



Volume 8, Issue 2, February 2018

Evidence About Programmers for Programming Language Design (Dagstuhl Seminar 18061) <i>Andreas Stefk, Bonita Sharif, Brad. A. Myers, and Stefan Hanenberg</i>	1
Planning and Operations Research (Dagstuhl Seminar 18071) <i>J. Christopher Beck, Daniele Magazzeni, Gabriele Röger, and Willem-Jan Van Hoeve</i>	26
Designing and Implementing Algorithms for Mixed-Integer Nonlinear Optimization (Dagstuhl Seminar 18081) <i>Pierre Bonami, Ambros M. Gleixner, Jeff Linderoth, and Ruth Misener</i>	64
Formal Methods for the Synthesis of Biomolecular Circuits (Dagstuhl Seminar 18082) <i>Yaakov Benenson, Neil Dalchau, Heinz Koepl, and Oded Maler</i>	88
Data Consistency in Distributed Systems: Algorithms, Programs, and Databases (Dagstuhl Seminar 18091) <i>Annette Bieniusa, Alexey Gotsman, Bettina Kemme, and Marc Shapiro</i>	101
The Logical Execution Time Paradigm: New Perspectives for Multicore Systems (Dagstuhl Seminar 18092) <i>Rolf Ernst, Stefan Kuntz, Sophie Quinton, and Martin Simons</i>	122

ISSN 2192-5283

Published online and open access by

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

Publication date

August, 2018

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 <http://dnb.d-nb.de>.

License

This work is licensed under a Creative Commons Attribution 3.0 DE license (CC BY 3.0 DE).



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

- Gilles Barthe
- Bernd Becker
- Daniel Cremers
- Stephan Diehl
- Reiner Hähnle
- Lynda Hardman
- Hannes Hartenstein
- Oliver Kohlbacher
- Bernhard Mitschang
- Bernhard Nebel
- Bernt Schiele
- Albrecht Schmidt
- Raimund Seidel (*Editor-in-Chief*)
- Emmanuel Thomé
- Heike Wehrheim
- Verena Wolf

Editorial Office

Michael Wagner (*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
<http://www.dagstuhl.de/dagrep>

Digital Object Identifier: 10.4230/DagRep.8.2.i

Evidence About Programmers for Programming Language Design

Edited by

Andreas Stefk¹, Bonita Sharif², Brad A. Myers³, and
Stefan Hanenberg⁴

1 Univ. of Nevada – Las Vegas, US, stefika@gmail.com

2 Youngstown State University, US, shbonita@gmail.com

3 Carnegie Mellon University – Pittsburgh, US, bam@cs.cmu.edu

4 Univ. of Duisberg-Essen, DE, stefan.hanenberg@uni-due.de

Abstract

The report documents the program and outcomes of Dagstuhl Seminar 18061 “Evidence About Programmers for Programming Language Design”. The seminar brought together a diverse group of researchers from the fields of computer science education, programming languages, software engineering, human-computer interaction, and data science. At the seminar, participants discussed methods for designing and evaluating programming languages that take the needs of programmers directly into account. The seminar included foundational talks to introduce the breadth of perspectives that were represented among the participants; then, groups formed to develop research agendas for several subtopics, including novice programmers, cognitive load, language features, and love of programming languages. The seminar concluded with a discussion of the current SIGPLAN artifact evaluation mechanism and the need for evidence standards in empirical studies of programming languages.

Seminar February 4–9, 2018 – <https://www.dagstuhl.de/18061>

2012 ACM Subject Classification Software and its engineering → Software notations and tools, Human-centered computing → HCI design and evaluation methods, Human-centered computing → Accessibility, Social and professional topics → Computing education

Keywords and phrases programming language design, computer science education, empirical software engineering, eye tracking, evidence standards

Digital Object Identifier 10.4230/DagRep.8.2.1

Edited in cooperation with Michael Coblenz

1 Executive summary

Michael Coblenz (Carnegie Mellon University – Pittsburgh, US)

License  Creative Commons BY 3.0 Unported license
© Michael Coblenz

Programming languages underlie and have significant impact on software development, especially in terms of the ability of programmers to achieve their goals. Although designers of programming languages can already reason about the *formal* properties of their languages, few tools are available to assess the impact of design decisions on programmers and software engineers.

At Dagstuhl Seminar 18061, a diverse set of participants gathered to review the existing body of evidence about programmers that has implications on programming language design.



Except where otherwise noted, content of this report is licensed under a Creative Commons BY 3.0 Unported license

Evidence About Programmers for Programming Language Design, *Dagstuhl Reports*, Vol. 8, Issue 02, pp. 1–25
Editors: Andreas Stefk, Bonita Sharif, Brad A. Myers, and Stefan Hanenberg



Dagstuhl Reports

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

Participants also reviewed existing research methods, such as eye tracking, that may help better understand the impact of language design decisions on programmers. Participants brainstormed a long list of possible research questions for investigation (§4), and then divided into working groups (§5) to focus on several areas of research interest, including novices, context switching and cognitive load, language features, emotional attachment to languages, and representativeness of subjects in studies. In each area, participants proposed research methods and questions that they felt would be valuable to address in the future. Then, the group discussed and prioritized these research questions.

