 1 themselves.

Long-term plan

We think of starting with quarterly rounds.

We would like to base the rounds on the existing benchmarks as much as possible till the benchmarking rounds become more stable to keep the work around running these benchmarks low.

To check the correctness of the output of the benchmark queries, we will double-check and use the results generated by Postgres. Furthermore, we plan to use Postgres as the baseline during the benchmark rounds, since it is a highly mature and popular database system.

4.1.3 Themes to consider

- OLAP + unseen queries
- OLTP + unseen transactions
- OLAP + add new / many tables
- OLAP + resource-constrained hardware
- OLTP + multitenancy / collocation
- Key-value stores + data skew
- Vector databases + different data characteristics
- Data loading & new table creation
- Cloud (with all of the above)
- Full surprise with mix & match
- Multimedia analysis
- Poorly factored microservices

4.1.4 Metrics to consider

- Throughput
- Latency and latency distribution
- Energy
- Carbon footprint
- Cost (bare-metal vs cloud)
- Resource consumption (memory, disk, CPUs)
- Micro-architectural (IPC, cache misses, etc.)
- Robustness
- Scalability
- Availability

Challenges

- *Automation* for both the generation of the surprises and the benchmark runs.
- What would be the *incentive* for companies and academics to run and/or enter these benchmark rounds?
- What would be the *hardware infrastructure* to support the benchmark runs? Who would provide it? Furthermore, when we would like a surprise hardware element, how much of the hardware infrastructure should be revealed beforehand?
- How do we identify and quantify some of the less traditional *metrics* such as robustness, carbon footprint, etc.?

Addressing all these challenges will be crucial for the success of the benchmark. While we aim to tackle these as much as possible with Round 0, some challenges have to be resolved as we have more rounds of benchmarking. For example, we won't be able to automate surprise generation across many workloads and deployment on different types of hardware infrastructures starting from Round 1.

4.2 Working Group on Transactions and Concurrency

Wensheng Dou (Chinese Academy of Sciences – Beijing, CN), Adam Dickinson (Snowflake Computing Inc. – Seattle, US), Burcu Kulahcioglu Ozkan (TU Delft, NL), Umang Mathur (National University of Singapore, SG), Everett Maus (Google – Seattle, US), Stan Rosenberg (Cockroach Labs – New York, US), Gambhir Sankalp (EPFL – Lausanne, CH), Caleb Stanford (University of California – Davis, US), and Cheng Tan (Northeastern University – Boston, US)

License © Creative Commons BY 4.0 International license
 © Wensheng Dou, Adam Dickinson, Burcu Kulahcioglu Ozkan, Umang Mathur, Everett Maus, Stan Rosenberg, Gambhir Sankalp, Caleb Stanford, and Cheng Tan

This working group focused on the challenges involved in testing and diagnosing transactional database systems, which support complex SQL operations and varying consistency and isolation levels, in a concurrent and distributed setting.

We investigate methods to evaluate the transactional correctness of database systems, i.e., whether the database system satisfies its guarantees under all possible execution scenarios. We excluded problems related to non-concurrent database testing problems (such as ensuring individual or non-concurrent queries, reads, or transactions are correct) and instead focused on problems which are specific to concurrent transaction workloads.

Discussed Clusters

Our discussions identified problems of interest in this space, and found that they can be divided into 3 major clusters:

1. Concurrent test oracles: how can we design test oracles to verify concurrent transactions?
2. Coverage metrics and test generation: How can we identify coverage metrics for testing, in order to generate minimal and sufficient test cases?
3. Bug and failure reproduction: After bugs are found (either by users or in production), how can we effectively reproduce, root-cause, and diagnose them?

In each cluster, we identify the state of the art in practice, problems to be solved that we view as open, related work, and potential solutions.

In addition to the three clusters, there are other dimensions of interest. For the target database system, it can vary from key-value stores (at the simplest level), to relational databases, to supporting client-level transactions. Consistency and isolation levels are another important dimension, ranging from strong sequential consistency to weaker consistency models or only eventual consistency. Problems can also be classified based on the type of bug that is considered: safety bugs such as data races and race conditions; violations of serializability, linearizability, and atomicity; deadlock freedom; liveness violations such as non-termination; and performance bugs. Many of these bugs manifest concretely as either *wrong results* returned to the user, *data corruption* in the database itself, or *crashes* (which is the best case scenario of the three).

Cluster 1 – Concurrent Test Oracle

4.2.1 Motivation

It is notoriously hard to test concurrent problems in databases as they (the bugs/anomalies) rely on various inter-transaction interactions, transaction interleavings, and transactional concurrency guarantees (isolation guarantees promised with the database systems).

Today's approach is to test a database by repeatedly running concurrent workloads, hoping to cover more interleavings, transactional interactions, and database internal schedules. However, given an isolation guarantee (e.g., serializability), the valid states are huge; then, how to examine if a given state is valid is challenging. Note that this is straightforward if a database provides linearizability and outputs a commit timestamp – we re-execute transactions in the order that they committed at, and can check the database state at each point as is done by Spanner at Google. The problem is more complex (from a algorithm complexity point of view) for some weaker isolation levels (e.g., non-strict serializability and snapshot isolation) and for the databases without accurate commit timestamps.

So, we form this problem into the *concurrent test oracle problem*: given some workload of concurrent transactions, how to validate that the responses and the final state of the database is correct with respect to the isolation and concurrency specification.

4.2.2 Batch Formalization

The batch formulation assumes we have executed some set of concurrent workloads, and now want to validate that the outcome was correct.

Given:

Input:

- $t_0 \leftarrow$ start time
- $t_1 \leftarrow$ end time
- {transactions, ...}
- database state at t_0
- concurrency + isolation specification

We assume input state of the database is valid.

The transactions are assumed to contain the sequence of actions they executed (reads, writes, mutations, but also queries and DML statements), the transaction’s start and end times (synonymously: arrival / commit time), and the results of reads or queries. (Optionally, we believe it is reasonable to also require the actual values written, if it simplifies the problem.) Note that we require actions to be atomic. This is identical to the standard way of describing concurrency / isolation models for reads / writes, but we also require things such as a query (which may touch multiple tables) read from the same version of the database (and similarly, DML may read/modify/write multiple rows).

For the batch case, we want to be able to validate that the database state at t_1 is valid after the transactions provided have all executed, and that the values read (or queried) by all transactions are correct at each point.

4.2.3 Incremental Formalization

The incremental formalization is aimed at the problem of validating transaction outcomes in an “online” way – as part of an ongoing test, for cases where the batch validation case may not scale (e.g. where hundreds of thousands of transactions might be run over the life of a test, but the number of transactions running concurrently might be quite low).

Given:

Input:

- $t_0 \leftarrow$ start time
- $t_1 \leftarrow$ end time
- transaction to validate starting at t_0 and ending at t_1
- {transactions + metadata, ...} all transactions with overlapping start or end times with the transaction to validate
- database state at t_0
- concurrency + isolation specification

We assume the database state at t_0 is correct. The transactions and metadata are similar to above.

Given this, we want to be able to do one of a few (largely equivalent) things:

- Determine if the actual database state at t_1 is consistent, and that the read/query results from that transaction are valid
- Compute the set of all possible database states at t_1 and the set of all possible read/query results from the transaction

4.2.4 Problem

We define a transaction as a sequence of SQL statements, decomposable into read and write operations. A set of all possible executions of transactions forms a schedule. Thus, an isolation level can be seen as a set of constraints, I , over all possible schedules. We say

that a schedule is allowed under an isolation level if it adheres to all constraints in I. E.g., serializability isolation is a set of constraints which ensures that every schedule corresponds to a serial execution of transactions.

There are two mainstream approaches to verifying an isolation level, namely black box and white box oracles. A black box oracle observes only external state changes, same as a database client; i.e., results of individual operations are invisible until a transaction is committed. Conversely, a white box oracle can observe internal states; i.e., results of read/write operations. Each approach is fraught with scalability and expressiveness challenges. The latter arises when transactions are composed of SQL statements. Read and write effects of a SQL statements are in some sense a prerequisite for expressing (transaction) conflicts. Scalability challenge is a function of transaction size (i.e., number of operations per transaction) and transaction throughput (i.e., number of committed transactions per second). Both online and offline approaches exist. An online approach is desirable especially if it yields a mechanism whereby a transaction in violation of the isolation level can be aborted. For practical reasons, e.g., high transaction throughput, offline verification remains a popular approach.

4.2.5 Challenges

There are multiple challenges to address the concurrent test oracle problem in practice.

Challenge 1: efficiency and scalability

Checking efficiency is crucial in practice. Some of the problem variants have been proved to be computationally expensive to check; for example, black-box checking serializability and snapshot isolation is known to be NP-Complete. Even for the white-box checking isolation levels that have polynomial checking algorithms, it is unclear if the checking can be efficiently implemented to catch up with the throughput of today's databases.

In practice, a DBMS is expected to meet its consistency/isolation guarantees, and is expected to support high levels of concurrent usage over long periods of time. In practice, this is often checked by long-running tests that aim to provoke a wide array of behaviors over a long (hours to days) length of time.

Challenge 2: beyond reads and writes

The state-of-the-art test oracles almost all work on the transactional key-value model; that is, the available operators are reads and writes to a key once at a time. However, in practice, real-world key-value stores have many advanced operators, including range queries, max, min, and many other aggregation operators. In addition, we barely know how to handle concurrent SQL statements in the concurrent test oracle problem.

Challenge 3: providing evolvability and debuggability

Developer friendliness is also a strong requirement. In particular, if the DBMS itself is extended (e.g. new features are added to the SQL dialect the DBMS supports) then either the underlying solution should not need new work or any extension required should be straightforward for the average developer. For example: If adding support for a new aggregate, such as MIN/MAX, to the test system requires significant work to update an SMT representation, it's unlikely to be a viable solution for industry use.

Similarly, the debuggability of a solution would be important to allow issues found by these systems to be reproduced, understood, and addressed by developers.

4.2.6 Existing approaches

Differential testing

For deterministic (statement-by-statement execution, that is, we can obtain a deterministic execution trace for each statement in the concurrent transactions) transaction test cases, we can compare the transactions' execution results between database systems. However, differential testing faces some issues. First, it cannot compare concurrent transaction executions. Second, different database systems may contain inconsistent transaction semantics, and differential testing can report false positives. Third, different database systems support different isolation levels, which cannot be compared together.

Infer transaction execution result

For some simple and deterministic transaction test cases, we can potentially infer the execution results of each statement in the transactions according to their transaction semantics. This approach can also have some limitations. First, it cannot infer the test oracle for concurrent transactions in which we cannot obtain their statement-by-statement execution. Second, it cannot support some complex SQL queries, e.g., aggregation functions (may not be important).

Key-Value Store Oracles.

This problem has largely been solved for simple key-value stores (i.e., supporting only point reads/writes). However, supporting the more complex semantics that languages like SQL support is an open problem. (Consider that the values read from one table may depend on the rows read from another table within the same query.)

Cluster 2: Coverage Metrics and Test Generation

4.2.7 Current industrial practice

A common way to test the concurrency/transactional correctness of a database is to run a set of concurrent transactions and observe that the database behaves correctly. However, it is non-trivial to determine what cases uncover net-new behavior in the system versus simply validating the same case repeatedly.

The current state of practice in this space tends to simply run a set of tests repeatedly (either for a test budget of a certain amount of compute time or a certain number of test executions) with no clear indication of coverage. This does not provide any confidence to the practitioners about the effectiveness of their testing strategy, leaving the following questions unanswered:

1. Given a particular test case (e.g. a set of concurrent transactions) at what point will running more tests stop providing additional information about the system's behavior?
2. Given two test cases and information about the events that occurred, do they exercise redundant behavior?
3. Running a set of text executions, at what point do additional test cases/executions not provide information about the system? (At what point has this metric been saturated?)
4. At what point have these tests covered enough of the system that we can be confident in the overall concurrent behavior of the system?
5. Is it possible to automatically construct net-new test cases that provide new information about the concurrent behavior of the system?

4.2.8 Problem Statement

To address these questions, we would like to be able to determine:

Defining Coverage How can we define proper coverage metrics that can reflect developers' intuition of test coverage?

Measuring Coverage Given a particular set of test cases, how can we efficiently measure the amount of exercised behaviors of the database systems?

Detecting Saturation Running a set of text executions, at what point do additional test cases not provide information about the system? At what point has the coverage been saturated?

Coverage-Guided Test Generation How can we automatically construct new test cases that provide new information about the concurrent behavior of the database system?

4.2.9 Coverage Metrics

The set of covered behaviors in database systems depends on both (i) test input transactions, which may trigger operations on different replicas and partitions (data-centric behaviors), and (ii) concurrent interactions and partial failures in the database system (concurrency-centric behaviors). The set of transactions running in the test case may trigger different features of the database system exercising different system behaviors (optimization engines, rollbacks, failure recovery, etc.). On the other hand, the same set of transactions can trigger different behaviors of the system, depending on the interleavings of the concurrent interactions in the system (e.g., communication between database replicas, the interleavings of the faults w.r.t. the communication).

Therefore, an ideal coverage metric should be able to capture the coverage in both aspects and also how novel behaviors in one aspect trigger novel behaviors in the other aspect. Even when a given set/sequence of transactions has the potential of exposing subtle behaviors and bugs in the system, not all interleavings may be able to expose them. Likewise, simply exploring the entire space of concurrent interactions between a given set of transactions may not exhaustively exercise features that may otherwise be exposed by different transactions. Indeed, a subtle choreography between transactions and their concurrent interactions is often required to expose bug inducing behaviors.

The existing coverage metrics widely used in practice (such as line or branch code coverage) does not capture the temporal order and the interactions between the transactions. It remains an open research question to define proper coverage metrics to address the problem in the database systems settings running concurrent transactions.

4.2.10 Measuring Coverage

Given the insufficiency of traditional coverage metrics like line and branch coverage, more exhaustive metrics that distinguish temporal behaviors may be more appropriate in our setting. However, the precise granularity and level of abstraction may affect the efficiency of determining the amount of coverage that has been achieved. The efficiency of coverage tracking also crucially affects the runtime of the testing campaign and, thus, the number of behaviors covered within the same testing budget. We think a thorough investigation of coverage metrics that achieve different tradeoffs between abstraction (i.e., does not distinguish between similar inputs/interleavings) and efficiency of tracking coverage may be required to assess their suitability to different settings.

4.2.11 Detecting Saturation

While measuring coverage of a set of tests provides information about the explored set of system behaviors, it does not indicate how much of the possible system behaviors have been explored.

To provide confidence in the testing strategy, the coverage metric should quantify the set of (all) possible system behaviors and how much of those are covered by a set of test executions seen so far. Such a quantification has the potential of providing concrete feedback to developers about how much more computational resources to continue to dedicate and whether or not to terminate the exploration of new behaviors in case the coverage is close to saturation and the marginal utility of continuing is expected to be low.

4.2.12 Coverage guided transaction test generation

The coverage information can be used to guide the generation of new test cases to extend the search to increase coverage. In order to expose concurrency-centric behaviors (in conjunction with other data-centric behaviors), we envision that *scheduling hints* and *hints on partial failures at runtime* can increase the effectiveness of otherwise vanilla transaction-based tests. On the one hand, no control over the scheduling and the injection of partial faults may empirically resemble naive random testing with poor effectiveness. On the other hand, full scheduling hints that can control precise interleavings of processes can be impractical. Further, retrofitting controlled testing into existing large-scale testing frameworks is likely to be not ergonomic and may not be adopted in practice.

The space of granularity of data-centric and scheduling hints should be investigated to build effective and ergonomic guidance methodologies for test generation. The feedback information can then be used in a similar approach to popular feedback-driven fuzzing methods.

Cluster 3: Bug and Failure Reproduction

Suppose that a system has encountered an unexpected bug or crash – either in a complex test or in production. We have the existence of the bug, the logging provided by the system prior to encountering the bug, and any other diagnostic information collected on failure. How can we re-create the sequence of events that led up to this bug?

In our discussions, we identified four major questions that we would like to see more work addressing. This section contains a summary of current industrial practice, a discussion of the four identified questions, and related areas of work.

4.2.13 Current industrial practice

Our industry attendees emphasized that this is a common scenario in practice, and it can be very difficult to reproduce failures. The current state-of-the-art is often simply rerunning a test a large number of times (e.g., 1000x), potentially after augmenting the system with additional logging, a slightly modified configuration, runtime sanitizers, or with changes to the system or test workload to increase the likelihood of perturbing the issue. Due to the lack of support for more targeted reproduction techniques, each test run can require running on 10s to 100s of virtual machines, and the time to run each test takes anywhere from a minimum of 30 minutes to run to a maximum of about 16 hours.

Due to all of these challenges, *known* bugs in production databases can remain undiagnosed for years. For example, one anecdote described a transactional database system with 6 long-standing concurrency-related bugs in production, and reproducing a single one of these bugs in a test environment and subsequently fixing it eventually took 2.5 years.

Whether it is more important to reproduce bugs found during testing vs. bugs found in production, it was answered that both are very useful, but roughly 20x more bugs are found during testing vs. in production. Therefore, new techniques that only work during testing would still be useful, even if the overhead is too high to be deployed in production.

Finally, some industries are leveraging systems for deterministic record-and-replay (see related work below), such as FoundationDB's testing framework [56], Megastore at Google [7] and Hermit at Meta [37]. See also Thomson and Abadi's 2010 paper making the case for determinism [50]. However, there was a feeling that these techniques may not be mature enough to scale to the distributed setting and reproduce transactional database bugs at scale. They require both an upfront design cost and runtime overhead, both of which can be prohibitive, especially when running on multiple machines.

4.2.14 Problems identified

Q1: Reconstructing the system trace

Suppose that some system events and failures are logged, but others are not. Given the set or trace of logged events, how can we reconstruct one possible system execution trace which is consistent with the log?

Note that there may be multiple system execution traces that are consistent with the log; we only need to find one of them in order to demonstrate a bug.

Approaches to the problem can be placed on a spectrum, trading the amount of additional logging required with additional effort in reconstructing the execution:

- On one end, logging every event in the system with a trusted clock (faults, scheduling, message ordering, message deliveries, internal non-determinism) can allow for viable reproduction of the explicit system state, but imposes a large burden on developers to explicitly track any choices that affect state as well as on the logging framework to store and manage the large amount of information.
- On the other end, minimally logging transactions, start/commit times, and the final state can allow the use of an oracle as in Cluster 1 to produce a set of viable executions explaining the behavior. However, the set of executions to explore in this scenario may be too large to reasonably compute, given that logs leading up to a failure may contain thousands to tens of thousands of transactions.

Q2: Amount of logging

The discussion above for Q1 raises the following central question: what is the *minimal* set of information that needs to be logged in order to answer question 1? That is, what is the minimum amount of logging which is sufficient to reconstruct the system state at time of failure, within a given computational means?

Here, it is not necessary that the logged information allows us to reconstruct the execution uniquely; it is sufficient to be able to reconstruct an execution leading up to the final state within a small enough search bound from the input logs. The logged information is then used to filter the space of possible executions.

One caveat that makes this difficult in practice is that sometimes when logging is added, it can make it impossible to reproduce some states (as logging can affect the timing of events).

Q3: Controlling environmental non-determinism

How can we ensure that the environment simulated in a controlled environment matches the production environment? In other words, every behavior that is possible in production must be possible in the simulated environment.

Simulated environments, including e.g. record-and-replay tools as well as fault injection tools, provide a partial solution to the failure-reproduction problem, albeit with significant overhead. The goal of such tools is to provide a highly controlled testing environment, wherein all non-determinism choices are controlled and determinized by the environment. However: 1) the controlled testing environment may not faithfully reproduce the behavior and semantics of the production environment, and 2) the determinization of behavior may preclude discovery of traces that reproduce the failing behavior.

Q4: Retrofitting determinism

How can we add determinism guarantees onto an existing database system that was not built to support deterministic execution?

The challenge here is a little different than deterministic replay; the question is about making the database implementation itself deterministic. For the distributed case, however, we also have to ensure determinism of the distributed aspects through, e.g., simulating the network and injecting controlled node failures.

4.2.15 Related Work

Several lines of related work are relevant to the problems identified above. *Flaky tests* (i.e. tests that may fail nondeterministically), are extensively studied in the software engineering community, (e.g., [35, 8, 21, 31, 42]). Bugs related to concurrency, distribution, consistency, and isolation levels often manifest as flaky tests. Work on *debugging big data applications* is also relevant; the most prominent tools in this space include Inspector Gadget [39] and BigDebug [27]. Finally, *record-and-replay* tools, which include Mozilla RR [38], Hermit [37], and DetTrace [36], provide controlled environments for determinized execution and debugging.

4.3 Working Group on Query Languages and Debugging

Denis Hirn (Universität Tübingen, DE), Moritz Eyssen (Snowflake – Berlin, DE), Tim Fischer (Universität Tübingen, DE), Torsten Grust (Universität Tübingen, DE), Muhammad Ali Gulzar (Virginia Polytechnic Institute – Blacksburg, US), Hannes Mühleisen (CWI – Amsterdam, NL), Thomas Neumann (TU München – Garching, DE), and Mark Raasveldt (DuckDB Labs – Amsterdam, NL)

License © Creative Commons BY 4.0 International license
© Denis Hirn, Moritz Eyssen, Tim Fischer, Torsten Grust, Muhammad Ali Gulzar, Hannes Mühleisen, Thomas Neumann, and Mark Raasveldt

SQL-level Plan Specification

A SQL query provides a *declarative* specification of a computation over relational tables: it is the database system’s task (and freedom!) to explore a space of possible algebraic plans that will implement the query. Query authors need not and (largely) *cannot* influence the system’s choice of execution strategy. Declarativity is a core virtue of the SQL language –

yet, occasionally, the declarative nature can stand in the way, in particular when precise control over the shape and operators in a physical algebraic plan is desirable, for example in plan-centric database engine testing.

The working group on *Query Languages & Debugging* tackled the challenge of formulating algebraic query plans in a human-readable (and thus: human-writable) fashion. Ideally, this plan format could also serve as a means to exchange physical plans across database engines. In the context of database engine testing, precise control over algebraic plan construction can help to ensure that even the most obscure operator constellations and remote engine code paths are covered – a feature that is hard to realize solely in terms of SQL-level query fuzzing.

The primary guiding principle of the working group’s discussion was pragmatics: we settled on (a carefully selected subset of) *SQL itself* as the format in which such algebraic plans are to be written and communicated. Refer to Figure 1 for one sample instance of such a SQL-encoded physical algebraic plan:

- We use a common table expression (CTE) to isolate SQL queries, each of which realize the semantics of a given algebraic operator: for example, query `INDEXSCAN_1` reads columns `a` and `b` from table `r` while evaluating the predicate `a = 42 ∧ b ≤ 5`, while query `TABLESCAN_1` scans columns `c` and `d` of table `s`.
- Naming conventions are used to communicate specific algebraic plan operator kinds: `INDEXSCAN_*` advises the engine to employ an *index scan* (here: over a (compound) `a, b` index), while `HASHJOIN_*` represents a *hash join* over given build and probe inputs (see the conventionally named `build` and `probe` row aliases in the query’s `FROM` clause).
- CTE-provided query names are then used to assemble tree- or DAG-shaped complex query plan. The algebraic plan encoded by the SQL query in Figure 1 is shown in Figure 2.
- The family of admissible physical plan operators is readily extended: we require a canonical SQL-based formulation of the operators’ semantics as well as a recognizable naming convention for the operator itself (as well as its n inputs if the operator is n -ary). We are convinced that a rather small SQL subset will suffice to encode the behavior of even the most exotic algebraic operators.

We envision a query-specific pragma or configuration setting – possibly embedded in a “magic comment” like `--#ENCODED_PLAN` or similar – to switch the database engine into plan-decoding mode. In this mode, the engine detects an encoded physical operator based on (1) its CTE naming convention and (2) a match against the operator’s known canonical SQL AST pattern. Table and row aliases are used to wire the physical plan tree.

The rationale of the working group’s proposal is as follows:

- Relational already engines come with the required facilities to parse, represent, and serialize SQL queries. No extra plan parser, validator, or pretty-printer needs to be implemented.
- Likewise, query authors and engine developers have already mastered the skill to read and write SQL queries and thus plans. No new syntactic oddities are to be learned and the cognitive overhead is minimized.
- This SQL-based plan format will continue to work even if a database engine only implements a selection of the algebraic operator patterns: CTEs that were not detected as a known algebraic operator can still be wired into the overall plan tree. Their evaluation will yield an intermediate result that can be consumed by the residual plan.
- Indeed, *any database engine* – including those fully unaware of our operator naming and encoding conventions – will still be able to evaluate such a SQL-encoded plan. In a testing environment, the query result may be used as an oracle that provides the expected outcome of the algebraic plan.

```

WITH
INDEXSCAN_1(a,b) AS (
  SELECT a, b FROM r WHERE a = 42 AND b <= 5
),
TABLESCAN_1(c,d) AS (
  SELECT c, d FROM s
),
HASHJOIN_1(a,b,c,d) AS (
  SELECT *
  FROM   INDEXSCAN_1 AS build JOIN TABLESCAN_1 AS probe ON (a = c)
),
RADIXSORT(a,b,c,d) AS (
  SELECT * FROM HASHJOIN_1 ORDER BY a DESC;
)
TABLE SORT;

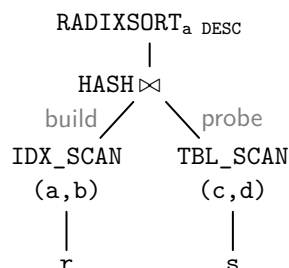
```

■ **Figure 1** A SQL-encoding of the algebraic plan of Figure 2.

The SQL Acid Test Project

Although SQL is an ISO standard, in practice, systems rarely comply *completely* with the standard. Countless semantic and syntactic differences between systems, often of rather subtle nature, are a sad fact of today’s SQL reality. This makes it difficult to write SQL that is portable between systems, and limits the ability to apply SQL skills across database engines – practitioners who are able to write SQL for one database system will face challenges when writing SQL for another database system. Similarly, SQL tooling needs to take these different query language dialects into account, which in practice results in tools being created only for a specific dialect or system, rather than widely applicable tools that can be used for SQL systems in general.

In the world of web browsers, this problem has been encountered before, where browsers (such as Microsoft’s Internet Explorer IE6) were found to not be truly standards-compliant. The *Web Standards Compliance Acid Test* [44] was created as an attempt to combat this conundrum. The idea was that users could visit a web page, and if the browser was indeed standards-compliant, it would render a smiley face. This was intended to be an incentive for browser vendors to strive for true standards-compliance. After all, if they were not, their users would see that their browser would not be “smiling at them.”



■ **Figure 2** Plan tree of physical operators, derived from the SQL encoding of Figure 1.

```

+-----+
| .....#####.....|
| ....##.....##....|
| ...#.....#...|
| ..#..()...()..#..|
| ..#.....o.....#..|
| .#.....#.|
| ..#\...../..#..|
| ..#...-----..#..|
| ..#.....#...|
| .....##.....##....|
| .....#####.....|
+-----+

```

■ **Figure 3** Successfully rendered smiley by a SQL Acid Test compliant database system.

The *SQL Acid Test Project* was launched during this seminar to attempt to create a similar compliance test for SQL database systems. The basic idea revolved around a (large) SQL query that users can execute using the database engine of their choice. If the system is standards-compliant, the query output shows the text-based rendering of a smiley face (see Figure 3). Otherwise, the system will either return an error if parts of the test query are not supported, or parts of the smiley rendering will be replaced with an “X” to indicate a deviation from the expected query result.

One problem with the SQL standard is that it is under-specified in many areas, and many behaviors are left up to the implementation. Therefore, while testing for SQL standard compliance definitely is important, it is not sufficient to guarantee that the system will behave properly in all scenarios. For this reason, we have decided to split up the acid tests into two sets:

Compliance tests strictly test if the system complies to the SQL standard.

Convention tests check if the system supports a *sane* SQL dialect as determined by a panel of experts (as of now these have been the members of the working group – a broader community representation clearly is desirable).

test	Db2	Oracle	MSSQL	MySQL	PostgreSQL	Umbra	DuckDB	Snowflake	Spark	GoogleSQL	SQLite3	Presto	Tableau	Hyper	MonetDB
01	✓	✓	✓	⚡	✓	✓	✓	✓	✓	⚡	⚡	✓	✓	✓	✓
02	⚡	✓	✗	⚡	✓	✓	✓	✓	✗	✓	✗	✓	✓	✓	✓
03	✗	⚡	⚡	⚡	✓	✓	✓	✓	⚡	⚡	⚡	✓	✓	✓	✓
04	⚡	✗	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
05	⚡	⚡	⚡	⚡	✓	✓	⚡	✓	⚡	⚡	⚡	⚡	✓	✓	⚡
06	✓	✓	✓	⚡	✓	✓	✓	✓	✓	⚡	⚡	✗	✗	✗	✗
07	⚡	⚡	⚡	✓	✓	✓	✓	⚡	✓	⚡	⚡	✗	✓	✓	⚡
08	⚡	✓	⚡	✓	✓	✓	✓	⚡	✓	⚡	⚡	✓	✓	✓	✓
09	✗	✓	✗	⚡	✓	✓	✓	✓	⚡	⚡	⚡	✓	✓	✓	✗
10	⚡	✓	✗	⚡	✓	✓	✗	✓	⚡	⚡	✗	✓	✓	✓	✗
11	⚡	⚡	⚡	⚡	✓	✓	✓	✓	✓	✓	✓	✓	⚡	✓	✓
12	⚡	⚡	✓	✓	✓	✓	✓	⚡	✓	⚡	⚡	⚡	⚡	✓	✓
13	⚡	⚡	⚡	⚡	✓	✓	✓	⚡	⚡	⚡	⚡	⚡	✓	✓	⚡
14	⚡	⚡	⚡	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
15	✓	✓	⚡	⚡	✓	✓	✓	✓	⚡	⚡	✓	✓	✓	✓	✓
16	⚡	⚡	⚡	⚡	✓	✓	✓	✓	⚡	⚡	✗	⚡	✓	✓	⚡
17	⚡	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
18	⚡	⚡	⚡	⚡	✓	✓	✓	✓	⚡	⚡	⚡	⚡	✓	✓	✓
19	✓	✓	✓	⚡	✓	✓	✓	✓	✗	⚡	⚡	⚡	⚡	⚡	⚡
20	✓	⚡	✓	⚡	✓	✓	✓	⚡	⚡	⚡	⚡	✓	✓	✓	✓
21	✓	⚡	✓	⚡	✓	✓	✓	✓	✗	⚡	⚡	✓	✓	✓	✓
22	✓	⚡	⚡	⚡	✓	✓	✓	✓	⚡	⚡	⚡	⚡	✓	✓	⚡
23	⚡	⚡	⚡	⚡	✓	✓	✓	✓	✓	⚡	⚡	⚡	✓	✓	⚡
24	⚡	⚡	⚡	⚡	✓	✓	✓	✓	✓	⚡	⚡	⚡	✓	✓	⚡
25	✓	✓	⚡	⚡	✓	✓	✓	✓	⚡	⚡	⚡	✓	✓	✓	✓
26	✓	⚡	⚡	⚡	✓	✓	✓	✓	✓	⚡	⚡	✓	✓	✓	⚡
27	⚡	✓	⚡	⚡	✓	✓	✓	✓	✓	✗	⚡	✓	✓	✓	✓
Score	33%	41%	26%	15%	100%	100%	93%	81%	52%	19%	19%	70%	89%	56%	

■ **Figure 4** Acid test compatibility matrix as of November 2023. The ✓ symbol indicates that a particular acid test is running successfully on the system. ✗ indicates that the system executes the test, but the result is unexpected. ⚡ means that the system was unable to execute the test. The highlighted test cases (■) are SQL compliance tests.

The *SQL Acid Test* project can be found at <https://github.com/sqlstandardsproject/sqlacidtest>.

4.4 Working Group on Testing “Analytical” Components of Databases

Manuel Rigger (National University of Singapore, SG), Jinsheng Ba (National University of Singapore, SG), Ankush Desai (Amazon – Cupertino, US), Adam Dickinson (Snowflake Computing Inc. – Seattle, US), Wensheng Dou (Chinese Academy of Sciences – Beijing, CN), Stefania Dumbrava (ENSIIE – Paris, FR), Moritz Eyssen (Snowflake – Berlin, DE), Florian Gerlinghoff (MotherDuck – Amsterdam, NL), Hong Hu (Pennsylvania State University – University Park, US), Zu-Ming Jiang (ETH Zürich, CH), Marcel Kost (Salesforce – München, DE), Everett Maus (Google – Seattle, US), Mark Raasveldt (DuckDB Labs – Amsterdam, NL), Andrei Satarin (Google – Mountain View, US), Thodoris Sotiropoulos (ETH Zürich, CH), and Chengyu Zhang (ETH Zürich, CH)

License © Creative Commons BY 4.0 International license

© Manuel Rigger, Jinsheng Ba, Ankush Desai, Adam Dickinson, Wensheng Dou, Stefania Dumbrava, Moritz Eyssen, Florian Gerlinghoff, Hong Hu, Zu-Ming Jiang, Marcel Kost, Everett Maus, Mark Raasveldt, Andrei Satarin, Thodoris Sotiropoulos, and Chengyu Zhang

The key problem we are interested in solving is: how can we make sure that a DBMS behaves correctly, without needing to manually enumerate every possible behavior? In particular, *how can we ensure that the behavior of a DBMS satisfies the expected behavior*

in terms of correctness, security, crashes, and performance guarantees, without requiring a DBMS developer to *expend exponential effort enumerating and testing every behavior and combinations of behaviors*? Although this is a general systems problem, DBMSs are more challenging than many other systems because of their (1) complexity (both of individual features and feature interactions), (2) expected correctness, and (3) persistent state.

Among the seminar attendees, we identified four common themes of interest, which relate to the core challenges of automatic testing:

1. Test case generation, including database and query generation;
2. Test oracles to validate the DBMSs' correctness and other properties;
3. Bug analysis, covering topics such as test-case reduction and deduplication;
4. Transactional testing, which covers aspects of the above challenges, but poses its own special challenges.

Cross-cutting Challenges

We identified various cross-cutting challenges (i.e., challenges that span across two or more of the general challenges that we have identified). We will outline them in this section.

4.4.1 Bug Studies

Various effective automated testing generation approaches have been proposed. However, it is unclear what kind of bugs that they overlook. This question could be studied based on reports in issue trackers or based on customer reports. Specifically, studies could be performed to investigate both characteristics of the features used in the bug-revealing test cases as well as whether existing test oracles could have found them, to identify gaps that could be addressed.

4.4.2 State of the art in the industry

Multiple companies have developed sophisticated testing frameworks. The approaches behind these frameworks are not widely shared, meaning that those unaware of them might reinvent the wheel or implement suboptimal approaches. It would be ideal if companies could share some of their tools or insights; one way forward could also be for academia to conduct interview studies with practitioners.

4.4.3 Interfaces between Existing Tools

Current testing tools typically consist of a database generator, query/workload generator, test oracle, and potentially a component to reduce and deduplicate test cases. These components are generally tightly coupled and it is currently not possible to easily integrate them. For example, it would be difficult to combine the query generator of one tool with a test oracle from another tool. Interfaces could be defined that would allow combining them, similar to existing work on formal verification tools [10].

4.4.4 Query Dialects

In the relational setting, despite the existence of an SQL standard since 1986, a known problem is that many very different variations of it have been implemented in commercial systems [30, 4]. The problem also exists in the non-relational setting. Indeed, for graph

databases, the current commercial eco-system is fragmented across dozens of vendors that each implement their own graph query language, as recently surveyed in [9], spurring interest in emerging standards, such as SQL/PGQ [18], which extends SQL with graph queries, and the GQL native graph query language.

Differences between query dialects are a challenge for test-case generation, the test oracle problem, and test-case reduction techniques. For test-case generation, it is difficult to design general generators that both produce valid databases and queries for various dialects, as well as dialect-specific features. For the test oracle, it is difficult to apply differential testing or implement reference engines due to differences between dialects. For test-case reduction, most current reducers are dialect-specific or rely on general text-based reductions.

Test Case Generation

Problem statement. In the context of DBMS testing, a test case consists of an initial database state (schema and data), some amount of workload (queries, transactions, schema changes), and the final expected state (expected results of queries, final database state, etc.). Traditional software development processes involve the creation of manually written test cases to ensure correctness. However, this approach leads to a fixed test suite that, even though it might be over a large area of the input space, still leaves many states unexplored. Therefore, an alternative approach is to generate test inputs automatically and randomly. An important challenge associated with randomized test input generation is: *how can we automatically generate diverse test cases that exercise interesting behaviors in the database engine under test, and thus uncover bugs?*

4.4.5 Existing Techniques

We identify and review the current state-of-the-art testing techniques as follows.

Manually written test cases. These sorts of tests are common across all (industry?) databases and consist of manually curated sets of initial states, workloads, and expected states. The problem with solely relying on manual testing is that:

- The space of feature interactions in a database grows exponentially as new features are added
- It is generally infeasible to enumerate all of the states that all users of DBMS will provoke

User query analysis. Another source of test cases and validation is using actual user workloads – provided by a DBMS consumer, observed in production (for databases that are also hosted by the database developers), or similar. These can be either run in the same environment (for hosted databases) or added to test suites maintained by the database developers. The primary limitations with user workloads are:

- New database features will not be exercised until they are available to the DBMS's users
- Most user's schema, data, and queries are often nonpublic and may not be available to a DBMS developer (for numerous reasons, including legal restrictions on sharing certain types of data)

Randomized testing. Randomized testing is a technique through which test data is randomly generated based on given adequacy criteria/requirements/specifications. The system under test (SUT) is then executed against the test data and the corresponding outputs are evaluated to assess whether they conform to the expected results. The phases of random testing comprise

data generation, execution, and evaluation. There are two main approaches to constructing test cases, each with its strengths, limitations and challenges: (1) random test case generation and (2) mutation-based generation.

Random test case generation. Using this approach, test cases (queries) are constructed completely from scratch. A key challenge here is the creation of syntactically- and semantically-valid queries to help uncover deep DBMS bugs. This is typically achieved by implementing a query generator that consults the grammar and the semantics of the DBMS under test. A fundamental limitation is that such query generators are often tailored to specific SQL dialects and DBMSs.

Mutation-based generation. This approach synthesizes queries by modifying existing ones, known as seeds. These modifications include replacements, additions, deletions of tokens, expressions, or complete statements. Mutation-based generation results in grammatically-valid queries by making small changes to existing, valid seeds. This helps exercise different behaviors in the database, while preserving much of the structure and characteristics of the given seed queries. A primary limitation of mutation-based generation is that its effectiveness relies on quality of the available seed programs.

Black-box vs. grey-box query generation. Test case generation techniques can be also grouped into two different categories based on whether the generation process leverages feedback from SUT. *Grey-box test generation* allows guiding the creation of new test inputs by considering feedback from previous executions. This feedback can include code coverage [55], query plans, newly-explored paths and code regions, or the state of SUT. *Black-box test case generation* treats SUT as a “black box” that simply takes an input and produces an output, without knowing anything about its internals.

Existing tooling. Popular query generators for relational databases include the following. SQLsmith [48] is a fuzzer for SQL that can target PostgreSQL by generating random queries from scratch. It however sacrifices complexity for validity, as it only generates one statement in each query, without analysing the dependencies between these statements. Conversely, SQUIRREL [55] can generate queries that contain multiple statements and infer dependencies between them but can produce invalid ones. An emerging challenge is accounting for the trade-off between the complexity and the validity of generated queries. To address this, recent approaches, such as DynSQL [28], combine query generation and execution and use the database state to inform query generation. Similarly, the GoogleSQL Random Query Generator used in testing at Google by Spanner, BigQuery, etc.) uses the latest database state and information about the current contents of the database tables to inform the generation of queries that are more likely to be successful.

As subsequently detailed, limitations of existing tooling include:

- The generated workload may not reflect how real users use the database
- It is challenging to ensure the workload generation tests the DBMS features it is expected to
- Avoiding generating spurious or less interesting test cases requires increasing the sophistication of the generator (e.g. queries that do not parse, type-check, are not valid for the database schema, or will not have any results given the contents of the database)

4.4.6 Open problems/challenges

4.4.6.1 Fuzzing Challenges

To get better coverage of possible inputs to a DBMS and potentially detect more bugs, randomized generation of test workload (for example, a schema, data to populate the schema, and a sequence of SQL queries or database transactions) is a common pattern that finds a large number of bugs.

- **Test Case Quality:** Given an infinite input space, fuzzers should ideally generate “interesting” queries or workloads, but it is not immediately clear what “interesting” entails. The reason for this is that it is not known beforehand where the bugs hide, hence it is important to test a large share of possible inputs.
- **Test Case Complexity:** Fuzzers like SQLancer were able to detect many bugs in various DBMSs by constructing rather uncomplicated data and queries. To investigate the potential of finding bugs with more complex queries, e.g., involving a set of complicated join operations, insights into actual bugs that were found by customers would be helpful.
- **Feature Interaction & Feature Targetting:** A common source of errors is cross-feature interactions. Features in this sense are developer-defined and range from different physical operators to interactions between subsystems to specific code paths taken. It is desirable that a test suite validates interactions between combinations of those features. Most DBMS fuzzing tools do not provide a way to steer generation in a direction that targets a specific set of features. When that capability exists, it is generally not a guaranteed behavior, but instead involves careful tuning of probabilities. Generating queries for a particular feature would especially be helpful in testing new features, which tend to be less well-tested and less mature, and thus a potential source of bugs. Furthermore, by constructing queries that target specific features, such fuzzers could give confidence that relevant parts of the DBMS are tested.