The seminar included a discussion of the need for an evidence standard in empirical studies of programming languages, focusing on content of the evidence standard, adoption mechanisms, and criteria for what it might include in our field. Finally, the seminar concluded with a discussion of future directions for research, including a list of research questions that the participants were planning on collaborating on in the near future.

2 Contents

Executive summary

<i>Michael Coblenz</i>	1
----------------------------------	---

Overview of Talks

How do PL Researchers Think? and Evidence about Software Engineers for PL Design <i>Jonathan Aldrich</i>	5
How does the Programming Language Community Design a Language? <i>Craig Anslow</i>	5
Thinking about Programmers with Disabilities <i>Ameer Armaly</i>	6
Enhancing Compiler Error Messages: What's been done, what needs doing <i>Brett A. Becker</i>	6
Visualizing and Interpreting Multimodal Data <i>Tanja Blascheck</i>	8
Methodological Considerations <i>Brian Dorn and Briana B. Morrison</i>	9
I Don't Understand Program Comprehension <i>Johannes C. Hofmeister</i>	9
What evidence do we have about programming language design? <i>Antti-Juhani Kaijanaho and Andrew J. Ko</i>	9
Scientists Programming <i>Amelia A. McNamara</i>	10
Types of Studies <i>Brad A. Myers, Jonathan Aldrich, Michael Coblenz, and Joshua Sunshine</i>	10
Programming language cultures are relevant <i>Lutz Prechelt</i>	11
Eye Tracking in Program Comprehension <i>Bonita Sharif</i>	11
Early Experimental Studies of Conditionals in Programming Languages <i>Walter F. Tichy</i>	12

Future Research Questions and Studies

Impacts in the field	14
Methodology	15
Language features	15
Novices and learning	16
Cognition and affect	18

Working groups

Novices: polyglot questions <i>Johannes Bechberger, Scott Fleming, Briana B. Morrison, Bonita Sharif, Andreas Stefik</i>	19
Context Switching and Cognitive Load <i>Ameer Armaly, Andrew Begel, Tanja Blascheck, John Daughtry, Rob DeLine, Ciera Jaspán, Philip Merlin Uesbeck</i>	19
Language features and error-proneness <i>Jonathan Aldrich, Michael Coblenz, Andrew J. Ko, Thomas LaToza, Sibylle Schupp, and Walter Tichy</i>	20
Why do people love programming languages? <i>Andrew Duchowski, Matthias Hauswirth, Brad A. Myers, Craig Anslow, Brian Dorn, Lutz Prechelt, and Antti-Juhani Kaijanaho</i>	21
Representativeness of subjects in studies <i>Felienne Hermans, Johannes C. Hofmeister, Amelia A. McNamara, Lea Verou</i> . . .	22
Open problems	
Evidence standards	23
Participants	25

3 Overview of Talks

3.1 How do PL Researchers Think? and Evidence about Software Engineers for PL Design

Jonathan Aldrich (Carnegie Mellon University – Pittsburgh, US)

License © Creative Commons BY 3.0 Unported license

© Jonathan Aldrich

Joint work of Michael Coblenz, Joshua S. Sunshine, Brad A. Myers

Evidence about programmers is of increasing interest in programming language design, and my colleagues and I have argued for incorporating this evidence in an interdisciplinary approach to programming language design [1]. However, PL design is a domain quite different from other domains in which human behavior has been studied. This talk is an introduction, aimed at HCI experts and others studying human behavior, to the field of programming language design. I'll give an example of programming language design, discuss the goals and potential impact of language design, and talk about how PL researchers currently think about language design. I'll talk about what makes a programming language viable for large-scale development, and about how ideas from software engineering (which are ultimately derived from empirical research) could influence improve future language designs. Finally, I'll suggest some strategies that empirically-focused researchers can use to maximize their impact on PL design.

References

- 1 Michael Coblenz, Jonathan Aldrich, Joshua Sunshine, and Brad Myers. *Interdisciplinary Programming Language Design*. Working draft, available at <http://www.cs.cmu.edu/~aldrich/papers/interdisciplinary-pl-design.pdf>, February 2018.

3.2 How does the Programming Language Community Design a Language?

Craig Anslow (Victoria University – Wellington, NZ)

License © Creative Commons BY 3.0 Unported license

© Craig Anslow

Designing a programming language is hard. The design process is complicated and different for each language. In this talk, the results from a study are presented from interviewing expert programming language designers (n=25) in the SIGPLAN community on how they designed their languages. The study also asked if they have conducted user experiments on their language and how they value these types of experiments. Results show that designing a language is very hard and a complex process; very few have adopted human-centered design methods or conducted user experiments. However, most language designers felt that conducting user experiments is of great value but very hard to perform. Encouraging programming language designers to adopt human-centered design methods and conduct user experiments will help to improve the usability and effectiveness of their languages.

3.3 Thinking about Programmers with Disabilities

Ameer Armaly (*University of Notre Dame, US*)

License © Creative Commons BY 3.0 Unported license
© Ameer Armaly

Programmers with disabilities face a variety of problems depending on the exact nature of their disability. Consequently, programmers with disabilities cannot be addressed as a monolithic group; researchers must instead examine the problems and impacts of individual disabilities. In this talk, I introduce the different ways disability has been thought of in society. I then present two research projects relating to programmers who are blind as examples of the different ways researchers can effectively examine and address the experience of programmers with disabilities. I also demonstrate in detail the experience of programming while blind and discuss ideas for future exploration.

3.4 Enhancing Compiler Error Messages: What's been done, what needs doing

Brett A. Becker (*University College Dublin, IE*)

License © Creative Commons BY 3.0 Unported license
© Brett A. Becker

Joint work of Brett A. Becker, Graham Glanville, Ricardo Iwashima, Claire McDonnell, Kyle Goslin, Catherine Mooney

Main reference Brett A. Becker, Graham Glanville, Ricardo Iwashima, Claire McDonnell, Kyle Goslin, Catherine Mooney: "Effective compiler error message enhancement for novice programming students", *Computer Science Education*, Vol. 26(2-3), pp. 148-175, 2016.

URL <http://dx.doi.org/10.1080/08993408.2016.1225464>

Programming is an essential skill that all computing students are expected to master. However, programming can be difficult to learn. Successfully interpreting compiler error messages is crucial for correcting errors and progressing toward success in learning programming. Yet these messages are often difficult to understand and pose a barrier to progress for many novices, with struggling students often exhibiting high frequencies of errors, particularly repeated errors [2]. The area of compiler error enhancement has seen increased activity in the last several years, particularly in presenting empirical evidence of effects. This talk focuses on these recent results. After discussing the importance of compiler error messages and the problems they present, empirical motivating evidence demonstrating a need for enhanced compiler error messages is reviewed. Then, recent empirical evidence on enhanced compiler error messages is discussed, including a debate on the effects and effectiveness of enhancing compiler error messages. In light of these recent results, the current state of play is then presented, organized in three areas: metrics and signals; compilers and languages; and frameworks, guidelines and principles. Finally, directions for research going forward are discussed.

References

- 1 Titus Barik, Justin Smith, Kevin Lubick, Elisabeth Holmes, Jing Feng, Emerson Murphy-Hill and Chris Parnin. *Do developers read compiler error messages?* Proceedings of the 39th International Conference on Software Engineering (ICSE 2017), Buenos Aires, Argentina, May 2017, IEEE Press. DOI: 10.1109/ICSE.2017.59
- 2 Brett A. Becker, Graham Glanville, Ricardo Iwashima, Claire McDonnell, Kyle Goslin and Catherine Mooney. *Effective compiler error message enhancement for novice*