4.4.6.2 Customer-Representativeness of Test Cases

In addition, the goal of testing can influence the desired properties of the generated test workload. For regression tests, a desirable property could be to resemble the workloads of typical customers. Generating these can be challenging since actual customer workloads are often not accessible directly, and accurately modeling them in synthetic workloads is difficult. On the other hand, for teams whose goal is to find as many bugs as possible (even if a customer might never encounter them), a focus on unusual or unexpected inputs, or particularly complex workloads, might be preferable to cover possible edge cases. In practice, there are no tools that provide both, and no tools provide a smooth way to transition between “closer to customer” and “closer to edge cases”).

4.4.6.3 Measuring Coverage

As with every testing method, fuzzers need to strike a balance between the test coverage and the cost of testing (time to create and maintain tests and test infrastructure, test execution time, and compute). We want to point out that test coverage here should be understood in a broader sense than simple code coverage. Determining a point of saturation is an open problem – at what point have we run enough tests?

4.4.6.4 Portability

Because different DBMSs come with different SQL dialects and hence differences in syntax and semantics, randomized query generators must often be adapted to be able to generate valid queries that cover all potential features of the system. This involves more work in the development and maintenance of the query generator. An open problem is the development of a portable query generator that can be applied to a variety of DBMSs while keeping the ability to generate complex queries.

An adjacent problem is the lack of compatibility/interoperability between existing query generators (and other parts of a complete test system, as discussed in cross-cutting problems). Currently, ideas from one research group have to be reimplemented by other research groups in order to integrate them into their own solution, and in the industry when these techniques are used, they usually are implemented per-database or per-SQL dialect.

4.4.7 Future Directions

Study of database differences. Another important future direction is to investigate the differences between database engines. This study should answer (1) to the which extent mainstream database engines differ, (2) what are the main sources of their differences, and (3) how do these differences affect the effectiveness of bug-finding tools?

Portable/Extensible query generation. Database engines exhibit distinct differences in their syntax and semantics. To address this challenge, there are numerous proposed approaches:

- *General query generator:* There is some evidence from the industrial partners that highlight that most of the database bugs are found via simple test cases and features, e.g., test cases that only contain `boolean` or `integer` data types. Therefore, it might be sufficient to come up with a general query generator that only covers the core part of SQL, which is commonly supported by databases. This general query generator should be extensible and support configuration options to (1) enable or disable database-specific features, or (2) combine simple test cases into more complex ones [28].
- *Learning-based approach:* A learning-based approach does not guarantee that the synthesized queries are valid. When a semantic or a syntax error is encountered, the approach examines the error message raised by the database engine, and adapts the generation process accordingly. For example, such a learning-based approach stops producing an erroneous combination of features.
- *Encoding database semantics (or part of it) into a specification:* It could be possible to encode part of database semantics through a specification. The benefit is that the constraints and the semantics of each database is described declaratively, rather than being hardcoded into the underlying generation process. Therefore, the input of a future and general query generator could be a specification that describes both the grammar and the semantics of a database under test. In this way, the generator produces queries that are both grammatically and semantically valid with regard to the given specification. For example, in a similar manner to the work of Dewey et al., [19] that relies on *constraint logic programming (CLP)*, one could encode the type system of each database engine into Datalog relations [12]. Based on these relations, we could then produce valid queries that adhere to the typing rules of each database. Notably, Datalog has been the foundational basis of many existing (relational) and emerging (native graph) query languages. As such, another research direction would consist of using it as a system-agnostic basis for specifying the semantics of classes of query languages, ranging from relational (recursive) queries to navigational ones for querying semi-structured data [45, 3] that rely on Regular

Path Queries (RPQs) and extensions (CRPQ, ECRPQ, DRPQ, GXPath, etc). Moreover, due to its generality, Datalog can also encode both syntactic (schema) constraints, as well as semantic ones, i.e., key constraints and functional dependencies. Hence, starting from the specification of a Datalog-based engine, one could generate semantically-aware test cases and leverage these for automated testing. Such test cases could be obtained from the Datalog specification, using property-based testing (PBT) techniques. Moreover, such specifications can also be formally verified, together with the correctness of the generators themselves, for example by leveraging existing plug-ins for automated theorem provers, such as ACL2 [14], and proof assistants, such as Isabelle [11], Agda [20], and, more recently, Coq [40]. Indeed, recent work has shown the promise of using PBT tools, such as the QuickChick plugin for Coq, to *automatically generate reliable generators* for programming language specifications, with high correctness guarantees. The challenges for adopting such methods for database testing are three-fold. The first issue concerns their *usability*, as identifying suitable properties to test, encoding theses, and integrating PBT into real-world database management system architectures is non-trivial. Second, their *extensibility* is also a concern, as such generators would need to be adapted to account for the idiosyncrasies (both syntactic and semantics) of each SUT. Third, there is an *expressiveness vs. performance trade-off* regarding the properties that such formal PBT tools support. An interesting future direction could be to combine such methods with other testing techniques, e.g., coverage-guided fuzzing [32] and combinatorial testing [23]. In addition to Datalog, another recent example is ISLa [49]. ISLa is a declarative specification language that extends context-free grammars and grammar-based fuzzers by constraining the generated inputs (e.g., a variable has to be defined before it is used).

Techniques/Metrics for guiding and evaluating the query generation process. Traditional code coverage metrics, such as line coverage, branch coverage, have been insufficient in effectively evaluating modern testing approaches [34, 13]. There is a need to devise new ways to guide and evaluate query generators. For example, we could use a property vector that indicates whether a certain query is interesting and expressive. Such a property vector could include both static (black-box) features, (e.g., the type and the order of SQL operators), and dynamic (grey-box) features (e.g., configuration parameters, data read/written). It is worth noting that this approach enables feature-aware test case generation. This means that by limiting or expanding the domain of property vectors, we can generate test cases that exhibit specific features, e.g., feature A and B, or feature A and not feature B. This could be useful for swarm testing [26].

Another example of an approach that goes beyond traditional code coverage metrics is inspired by the work of Park et al. [41] on conformance testing of JavaScript. We should investigate whether their feature-sensitive approach is applicable to the database world.

Running queries in different database states. In contrast to compiler testing, where the primary focus is on generating interesting inputs (programs), in database testing, we need to consider both test case generation *and* database state generation. This is because, certain database bugs might be triggered only when the database is in a certain state. Therefore, a future direction should run the generated queries in various database states. A database state includes database schema, logical and physical data, or database configuration.

Test Oracles

One of the core issues with automatically running test cases and automatically finding bugs is checking whether or not the result of a query is correct or expected. There are different approaches with validating that the result is correct.

4.4.8 Existing Techniques

4.4.8.1 Test Suites

Test suites manually specify the test oracle as assertions for a specific test input or scenario [24]. Virtually every DBMS has manually written test suites as part of their testing efforts. They are certainly beneficial to provide fast and reliable feedback to developers on the quality of critical features.

Historically these test suites are hard to reuse across engines:

- they are strongly tied to particular DBMS SQL syntax and semantics;
- they use custom test runners incompatible with other DBMSs or tech stacks;
- alternative test runners are hard to build because test case representation is not specified

4.4.8.2 Differential Testing

Differential testing compares the results of multiple systems and checks them for discrepancies. These systems could be:

- Actual different DBMSs supporting different SQL dialects, which makes their comparison difficult (see the RAGS system by Microsoft [47]);
- A reference engine implemented for that purpose, which can serve as an effective test oracle, but requires significant implementation effort (see more details below);
- the same DBMS, but with different configuration/optimization options.
- Customer query replay [53, 22, 52], where historical customer queries are replayed with a new release or feature enabled. Query results, performance, and cost can then be compared. Queries could be sampled for replay by a combination of random sampling, analysis of feature applicability or plan changes, or cost.
- a previous version of the same DBMS, to prevent regressing in behaviour. Note, that new features can't be tested with such an approach and you carry existing behavior into the future even if it's incorrect. Another variant of this approach is the use of expected results stored with tests [24] or in a database.

Comparing results between two systems is not free of challenges itself. Even very close systems (e.g. previous and current versions) might yield different results due to:

- inherit non-determinism or under specification in the SQL language;
- non-determinism in the query;
- floating point rounding errors due to different accumulation order;
- difference in corner case semantics (e.g. handling of NULL, rounding).

4.4.8.3 Reference Engine

A reference engine is a general-purpose oracle for a database that re-implements the database's semantics in a simpler way, greatly simplifying correct implementation.

Advantages.

- Checks result correctness, rather than just result differences
- Decouples test case generation from validation
- Can validate any supported workload
- Can handle non-determinism and imprecision
- Developers don't need to learn new skills to extend the engine
- Feature support added incrementally during feature development
- Have been extended to strictly serializable workload validation [17]
- Customers could develop and test against the reference
- Has been used for cross-database semantic compliance comparison [6]
- Workload / reference partitioning can improve scale [17]

Disadvantages.

- Large investment to build for existing engine
- Different DBMSs require different reference engines for their dialects
- Requires continuous maintenance to keep up with changes
- Limited scalability in terms of data size, intermediate result size, and query result size
- Limited published research and examples

The reference engine implements either all or a subset of the database API and semantics. For example, some reference engines only support query execution, and some support DDL, DML, indexes, queues, full-text search, etc. It is also possible to reuse some components that will not be verified by the reference, for example, the parser/resolver or scalar functions.

The reference engine should also encode non-determinism, ordering, and imprecision in its values and tuples. This enables the reference to logically return a set of possible correct results and test that the real engine result is in that set. The sophistication can vary from just a special Imprecise value to values supporting epsilons, complex partial ordering, or even multiple logical intermediate result sets.

Open Questions.

- Can we create a general or shared reference? Possibly define an algebra with configurable semantic options and each database family implements a parser and resolver to the shared algebra. Then operators, expressions, etc can be shared when possible.
- Can a reference engine be used to validate weaker transactional semantics than strict serializability?

Examples. Google Spanner has been system tested [17] using a reference engine [25]. ZetaSQL [25] engine's language compliance was also verified using a reference engine [6] (see 6. COMMON SQL DIALECT).

4.4.8.4 Metamorphic Testing

Metamorphic testing generates two related test inputs and checks a relation between their output. For example, many test oracles check that equivalent queries compute the same result.

Advantages include.

- Applicable to DBMSs using different dialects
- Multiple metamorphic relations can be proposed to test different SQL features.

Disadvantages include.

- Missing bugs when both two queries produce the same incorrect results
- Not always possible to cover all features

Logically Equivalent Queries. Generating logically equivalent queries that should return the same result but run a different code path is a way of finding bugs without needing to involve another engine. Examples of this include the work in SQLancer, where queries can be split up into three different queries that are combined through UNION ALL. Similarly, GROUP BY queries can be partitioned along the group, and certain aggregates can be split up (e.g. SUM(x) can have its child tables partitioned and then unified).

One issue with generating logically equivalent queries is that the same code path might be triggered within the same engine so that the logically equivalent queries both suffer from the same bug and produce the wrong result. A potential way of discovering this is to look at e.g. code coverage and verify that the queries actually take different code paths.

Physically Equivalent Queries. Queries that are not logically equivalent might still be equivalent given a specific data set. For example, a query that returns 50 rows, will not return a different data set if a LIMIT 100 is added to the end. Similarly, we can add a filter that has no effect, a filter that filters out rows that would have been filtered out in a later step, or a LEFT JOIN that has no matches on the right side.

Rerunning the Same Query. When running a query on a given schema, the same result should always be produced (except for issues of determinism as discussed later). It can be useful to run the same queries on a database while changing other variables.

Equivalent Schema. We can execute the same query on the same schema, but where the data sources are stored in a different manner. For example, the table can be stored in a different data format (e.g. a Parquet file vs the databases' native format, splitting data into multiple files, etc). We can also run the query over a *view* versus the result of that view stored in a table.

Modifying the Environment. Another variable that can be changed is the environment. For example, we can run the same query on different operating systems or different processors and the same result should be returned.

Toggling Feature Flags. A query can be run with specific features enabled or disabled and the query should still produce the same result in both scenarios. For example, we can enable/disable the optimizer, enable/disable certain types of operators (e.g. hash joins), enable/disable parallelism, enable/disable spilling data to disk, etc.

Nop Rows. We can insert rows into the base table that should be filtered out by the query and not be part of the end result. For example, we can insert rows that should be filtered out by a WHERE clause or by a join condition, or that input a NULL into an aggregate or 0 into a SUM. These rows should not affect the query – allowing us to re-run the query both with and without the nop rows.

Future Problems

4.4.9 Study what the missed bugs are

Various test oracles have been proposed that were effective in finding bugs. However, it has not been systematically investigated what classes of bugs existing test oracles overlook. We

believe that systematic empirical studies based on issue trackers and postmortems could shed some light on bugs existing test oracles fail to find. This would be the first step to identify gaps for new test oracles.

4.4.10 Compose or generate test oracles

Existing approaches have proposed many manually-crafted metamorphic relations of SQL (e.g., Ternary Logic Partitioning) as test oracles to detect different kinds of logic bugs. However, there should be many other metamorphic relations in standard SQL and DBMS-specific features, which we have not explored. This raises an interesting research question: How to automatically mine possible metamorphic relations from SQL specification, execution traces of queries, etc.?

From another perspective, a query should return the same result on a logically equivalent data (e.g., the same database with different configuration, and the same view with different physical schemas). This raises another interesting research question: Given a query q and database db that the query operates on, how to generate interesting equivalent databases on which q can still return the same result?

An example might include generating a query which is semantically equivalent to `SELECT * FROM T1`. Then that generated query could be used to create a view or materialized view $V1$ which is equivalent to the base table. Further objects could then be generated based on both $T1$ and $V1$ to create even more complex test objects which are equivalent to $T1$.

Assume that we have a couple of equivalent patterns for specific SQL features, is it necessary to compose individual patterns in complex ones? In ideal cases, complex test oracles can quickly trigger more bugs in a single testing process, instead of trying individual patterns one by one.

4.4.11 Coverage with respect to oracles

With many generated queries and many test oracles per query, execution time for the test grows dramatically. It is essential to understand that a particular test oracle provides more (useful) coverage for a given query. It is also important to evaluate if sufficient validated coverage has been achieved by all oracles prior to a release. This could be seen as a prioritization problem. We would ideally want to run all queries with all possible oracles, but this could be prohibitively expensive. Picking first test oracles for query which will yield “better” coverage is crucial.

We also want to evaluate if the oracles we choose can validate all aspects of the database and identify validation gaps that need to be addressed.

Prioritization might depend on

- prior knowledge about coverage, e.g. we want to verify certain oracles (invariants) for majority of the test queries;
- or the query itself, e.g. we expect certain query shapes provide more coverage with certain oracles.

Another approach is oracle-aware test generation. How can we generate queries for a given test oracle to quickly yield “better” coverage?

One potential method to generate oracle coverage for differential testing-based approaches would be to track coverage on each side and identify the different features or code paths covered. Coverage in a reference engine might be achieved similarly, as the difference would exclude any shared code, for example, a shared parser or scalar function implementations.

4.4.12 Designing for Testability

Many current testing tools and their test oracles have been designed independently from the DBMS under test. Certain additional features in DBMS could be useful in testing only and relatively cheap to implement or expose:

- SQL parser as a separate component,
- SQL analyzer as a separate component,
- ability to get metadata about query from the engine, e.g. cost estimate, determinism property, etc. This might include things that can be determined statically or dynamically during execution.
- clearly distinguish between different error types. E.g. syntax, semantics (constraint violation) or internal.
- provide more details for internal errors for tests only. Intention is to help with debugging of failed tests.
- allow to extract variable information from error messages programmatically, instead of using RegEx for that.
- Enforce alternative query plans (all plans vs. choosing other plausible plans dependent on the data distribution; e.g., select top N plans)
- ability to enable / disable optimization flags. This provides test oracles and aids debugging.
- Self-check options (e.g., https://www.sqlite.org/prAGMA.html#prAGMA_integrity_check) to expose potential internal errors masked or invariants only enabled in self-check mode (e.g. debug asserts).
- Deterministic execution option
- Deterministic execution/simulation testing
- allow to change table statistics (e.g. histograms) not used for correctness (e.g., often required for MIN, MAX, is NULL). This will help to steer queries into different plans without changing data itself, which might be costly.
- white-box testing of the optimizer (e.g., plan stability). Exposing optimizer API for testing will help with tests which target query optimization tests directly without paying execution costs. Query optimizer is large and important enough component to merit implementation and maintenance costs for this test-only API.

4.4.13 Non-deterministic queries

Queries can be ambiguous and yield different results. Common examples include insufficient ordering with limit/offset, floating point imprecision in aggregation, non-deterministic functions, and runtime errors which are avoided due to short-circuiting during execution. This is a problem when test oracles rely on a specific result or compare the results of SQL semantically equivalent operations without modeling this non-determinism. Test oracles would ideally be free of false positives, which is why such ambiguous queries should be identified or avoided. (Or the test oracle should model these sorts of non-determinism, which adds additional complexity to a test oracle.)

4.4.14 Hitting duplicate bugs

Testing tools might frequently hit the same underlying bug. To continue exploring the state space and find additional issues it is valuable to build tools to identify and ignore duplicate issues. This problem is inherently difficult, as deciding whether two bug-inducing test cases are duplicates or not often requires understanding the bug at a source code level. In practice,

matchers are used that classify a bug-inducing test case based on an error code, configuration, SQL text, or query plan. Often, constructing these matchers can be time consuming and manual. Some databases have implemented failure signature or feature extraction with automated clustering, which is integrated with bug management to speed up bug detection, provide prioritization data, and accumulate test cases for each issue.

Another interesting direction is to prevent testing tools from repeatedly triggering the same bugs. For example, we could try to guide the query generator not to generate test cases similar to the cases that have already triggered bugs (code-coverage guiding and query-plan guiding can do it to some extent?). It seems like a cross-cutting issue in test-case generation, test oracle, and test-case reduction.

4.4.15 Dealing with Semantic and Runtime Errors

It can often be difficult to avoid statements that result in semantic errors. For example, avoiding an `INSERT` statement to a table with `UNIQUE` constraints might require encoding constraints, which can be complicated (e.g., collation handling with strings). Another common example is generated expressions evaluating to zero, resulting in division by zero errors. Current testing tools typically tackle this in a simpler way, by generating potentially semantically-invalid statement, and annotating the statements using a list of so-called *expected errors*. For example, an `INSERT` statement might be annotated with an expected error “`UNIQUE constraint failed`”. More systematic ways could be explored; for example, the DBMSs could expose different list of expected errors (e.g., syntax errors, run-time errors, or internal errors). For example, this is done by the ZetaSQL engine in code here: https://github.com/google/zetasql/blob/master/zetasql/compliance/runtime_expected_errors.cc – a common model for enumerating these sorts of errors and categorizing them would help in creating any general testing solution. Systematically exploring unexpected errors may be another solution for this challenge.

4.4.16 Monolithic/Coupled Testing Tools

Existing testing tools couple their test generation and test oracles, which limits them from finding more bugs triggered by uncovered test case patterns. For example, query partitioning requires that the test case contain predicates to be partitioned. If the test case does not contain predicates (e.g., `INSERT` statements), query partitioning does not work anymore. Decoupling test-case generation and test oracles is not easy, as the test oracles usually require designated test cases to be generated. Using reference engines helps decouple test-case generation and test oracles if the reference engines can process general queries. However, implementing reference engines consumes lots of effort. Another interesting direction is to make testing tools provide interfaces for different test-case generation and different test oracles. The test oracles can choose their suitable test-case generation. It helps the decoupling, but the question is how to design interfaces that are extensible for both test-case generation and test oracles. We may develop a domain specification language that specifies what features are supported by a test oracle.

4.4.17 Non-functional issues

Most existing test oracles focus on logic bugs, which cause the DBMS to compute an incorrect result. However, DBMSs can also be affected by other kinds of issues such as performance issues, issues in cost models, memory consumption, or metastable failures. In addition, for distributed systems, errors could be masked by retries/replication and other techniques,

meaning that they do not surface as logic bugs. New test oracles are needed to tackle such issues, while preventing test oracles from being coupled too tightly with the DBMS under test.

Additionally to different non-functional issues, DBMS needs to be further developed and operated. This brings its own set of non-functional challenges. The next step in the development life cycle is deployment. Deployment requires choosing the deployment model, e.g. blue-green deployment, rolling update, full stop and restart, etc. Any chosen deployment approach should be covered by testing to make sure deployment does not introduce issues in production, most notably different versions of a DBMS talking to each other should work correctly.

DBMSs usually have additional features to support operations. These might include backup, restore, various methods of introspection, integration with cluster management for distributed systems and alike. The full list of these features is highly specific to a particular DBMS and environment it targets to run in.

Backward / forward compatibility is another non-functional requirement. Which might extend to file formats, configuration formats, different client versions.

Non-functional requirements are often orthogonal to correctness requirements and test oracles can be reused.

Bug Analysis

4.4.18 Problem Statement

The previous chapters talk about how to generate a large volume of test cases to hit as many bugs as possible. But, because debugging is still a manual task that is done by developers and cannot be automated, dealing with these failing test cases has two major problems.

1. Many different test cases will fail due to the same underlying root cause, making the number of failing tests in orders of magnitudes higher than the number of bugs.
2. The failing test cases are likely to contain very complex queries (and a lot of data) which makes it hard for developers to find the issue.

Ideally the tester only reports a single minimal test case per bug.

4.4.19 Existing Techniques

4.4.19.1 Test Case Reduction Techniques and Tools

We have witnessed various techniques proposed to reduce the complexity of bug-triggering queries and databases, summarized as follows.

1. **String-level manipulation:** This kind of technique involves simple string operations without understanding the SQL grammar. String-level manipulation techniques can be general and relatively easy to implement. The general test case reduction methods like delta debugging [54] can be used for performing the reduction for SQL queries. This kind of technique is simple and already pretty effective in practice [5] but it might cause many invalid queries during the reduction.
2. **Syntax-level manipulation:** This kind of technique tokenizes the SQL queries or parses the SQL queries and operates on the Abstract Syntax Trees (AST). It is a more fine-grain technique than string-level manipulation, which can always generate syntax-valid queries. There are several problems with this kind of technique. First, it is challenging to be

portable to the SQL dialects. Engineering efforts will be required to implement the corresponding parsers and tokenizers. Second, it is still possible to mess up the semantics of the queries, but it often works fine when just removing the leaf nodes or sub-trees on the AST. Third, it may not be able to get to the global minimum by

3. **Semantic-level manipulation:** This kind of technique parses and semantically analyzes the queries to reduce the test cases. It can always create semantically valid queries. The high-level idea is to transform the query into a simplified version while handling the schema and the data dependency. Such transformations are not necessarily to be semantic preserving but need to generate executable queries. It requires extensive efforts to analyze the queries, which may reuse some components of the database management systems, especially the components related to the query optimizations.

The deeper the understanding of the query is (string-level, syntax-level, semantic-level), the easier it is to generate a valid query (syntactically and semantically valid) and to find a global instead of a local minimum. However, at one point the reducer almost reached the complexity of half a DBMS and also is not portable anymore. However, such a reducer can reuse parts of the actual DBMS (if they are generic enough).

Following these techniques, practitioners in industry and academia have implemented diverse tools for various purposes. We collect a set of representative tools as follows.

- **SQLreduce:**¹ This reducer tries to remove tokens by parsing SQL statements into Abstract Syntax Trees (ASTs) and is specific to PostgreSQL. It belongs to the second method *Syntax-level manipulation*.
- **C-Reduce:**² This reducer is one of the most commonly used reducers for the C/C++ languages and also works well for SQL statements. It deploys various heuristics to conduct a *String-level manipulation* by gradually removing strings without understanding the semantics of test cases.
- **SQLright Minimizer:**³ This reducer minimizes test cases by reducing the tokens identified in a customized Intermediate Representation (IR). This reducer was implemented as a component for the database testing work *SQLRight* [33]. It belongs to the method *Syntax-level manipulation*.
- **SQLMin:**⁴ Similar to *SQLreduce*, this reducer also minimizes test cases by removing the tokens based on ASTs. This tool was implemented as a component for a database testing work *APOLLO* [29] for finding regression performance bugs. This technique belongs to the method *Syntax-level manipulation*.
- **SQLancer Reducer:**⁵ Similar to *SQLreduce* and *SQLMin*, this reducer also parses SQL statements into ASTs and reduce them by dropping the tokens identified on ASTs. This method was implemented as a debugging tool for the database testing framework *SQLancer*. This technique belongs to the method *Syntax-level manipulation*.
- **Percona Reducer:**⁶ This reducer adopts several predefined heuristic rules to identify the tokens from SQL statements without understanding SQL semantics. It is included in the toolbox Percona-QA for database quality assurance, and belongs to the method *Syntax-level manipulation*.

¹ <https://github.com/credativ/sqlreduce>

² <https://github.com/csmith-project/creduce>

³ <https://github.com/PSU-Security-Universe/sqlright/blob/main/SQLite/scripts/minimize.py>

⁴ https://github.com/sslabs-gatech/apollo/blob/master/src/sqlfuzz/sql_minimizer.py

⁵ <https://github.com/sqlancer/sqlancer/blob/main/docs/testCaseReduction.md>

⁶ <https://github.com/Percona-QA/percona-qa/blob/master/reducer.sh>

4.4.19.2 Test Case Deduplication Techniques and Tools

Depending on the failure type of the failing test case, different deduplication techniques can be used. While some techniques are completely unaware of both the test case (query and dataset) and the failure, others use the test case or the failure to determine whether two failures have the same root cause.

Here is a list of known techniques to help identify duplicated bug reports in the context of SQL test cases.

1. **Query Reduction:** Two failing SQL test cases have (likely) the same root cause if they can be reduced to the same (minimal) test case. This requires both test cases being reduced to the same minimum, which we mentioned earlier can be hard (depending on the mutation level being used). If the reducer works perfectly, all queries with the same root cause can be reduced to the same minimal reproduction test case and there are no false positives.
2. **Feature-based Clustering:** Given the test case that triggers the bug, record what features have been triggered along running the query. If the test case triggers new combination of features, it should be put into a new bucket. This work may leverage error information to extract partial features to help achieve simple implementation.
3. **Distance-based Ranking:** This technique doesn't deduplicate the test failures, but instead tries to order the test failures such that diverse, interesting test cases are highly ranked. The idea of distance-based ranking is to check what code/feature is covered by a bug-triggering test case and try to build a vector to describe each test case. Only if the vector has a long-enough distance from existing ones, we will put the test case into a new bucket for further analysis. One example for distance-based ranking is Query Taming [15].
4. **Change Bisection:** Two failing test cases have (likely) the same root cause if two queries appear the first time in the same change (e.g. git commit). This is a generic technique that doesn't need to be aware of the failure or the test case, making it simple to implement across different systems. However, it is orders of magnitudes more time and resource consuming than other techniques.

The change found might be the one that surfaced the problem, not the one introducing it. (An idea is to combine the change diff with coverage information of the running query to decide whether the change introduced the bug or just surfaced it.) For the purpose of bug deduplication however, that is usually good enough.

This method can have both false positives (commit introduces two different bugs) and false negatives (it's the same root cause, but one surfaces at a different commit). Prior query reduction (even best-effort) useful to remove unneeded features that (a) might not be supported in earlier versions or (b) trigger other bugs in older versions.

5. **Crash Stack Trace Matching:** For hard system crashes the stack traces can be matched to figure out if the crashes happened in the same place. Many such algorithms do already exist, like TraceSim [51] and FaST [46] that are already used for deduplicating fuzzer crashes [1]. For code generating DBMSs however, the stack trace information might be incomplete or without comparable symbols/function names.

A common practice in industry (like Google) is to combine the feature-based clustering and committing-based bisecting. The former is efficient but less effective, due to the complicated relations between bugs and triggered features, like many-to-many relations. The latter is more accurate but costly, since we need to run different code versions and set up various environments in a binary-search style to locate the first commit or unique parameter that leads to the bug.

Bisecting is also effective and in addition allows to directly assign the found bug to a developer (the owner of the change), however requires many resources and therefore is only doable for big companies.

Unfortunately, we did not find any automated tools for SQL test case deduplication, only for ranking them. The failures of most generic fuzzers are crashes, which can be deduplicated using their stack traces. In a DBMS however, many failures are not crashes but correctness issues.

4.4.20 Open Questions and Future Directions

4.4.20.1 Test case reduction

We identify the following questions of test case reduction are open and have no widely accepted solutions.

1. **Metric.** What is a good metric for the minimum test case and how hard is it to reach the global minimum? (global minimum required for test case deduplication) It is challenging to know the global minimum, so it is valuable to have a metric to evaluate how good a minimum test case is.
2. **Evaluation.** How to fairly evaluate reduction methods? There are already several reduction tools for SQL, but it's hard to compare them. Which reduction methods do they use, how minimal do the test cases get and what are their weaknesses?
3. **Generality.** Can we build a general reduction tool across database systems? Different database systems have various different SQL dialects, which affects the semantic analysis for reduction. Snowflake built a semantic-level reducer that might get open-sourced at one point. How useful will it be for other DBMSs due to dialect and semantic differences?
4. **Complexity.** Is writing (semantic-level) mutation rules a trivial task or as complex as writing a whole optimizer?

4.4.20.2 Test case deduplication

1. **Metric.** What metrics do we use for evaluating the uniqueness of bugs? Existing methods evaluate duplicated bugs by examining their input shapes and first-introduce-commits. However, these metrics are approximations, and false alarms exist. Towards to a clearly defined goal of test case deduplication and a fair evaluation method, it is expected to have more accurate metrics to evaluate the uniqueness of test cases.
2. **False Alarms.** How would false alarms affect the test case deduplication? Can we avoid false alarms? False alarms refer to either duplicated bugs that are not deduplicated or unique bugs that are deduplicated. False alarms may happen, and it is unclear about its impact on bug analysis. If false alarms have a significant impact, can we fully avoid the false alarms?
3. **Efficiency.** How to deduplicate test cases efficiently? Bisection, an existing method that deduplicates bugs, requires hundreds of thousands of builds for different commits of a database, which takes non-trivial time and computing resources. How can we deduplicate test cases within fewer resources?
4. **Compatibility with Test Oracles.** How to duplicate the test cases found by non-crash test oracles? Existing methods, such as bisection and stack trace, only work for crash bugs, which can be found by a single SQL statement. While, for the bugs that require a test oracle to check multiple SQL statements, how do we do the deduplication?

References

- 1 Rui Abreu, Franjo Ivančić, Filip Nikšić, Hadi Ravanbakhsh, and Ramesh Viswanathan. Reducing time-to-fix for fuzzer bugs. In *2021 36th IEEE/ACM International Conference on Automated Software Engineering (ASE)*, pages 1126–1130. IEEE, 2021.
- 2 Renzo Angles, János Benjamin Antal, Alex Averbuch, Peter A. Boncz, Orri Erling, Andrey Gubichev, Vlad Haprian, Moritz Kaufmann, Josep Lluís Larriba-Pey, Norbert Martínez-Bazan, József Marton, Marcus Paradies, Minh-Duc Pham, Arnau Prat-Pérez, Mirko Spasic, Benjamin A. Steer, Gábor Szárnyas, and Jack Waudby. The LDBC social network benchmark. *CoRR*, abs/2001.02299, 2020.
- 3 Renzo Angles, Marcelo Arenas, Pablo Barcelo, Peter Boncz, George Fletcher, Claudio Gutierrez, Tobias Lindaaker, Marcus Paradies, Stefan Plantikow, Juan Sequeda, Oskar van Rest, and Hannes Voigt. G-CORE: A Core for Future Graph Query Languages. In *SIGMOD*, pages 1421–1432, 2018.
- 4 T. Arvin. Comparison of different sql implementations. <http://troels.arvin.dk/db/rdbms> (visited: 2023-11), 2017.
- 5 Jinsheng Ba and Manuel Rigger. Testing database engines via query plan guidance. In *The 45th International Conference on Software Engineering (ICSE'23)*, May 2023.
- 6 David F. Bacon, Nathan Bales, Nico Bruno, Brian F. Cooper, Adam Dickinson, Andrew Fikes, Campbell Fraser, Andrey Gubarev, Milind Joshi, Eugene Kogan, Alexander Lloyd, Sergey Melnik, Rajesh Rao, David Shue, Christopher Taylor, Marcel van der Holst, and Dale Woodford. Spanner: Becoming a sql system. In *Proceedings of the 2017 ACM International Conference on Management of Data*, SIGMOD '17, page 331–343, New York, NY, USA, 2017. Association for Computing Machinery.
- 7 Jason Baker, Chris Bond, James C Corbett, JJ Furman, Andrey Khorlin, James Larson, Jean-Michel Leon, Yawei Li, Alexander Lloyd, and Vadim Yushprakh. Megastore: Providing scalable, highly available storage for interactive services. 2011.
- 8 Jonathan Bell, Owolabi Legunsen, Michael Hilton, Lamyaa Eloussi, Tiffany Yung, and Darko Marinov. Deflaker: Automatically detecting flaky tests. In *Proceedings of the 40th international conference on software engineering*, pages 433–444, 2018.
- 9 Maciej Besta, Robert Gerstenberger, Emanuel Peter, Marc Fischer, Michal Podstawski, Claude Barthels, Gustavo Alonso, and Torsten Hoefler. Demystifying graph databases: Analysis and taxonomy of data organization, system designs, and graph queries. *ACM Comput. Surv.*, 56(2):31:1–31:40, 2024.
- 10 Dirk Beyer, Jan Haltermann, Thomas Lemberger, and Heike Wehrheim. Decomposing software verification into off-the-shelf components: An application to cegar. In *Proceedings of the 44th International Conference on Software Engineering*, ICSE '22, page 536–548, New York, NY, USA, 2022. Association for Computing Machinery.
- 11 Lukas Bulwahn. The new quickcheck for isabelle – random, exhaustive and symbolic testing under one roof. In *CPP*, volume 7679 of *Lecture Notes in Computer Science*, pages 92–108. Springer, 2012.
- 12 S. Ceri, G. Gottlob, and L. Tanca. What You Always Wanted to Know About Datalog (And Never Dared to Ask). *IEEE Transactions on Knowledge and Data Engineering*, pages 146–166, 1989.
- 13 Stefanos Chaliasos, Thodoris Sotiropoulos, Diomidis Spinellis, Arthur Gervais, Benjamin Livshits, and Dimitris Mitropoulos. Finding typing compiler bugs. In *Proceedings of the 43rd ACM SIGPLAN International Conference on Programming Language Design and Implementation*, PLDI 2022, page 183–198, New York, NY, USA, 2022. Association for Computing Machinery.

- 14 Harsh Raju Chamarthi, Peter C. Dillinger, Matt Kaufmann, and Panagiotis Manolios. Integrating testing and interactive theorem proving. In *ACL2*, volume 70 of *EPTCS*, pages 4–19, 2011.
- 15 Yang Chen, Alex Groce, Chaoqiang Zhang, Weng-Keen Wong, Xiaoli Fern, Eric Eide, and John Regehr. Taming compiler fuzzers. In *Proceedings of the 34th ACM SIGPLAN conference on Programming language design and implementation*, pages 197–208, 2013.
- 16 Brian F. Cooper, Adam Silberstein, Erwin Tam, Raghu Ramakrishnan, and Russell Sears. Benchmarking cloud serving systems with ycsb. In *Proceedings of the 1st ACM Symposium on Cloud Computing*, SoCC '10, page 143–154, 2010.
- 17 Jay Corbett. Randomized testing of cloud spanner. https://medium.com/@jcorbett_26889/randomized-testing-of-cloud-spanner-5286f1eaba75 (visited: 2023-11), 11 2023.
- 18 Alin Deutsch, Nadime Francis, Alastair Green, Keith Hare, Bei Li, Leonid Libkin, Tobias Lindaaker, Victor Marsault, Wim Martens, Jan Michels, Filip Murlak, Stefan Plantikow, Petra Selmer, Oskar van Rest, Hannes Voigt, Domagoj Vrgoc, Mingxi Wu, and Fred Zemke. Graph pattern matching in GQL and SQL/PGQ. In *SIGMOD Conference*, pages 2246–2258. ACM, 2022.
- 19 Kyle Dewey, Jared Roesch, and Ben Hardekopf. Fuzzing the Rust typechecker using CLP. In *Proceedings of the 30th IEEE/ACM International Conference on Automated Software Engineering*, ASE '15, page 482–493. IEEE Press, 2015.
- 20 Peter Dybjer, Qiao Haiyan, and Makoto Takeyama. Verifying haskell programs by combining testing, model checking and interactive theorem proving. *Inf. Softw. Technol.*, 46(15):1011–1025, 2004.
- 21 Moritz Eck, Fabio Palomba, Marco Castelluccio, and Alberto Bacchelli. Understanding flaky tests: The developer’s perspective. In *Proceedings of the 2019 27th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering*, pages 830–840, 2019.
- 22 Leonidas Galanis, Supiti Buranawanachoke, Romain Colle, Benoît Dageville, Karl Dias, Jonathan Klein, Stratos Papadomanolakis, Leng Leng Tan, Venkateshwaran Venkataramani, Yujun Wang, and Graham Wood. Oracle database replay. In *Proceedings of the 2008 ACM SIGMOD International Conference on Management of Data*, SIGMOD '08, page 1159–1170, New York, NY, USA, 2008. Association for Computing Machinery.
- 23 Harrison Goldstein, John Hughes, Leonidas Lampropoulos, and Benjamin C. Pierce. Do judge a test by its cover – combining combinatorial and property-based testing. In *ESOP*, volume 12648 of *Lecture Notes in Computer Science*, pages 264–291. Springer, 2021.
- 24 Google. Zetasql compliance tests. <https://github.com/google/zetasql/tree/master/zetasql/compliance> (visited: 2023-11), 11 2023.
- 25 Google. Zetasql reference engine. https://github.com/google/zetasql/tree/master/zetasql/reference_impl (visited: 2023-11), 11 2023.
- 26 Alex Groce, Chaoqiang Zhang, Eric Eide, Yang Chen, and John Regehr. Swarm testing. In *Proceedings of the 2012 International Symposium on Software Testing and Analysis*, ISSTA 2012, page 78–88, New York, NY, USA, 2012. Association for Computing Machinery.
- 27 Muhammad Ali Gulzar, Matteo Interlandi, Seunghyun Yoo, Sai Deep Tetali, Tyson Condie, Todd Millstein, and Miryung Kim. Bigdebug: Debugging primitives for interactive big data processing in spark. In *Proceedings of the 38th International Conference on Software Engineering*, pages 784–795, 2016.
- 28 Zu-Ming Jiang, Jia-Ju Bai, and Zhendong Su. DynSQL: Stateful fuzzing for database management systems with complex and valid SQL query generation. In *32nd USENIX Security Symposium (USENIX Security 23)*, pages 4949–4965, Anaheim, CA, August 2023. USENIX Association.

- 29 Jinho Jung, Hong Hu, Joy Arulraj, Taesoo Kim, and Woonhak Kang. APOLLO: Automatic Detection and Diagnosis of Performance Regressions in Database Systems. In *Proceedings of the 46th International Conference on Very Large Data Bases (VLDB 2020)*, Tokyo, Japan, aug 2020.
- 30 Kevin E. Kline, Daniel Kline, and Brand Hunt. *SQL in a nutshell – a desktop quick reference (3. ed.)*. O’Reilly, 2008.
- 31 Wing Lam, Patrice Godefroid, Suman Nath, Anirudh Santhiar, and Suresh Thummalapenta. Root causing flaky tests in a large-scale industrial setting. In *Proceedings of the 28th ACM SIGSOFT International Symposium on Software Testing and Analysis*, pages 101–111, 2019.
- 32 Leonidas Lampropoulos, Michael Hicks, and Benjamin C. Pierce. Coverage guided, property based testing. *Proc. ACM Program. Lang.*, 3(OOPSLA):181:1–181:29, 2019.
- 33 Yu Liang, Song Liu, and Hong Hu. Detecting Logical Bugs of DBMS with Coverage-based Guidance. In *Proceedings of the 31st USENIX Security Symposium (USENIX 2022)*, Boston, MA, aug 2022.
- 34 Vsevolod Livinskii, Dmitry Babokin, and John Regehr. Random testing for C and C++ compilers with YARPGen. *Proc. ACM Program. Lang.*, 4(OOPSLA), November 2020.
- 35 Qingzhou Luo, Farah Hariri, Lamyaa Eloussi, and Darko Marinov. An empirical analysis of flaky tests. In *Proceedings of the 22nd ACM SIGSOFT international symposium on foundations of software engineering*, pages 643–653, 2014.
- 36 Omar S. Navarro Leija, Kelly Shiptoski, Ryan G. Scott, Baojun Wang, Nicholas Renner, Ryan R. Newton, and Joseph Devietti. Reproducible containers. In *Proceedings of the Twenty-Fifth International Conference on Architectural Support for Programming Languages and Operating Systems, ASPLOS ’20*, page 167–182, New York, NY, USA, 2020. Association for Computing Machinery.
- 37 Ryan Rhodes Newton. Hermit: Deterministic linux for controlled testing and software bug-finding. <https://web.archive.org/web/20231102092943/https://developers.facebook.com/blog/post/2022/11/22/hermit-deterministic-linux-testing/>, 2022. Accessed 2023-11-02.
- 38 Robert O’Callahan, Chris Jones, Nathan Froyd, Kyle Huey, Albert Noll, and Nimrod Partush. Engineering record and replay for deployability. In *2017 USENIX Annual Technical Conference (USENIX ATC 17)*, pages 377–389, 2017.
- 39 Christopher Olston and Benjamin Reed. Inspector gadget: A framework for custom monitoring and debugging of distributed dataflows. In *Proceedings of the 2011 ACM SIGMOD International Conference on Management of data*, pages 1221–1224, 2011.
- 40 Zoe Paraskevopoulou, Catalin Hritcu, Maxime Dénès, Leonidas Lampropoulos, and Benjamin C. Pierce. Foundational property-based testing. In *ITP*, volume 9236 of *Lecture Notes in Computer Science*, pages 325–343. Springer, 2015.
- 41 Jihyeok Park, Dongjun Youn, Kanguk Lee, and Sukyoung Ryu. Feature-sensitive coverage for conformance testing of programming language implementations. *Proc. ACM Program. Lang.*, 7(PLDI), jun 2023.
- 42 Owain Parry, Gregory M Kapfhammer, Michael Hilton, and Phil McMinn. A survey of flaky tests. *ACM Transactions on Software Engineering and Methodology (TOSEM)*, 31(1):1–74, 2021.
- 43 Transaction Processing and Performance Council. Transaction processing and performance council. <https://tpc.org/>.
- 44 The Web Standards Project. The web standards compliance acid tests.
- 45 Juan L. Reutter, Miguel Romero, and Moshe Y. Vardi. Regular Queries on Graph Databases. *Theory of Computing Systems*, 61(1):31–83, 2017.