- programming students*. Computer Science Education 26 (2-3), (2016) 148-175. DOI: 10.1080/08993408.2016.1225464
- 3 Brett A. Becker, Kyle Goslin, and Graham Glanville. *The effects of enhanced compiler error messages on a syntax error debugging test*. Proceedings of the 48th ACM Technical Symposium on Computer Science Education (SIGCSE 2018), Baltimore, Maryland, USA, February 2018. ACM. DOI: 10.1145/3159450.3159461
 - 4 Brett A. Becker, Cormac Murray, Tianyi Tao, Changheng Song, Robert McCartney and Kate Sanders. *Fix the first, ignore the rest: Dealing with multiple compiler error messages*. Proceedings of the 48th ACM Technical Symposium on Computer Science Education (SIGCSE 2018), Baltimore, Maryland, USA, February 2018. ACM. DOI: 10.1145/3159450.3159453
 - 5 Brett A. Becker. *An effective approach to enhancing compiler error messages*. Proceedings of the 47th ACM Technical Symposium on Computer Science Education (SIGCSE 2016), Memphis, Tennessee, USA, March 2016. ACM. DOI: 10.1145/2839509.2844584
 - 6 Brett A. Becker and Catherine Mooney. *Categorizing compiler error messages with principal component analysis*. 12th China-Europe International Symposium on Software Engineering Education, Shenyang, China, May 2016.
 - 7 Brett A. Becker. *A new metric to quantify repeated compiler errors for novice programmers*. Proceedings of the 21st Annual Conference on Innovation and Technology in Computer Science Education (ITiCSE 2016), Arequipa, Peru, July 2016. ACM. DOI: 10.1145/2899415.2899463
 - 8 Brett A. Becker. *An exploration of the effects of enhanced compiler error messages for computer programming novices*. MA Thesis, Dublin Institute of Technology, Dublin, Ireland, November 2015.
 - 9 Paul Denny, Andrew Luxton-Reilly and Dave Carpenter. *Enhancing syntax error messages appears ineffectual*. In Proceedings of the 19th Annual Conference on Innovation and Technology in Computer Science Education (ITiCSE 2014), Arequipa, Peru, July 2014. ACM. DOI: 10.1145/2591708.2591748
 - 10 Devon Harker. *Examining the effects of enhanced compilers on student productivity*. MSc Thesis, University of Northern British Columbia, Prince George, British Columbia, Canada, 2018.
 - 11 Michael J. Lee and Andy J. Ko. *Personifying programming tool feedback improves novice programmers learning*. In Proceedings of the 7th International Workshop on Computing Education Research (ICER 2011), Seattle, Washington, USA, August 2011, ACM. DOI: 10.1145/2016911.2016934
 - 12 Raymond S. Pettit, John Homer and Roger Gee. *Do enhanced compiler error messages help students?: Results inconclusive*. Proceedings of the 47th ACM Technical Symposium on Computer Science Education (SIGCSE 2017), Seattle, Washington, USA, March 2017, ACM. DOI: 10.1145/3017680.3017768
 - 13 James Prather, Raymond Pettit, Kayla Holcomb McMurray, Alani Peters, John Homer, Nevan Simone, and Maxine Cohen. *On novices' interaction with compiler error messages: A human factors approach*. Proceedings of the 2017 ACM Conference on International Computing Education Research (ICER 2017), Tacoma, Washington, USA, August 2018, ACM. DOI: 10.1145/3105726.3106169

3.5 Visualizing and Interpreting Multimodal Data

Tanja Blascheck (INRIA Saclay – Orsay, FR)

License © Creative Commons BY 3.0 Unported license
© Tanja Blascheck

Main reference Tanja Blascheck, Kuno Kurzhals, Michael Raschke, Michael Burch, Daniel Weiskopf, Thomas Ertl: “Visualization of Eye Tracking Data: A Taxonomy and Survey”, *Comput. Graph. Forum*, Vol. 36(8), pp. 260–284, 2017.

URL <http://dx.doi.org/10.1111/cgf.13079>

In program language design, evaluation plays a crucial role. In recent years, eye tracking has become one means to analyze how participants perceive and understand a programming language. Because programming is highly interactive, a study should take interaction into account as well. In addition, think-aloud data gives insights into cognitive processes of participants using a programming language. Typically, researchers evaluate these data sources separately. However, it is beneficial to correlate eye tracking, interaction, and think aloud data for deeper analyses. I present challenges and possible solutions in triangulating behavior using multiple evaluation data sources and how these approaches can be used to analyze programming languages. Overall, the objective of this talk is to provide methods and techniques that contribute to more holistic evaluation methods that, in turn, leads to an improved understanding of programming languages. Starting by exploring existing evaluation methodologies, a description of the state-of-the-art of visualization techniques [5] that are available for studying eye movement data are provided and it is discussed, to what extent the existing techniques can be applied. To integrate the use of eye movement data with log files and think-aloud protocols, three novel visualization techniques are contributed: the Radial Transition Graph [6], the AOI Sequence Chart [1], the AOI hierarchy approach [2], and a Visual Coding Approach [3], which embeds Word-sized Eye Tracking Visualizations [4].

References

- 1 T. Blascheck, M. John, K. Kurzhals, S. Koch, and T. Ertl. *VA²: A visual analytics approach for evaluating visual analytics applications*. *IEEE Transactions on Visualization and Computer Graphics*, 22(1): 61–70, 2016.
- 2 T. Blascheck, K. Kurzhals, M. Raschke, S. Strohmaier, D. Weiskopf, and T. Ertl. *AOI hierarchies for visual exploration of fixation sequences*. In *Proceedings of the Symposium on Eye Tracking Research & Applications*, pages 111–118. ACM, 2016.
- 3 T. Blascheck, F. Beck, S. Baltes, T. Ertl, and D. Weiskopf. *Visual analysis and coding of data-rich user behavior*. In *Proceedings of the IEEE Conference on Visual Analytics Science and Technology*, pages 141–150. IEEE Computer Society Press, 2016.
- 4 F. Beck, T. Blascheck, T. Ertl, and D. Weiskopf. *Word-sized eye tracking visualizations*. In *Eye Tracking and Visualization*, pages 113–128. Springer, 2017.
- 5 T. Blascheck, K. Kurzhals, M. Raschke, M. Burch, D. Weiskopf, and T. Ertl. *Visualization of eye tracking data: A taxonomy and survey*. *Computer Graphics Forum*, 2017.
- 6 T. Blascheck, M. Schweizer, F. Beck, and T. Ertl. *Visual comparison of eye movement patterns*. *Computer Graphics Forum*, 36(3): 87–97, 2017.