- 46 Irving Muller Rodrigues, Daniel Aloise, and Eraldo Rezende Fernandes. Fast: A linear time stack trace alignment heuristic for crash report deduplication. In *Proceedings of the 19th International Conference on Mining Software Repositories*, pages 549–560, 2022.
- 47 Donald R Slutz. Massive stochastic testing of sql. In *VLDB*, volume 98, pages 618–622, 1998.
- 48 SQLsmith. Sqlsmith. <https://github.com/anse1/sqlsmith> (visited: 2023-11), 11 2023.
- 49 Dominic Steinhöfel and Andreas Zeller. Input invariants. In *Proceedings of the 30th ACM Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering*, ESEC/FSE 2022, page 583–594, New York, NY, USA, 2022. Association for Computing Machinery.
- 50 Alexander Thomson and Daniel J Abadi. The case for determinism in database systems. *Proceedings of the VLDB Endowment*, 3(1-2):70–80, 2010.
- 51 Roman Vasiliev, Dmitriy Koznov, George Chernishev, Aleksandr Khvorov, Dmitry Luciv, and Nikita Povarov. Tracesim: a method for calculating stack trace similarity. In *Proceedings of the 4th ACM SIGSOFT International Workshop on Machine-Learning Techniques for Software-Quality Evaluation*, pages 25–30, 2020.
- 52 Ming-Chuan Wu, Jingren Zhou, Nicolas Bruno, Yu Zhang, and Jon Fowler. Scope playback: Self-validation in the cloud. In *Proceedings of the Fifth International Workshop on Testing Database Systems*, DBTest '12, New York, NY, USA, 2012. Association for Computing Machinery.
- 53 Jiaqi Yan, Qiuye Jin, Shrainik Jain, Stratis D. Viglas, and Allison Lee. Snowtrail: Testing with production queries on a cloud database. In *Proceedings of the Workshop on Testing Database Systems*, DBTest'18, New York, NY, USA, 2018. Association for Computing Machinery.
- 54 Andreas Zeller and Ralf Hildebrandt. Simplifying and isolating failure-inducing input. *IEEE Transactions on Software Engineering*, 28(2):183–200, 2002.
- 55 Rui Zhong, Yongheng Chen, Hong Hu, Hangfan Zhang, Wenke Lee, and Dinghao Wu. SQUIRREL: testing database management systems with language validity and coverage feedback. In *CCS*, pages 955–970. ACM, 2020.
- 56 Jingyu Zhou, Meng Xu, Alexander Shraer, Bala Namasivayam, Alex Miller, Evan Tschannen, Steve Atherton, Andrew J Beamon, Rusty Sears, John Leach, et al. Foundationdb: A distributed unbundled transactional key value store. In *Proceedings of the 2021 International Conference on Management of Data*, pages 2653–2666, 2021.

Participants

- Jinsheng Ba
National University of
Singapore, SG
- Lawrence Benson
Hasso-Plattner-Institut,
Universität Potsdam, DE
- Carsten Binnig
TU Darmstadt, DE
- Ankush Desai
Amazon – Cupertino, US
- Adam Dickinson
Snowflake Computing Inc. –
Seattle, US
- Wensheng Dou
Chinese Academy of Sciences –
Beijing, CN
- Stefania Dumbrava
ENSIEE – Paris, FR
- Moritz Eyssen
Snowflake – Berlin, DE
- Tim Fischer
Universität Tübingen, DE
- Florian Gerlinghoff
MotherDuck – Amsterdam, NL
- Torsten Grust
Universität Tübingen, DE
- Muhammad Ali Gulzar
Virginia Polytechnic Institute –
Blacksburg, US
- Denis Hirn
Universität Tübingen, DE
- Hong Hu
Pennsylvania State University –
University Park, US
- Zu-Ming Jiang
ETH Zürich, CH
- Marcel Kost
Salesforce – München, DE
- Burcu Kulahcioglu Ozkan
TU Delft, NL
- Federico Lorenzi
TigerBeetle – Cape Town, ZA
- Umang Mathur
National University of
Singapore, SG
- Everett Maus
Google – Seattle, US
- Hannes Mühleisen
CWI – Amsterdam, NL
- Thomas Neumann
TU München – Garching, DE
- Danica Porobic
Oracle Switzerland – Zürich, CH
- Mark Raasveldt
DuckDB Labs – Amsterdam, NL
- Tilmann Rabl
Hasso-Plattner-Institut,
Universität Potsdam, DE
- Manuel Rigger
National University of
Singapore, SG
- Stan Rosenberg
Cockroach Labs – New York, US
- Anupam Sanghi
IBM India – Bangalore, IN
- Gambhir Sankalp
EPFL – Lausanne, CH
- Andrei Satarin
Google – Mountain View, US
- Russell Sears
Crystal DB – San Francisco, US
- Thodoris Sotiropoulos
ETH Zürich, CH
- Caleb Stanford
University of California –
Davis, US
- Cheng Tan
Northeastern University –
Boston, US
- Pinar Tözün
IT University of
Copenhagen, DK
- Chengyu Zhang
ETH Zürich, CH



Approaches and Applications of Inductive Programming

Luc De Raedt^{*1}, Ute Schmid^{*2}, and Johannes Langer^{†3}

1 KU Leuven, BE. luc.deraedt@cs.kuleuven.be

2 Universität Bamberg, DE. ute.schmid@uni-bamberg.de

3 Universität Bamberg, DE. johannes.langer@uni-bamberg.de

Abstract

The Dagstuhl Seminar “Approaches and Applications of Inductive Programming” (AAIP) has taken place for the sixth time. The Dagstuhl Seminar series brings together researchers concerned with learning programs from input/output examples from different areas, mostly from machine learning and other branches of artificial intelligence research, cognitive scientists interested in human learning in complex domains, and researchers with a background in formal methods and programming languages. Main topics addressed in the AAIP 2023 seminar have been neurosymbolic approaches to IP bringing together learning and reasoning, IP as a post-hoc approach to explaining decision-making of deep learning blackbox models, and exploring the potential of deep learning approaches, especially large language models such as OpenAI Codex for IP. Topics discussed in working groups were Large Language Models and inductive programming in cognitive architectures, avoiding too much search in inductive programming, finding suitable benchmark problems, and evaluation criteria for interpretability and explainability of inductive programming.

Seminar October 29 – November 3, 2023 – <https://www.dagstuhl.de/23442>

2012 ACM Subject Classification Computing methodologies → Artificial intelligence; Human-centered computing; Computing methodologies → Machine learning


Keywords and phrases explainable ai, human-like machine learning, inductive logic programming, interpretable machine learning, neuro-symbolic ai

Digital Object Identifier 10.4230/DagRep.13.10.182

1 Executive Summary

Ute Schmid (Universität Bamberg, DE)

Luc De Raedt (KU Leuven, BE)

License  Creative Commons BY 4.0 International license
© Ute Schmid and Luc De Raedt

Inductive programming (IP) is a special perspective on program synthesis, addressing learning programs from incomplete specifications such as input/output examples. The seminar “Approaches and Applications of Inductive Programming” (AAIP) took place in Dagstuhl for the sixth time. This Dagstuhl Seminar brings together researchers from different areas of artificial intelligence research, machine learning, formal methods, programming languages, cognitive science, and human-computer-interaction interested in methods and applications of IP. Focus topics of AAIP’23 have been neurosymbolic approaches to IP bringing together learning and reasoning, IP as a post-hoc approach to explaining decision-making of deep learning blackbox models, and exploring the potential of deep learning approaches, especially large language models such as OpenAI Codex for IP.

* Editor / Organizer

† Editorial Assistant / Collector



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

Approaches and Applications of Inductive Programming, *Dagstuhl Reports*, Vol. 13, Issue 10, pp. 182–211

Editors: Luc De Raedt and Ute Schmid



DAGSTUHL
REPORTS

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

The focus topics have been introduced and discussed in a series of talks addressing neuro-symbolic IP, IP for learning in planning, explainable AI and IP, and IP and generative AI. Furthermore, a series of talks were dedicated to the relation of cognitive science to IP: Human-like few-shot learning via Bayesian reasoning over natural language, the child as hacker, using program synthesis to model strategy diversity in human visual reasoning, a neurodiversity-inspired solver for the Abstraction and Reasoning Corpus (ARC) using visual imagery and program synthesis, and using natural language for self-programming in cognitive architectures. The relation between IP and explainability has been highlighted with talks about explainable models via compression of relational ensembles, and effects of explaining machine-learned logic programs for human comprehension and discovery. Relations between IP and knowledge based methods have been addressed in a talk about learning disjointness axioms for knowledge graph refinement and for making knowledge graph embedding methods more robust. Methods of IP as an approach to learning interpretable rules have been presented with a focus on inductive logic programming (ILP), deep-rule learning, relational program synthesis with numerical reasoning, improving rule classifiers learned from quantitative data by recovering information lost by discretisation, meta-interpretive learning for generalised planning, probabilistic inductive logic programming, abstraction for answer set programs, anti-unification and generalization, programmatic reinforcement learning, and making program synthesis fast on a GPU. These talks have been complemented by several system demos presenting the ILP systems Popper and Louise, an RDF rules learner, and learning rules to sort e-mails into folders (EmFORE).

We identified four relevant research problems for current and future research in IP which were addressed in in-depth discussions in working groups and afterwards discussed in plenary sessions: (1) Large Language Models and Inductive Programming in Cognitive Architectures: one main outcome has been that combining learning and reasoning by integrating LLMs and reasoners in a cognitive architecture could be an enabler for validating programs that get executed by the overall architecture and to possibly get nearer to human performance. (2) Avoiding too much search in Inductive Programming: It was noted that for IP in general we do need to learn structure as well as probabilities. Classic IP approaches focus on structure learning and – in contrast to neural network architectures – can learn recursion explicitly. The main result has been that suitable problem domains should be identified for systematic evaluation, such as string transformation which combine syntactic (e.g. return first letter) and semantic (e.g. give the capital of a country) transformations. (3) Finding Suitable Benchmark Problems for Inductive Programming: Here, the discussion from the second topic has been extended and systematised with the formulation of several relevant criteria for benchmark problems to evaluate IP approaches, among them problem domains which are not solvable by LLMs and solvable efficiently by humans. (4) Evaluation Criteria for Interpretability and Explainability of Inductive Programming: The main insight has been that the degree of interpretability and the quality of explanations is strongly context-dependent, being influenced by the recipient (who), the content (what), the information need and reason for an explanation (why), and the form of the explanation (how). Different candidates for metrics were identified, such as complexity measures, semantic coherence, and reliability of generated code.

In a final discussion round, several outcomes have been summarized and action points have been discussed. A crucial problem which might impact scientific progress as well as visibility could be that there is no core general approach to IP (such as gradient descent for neural networks). Relevant use cases might not have a focus on learning recursion/loops but on relations (e.g. in medicine and biology). The focus on learning programs (including

recursion) might profit from using Python as the target language instead of more specific languages such as Prolog. Furthermore, current IP systems are mostly not easy to find and to use. Providing a toolbox which can be easily used (such as Weka for standard ML) might be helpful. There was a general agreement among the participants that the format of Dagstuhl Seminars is especially fruitful for bringing together the different perspectives on IP from machine learning, cognitive science, and program language research.

2 Table of Contents

Executive Summary

<i>Ute Schmid and Luc De Raedt</i>	182
--	-----

Overview of Talks

Effects of explaining machine-learned logic programs for human comprehension and discovery <i>Lun Ai</i>	187
Making program synthesis fast on a GPU <i>Martin Berger</i>	188
Anti-unification and Generalization: What's next? <i>David Cerna</i>	189
On the Need of Learning Disjointness Axioms for Knowledge Graph Refinement and for Making Knowledge Graph Embedding Methods more Robust <i>Claudia d'Amato</i>	189
How to make logics neurosymbolic <i>Luc De Raedt</i>	190
What should we do next in ILP? <i>Sebastijan Dumančić</i>	191
Human-like Few-Shot Learning via Bayesian Reasoning over Natural Language <i>Kevin Ellis</i>	191
Towards Programmatic Reinforcement Learning <i>Nathanaël Fijalkow</i>	192
Inductive Programming for Explainable Artificial Intelligence (IP for XAI) <i>Bettina Finzel</i>	192
On Deep Rule Learning <i>Johannes Fürnkranz</i>	193
Three Learning Problems in Planning <i>Hector Geffner</i>	194
A tutorial on Popper <i>Céline Hocquette</i>	194
Relational program synthesis with numerical reasoning <i>Céline Hocquette</i>	195
On the role of natural language for self-programming in cognitive architectures <i>Frank Jäkel</i>	196
QCBA: improving rule classifiers learned from quantitative data by recovering information lost by discretisation <i>Tomáš Kliegr</i>	196
RDFrules: A Swiss knife for relational association rule learning, classification and knowledge graph completion <i>Tomáš Kliegr</i>	197

The Child as Hacker <i>Josh Rule</i>	198
Abstraction for Answer Set Programs <i>Zeynep G. Saribatur</i>	199
Explanatory Inductive Programming (XAI for IP) <i>Ute Schmid</i>	200
Explainable models via compression of tree ensembles <i>Sriraam Natarajan</i>	201
Inductive Programming meets Large Language Models <i>Gust Verbruggen</i>	202
Inductive Programming meets Real User Problems <i>Gust Verbruggen</i>	202
Probabilistic Logic Programming: Quo Vadis? <i>Felix Weitzkämper</i>	203
Working groups	
Large Language Models and Inductive Programming in Cognitive Architectures <i>Bettina Finzel and Frank Jäkel</i>	204
Avoiding too much search in Inductive Programming <i>Ute Schmid, David Cerna, and Hector Geffner</i>	204
Evaluation Criteria for Interpretability and Explainability of Inductive Programming <i>Ute Schmid, Lun Ai, Claudia d’Amato, and Johannes Fürnkranz</i>	205
Finding Suitable Benchmark Problems for Inductive Programming <i>Ute Schmid, Martin Berger, Sebastijan Dumancic, Nathanaël Fijalkow, and Gust Verbruggen</i>	207
Panel discussions	
Inductive Programming – How to Go On? <i>Ute Schmid, Claudia d’Amato, Hector Geffner, Sriraam Natarajan, and Josh Rule</i>	209
Participants	211

3 Overview of Talks

3.1 Effects of explaining machine-learned logic programs for human comprehension and discovery

Lun Ai (Imperial College London, GB)

License © Creative Commons BY 4.0 International license
© Lun Ai

Joint work of Lun Ai, Johannes Langer, Stephen H. Muggleton, Ute Schmid

Main reference Lun Ai, Johannes Langer, Stephen H. Muggleton, Ute Schmid: “Explanatory machine learning for sequential human teaching”, *Mach. Learn.*, Vol. 112(10), pp. 3591–3632, 2023.

URL <https://doi.org/10.1007/S10994-023-06351-8>

The talk focused on the assumption in the Logic Programming community: logic programs are human-comprehensible. This had resulted in very few empirical assessments on the effects of explaining machine-learned logic programs. Empirical results by the authors showed explaining logic programs do not always lead to improved human performance. In addition, the authors stressed the need for objective and operational measurements of explainability. Their results provided novel insights on the explanatory effects of curriculum order and the presence of machine-learned explanations for sequential problem-solving.

The topic of comprehensibility of machine-learned theories has recently drawn increasing attention. Inductive logic programming uses logic programming to derive logic theories from small data based on abduction and induction techniques. Learned theories are represented in the form of rules as declarative descriptions of obtained knowledge. In earlier work, the authors provided the first evidence of a measurable increase in human comprehension based on machine-learned logic rules for simple classification tasks. In a later study, it was found that the presentation of machine-learned explanations to humans can produce both beneficial and harmful effects in the context of game learning.

The talk concentrated on a most recent investigation on the effects of the ordering of concept presentations and logic program explanations. The authors proposed a framework for the effects of sequential teaching based on an existing definition of comprehensibility. This empirical study involved curricula that teach novices the merge sort algorithm. They provided performance-based and trace-based evidence for support. Results show that sequential teaching of concepts with increasing complexity (a) has a beneficial effect on human comprehension and (b) leads to human re-discovery of divide-and-conquer problem-solving strategies, and (c) allows adaptations of human problem-solving strategy with better performance when machine-learned explanations are also presented.

Several open questions were discussed during and after the talk. For instance, the audience suggested an investigation on “learning how to learn” and comparisons between the human traces and the machine learner (ILP) trace. In the context of increasing the popularity of logic programs, some challenges in higher-education curricula were discussed showing the significance of how to best design Logic Programming teaching interactions. Importantly, this talk highlighted the limitations to performance-based evaluations. This led to an extended discussion on computable and objective assessments for various perspectives of explainability.

3.2 Making program synthesis fast on a GPU

Martin Berger (University of Sussex – Brighton, GB)

License © Creative Commons BY 4.0 International license
© Martin Berger

Joint work of Mojtaba Valizadeh, Martin Berger

Main reference Mojtaba Valizadeh, Martin Berger: “Search-Based Regular Expression Inference on a GPU”, Proc. ACM Program. Lang., Vol. 7(PLDI), pp. 1317–1339, 2023.

URL <https://doi.org/10.1145/3591274>

Inductive programming is stuck!

GPUs are the work-horses of computing. Applications that fit the GPU style of programming typically run orders of magnitude faster on GPUs than on CPUs. This gives opportunities for scaling not achievable with CPUs. The recent success of deep learning amply demonstrates this. Unfortunately, large classes of applications are not known to benefit from GPU acceleration. That includes most tools in program synthesis, inductive programming, theorem proving, ... (from now on: automated reasoning) such as SAT and SMT solvers. How can we change this? Simplifying a bit, a GPU can only accelerate applications if they are “GPU-friendly”, meaning they are

- highly parallel,
- have little to no data-dependent branching, and have
- predictable data-movement, and high temporal and spatial data locality.

Algorithms in automated reasoning, as implemented today, mostly lack those properties. Many are extremely branching heavy, for example because they branch on syntactic structure. Some are seemingly sequential (e.g. unit propagation, a core step modern SAT solvers for simplifying formulae). This might be because an algorithmic problem is intrinsically sequential, or because a way of making an algorithmic problem GPU-friendly has not yet been found.

Research question: Can we identify workloads arising in industrial automatic reasoning practise, and scale them up on GPUs by developing suitable, GPU-friendly algorithms? The GPU-based algorithms should give at least 100x speedup (for comparable problem instances), and be able to handle at least 1000x bigger problem instances, both in comparison with state-of-the-art open (= non-proprietary) software for the same problem domain.

Preliminary answer, based on [1]: all program synthesis that uses the generate-and-test approach can see orders of magnitude speedup on GPUs.

Recommendation to the ILP community: stop what your are doing and implement your ideas on a GPU.

References

- 1 Mojtaba Valizadeh, Martin Berger: *Search-Based Regular Expression Inference on a GPU*. Proc. ACM Program. Lang. 7(PLDI): 1317-1339 (2023), <https://doi.org/10.1145/3591274>

3.3 Anti-unification and Generalization: What's next?

David Cerna (*The Czech Academy of Sciences – Prague, CZ*)

License © Creative Commons BY 4.0 International license
© David Cerna

Joint work of David M. Cerna, Temur Kutsia

Main reference David M. Cerna, Temur Kutsia: “Anti-unification and Generalization: A Survey”, in Proc. of the Thirty-Second International Joint Conference on Artificial Intelligence, IJCAI-23, pp. 6563–6573, International Joint Conferences on Artificial Intelligence Organization, 2023.