3.6 Methodological Considerations

Brian Dorn (University of Nebraska – Omaha, US) and Briana B. Morrison (University of Nebraska – Omaha, US)

License © Creative Commons BY 3.0 Unported license
© Brian Dorn and Briana B. Morrison

In this talk we review conceptual and practical elements of empirical study design. Multiple epistemological stances on evidence are introduced, and we frame the work of programming as activity to be understood holistically using the activity theory framework. We then discuss specific methods and tools from computer science education and the learning sciences that may be useful to the programming language design community when designing studies.

3.7 I Don't Understand Program Comprehension

Johannes C. Hofmeister

License © Creative Commons BY 3.0 Unported license
© Johannes C. Hofmeister

How do we understand programs? When we ask this question, it is worthwhile to think about what comprehension is and how we can make it measurable. If we don't, we might end up asking the wrong questions. For example, several studies had asked how syntax highlighting affects comprehension. But why should recall questions be answered differently depending on the color of the words? I proposed that highlighting does not affect memory and reasoning, and instead affects perceptual processes. This idea was demonstrated visually. The talk showed how psychologists reason about such ideas, by introducing explaining variables, named constructs.

3.8 What evidence do we have about programming language design?

Antti-Juhani Kaijanaho (University of Jyväskylä, FI) and Andrew J. Ko (University of Washington – Seattle, US)

License © Creative Commons BY 3.0 Unported license
© Antti-Juhani Kaijanaho and Andrew J. Ko

In this talk we present an informal review of the body of evidence about programming language design. The talk discusses the results of two mapping studies of empirical studies about programming languages, and a literature review of evidence about the effect of programming languages on productivity and errors. The talk concludes that we don't know very much (there are only about 100 studies evaluating language features), that the number of studies is accelerating, but very slowly, and that most studies focus on evaluating novel language features, rather than systematically comparing language features in broad use. The researchers doing these studies tend to be trained in labs with expertise on running human subjects experiments.

3.9 Scientists Programming

Amelia A. McNamara (Smith College – Northampton, US)

License © Creative Commons BY 3.0 Unported license
© Amelia A. McNamara

Main reference Julia S. Stewart Lowndes, Benjamin D. Best, Courtney Scarborough, Jamie C. Afflerbach, Melanie R. Frazier, Casey C. O'Hara, Ning Jiang, and Benjamin S. Halpern: "Our path to better science in less time using open data science tools." *Nature Ecology & Evolution*, 1, 2017

URL <https://doi.org/10.1038/s41559-017-0160>

Scientists are an important important sub-category of programmers, because their work has implications in government policy, medicine and general knowledge. The three main categories of computer use by scientists are "make models, generate data" "generate data, make models," and general application programming. We will not consider the last category in this talk, as it is the closest to traditional software engineering. Common tools for making models to generate data are Matlab, Sage, Maple, and Mathematica. Scientists who generate data (by field collection, experimentation, etc) and make models, often use Excel, Stata, SPSS, SAS, R or Python. The transition from non-reproducible research to more robust analysis can be challenging, but the benefits are broad. We consider as a case study a marine science group who wrote "Our path to better science in less time using open data science tools" about their movement from Excel to a toolkit including git and GitHub for collaboration and version control, R in the IDE RStudio for analysis, and RMarkdown for reproducibility and narrative.

3.10 Types of Studies

Brad A. Myers (Carnegie Mellon University – Pittsburgh, US), Jonathan Aldrich (Carnegie Mellon University – Pittsburgh, US), Michael Coblenz (Carnegie Mellon University – Pittsburgh, US), and Joshua Sunshine (Carnegie Mellon University – Pittsburgh, US)

License © Creative Commons BY 3.0 Unported license
© Brad A. Myers, Jonathan Aldrich, Michael Coblenz, and Joshua Sunshine

Main reference Brad A. Myers, Andrew J. Ko, Thomas D. LaToza, YoungSeok Yoon: "Programmers Are Users Too: Human-Centered Methods for Improving Programming Tools", *IEEE Computer*, Vol. 49(7), pp. 44–52, 2016.

URL <http://dx.doi.org/10.1109/MC.2016.200>

Traditional programming language design approaches center around theoretical and performance-oriented evaluation. Recently, researchers have been considering more approaches to language design, including the use of quantitative and qualitative user studies, to evaluate how different designs affect users. In this talk, we define "design" to encompass both the activities before and during creation of a system, including needs finding, and the activities involved in the evaluation after the design has been created. We list desiderata of programming languages, including software engineering quality attributes such as correctness, performance of the resulting code, expressiveness, speed of compiling, understandability, ease of reasoning, modifiability and learnability. We identify the different kinds of people or roles involved, such as logician, industrialist, empiricist and teacher. Then, we list many different methods that can be used to help with the design, most of which we have used in our group. These include: contextual inquiry, interviews, surveys, corpus studies, natural programming, rapid prototyping, programming language and software engineering theory, qualitative usability studies, case studies, expert evaluation, performance evaluation, user experiments, and

formalism and proof. We argue for taking an interdisciplinary approach using mixed methods to answer questions. Finally, we argue against having unsubstantiated claims, using methods incorrectly, assuming the conventional wisdom, and inadequate reporting of results.

3.11 Programming language cultures are relevant

Lutz Prechelt (*FU Berlin, DE*)

License © Creative Commons BY 3.0 Unported license
© Lutz Prechelt

Main reference Lutz Prechelt: “Plat_Forms: A Web Development Platform Comparison by an Exploratory Experiment Searching for Emergent Platform Properties”, IEEE Trans. Software Eng., Vol. 37(1), pp. 95–108, 2011.

URL <http://dx.doi.org/10.1109/TSE.2010.22>

We will contrast glimpses of possible cultures (found in self-presentations) of different programming languages: Haskell vs. C++ vs. Go; PHP vs. Ruby; Perl vs. Python. These suggest that distinct cultures may exist for at least many languages. Then we discuss two pieces of actual evidence that such cultures have relevant impact: First, in one experiment the Java participants all chose either a simple, very inefficient or a super-efficient, but very cumbersome solution, while the Perl and Python participants chose a third one, based on HashMaps, that is simple *and* efficient. Second, in another experiment 4 of 6 Perl and PHP teams chose the most straightforward and by far most maintenance-friendly approach, while the Java teams picked a conventional one involving property files, although that is cumbersome and its only advantage (internationalization) was explicitly not required.

References

- 1 Lutz Prechelt. *An empirical comparison of seven programming languages*. IEEE Computer 33(10), October 2000.
- 2 Lutz Prechelt. *An empirical comparison of C, C++, Java, Perl, Python, Rexx, and Tcl for a search/string-processing program*. Technical Report 2000-5, 34 pages, Fakultät für Informatik, Universität Karlsruhe, Germany, March 2000.
- 3 Lutz Prechelt. *Plat_Forms: A Web Development Platform Comparison by an Exploratory Experiment Searching for Emergent Platform properties*. IEEE Trans. on Software Engineering 37(1):95-108, Jan-Feb 2011.
- 4 Lutz Prechelt. *Plat_Forms 2007: The Web Development Platform Comparison – Evaluation and Results*. Technical Report TR-B-07-10, 118 pages, Freie Universität Berlin, Institut für Informatik, June 2007.

3.12 Eye Tracking in Program Comprehension

Bonita Sharif (*Youngstown State University, US*)

License © Creative Commons BY 3.0 Unported license
© Bonita Sharif

Main reference Timothy R. Shaffer, Jenna L. Wise, Braden M. Walters, Sebastian C. Müller, Michael Falcone, Bonita Sharif: “iTrace: enabling eye tracking on software artifacts within the IDE to support software engineering tasks”, in Proc. of the 2015 10th Joint Meeting on Foundations of Software Engineering, ESEC/FSE 2015, Bergamo, Italy, August 30 – September 4, 2015, pp. 954–957, ACM, 2015.

URL <http://dx.doi.org/10.1145/2786805.2803188>

Eye tracking has been used since the 1800s for various tasks in psychological research. An eye tracker tells us what a person is looking at while they are doing a task. This information is invaluable for researchers and educators. Educators can see the thought processes that

go into reading and comprehending code. This can help them devise better intervention strategies. Researchers can see how developers work while fixing a bug or adding a feature and use this information to devise better methods and tools for them. In this talk, I will describe eye tracking technology, introduce three eye tracking studies related to program comprehension and describe how we can conduct empirical studies at scale using iTrace, a tool that implicitly embeds eye tracking into the developer work environment. In conclusion, a discussion about how these results could help PL designers and educators is presented.

References

- 1 Bonita Sharif, Timothy Shaffer, Jenna L. Wise and Jonathan I. Maletic. *Tracking Developers' Eyes in the IDE*. IEEE Software, 33:3, 2016
- 2 Bonita Sharif and Jonathan Maletic. *iTrace: Overcoming the Limitations of Short Code Examples in Eye Tracking Experiments*. IEEE International Conference on Software Maintenance and Evolution, ICSME 2016, Raleigh, NC, USA, October 2 – 7, 2016
- 3 Katja Kevic, Braden M. Walters, Timothy R. Shaffer, Bonita Sharif, David C. Shepherd and Thomas Fritz. *Tracing software developers' eyes and interactions for change tasks*, ESEC/FSE, Aug 30 – Sept 4, 2015
- 4 Teresa Busjahn, Roman Bednarik, Andrew Begel, Martha E. Crosby, James H. Paterson, Carsten Schulte, Bonita Sharif and Sascha Tamm. *Eye movements in code reading: relaxing the linear order*, IEEE International Conference on Program Comprehension, ICPC, Florence, Italy, May 16 – 24, 2015
- 5 Bonita Sharif and Jonathan Maletic. *An Eye Tracking Study on camelCase and under_score Identifier Styles..* IEEE International Conference on Program Comprehension, ICPC, Braga, Portugal, USA, Jun 30 – Jul 2, 2010