URL <https://doi.org/10.24963/ijcai.2023/736>

Anti-unification (AU) is a fundamental operation for the computation of symbolic generalizations useful for inductive inferencing [1]. It is the dual operation to *unification*, an operation at the foundation of automated theorem proving. In contrast to unification, where one is interested in constructing *most general unifiers (mgus)*, anti-unification is concerned with the construction of *least general generalizations (lggs)*; that is, expressions capturing the commonalities shared between members of a set of symbolic expressions.

The operation was introduced by Plotkin and Reynolds and found many applications within the area of *Inductive synthesis* and, in particular, early inductive logic programming (ILP) systems. However, since their seminal work, the number of applications has grown tremendously with uses in program analysis, program repair, automated reasoning, and beyond. With the growing number of applications, several investigations have developed anti-unification methods over various symbolic objects, such as the simply-typed lambda calculus, term graphs, and hedge expression, to name a few. In particular, there has been significant progress in understanding equational anti-unification and the cardinality of the set of solutions (set of lggs). In many cases, the solution sets are either infinitely large or do not exist (every generalization allows a more specific generalization).

We ask, is *least general generalization* the right characterization of a solution to an anti-unification problem? In particular, is there a characterization of a solution more amenable to modern approaches to inductive synthesis? Secondly, what does the inductive synthesis community need from symbolic generalization techniques, which is currently missing?

References

- 1 David M. Cerna, Temur Kutsia: *Anti-unification and Generalization: A Survey*. IJCAI 2023: 6563–6573, <https://doi.org/10.24963/IJCAI.2023/736>

3.4 On the Need of Learning Disjointness Axioms for Knowledge Graph Refinement and for Making Knowledge Graph Embedding Methods more Robust

Claudia d’Amato (*University of Bari, IT*)

License © Creative Commons BY 4.0 International license
© Claudia d’Amato

Joint work of Giuseppe Rizzo, Claudia d’Amato, Nicola Fanizzi

Main reference Giuseppe Rizzo, Claudia d’Amato, Nicola Fanizzi: “An unsupervised approach to disjointness learning based on terminological cluster trees”, *Semantic Web*, Vol. 12(3), pp. 423–447, 2021.

URL <https://doi.org/10.3233/SW-200391>

Knowledge Graphs (KGs) are multi-relational graphs designed to organize and share real-world knowledge where nodes represent entities of interest and edges represent different types of relationships between such entities [1]. Despite the large usage, it is well known that KGs suffer from incompleteness and noise. For tackling these problems, solutions to the link

prediction task, that amount at predicting an unknown component of a triple, have been investigated. Mostly, Knowledge Graph Embedding methods (KGE) have been devised since they have been shown to scale even to very large KGs. KGE convert the data graph into an optimal low dimensional space where structural graph information is preserved as much as possible. Embeddings are learned based on the constraint that a valid (positive) triple score has to be lower than the invalid (negative) triple score. As KGs mainly encode positive triples, negative triples are obtained by randomly corrupting true/observed triples [2], thus possibly injecting false negatives during the learning process.

In this talk we present a solution for an informed generation of negative examples that, by exploiting the semantics of the KGs and reasoning capabilities, is able to limit false negatives. A key element is represented by disjointness axioms, that are essential for making explicit the negative knowledge about a domain. Yet, disjointness axioms are often overlooked during the modeling process [3]. For the purpose, a symbolic method for discovering disjointness axioms from the data distribution is illustrated. Moving from the assumption that two or more concepts may be mutually disjoint when the sets of their (known) instances do not overlap, the problem is cast as a conceptual clustering problem, where the goal is both to find the best possible partitioning of the individuals in (a subset of) the KG and also to induce intensional definitions of the corresponding classes expressed in the standard representation languages.

The talk will conclude with the analysis of some open challenges related to the presented solutions.

References

- 1 Aidan Hogan, Eva Blomqvist, Michael Cochez, Claudia d’Amato, Gerard de Melo, Claudio Gutierrez, Sabrina Kirrane, José Emilio Labra Gayo, Roberto Navigli, Sebastian Neumaier, Axel-Cyrille Ngonga Ngomo, Axel Polleres, Sabbir M. Rashid, Anisa Rula, Lukas Schmelzeisen, Juan Sequeda, Steffen Staab, Antoine Zimmermann: *Knowledge Graphs*. Synthesis Lectures on Data, Semantics, and Knowledge, Morgan & Claypool Publishers 2021, ISBN 978-3-031-00790-3, pp. 1-257. <https://doi.org/10.2200/S01125ED1V01Y202109DSK022>
- 2 Hongyun Cai, Vincent W. Zheng, Kevin Chen-Chuan Chang: *A Comprehensive Survey of Graph Embedding: Problems, Techniques, and Applications*. IEEE Trans. Knowl. Data Eng. 30(9): 1616-1637 (2018). <https://doi.org/10.1109/TKDE.2018.2807452>
- 3 Taowei David Wang, Bijan Parsia, James A. Hendler: *A Survey of the Web Ontology Landscape*. ISWC 2006: 682-694. https://doi.org/10.1007/11926078_49

3.5 How to make logics neurosymbolic

Luc De Raedt (KU Leuven, BE)

License © Creative Commons BY 4.0 International license
© Luc De Raedt

Joint work of Giuseppe Marra, Sebastijan Dumančić, Robin Manhaeve, Luc De Raedt
Main reference Giuseppe Marra, Sebastijan Dumančić, Robin Manhaeve, Luc De Raedt: “From Statistical Relational to Neural Symbolic Artificial Intelligence: a Survey”, CoRR, Vol. abs/2108.11451, 2021.
URL <https://arxiv.org/abs/2108.11451>

Neurosymbolic AI (NeSy) is regarded as the third wave in AI. It aims at combining knowledge representation and reasoning with neural networks. Numerous approaches to NeSy are being developed and there exists an ‘alphabet-soup’ of different systems, whose relationships are often unclear. I will discuss the state-of-the art in NeSy and argue that there are many similarities with statistical relational AI (StarAI).

Taking inspiration from StarAI, and exploiting these similarities, I will argue that Neurosymbolic AI = Logic + Probability + Neural Networks. I will also provide a recipe for developing NeSy approaches: start from a logic, add a probabilistic interpretation, and then turn neural networks into “neural predicates”. Probability is interpreted broadly here, and is necessary to provide a quantitative and differentiable component to the logic. At the semantic and the computation level, one can then combine logical circuits (ako proof structures) labeled with probability, and neural networks in computation graphs.

I will illustrate the recipe with NeSy systems such as DeepProbLog, a deep probabilistic extension of Prolog, and DeepStochLog, a neural network extension of stochastic definite clause grammars (or stochastic logic programs).

3.6 What should we do next in ILP?

Sebastija Dumančić (TU Delft, NL)

License © Creative Commons BY 4.0 International license
© Sebastijan Dumančić

This talk consists of two parts. In the first part, I provide a brief introduction to Inductive Logic Programming: what is it, why is it interesting, and what interesting has recently happened. In the second part, I will explore what I think we should do next in ILP and program synthesis to further advance the field, all centered around the idea of avoiding search.

3.7 Human-like Few-Shot Learning via Bayesian Reasoning over Natural Language

Kevin Ellis (Cornell University – Ithaca, US)

License © Creative Commons BY 4.0 International license
© Kevin Ellis

Main reference Kevin Ellis: “Modeling Human-like Concept Learning with Bayesian Inference over Natural Language”, CoRR, Vol. abs/2306.02797, 2023.

URL <https://doi.org/10.48550/ARXIV.2306.02797>

A core tension in models of concept learning is that the model must carefully balance the tractability of inference against the expressivity of the hypothesis class. Humans, however, can efficiently learn a broad range of concepts. We introduce a model of inductive learning that seeks to be human-like in that sense. It implements a Bayesian reasoning process where a language model first proposes candidate hypotheses expressed in natural language, which are then re-weighted by a prior and a likelihood. By estimating the prior from human data, we can predict human judgments on learning problems involving numbers and sets, spanning concepts that are generative, discriminative, propositional, and higher-order.

3.8 Towards Programmatic Reinforcement Learning

Nathanaël Fijalkow (CNRS – Talence, FR)

License  Creative Commons BY 4.0 International license
© Nathanaël Fijalkow

This short talk was a pitch for a new problem, called Programmatic Reinforcement Learning: assuming that the environment is given as a program, the goal is to construct an optimal policy in the form of a program. Some motivations, basic examples, and preliminary experimental results were presented and discussed.

3.9 Inductive Programming for Explainable Artificial Intelligence (IP for XAI)

Bettina Finzel (Universität Bamberg, DE)

License  Creative Commons BY 4.0 International license
© Bettina Finzel

Methods of explainable artificial intelligence (XAI) and of inductive programming (IP) can profit from each other in two ways: (1) Inductive programming results in symbolic models (programs) which are inherently interpretable. These programs can provide expressive, relational explanations for learned black box models, for instance Convolutional Neural Networks for image classification. This perspective (IP for XAI) is addressed in this summary. (2) On the other hand, there might be a need for explainability of IP programs to humans. This perspective (XAI for IP) is addressed in the contribution of U. Schmid in this report.

End-to-end and data-driven approaches to learning, like deep convolutional neural networks in image classification, have become prevalent and the center of attention in many research and application areas. However, some research objectives and real world problems may not be solvable by just processing large amounts of data. In some cases, like medical diagnostics, “big data” simply may not be available [2]. At the same time, deep learning models are not inherently transparent opposed to those generated by interpretable machine learning algorithms, such as Inductive Logic Programming (ILP) [6]. This may be a crucial deficiency and a barrier to high stakes applicability of deep learning. At the same time, ILP frameworks provide symbolic representations in the form of predicates in First-Order-Logic, tracing capabilities and the integration of relational background knowledge by design, e.g., from human expertise and domain knowledge [3]. Moreover, their learning process is data-efficient in comparison to deep learning. In addition, being a relational learning approach qualifies ILP for explainability [1], e.g., in complex knowledge domains like medicine [2] and AI evaluation in general [5]. Deep learning may therefore profit from being combined with ILP for explanation, validation and a bi-directional interaction between a human and an AI system [3]. A crucial part of this avenue is the design of interfaces between internal representations of what a deep learning model has learned and the relational background knowledge of IP systems, like ILP, to provide human-understandable surrogate models, explanations and interactions. First attempts to bridge this gap have already been proposed [4]. However, several open questions remain to date: How can we find and extract relevant internal representations from deep learning models and present them in a human-understandable manner? How can we disambiguate representations? Which relations should be included in the IP module and satisfied by the deep learning model? How can we implement a knowledge exchange between IP and deep learning models to support the interplay of learning and reasoning in

knowledge discovery and AI evaluation? In my opinion, to build such systems is the way toward approximating the strengths of the human inductive bias and adaptability of AI systems to the real world.

References

- 1 Gesina Schwalbe, Bettina Finzel: *A comprehensive taxonomy for explainable artificial intelligence: a systematic survey of surveys on methods and concepts*. Data Mining and Knowledge Discovery: 1-59 (2023). <https://doi.org/10.1007/s10618-022-00867-8>
- 2 Sebastian Bruckert, Bettina Finzel, Ute Schmid: The Next Generation of Medical Decision Support: A Roadmap Toward Transparent Expert Companions. *Frontiers Artif. Intell.* 3: 507973 (2020). <https://doi.org/10.3389/FRAI.2020.507973>
- 3 Ute Schmid, Bettina Finzel: *Mutual Explanations for Cooperative Decision Making in Medicine*. *Künstliche Intell.* 34(2): 227-233 (2020). <https://doi.org/10.1007/S13218-020-00633-2>
- 4 Johannes Rabold, Michael Siebers, Ute Schmid: *Explaining Black-Box Classifiers with ILP – Empowering LIME with Aleph to Approximate Non-linear Decisions with Relational Rules*. *ILP 2018*: 105-117. https://doi.org/10.1007/978-3-319-99960-9_7
- 5 José Hernández-Orallo: *The Measure of All Minds: Evaluating Natural and Artificial Intelligence*. Cambridge University Press 2017, ISBN 9781316594179. <https://doi.org/10.1017/9781316594179>
- 6 Sumit Gulwani, José Hernández-Orallo, Emanuel Kitzelmann, Stephen H. Muggleton, Ute Schmid, Benjamin G. Zorn: Inductive programming meets the real world. *Commun. ACM* 58(11): 90-99 (2015). <https://doi.org/10.1145/2736282>

3.10 On Deep Rule Learning

Johannes Fürnkranz (Johannes Kepler Universität Linz, AT)

License  Creative Commons BY 4.0 International license
© Johannes Fürnkranz

Joint work of Florian Beck, Johannes Fürnkranz

Main reference Florian Beck, Johannes Fürnkranz: “An Empirical Investigation Into Deep and Shallow Rule Learning”, *Frontiers in Artificial Intelligence*, Vol. 4, 2021.

URL <https://doi.org/10.3389/frai.2021.689398>

Rule learning algorithms form the basis of classic inductive logic programming algorithms such as FOIL or PROGOL. Studying them in a propositional logic setting allows to focus on the algorithmic aspects. A key limitation of the current state-of-the-art such as the LORD algorithm recently developed in our group [1], is that they are all limited to learning rule sets that directly connect the input features to the target feature. In a logical setting, this corresponds to learning a DNF expression. While every logical function can be expressed as a DNF formula, we argue in this talk that learning deeply structured theories may be beneficial, by drawing an analogy to (deep) neural networks [3], and recapitulating some recent empirical results [2].


References

- 1 Phuong Huynh Van Quoc, Johannes Fürnkranz, Florian Beck: *Efficient learning of large sets of locally optimal classification rules*. *Machine Learning* 112(2): 571-610 (2023) <https://doi.org/10.1007/s10994-022-06290-w>
- 2 Florian Beck, Johannes Fürnkranz, Phuong Huynh Van Quoc: *Layerwise Learning of Mixed Conjunctive and Disjunctive Rule Sets*. *Proceedings of the 7th International Joint Conference on Rules and Reasoning (RuleML+RR)*, 2023, 2023:95-109. https://doi.org/10.1007/978-3-031-45072-3_7

- 3 Florian Beck, Johannes Fürnkranz: *An Empirical Investigation Into Deep and Shallow Rule Learning*. *Frontiers in Artificial Intelligence* 4: 689398 (2021). <https://doi.org/10.3389/frai.2021.689398>

3.11 Three Learning Problems in Planning

Hector Geffner (RWTH Aachen, DE)

License  Creative Commons BY 4.0 International license
© Hector Geffner

Joint work of Hector Geffner, Simone Ståhlberg, Blai Bonet, Dominik Drexler, RLeap team

I'll talk about three learning problems in planning: learning lifted action models, learning generalized policies, and learning general problem decomposition or sketches. We have been approaching these problems in a top-down fashion, making a clear distinction between what is to be learned and how it is to be learned. Indeed, we have been pursuing two types of approaches in parallel: formulations that rely on combinatorial optimization solvers on the one hand, and deep (reinforcement) learning approaches on the other. I'll also discuss the relation between the two approaches which in the common form are limited by the expressive power of C2 logic; first-order logic with two variables and counting, and challenges to get beyond C2.

References

- 1 Blai Bonet, Hector Geffner: *General Policies, Subgoal Structure, and Planning Width*. *CoRR* abs/2311.05490 (2023). <https://doi.org/10.48550/ARXIV.2311.05490>
- 2 Simon Ståhlberg, Blai Bonet, Hector Geffner: *Learning General Policies with Policy Gradient Methods*. *KR 2023*: 647-657. <https://doi.org/10.24963/KR.2023/63>
- 3 Dominik Drexler, Jendrik Seipp, Hector Geffner: *Learning Sketches for Decomposing Planning Problems into Subproblems of Bounded Width*. *ICAPS 2022*: 62-70 <https://doi.org/10.1609/icaps.v32i1.19786>
- 4 Ivan D. Rodriguez, Blai Bonet, Javier Romero, Hector Geffner: *Learning First-Order Representations for Planning from Black Box States: New Results*. *KR 2021*: 539-548. <https://doi.org/10.24963/KR.2021/51>

3.12 A tutorial on Popper

Céline Hocquette (University of Oxford, GB)

License  Creative Commons BY 4.0 International license
© Céline Hocquette

Joint work of Andrew Cropper, Céline Hocquette

Main reference Andrew Cropper, Céline Hocquette: “Learning Logic Programs by Combining Programs”, in Proc. of the ECAI 2023 – 26th European Conference on Artificial Intelligence, September 30 – October 4, 2023, Kraków, Poland – Including 12th Conference on Prestigious Applications of Intelligent Systems (PAIS 2023), *Frontiers in Artificial Intelligence and Applications*, Vol. 372, pp. 501–508, IOS Press, 2023.

URL <https://doi.org/10.3233/FAIA230309>

Inductive logic programming (ILP) is a form of program synthesis. The goal is to induce a logic program that generalises training examples. Popper is a recent ILP system which frames the ILP problem as a constraint satisfaction problem [1, 2]. Popper continually generates hypotheses and tests them on the training examples. If a hypothesis is not a solution, Popper

builds constraints to prune hypotheses which are also provably no solutions. Popper supports learning of recursive programs, predicate invention and learning moderately large programs. We present a recent extension of Popper which supports learning minimal description length programs from noisy data [3]. Our approach leverages recent progress in MaxSAT solvers to efficiently find an optimal program.

References

- 1 Andrew Cropper, Rolf Morel: *Learning programs by learning from failures*. Mach. Learn. 110(4): 801-856 (2021). <https://doi.org/10.1007/S10994-020-05934-Z>
- 2 Andrew Cropper, Céline Hocquette: *Learning Logic Programs by Combining Programs*. ECAI 2023: 501-508 <https://doi.org/10.3233/FAIA230309>
- 3 Céline Hocquette, Andreas Niskanen, Matti Järvisalo, Andrew Cropper: *Learning MDL logic programs from noisy data*. CoRR abs/2308.09393 (2023). <https://doi.org/10.48550/ARXIV.2308.09393>

3.13 Relational program synthesis with numerical reasoning

Céline Hocquette (University of Oxford, GB)

License © Creative Commons BY 4.0 International license
© Céline Hocquette

Joint work of Céline Hocquette, Andrew Cropper

Main reference Céline Hocquette, Andrew Cropper: “Relational Program Synthesis with Numerical Reasoning”, in Proc. of the Thirty-Seventh AAAI Conference on Artificial Intelligence, AAAI 2023, Thirty-Fifth Conference on Innovative Applications of Artificial Intelligence, IAAI 2023, Thirteenth Symposium on Educational Advances in Artificial Intelligence, EAAI 2023, Washington, DC, USA, February 7-14, 2023, pp. 6425–6433, AAAI Press, 2023.

URL <https://doi.org/10.1609/AAAI.V37I5.25790>

Learning programs with numerical values is fundamental to many AI applications, including bio-informatics and drug design. However, current program synthesis approaches struggle to learn programs with numerical values. Program synthesis approaches based on enumeration of candidate numerical symbols cannot handle infinite domains. Recent program synthesis approaches also have difficulties reasoning from multiple examples, which is required for instance to identify numerical thresholds or intervals. To overcome these limitations, we introduce an inductive logic programming approach which combines relational learning with numerical reasoning [1]. Our approach uses satisfiability modulo theories solvers to efficiently identify numerical values. Our approach can identify numerical values in linear arithmetic fragments, such as real difference logic, and from infinite domains, such as real numbers or integers. Our results show our approach can outperform existing program synthesis approaches. However, our approach has limited scalability with respect to the complexity of the numerical reasoning stage.

3.14 On the role of natural language for self-programming in cognitive architectures

Frank Jäkel (TU Darmstadt, DE)

License © Creative Commons BY 4.0 International license
© Frank Jäkel

Human problem solvers are able to adapt their problem solving strategies to new situations. They program their own behavior. In order to do so, they introspect, test, debug, and optimize their problem solving algorithms. These metacognitive activities can be implemented in standard cognitive architectures that can store code in working memory and execute it with an interpreter that is implemented as a set of rules in a production system. Additional rules can then modify the code at runtime. Unfortunately, the programming language in which such mental code is written has remained elusive. Here, I will argue that it is time to revive the old idea that program code is directly given in natural language. Traditionally, research on cognitive architectures has mostly avoided natural language even though language is obviously an important aspect of human cognition. With the advent of large language models it seems more plausible than ever that natural language interpreters might become an essential part of a new generation of cognitive architectures. In particular, the metacognitive activity of modifying your own programs might simply consist of transforming one natural language expression into another – the task that transformers were developed for and have turned out to be quite successful at.

3.15 QCBA: improving rule classifiers learned from quantitative data by recovering information lost by discretisation

Tomáš Kliegr (University of Economics – Prague, CZ)

License © Creative Commons BY 4.0 International license
© Tomáš Kliegr

Main reference Tomáš Kliegr, Ebrou Izquierdo: “QCBA: improving rule classifiers learned from quantitative data by recovering information lost by discretisation”, *Appl. Intell.*, Vol. 53(18), pp. 20797–20827, 2023.

URL <https://doi.org/10.1007/S10489-022-04370-X>

Many rule-learning algorithms require prior discretization before they can effectively process datasets with numerical data. For example, consider a dataset with attributes such as temperature and humidity. Discretization (also called quantization) means binning their values into intervals. A simple equidistant algorithm would produce intervals such as (0;10], (10;20], and (20; 30]. If we consider rule learning algorithms based on association rule learning, such as Classification based on Associations [3], discretization is necessary to ensure fast pruning of the state space and also learning of sufficiently generalized rules. Only after the discretization is it possible to learn the rules of the type IF temperature=(20;30] and humidity=(50;60] THEN worker_comfort= good.

While some rule learning algorithms can directly work with numerical attributes, such as the recently proposed extension of the POPPER ILP system [4], for those based on association rule learning, integrating quantization may not be efficient as it could excessively slow down the candidate generation phase. A common approach is thus to apply prediscretization, e.g., following the Minimum Description Length Principle (MDLP)-based method proposed by [2]. However, as the determination of interval lengths is done globally (i.e., same intervals for all instances) and outside of the learning algorithm (e.g. CBA), information is lost, resulting in inefficiencies in the final classifier.

Following this problem, this talk introduced the Quantitative CBA (QCBA) algorithm for the subsequent processing of rule models learned on arbitrarily pre-discretized data (e.g., with equidistant binning, MDLP or other method). Extensive experiments have shown that the proposed algorithm consistently reduces the models' size and thus makes them more understandable. Additionally, in many cases, the predictive performance is also improved. The algorithm can be used to process the results of many rule learning algorithms, including CBA, Interpretable Decision Sets [1] and Scalable Bayesian Rule Lists [5]. The results are available in the R package qCBA available in CRAN. The method is described in detail in [6].

References

- 1 Himabindu Lakkaraju, Stephen H. Bach, Jure Leskovec: *Interpretable Decision Sets: A Joint Framework for Description and Prediction*. KDD 2016: 1675-1684 <https://doi.org/10.1145/2939672.2939874>
- 2 Usama M. Fayyad, Keki B. Irani: *Multi-Interval Discretization of Continuous-Valued Attributes for Classification Learning*. IJCAI 1993: 1022-1029
- 3 Bing Liu, Wynne Hsu, Yiming Ma: *Integrating Classification and Association Rule Mining*. KDD 1998: 80-86
- 4 Céline Hocquette, Andrew Cropper: *Relational Program Synthesis with Numerical Reasoning*. AAAI 2023: 6425-6433 <https://doi.org/10.1609/AAAI.V37I5.25790>
- 5 Hongyu Yang, Cynthia Rudin, Margo I. Seltzer: *Scalable Bayesian Rule Lists*. ICML 2017: 3921-3930
- 6 Tomáš Kliegr, Ebroul Izquierdo: *QCBA: improving rule classifiers learned from quantitative data by recovering information lost by discretisation*. Appl. Intell. 53(18): 20797-20827 (2023) <https://doi.org/10.1007/S10489-022-04370-X>

3.16 RDFrules: A Swiss knife for relational association rule learning, classification and knowledge graph completion

Tomáš Kliegr (University of Economics – Prague, CZ)

License  Creative Commons BY 4.0 International license
© Tomáš Kliegr

Joint work of Václav Zeman, Tomáš Kliegr, Vojtech Svátek

Main reference Václav Zeman, Tomáš Kliegr, Vojtech Svátek: “RDFrules: Making RDF rule mining easier and even more efficient”, Semantic Web, Vol. 12(4), pp. 569–602, 2021.

URL <https://doi.org/10.3233/SW-200413>

Many commonly used machine learning algorithms are limited to tabular data sets, but real-world data is often stored in relational databases and increasingly in knowledge graphs. Processing of such data with standard „tabular“ machine learning usually requires extensive data transformation and aggregations, resulting in a loss of information. As an alternative, relational Horn rules can be used to model complex relational structures naturally and use these in a range of machine learning tasks, including exploratory analysis, classification, and imputation of missing information.

The RDFrules system for learning rules from knowledge graphs is based on the high-performance AMIE+ algorithm [1] and includes a number of improvements based on more than 10 years of experience with the development of its sister tabular EasyMiner [2] rule learning system. While the AMIE+ algorithm was initially designed for a narrower exploratory task of discovery of rules with the potential to perform knowledge graph (KG) completion,

the current version of the RDRules system goes significantly beyond the original capabilities of the AMIE+ algorithm [1] as it now makes possible to perform the following tasks:

- load not only graph data in RDF but also relational databases described as SQL scripts,
- specify fine-grained patterns to limit the search space,
- preprocess numerical literals,
- cluster discovered rules,
- perform classification tasks,
- evaluate results using standard metrics adapted to graph data and open world assumption,
- support the KG completion task,

The new features make it possible to graph-based rule learning directly on complex real-world data.

The system is described in [3] and available at <https://github.com/propi/rdrules>.

References

- 1 Luis Galárraga, Christina Teflioudi, Katja Hose, Fabian M. Suchanek: *Fast rule mining in ontological knowledge bases with AMIE+*. VLDB J. 24(6): 707-730 (2015) <https://doi.org/10.1007/S00778-015-0394-1>
- 2 Stanislav Vojír, Vaclav Zeman, Jaroslav Kuchar, Tomáš Kliegr: *EasyMiner.eu: Web framework for interpretable machine learning based on rules and frequent itemsets*. Knowl. Based Syst. 150: 111-115 (2018) <https://doi.org/10.1016/J.KNOSYS.2018.03.006>
- 3 Václav Zeman, Tomáš Kliegr, Vojtech Svátek: *RDRules: Making RDF rule mining easier and even more efficient*. Semantic Web 12(4): 569-602 (2021) <https://doi.org/10.3233/SW-200413>

3.17 The Child as Hacker

Josh Rule (University of California – Berkeley, US)

License © Creative Commons BY 4.0 International license
© Josh Rule

Main reference Joshua S. Rule: “The child as hacker: building more human-like models of learning”, Doctoral dissertation, Massachusetts Institute of Technology (2020)

URL <https://hdl.handle.net/1721.1/129232>

I describe the *child as hacker* hypothesis, which relates program induction with aspects of human cognition, particularly learning [1]. By the deep relationship proposed to exist between knowledge and program-like structures, the child as hacker treats the activities and values of human programmers as hypotheses for the activities and values of many forms of human learning. After introducing this idea, I then look briefly at a project where we’ve begun to implement it in a system called HL (Hacker-Like) [2]. HL explains human behaviour better than some recent alternative program induction systems by representing a concept not only in terms of its object-level content but also in terms of the inferences required to produce that content. By searching over both kinds of representations, HL learns orders of magnitude faster than competing systems. I close by discussing three major areas ripe for future research: i) developing a better empirical understanding of how people solve hard search problems; ii) understanding the neural and psychological basis for human computational abilities; and iii) better understanding the goals and values of human programmers. All three areas have the potential to significantly improve both our understanding of human intelligence and our ability to use program induction systems to solve complex problems.

References

- 1 Joshua S. Rule, Joshua B. Tenenbaum, Steven T. Piantadosi: *The child as hacker*. Trends in cognitive sciences 24(11): 900-915 (2020) <https://doi.org/10.1016/j.tics.2020.07.005>
- 2 Joshua S. Rule: *The child as hacker: building more human-like models of learning*. Doctoral dissertation, Massachusetts Institute of Technology (2020) <https://hdl.handle.net/1721.1/129232>

3.18 Abstraction for Answer Set Programs

Zeynep G. Saribatur (TU Wien, AT)

License © Creative Commons BY 4.0 International license
© Zeynep G. Saribatur

Joint work of Zeynep G. Saribatur, Thomas Eiter, Peter Schüller

Main reference Zeynep G. Saribatur, Thomas Eiter, Peter Schüller: “Abstraction for non-ground answer set programs”, *Artif. Intell.*, Vol. 300, p. 103563, 2021.

URL <https://doi.org/10.1016/J.ARTINT.2021.103563>

In this talk, I present our notion of abstraction for answer set programming, a prominent rule-based language for knowledge representation and reasoning with roots in logic programming and non-monotonic reasoning. With the aim to abstract over the irrelevant details of answer set programs, we focus on two approaches of abstraction: (1) abstraction by omission [2], and (2) domain abstraction [1], and introduce a method to construct an abstract program with a smaller vocabulary, by ensuring that the original program is over-approximated. We provide an abstraction & refinement methodology that makes it possible to start with an initial abstraction and upon encountering spurious solutions automatically refining the abstraction until an abstract program with a non-spurious solution is reached. Experiments based on the prototypical implementations reveal the potential of the approach for problem analysis by focusing on the parts of the program that cause the unsatisfiability, some even matching a human-like focus shown by a user study, and by achieving generalization of the answer sets that reflect relevant details only. This makes abstraction an interesting topic of research whose further use in human-understandability of logic programs remains to be explored.

References

- 1 Zeynep G. Saribatur, Thomas Eiter, Peter Schüller: *Abstraction for non-ground answer set programs*. *Artif. Intell.* 300: 103563 (2021) <https://doi.org/10.1016/J.ARTINT.2021.103563>
- 2 Zeynep G. Saribatur, Thomas Eiter: *Omission-Based Abstraction for Answer Set Programs*. *Theory Pract. Log. Program.* 21(2): 145-195 (2021) <https://doi.org/10.1017/S1471068420000095>

3.19 Explanatory Inductive Programming (XAI for IP)

Ute Schmid (Universität Bamberg, DE)

License © Creative Commons BY 4.0 International license
© Ute Schmid

Joint work of Johannes Rabold, Michael Siebers, Ute Schmid

Main reference Johannes Rabold, Michael Siebers, Ute Schmid: “Generating contrastive explanations for inductive logic programming based on a near miss approach”, *Mach. Learn.*, Vol. 111(5), pp. 1799–1820, 2022.

URL <https://doi.org/10.1007/S10994-021-06048-W>

Methods of explainable artificial intelligence (XAI) and of inductive programming (IP) can profit from each other in two ways: (1) Inductive programming results in symbolic models (programs) which are inherently interpretable. Nevertheless, there might be a need for explainability to humans – end-users or domain experts from other areas than computer science. This perspective (XAI for IP) is addressed in this summary. (2) On the other hand, expressive, relational explanations for learned black box models, for instance Convolutional Neural Networks for image classification, can be provided by IP. This perspective (IP for XAI) is addressed in the contribution of B. Finzel in this report.

The power of IP approaches lies in their ability to learn highly expressive models from small sets of examples [2]. Learned programs can support humans to get insights into complex relational or recursive patterns underlying a set of observed data. That is, IP might be an ultra-strong learning approach as defined by Donald Michie (see [3]) under the condition that the learning system can teach the learned model to a human, whose performance is consequently increased to a level beyond that of the human studying the training data alone. For programs which consist of several rules or for programs involving complex relations or recursion, different approaches to construct explanations might support human understanding. One possibility to reduce complexity is to introduce new predicates. For instance, the introduction of a predicate *parent/2* as generalization for *father/2* and *mother/2*, reduces four rules for the *grandparent/2* relation to one (see [3]). Another possibility is, to translate the rule which covers the current instant to a verbal explanation for humans without background in computer science. This can be realized by simple template-based methods [5]. Alternatively, Large Language Models could be used. For effective teaching a concept to humans, near miss explanations have been proposed by [4]. Winston showed in his early work on learning rules for relational perceptual concepts such as arcs, that providing near misses rather than arbitrary negative examples results in faster convergence of the learned model. In cognitive science it has been shown that teaching concepts by their difference to similar concepts is much more efficient than contrasting them with more distant concepts (for a discussion of these aspects and references, see [4]). In [4] an algorithm for constructing near miss explanations is presented and applied to different domains. Furthermore, an empirical study is presented where it could be shown that in pairwise comparisons, participants preferred near miss explanations over other types of explanations as more helpful.

Augmenting IP models with explanations can also be helpful to support medical decision making [1]. Here, it might be helpful to go beyond ultrastrong machine learning and bring the human expert in the loop for incremental model correction and adaptation. In contrast to standard interactive machine learning, human feedback might go beyond label correction and allow human domain experts to also correct explanations which might be right for the wrong reasons. Correcting explanations can be seen as a special case of knowledge injection in human-in-the-loop IP which exploits such corrections for efficient model adaptation.

References

- 1 Sebastian Bruckert, Bettina Finzel, Ute Schmid: *The Next Generation of Medical Decision Support: A Roadmap Toward Transparent Expert Companions*. *Frontiers Artif. Intell.* 3: 507973 (2020) <https://doi.org/10.3389/FRAI.2020.507973>
- 2 Sumit Gulwani, José Hernández-Orallo, Emanuel Kitzelmann, Stephen H. Muggleton, Ute Schmid, Benjamin G. Zorn: *Inductive programming meets the real world*. *Commun. ACM* 58(11): 90-99 (2015) <https://doi.org/10.1145/2736282>
- 3 Stephen H. Muggleton, Ute Schmid, Christina Zeller, Alireza Tamaddoni-Nezhad, Tarek R. Besold: *Ultra-Strong Machine Learning: comprehensibility of programs learned with ILP*. *Mach. Learn.* 107(7): 1119-1140 (2018) <https://doi.org/10.1007/S10994-018-5707-3>
- 4 Johannes Rabold, Michael Siebers, Ute Schmid: *Generating contrastive explanations for inductive logic programming based on a near miss approach*. *Mach. Learn.* 111(5): 1799-1820 (2022) <https://doi.org/10.1007/S10994-021-06048-W>
- 5 Ute Schmid: *Interactive Learning with Mutual Explanations in Relational Domains*. *Human-Like Machine Intelligence 2022*: 338-354 <https://doi.org/10.1093/OSO/9780198862536.003.0017>

3.20 Explainable models via compression of tree ensembles

Sriraam Natarajan (University of Texas at Dallas – Richardson, US)

License © Creative Commons BY 4.0 International license
© Sriraam Natarajan

Joint work of Siwen Yan, Sriraam Natarajan, Saket Joshi, Roni Khardon, Prasad Tadepalli

Main reference Siwen Yan, Sriraam Natarajan, Saket Joshi, Roni Khardon, Prasad Tadepalli: “Explainable Models via Compression of Tree Ensembles” *Mach. Learn.*: 1-26 (2023)

URL <https://doi.org/10.1007/s10994-023-06463-1>

We consider the problem of explaining learned (relational) ensemble models. Ensemble models (bagging and gradient-boosting) of relational decision trees have proved to be one of the most effective learning methods in the area of probabilistic logic models (PLMs). While effective, they lose one of the most important aspect of PLMs – interpretability.

Our key hypothesis in this work is that combining large number of logical decision trees would yield in a more compressed model compared to that of combining standard decision trees. This is due to the fact that unification of variables in logic would allow for effective and efficient compression.

To this effect, we propose CoTE – Compression of Tree Ensembles – that produces a single small decision list as a compressed representation. CoTE first converts the trees to decision lists and then performs the combination and compression with the aid of the original training set. Experiments on standard benchmarks demonstrate the value of this approach and justifies the hypotheses that compression is more effective in logical decision trees.

3.21 Inductive Programming meets Large Language Models

Gust Verbruggen (Microsoft – Keerbergen, BE)

License  Creative Commons BY 4.0 International license
© Gust Verbruggen

Joint work of Gust Verbruggen, Vu Le, Sumit Gulwani

Main reference Gust Verbruggen, Vu Le, Sumit Gulwani: “Semantic programming by example with pre-trained models”, Proc. ACM Program. Lang., Vol. 5(OOPSLA), pp. 1–25, 2021.

URL <https://doi.org/10.1145/3485477>

Both inductive programming (IP) and large language models (LLMs) are able to complete a task from a few examples. Instead of pitting them against each other, together they can achieve a lot more. One example of such integration is FlashGPT, which iteratively uses witness functions to break an inductive programming problem into smaller subproblems until all are solved (*FlashFill*) and leverages an LLM to solve the subproblems that cannot be solved symbolically (*GPT-3*). Instead of reiterating what has been discovered, this talk focused on (a non-exhaustive list of) next steps for combining IP and LLMs.

First, we discuss how the LLM can be used to improve learning in a fully symbolic IP system. Two approaches are (1) using the LLM to generate additional input-output examples for the IP system, or (2) using the LLM to generate candidate solutions to serve as seeds for initiating a search. The latter is a combination of component-based synthesis [1] and sketching, both of which rely on generating useful substructures over the grammar of the target language.

Second, we show how the LLM can be used to improve the experience of working with an IP system, by providing natural language descriptions of the learned programs.

Third, we show how the scope of IP can be improved with LLMS in systems that do not leverage witness functions. One potential method is masking semantic components, performing IP as usual and learning a program that emits masks, and then resolving the masks using an LLM.

Fourth, we show how operators that only use the embeddings from LLMs strike a balance between the inference speed of symbolic operations and the number of examples and capabilities of semantic operations. When the domain of a semantic relation is finite, or when the task is extraction of relevant parts of the input, we can use embeddings of tokens from the input to capture semantic relations between input and output.

References

- 1 Yoad Lustig, Moshe Y. Vardi: *Synthesis from component libraries*. Int. J. Softw. Tools Technol. Transf. 15(5-6): 603-618 (2013) <https://doi.org/10.1007/S10009-012-0236-Z>
- 2 Armando Solar-Lezama: *The Sketching Approach to Program Synthesis*. APLAS 2009: 4-13 https://doi.org/10.1007/978-3-642-10672-9_3

3.22 Inductive Programming meets Real User Problems

Gust Verbruggen (Microsoft – Keerbergen, BE)

License  Creative Commons BY 4.0 International license
© Gust Verbruggen

We show two novel applications of inductive programming that bring some unique challenges with respect to parsing user input. Both problems share some challenges: the need for speed, noisy input and labels, inferring constant values that adhere to a semantic bias, underspecification of the problem, and suppression of programs with low confidence.

First, we consider the problem of predicting the folder to which an email should be moved. Popular email clients offer to automate this functionality by setting rules, and the expected output of our learner is thus such a rule. An additional challenge with this problem is concept drift. Our approach [1] learns simple propositional rules in conjunctive normal form by generalizing (if an email is mistakenly not covered) or specializing (if an email is mistakenly covered) the rule corresponding to a folder. Because we guarantee that all historical emails are correctly classified, we easily adapt to concept drift. This classic inductive programming approach performs better than many neural and hybrid baselines.

Second, we consider the problem of learning conditional formatting rules in spreadsheets. An additional challenge is the scope of functions that can be used. Our approach [2, 3] uses semi-supervised clustering of input values to tackle underspecification, then learns different rules as decision trees, and ranks them with a learned ranker. Our corpus of 102K rules from real spreadsheets allows this ranker to encode the semantic bias, which allows us to outperform many neural and symbolic approaches, even if they have access to the same set of base predicates.

References

- 1 Mukul Singh, José Cambronero, Sumit Gulwani, Vu Le, Gust Verbruggen: *EmFore: Online Learning of Email Folder Classification Rules*. CIKM 2023: 2280-2290 <https://doi.org/10.1145/3583780.3614863>
- 2 Mukul Singh, José Pablo Cambronero Sánchez, Sumit Gulwani, Vu Le, Carina Negreanu, Mohammad Raza, Gust Verbruggen: *CORNET: Learning Table Formatting Rules By Example*. Proc. VLDB Endow. 16(10): 2632-2644 (2023) <https://doi.org/10.14778/3603581.3603600>
- 3 Mukul Singh, José Pablo Cambronero Sánchez, Sumit Gulwani, Vu Le, Carina Negreanu, Gust Verbruggen: *CORNET: Learning Spreadsheet Formatting Rules By Example*. Proc. VLDB Endow. 16(12): 4058-4061 (2023) <https://doi.org/10.14778/3611540.3611620>

3.23 Probabilistic Logic Programming: Quo Vadis?

Felix Weitekämper (LMU München, DE)

License © Creative Commons BY 4.0 International license
© Felix Weitekämper


Probabilistic Inductive Logic Programming refers to learning probabilistic relational programs from “examples”. These could be probabilistic logic programs, but many considerations also apply to learning other statistical relational models. From the perspective of statistical relational artificial intelligence, this is usually referred to as *structure learning*. Probabilistic Inductive Logic Programming is key in several areas of artificial intelligence, including knowledge discovery in stochastic, relational domains and causal structure discovery in a Boolean relational setting. Probabilistic inductive logic programming is traditionally considered difficult, since it adds another dimension to the classical ILP problem. Current approaches are still based on traditional ILP approaches developed in the 1990s, while the field of ILP has since made huge progress: Metainterpretive learning provides a new conceptual framework for rethinking Inductive Logic Programming, Constraints and learning from failures can help prune the search space, and Powerful ASP encodings can be leveraged to achieve more consistent outcomes.