3.13 Early Experimental Studies of Conditionals in Programming Languages

Walter F. Tichy (KIT – Karlsruher Institut für Technologie, DE)

License  Creative Commons BY 3.0 Unported license
© Walter F. Tichy

Eight controlled experiments about conditionals in programming languages are surveyed. The experiments were identified in a mapping study by Kaijanaho as the only controlled studies on conditionals between 1973 and 1993. They evaluated the goto statement, arithmetic if, nested if-then-else, scoping keywords such as **begin** and **end**, decision tables, **if-then** rule sets, and indentation. The first paper, Sime et al's study of 1973, compared goto statements with nested if-then-else without scoping constructs. The experiment used a specially designed micro language that isolated the language feature under study, a technique that was taken up by later experiments. A 1977 study by the same authors added scoping and indentation, while a 1984 paper by Vessey et al used decision tables rather than text to specify the statements to be written. A 1978 paper by Embley advocated using empirical studies in conjunction with considerations of the complexity of the associated proof rules when designing new control constructs. Two studies compared rule sets with **if-then-else** and **goto**. Even the effect of the true/false direction of Boolean expressions in conditionals was tested.

From the standpoint of experimental methodology, these studies were well constructed, but they appear to have had little impact on programming language design. For example, the nested if-then-else statement was already present in the Algol programming language in 1960, and Dijkstra's famous letter "The Go To Statement Considered Harmful" (which called for the abolishment of the goto statement) appeared in 1968, years before Sime's controlled

trial of 1973. The surveyed papers pay scant attention to the intense debate about structured programming at the time, which started with Dijkstra's 1968 letter. Knuth published a thorough study of control constructs, including the goto and its alternatives, in 1974. By 1978, the C language provided an adequate set of control constructs, including `if-then-else`, curly braces instead of unwieldy keywords, short-circuit evaluation of Boolean expressions, as well as `switch`, `continue`, `break`, and `return` statements, which are constrained forms of `goto`. This set has stood the test of time and is still in use in more recent languages such as Java. Apparently, language designers, by personal programming experience, introspection, or intuition, found adequate forms of conditionals without recourse to controlled trials. However, not all questions about programming languages can be settled that way. For example, the questions of whether to prefer dynamic over static type-checking, or functional over imperative programming, need evidence to be answered reliably.

References

- 1 Edsger W. Dijkstra, Letters to the Editor: Go To Statement Considered Harmful, *Communications of the ACM*, 11(3), March 1968, 147-148. DOI: 10.1145/362929.362947
- 2 David W. Embley, Empirical and formal language design applied to a unified control construct for interactive computing, *Int. J. Man-Machine Studies* (1978) 10 (2), 197-216. DOI: 10.1016/S0020-7373(78)80012-1
- 3 D. J. Gilmore and T. R. G. Green, Comprehension and recall of miniature programs, *Int. J. Man-Machine Studies* (1984), 21 (1), 31-48. DOI: 10.1016/S0020-7373(84)80037-1
- 4 R. Halverson, An Empirical Investigation Comparing IF-THEN Rules and Decision Tables for Programming Rule-Based Expert Systems, In *System Sciences, 1993, Proceedings of the Twenty-Sixth Hawaii International Conference on*. Vol. iii, 316–323 vol.3. DOI: 10.1109/HICSS.1993.284327
- 5 E. R. Iselin, Conditional statements, looping constructs, and program comprehension: an experimental study. *Int. J. Man-Machine Studies* (1988) 28 (1), 45–66. DOI: 10.1016/S0020-7373(88)80052-X
- 6 A.-J. Kaijanaho, Evidence-Based Programming Language Design, A Philosophical and Methodological Exploration, Dissertation, Faculty of Information Technology of the University of Jyväskylä, Finland, 2015
- 7 B. W. Kernighan and D. M. Ritchie, *The C Programming Language*, Prentice-Hall, 1978.
- 8 D. E. Knuth, Structured Programming with go to Statements, *ACM Computing Surveys*, 6(4), 1974, 261-301. DOI: 10.1145/356635.356640
- 9 P. Naur (ed.), Report on the Algorithmic Language Algol 60, *Comm. of the ACM*, 3(5), 1960, 299-314. DOI: 10.1145/367236.367262
- 10 B. Shneiderman, Exploratory experiments in programmer behavior. *Int. J. Computer and Information Sciences* (1976) 5 (2), 123–143. DOI:10.1007/BF00975629
- 11 B. Shneiderman, and R. Mayer, R. Syntactic/semantic interactions in programmer behavior: A model and experimental results. *Int. J. Parallel Programming* (1979) 8 (3), 219–238. DOI: 10.1007/BF00977789
- 12 M. E. Sime, T. R. G. Green, and D. J. Guest, Psychological evaluation of two conditional constructions used in computer languages. *Int. J. Man-Machine Studies* 5 (1), (1973) 105–113. DOI: 10.1016/S0020-7373(73) 8001-2
- 13 M. E. Sime, T. R. G. Green, and D. J. Guest. Scope marking in computer conditionals – a psychological evaluation. *Int. J. Man-Machine Studies* 9 (1), (1977) 107–118. DOI: 10.1016/S0020-7373(77)80045-X
- 14 I. Vessey and R. Weber, Conditional statements and program coding: an experimental evaluation. *Int. J. of Man-Machine Studies* 21 (2), (1984) 161–190. DOI: 10.1016/S0020-7373(84)80065-6