This raises the question: Can we leverage these modern techniques for PILP?

4 Working groups

4.1 Large Language Models and Inductive Programming in Cognitive Architectures

Bettina Finzel (Universität Bamberg, DE) and Frank Jäkel (TU Darmstadt, DE)

License  Creative Commons BY 4.0 International license
© Bettina Finzel and Frank Jäkel

Cognitive architectures provide frameworks to simulate and test principles of cognition [1]. There are different components in cognitive architectures that qualify for being enhanced by large language models (LLMs) [3] and inductive programming (IP) [2]. LLMs could be used in the production module to generate rules for execution, in the memory module as a compressor of information for more efficient access and possibly as the stimuli of a general cognitive architecture as a model that produces outputs on which further reasoning could be applied for decision making and learning. An open challenge remains in mimicking the abilities of humans to switch between modalities in the sense that they are able to dynamically choose between the representations they need. With respect to this, we were discussing about some form of reward or reinforcer to increase the response for certain signals or items in the process of inference and problem solving. Combining learning and reasoning by integrating LLMs and IP in a cognitive architecture could be an enabler for validating programs that get executed by the overall architecture and to possibly get nearer to human performance.

References

- 1 Paul Thagard: *Cognitive Architectures*. The Cambridge Handbook of Cognitive Science 2012: 50-70 <https://doi.org/10.1017/CB09781139033916.005>
- 2 Sumit Gulwani, José Hernández-Orallo, Emanuel Kitzelmann, Stephen H. Muggleton, Ute Schmid, Benjamin G. Zorn: Inductive programming meets the real world. *Commun. ACM* 58(11): 90-99 (2015) <https://doi.org/10.1145/2736282>
- 3 Naman Jain, Skanda Vaidyanath, Arun Shankar Iyer, Nagarajan Natarajan, Suresh Parthasarathy, Sriram K. Rajamani, Rahul Sharma: Jigsaw: Large Language Models meet Program Synthesis. *ICSE 2022*: 1219-1231 <https://doi.org/10.1145/3510003.3510203>

4.2 Avoiding too much search in Inductive Programming

Ute Schmid (Universität Bamberg, DE), David Cerna (The Czech Academy of Sciences – Prague, CZ), and Hector Geffner (RWTH Aachen, DE)

License  Creative Commons BY 4.0 International license
© Ute Schmid, David Cerna, and Hector Geffner

A crucial part of inductive programming (IP) is search. Since search is costly, an important question is how we can avoid to search so much or too much.

What we search for can be very different things: logic or functional programs, but also decision lists, policies, classifications, language representations or deep learned models. How search is performed can be also realized with many different approaches: enumeration, anti-unification, genetic programming, greedy strategies, combinatorial optimization, stochastic gradient descent, deep reinforcement learning, or Monte Carlo Tree Search. In general, we do need to learn structure as well as probabilities.

To evaluate the quality of the learned program, different aspects might be focused on alone or in combination which is a challenge for search. Obvious criteria are sample complexity and scalability. But one might also be interested in novelty of the learned program, how similar is the inductive strategy to human learning (humans do not enumerate first and then select but typically generalise over few examples).

To push research on becoming more search efficient, a set of benchmark problems and a competition should be introduced. Promising challenge data sets might come from the FlashFill domain (learning more complex Excel functions and string transformations), the abstract reasoning challenge (ARC) and the modified ILP version might be interesting, furthermore, we could look at problems from the International Math Olympiad Challenge.

We should critically evaluate for which problems deep learning/generative approaches are more successful and hopefully identify a class of problems where symbolic IP is superior. For instance, the IP system FlashFill performs better than the transformer-based SmartFill. The core difference between neural network approaches and symbolic IP is that IP returns explicit programs which give the intensional characterisation of the input/output-examples while neural networks are extensional representations. Therefore, one might postulate that neural networks cannot learn recursion.

Currently, string transformation problems are often either syntactic (return the first letter of a string) – which works very well for symbolic IP – or semantic (give the capital for a country) – where generative AI is very good at. Maybe we should look for transformation problems which combine syntactic and semantic transformations (return the first letter for every string which names a capital). As it is so often the case, it might be a good idea to combine symbolic IP and deep learning/generative approaches. A paper we should look at is [1].

References

- 1 Stéphane d’Ascoli, Pierre-Alexandre Kamienny, Guillaume Lample, François Charton: *Deep symbolic regression for recurrence prediction*. ICML 2022: 4520-453

4.3 Evaluation Criteria for Interpretability and Explainability of Inductive Programming

Ute Schmid (Universität Bamberg, DE), Lun Ai (Imperial College London, GB), Claudia d’Amato (University of Bari, IT), and Johannes Fürnkranz (Johannes Kepler Universität Linz, AT)

License © Creative Commons BY 4.0 International license
© Ute Schmid, Lun Ai, Claudia d’Amato, and Johannes Fürnkranz

Inductive programming results in symbolic models (programs) which are inherently interpretable. Nevertheless, there might be a need for explainability to humans – end-users or domain experts from other areas than computer science. In the discussion group we focused on the question of how to measure the quality of interpretable representations (programs) and of post-hoc generated explanations. The main challenge is to provide for assessment metrics which are not dependent on studies with humans but which can be evaluated directly for the interpretation/explanations. A core difficulty is that the quality of an explanation is context-dependent: it depends on what is explained to whom in what way (how) and for what reason (why). The way to explain something can be a set of symbolic rules (learned with IP or a rule learning system or extracted from a neural net), highlighting important

features (which is done by many XAI approaches such as LIME, SHAP or LRP), a natural language explanation, prototypical or near miss examples. Furthermore, explanations can be more abstract or give more details. Explanations can either be constructed to explain for what reason a learned (black box) model gave a specific output (mechanistic explanation) or to explain the learned content to a human (functional explanation, ultra-strong machine learning).

As candidates for assessment metrics we discussed (1) complexity measures (proposals for cognitive complexity measures, structural information theory, Kolmogorov complexity), (2) semantic coherence, (3) reliability of a component (of a program) which can result in abstracting this part away if a human has sufficient/justified trust. A further aspect for evaluation might be a suitable trade-off between the size of the explanation (memory) and the effort to interpret it (run time) as proposed, for instance by Donald Michie [1] or Lun Ai [2].

A program itself can be a good explanation, depending of its complexity. Abstraction might be a useful method to make explanations more comprehensible. Here approaches like predicate/function invention, anti-unification, introducing higher-orderness (such as map/fold), or compression might be helpful. A hierarchy of abstractions can be helpful for providing the ‘right’ level of detail for a given explanatory context. There are first approaches of explanations as a dialogue where more detailed or different forms of explanations can be presented to a human [3]. Learned (Prolog) programs are also suitable in this context: The highest level of abstraction refers to a single (left-hand/target/head) predicate, the next level of detail can be achieved by presenting the instantiated right-hand side of a rule (or a verbal description of it), continued by expanding predicates in the body until ground facts are reached.

Recently, an explainable version FlashFill has been developed. It showed that users sometimes reject a FlashFill rule which correctly covers the examples because they do not understand it. Here approximate symbolic regression has been applied to provide simpler explanations [4]. In the group of Josh Tenenbaum, the system LILO [5] has been developed which provides explanations by abstraction.

A final idea on assessing the quality of an interpretation/explanation has been to input explanations of a learned programs to a LLM, let the LLM generate a program from that and then compare the originally synthesized program with the one generated by the LLM (a kind of loss function). Comparison can be done by behavioral comparison for test cases or by comparing the code.

References

- 1 Donald Michie: *Experiments on the Mechanization of Game-Learning. 2-Rule-Based Learning and the Human Window*. Comput. J. 25(1): 105-113 (1982) <https://doi.org/10.1093/COMJNL/25.1.105>
- 2 Lun Ai, Stephen H. Muggleton, Céline Hocquette, Mark Gromowski, Ute Schmid: *Beneficial and harmful explanatory machine learning*. Mach. Learn. 110(4): 695-721 (2021) <https://doi.org/10.1007/S10994-020-05941-0>
- 3 Bettina Finzel, David Elias Tafler, Anna Magdalena Thaler, Ute Schmid: *Multimodal Explanations for User-centric Medical Decision Support Systems*. HUMAN@AAAI Fall Symposium 2021
- 4 Mukul Singh, José Cambronero, Sumit Gulwani, Vu Le, Carina Negreanu, Gust Verbruggen: *CodeFusion: A Pre-trained Diffusion Model for Code Generation*. EMNLP 2023: 11697-11708 <https://doi.org/10.48550/arXiv.2310.17680>

- 5 Gabriel Grand, Lionel Wong, Matthew Bowers, Theo X. Olausson, Muxin Liu, Joshua B. Tenenbaum, Jacob Andreas: *LILLO: Learning Interpretable Libraries by Compressing and Documenting Code*. CoRR abs/2310.19791 (2023) <https://doi.org/10.48550/ARXIV.2310.19791>

4.4 Finding Suitable Benchmark Problems for Inductive Programming

Ute Schmid (Universität Bamberg, DE), Martin Berger (University of Sussex – Brighton, GB), Sebastijan Dumancic (TU Delft, NL), Nathanaël Fijalkow (CNRS – Talence, FR), and Gust Verbruggen (Microsoft – Keerbergen, BE)

License © Creative Commons BY 4.0 International license
© Ute Schmid, Martin Berger, Sebastijan Dumancic, Nathanaël Fijalkow, and Gust Verbruggen

To advance progress as well as visibility of IP, a collection of suitable benchmarks, convincing use cases, and joint formats to represent problems, as well as starting an IP challenge have been identified as helpful. In the discussion group, we focussed on benchmark sets.

First we collected problems currently used in different groups: List problems, regular expressions (RegEx), boolean language inference, competitive programming, Math Olympiad Challenge, Reasoning/Theorem Proving, Planning, Knowledge Graphs, Zendo, Games, Navigation, Biology, standard ML benchmarks (UCML Repository), natural language to programs (NL2P), abstract reasoning challenge (ARC).

Then we discussed what characteristics benchmark problems should have: tunable, clear performance metrics, standard format, correct annotations, noise, social recognition/PR, breadth, not solvable by LLMs (alone), conceptual jumps, linkable to external resources, curriculum, dramatic finish line, doable by humans. Several of these characteristics were discarded. For instance, clear metrics (beyond just right or wrong) did not seem to be a good fit (but see discussion results about explainability). Format has also been seen as not relevant compared to having good environments to execute and evaluate learned programs and tools/environments which are easily usable.

For a selected set of characteristics, we identified that problems from the list above which fulfill the respective characteristic:

- Tunable: List problems, RegEx, Boolean languages, Knowledge Graphs, Games, Navigation, (competitive programming)
- Breadth: List problems, competitive programming, Games, NLP2P, Math Olymp, ARC
- Not solvable by LLM: List problems, Knowledge Graphs, Math Olymp, ARC
- Dramatic finish: Math Olymp, Biology, NLP2P, ARC
- Curriculum: List problems, RegEx, Math Olymp, Navigation
- Doable by (average) humans: List Problems, RegEx, Games, Navigation, ARC

Given the number of criteria which are met, the following problem domains have been identified as the most promising ones: List problems (including string transformations and other domain-specific language based approaches), RegEx, Math Olymp, ARC.

We then had a further critical look at the selected problem classes and evaluated the following aspects: Not suitable for application, lack of format, producible, lack of probabilistic benchmarks, tension between standard and generation, domain specific problems, not perceived as difficult/relevant, need/miss to have a relational core, loss of propositional benchmarks, lack of diversity of evolution, novelty (invent a new sorting algorithm, automated computer scientist), plugable.

After discussing these additional aspects, we came up with the following set of potentially interesting benchmark problems:

- The Automated Computer Scientist (from Andrew Cropper): learning novel (e.g. sorting) algorithms or novel data structures [1]
- Joint IP and KG (Knowledge Graph) problems, especially for combining syntactic and semantic transformations (e.g. give the capital for a country and then take the first letter of it), this can be list problems or Excel tables [5]
- Strategy learning (explicit compared to implicit policy learning in reinforcement learning): for human problem solving, planning (look at problems from the planning competition) [2, 3, 4]
- Online encyclopedia of integer sequences OEIS (not for all of them exists a closed formula)
- Expert domains: learning strategies/patterns for SAT-solvers, theorem provers
- Constructing ML pipelines (AutoML)

Links to benchmark data sets:

- Popper's (includes Zendo, many others)
<https://github.com/logic-and-learning-lab/Popper/tree/main/examples>
- SyGuS (includes list problems, FlashFill, phone numbers)
<https://github.com/SyGuS-Org/benchmarks>
- IP Repository (programming benchmarks, including problem solving like Tower of Hanoi)
<https://www.inductive-programming.org/repository.html>
- Regular expressions
<https://codalab.lisn.upsaclay.fr/competitions/15096>
- Boolean language inference
<https://www.iwls.org/contest/>
- Competitive programming
<https://github.com/openai/human-eval>
- Math Olympiad Challenge
<https://github.com/lupantech/dl4math#-mathematical-reasoning-benchmarks>
<https://github.com/openai/miniF2F/tree/v1>
- Planning
<https://github.com/AI-Planning/pddl-generators>
- Program synthesis benchmarks from genetic programming community
<https://cs.hamilton.edu/~thelmuth/PSB2/PSB2.html>
<https://zenodo.org/records/5084812>
- Standard ML benchmarks: UC Irvine ML Repository
<https://archive.ics.uci.edu/>
- Abstract Reasoning Challenge (ARC) [7, 8]
<https://github.com/fchollet/ARC>
<https://lab42.global/arc/>
- E-Mail Folder Classification
<http://www-2.cs.cmu.edu/~enron/>
<https://catalog.ldc.upenn.edu/LDC2015T03>
- Rule learning
<https://github.com/kliegr/arcbench>

References

- 1 Andrew Cropper: *The Automatic Computer Scientist*. AAAI 2023: 15434 <https://doi.org/10.1609/AAAI.V37I13.26801>
- 2 Mario Martín, Hector Geffner: *Learning Generalized Policies from Planning Examples Using Concept Languages*. Appl. Intell. 20(1): 9-19 (2004) <https://doi.org/10.1023/B:APIN.0000011138.20292.DD>
- 3 Ute Schmid, Emanuel Kitzelmann: *Inductive rule learning on the knowledge level*. Cogn. Syst. Res. 12(3-4): 237-248 (2011) <https://doi.org/10.1016/J.COGSYS.2010.12.002>
- 4 Jude W. Shavlik: *Acquiring Recursive and Iterative Concepts with Explanation-Based Learning*. Mach. Learn. 5: 39-40 (1990) <https://doi.org/10.1007/BF00115894>
- 5 Mukul Singh, José Pablo Cambronero Sánchez, Sumit Gulwani, Vu Le, Carina Negreanu, Mohammad Raza, Gust Verbruggen: *CORNET: A neurosymbolic approach to learning conditional table formatting rules by example*. CoRR abs/2208.06032 (2022) <https://doi.org/10.48550/ARXIV.2208.06032>
- 6 Mukul Singh, José Cambronero, Sumit Gulwani, Vu Le, Gust Verbruggen: *EmFore: Online Learning of Email Folder Classification Rules*. CIKM 2023: 2280-2290 <https://doi.org/10.1145/3583780.3614863>
- 7 James Ainooson, Deepayan Sanyal, Joel P. Michelson, Yuan Yang, Maithilee Kunda: *An Approach for Solving Tasks on the Abstract Reasoning Corpus*. CoRR abs/2302.09425 (2023) <https://doi.org/10.48550/ARXIV.2302.09425>
- 8 Jonas Witt, Stef Rasing, Sebastijan Dumancic, Tias Guns, Claus-Christian Carbon: *A Divide-Align-Conquer Strategy for Program Synthesis*. CoRR abs/2301.03094 (2023) <https://doi.org/10.48550/ARXIV.2301.03094>

5 Panel discussions

5.1 Inductive Programming – How to Go On?

Ute Schmid (Universität Bamberg, DE), Claudia d'Amato (University of Bari, IT), Hector Geffner (RWTH Aachen, DE), Sriraam Natarajan (University of Texas at Dallas – Richardson, US), and Josh Rule (University of California – Berkeley, US)

License © Creative Commons BY 4.0 International license
© Ute Schmid, Claudia d'Amato, Hector Geffner, Sriraam Natarajan, and Josh Rule

In a final discussion we addressed topics and activities to make scientific progress and make the topic more visible. A crucial problem might be that there is no core general approach to IP (such as gradient descent for neural networks). The most prominent IP task is to learn programs from input/output examples. Other approaches address learning programs from traces or constraints. Methods range from classic inductive generalization and folding for induction of functional programs over genetic and evolutionary programming to a collection of ILP methods (sequential covering, theta-subsumption, combining with tools from answer set programming). Relevant use cases might not have a focus on learning recursion/loops but on relations (e.g. in medicine and biology). The focus on learning programs (including recursion) might profit from using Python as target language.

Furthermore, current IP systems are mostly not easy to find and to use. Therefore, a toolbox which can be easily used (such as Weka for standard ML) might be helpful. Currently Sebastijan Dumancic is working on such a tool box (Herb.jl). A collection of data sets and benchmark problems (see summary of the discussion group about benchmarks) would also be very helpful, especially when they are given in a standardized, easy to parse format.

To make the field of IP less distributed, it might be helpful to write a primer to IP including the classic approaches of inductive functional programming and relate also to deductive and transformational program synthesis methods and genetic/evolutionary programming. Papers falling in this category are:

- Andrew Cropper, Sebastijan Dumancic:
Inductive Logic Programming At 30: A New Introduction. J. Artif. Intell. Res. 74: 765-850 (2022) <https://doi.org/10.1613/JAIR.1.13507>
- Pierre Flener, Ute Schmid:
Inductive Programming. Encyclopedia of Machine Learning and Data Mining 2017: 658-666 https://doi.org/10.1007/978-1-4899-7687-1_137
(updated in 2020, not online yet)
- Pierre Flener, Ute Schmid:
An introduction to inductive programming. Artif. Intell. Rev. 29(1): 45-62 (2008)
<https://doi.org/10.1007/S10462-009-9108-7>
- Sumit Gulwani, Oleksandr Polozov, Rishabh Singh: *Program Synthesis*. Found. Trends Program. Lang. 4(1-2): 1-119 (2017) <https://doi.org/10.1561/2500000010>
- Emanuel Kitzelmann:
Inductive Programming: A Survey of Program Synthesis Techniques. AAIP 2009: 50-73
https://doi.org/10.1007/978-3-642-11931-6_3

As outcome of the AAIP 2023 seminar we plan to publish a book “Inductive Programming” which contains systematic introductions into the core topics as well as a collection of recent work (from participants of the seminar plus an open call for contributions) addressing topics such as “New Approaches to IP”, “Cognitive Aspects of IP”, “Applications of IP”.

Furthermore, it has been proposed to apply for an IP workshop at IJCAI and to try to include IP as a topic at the next European Summer School on AI (ESSAI). It might also be helpful for visibility and community building to propose a COST Network on IP.

The website <https://www.inductive-programming.org/> should be kept but made more general and link from there to a github for IP. The Wikipedia Entry on Inductive Programming https://en.wikipedia.org/wiki/Inductive_programming should be updated by the community. We also might make at least a tag `inductiveprogramming` at linkedin which the IP community should include in posts. More persons of the community should give Inductive Programming as Keyword in their google scholar profile. We could sample all videos related to IP in a YouTube channel, and we could produce a 3 minute introductory video.

Participants

- Lun Ai
Imperial College London, GB
- Martin Berger
University of Sussex –
Brighton, GB
- David Cerna
The Czech Academy of Sciences –
Prague, CZ
- David J. Crandall
Indiana University –
Bloomington, US
- Claudia d’Amato
University of Bari, IT
- Luc De Raedt
KU Leuven, BE
- Sebastijan Dumančić
TU Delft, NL
- Kevin Ellis
Cornell University – Ithaca, US
- Nathanaël Fijalkow
CNRS – Talence, FR
- Bettina Finzel
Universität Bamberg, DE
- Johannes Fürnkranz
Johannes Kepler Universität
Linz, AT
- Hector Geffner
RWTH Aachen, DE
- Céline Hocquette
University of Oxford, GB
- Frank Jäkel
TU Darmstadt, DE
- Emanuel Kitzelmann
Technische Hochschule
Brandenburg, DE
- Tomáš Kliegr
University of Economics –
Prague, CZ
- Maithilee Kunda
Vanderbilt University –
Nashville, US
- Johannes Langer
Universität Bamberg, DE
- Sriraam Natarajan
University of Texas at Dallas –
Richardson, US
- Stassa Patsantzis
University of Surrey –
Guildford, GB
- Josh Rule
University of California –
Berkeley, US
- Zeynep G. Saribatur
TU Wien, AT
- Ute Schmid
Universität Bamberg, DE
- Gust Verbruggen
Microsoft – Keerbergen, BE
- Felix Weitkämper
LMU München, DE