4 Future Research Questions and Studies

The seminar included a brainstorming session where the goal was to enumerate research questions. The session was free-form and questions that were mentioned or written down by participants are listed here. To be clear on the intent here, our goal with this section was to document many of the research questions mentioned by participants.

We categorized the questions into broader themes after the event. We did this for organization in this document and to make it easier to see the big picture of what participants were interested in, but participants themselves did not have these categories during the session. Finally, while we did conduct some light editing, and removed a few questions that did not feel fleshed out enough to include, we did not substantively edit the original questions.

4.1 Impacts in the field

These questions appeared related to either how programming languages evolved, or could evolve, in regard to their impact in practice. We imagine many other questions on the topic could be asked.

4.1.1 Standardization

Research Question What aspects of programming languages have been largely standardized?

Description Over the last 50 years, while there has been some disagreement across programming languages, it is clear that some features have been more successful than others. Which parts have been standardized across multiple languages, and why?

4.1.2 Library design

Research Question Different software developers, given a library or API design problem, make different design choices. What can we learn about programming language and API design from the diversity of choices made by experts?

4.1.3 Polyglot programming

Research Question What is the impact of polyglot programming? Are there economic costs of the current polyglot state?

Description Often programmers need to embed one language in another (e.g. embedded SQL, or JavaScript inside HTML inside Ruby). What impact does this have?

4.1.4 Methods of language evolution

Research Question Characterize the in-use methods of programming language evolution among designers and maintainers of existing languages. To what extent are these consistent or inconsistent? How do these relate to the costs and benefits of programmers adopting these changes and using the new language features? Can we predict the costs/benefits of language changes?

Maybe eventually a given language is “done” and the cost of change outweighs any benefit. What is the effect of language feature creep on development?

4.2 Methodology

There was interest amongst participants in the methodological procedures under which we can test languages. This might include trying to garner a better understanding who we are, or perhaps who we should test in such studies, in addition to asking questions about the kinds of metrics we might use for evaluation.

4.2.1 Representativeness of participants

Research Question How representative are the students or subjects we use now? Which groups of people are we using now? We could compare new groups to the current group. This was an issue in psychology as well [1] and in medicine [2]. This also includes the diversity of the stakeholders.

References

- 1 http://www.slate.com/articles/health_and_science/science/2013/05/weird_psychology_social_science_researchers_rely_too_much_on_western_college.html
- 2 <http://www.theguardian.com/lifeandstyle/2015/apr/30/fda-clinical-trials-gender-gap-epa-nih-institute-of-medicine-cardiovascular-disease>

4.2.2 Measuring cognitive load

Research Question What empirical methods can be used to unobtrusively measure cognitive load? The goal would be measure and propose interventions that improve productivity.

Description We don't yet know how to measure cognitive load in programmers, but we suspect that programmers' ability to be productive decreases when cognitive load is too high. We should identify and standardize methods that can be used to (minimally-invasively) measure cognitive load. Then, we should show that cognitive load indeed interferes with programmer effectiveness. Finally, we should create effective interventions that help decrease cognitive load (or help people be more effective when it is high. Eye tracking methodology lends itself well for this type of analysis.).

4.3 Language features

Many programming languages have features in common. While some have unique philosophies or semantics, many have commonalities like loops, conditionals, functions, classes, or other features. Even if the underlying semantics are the same, or very similar, language designers often use alternative notations or representations for them. Other times, designers vary the semantics in subtle ways. A variety of questions were asked regarding individual language features or aspects of their paradigms.

4.3.1 Functional vs. procedural languages

Research Question How are people **reasoning differently in functional vs procedural languages**? Where are they spending their time? Can we characterize how they are working? How does this look different from the existing literature on procedural languages?

Study Type Exploratory Qualitative Work

Description How do we characterize how functional programmers work? We could study an expert functional programmer by giving them a design problem and asking them to solve it and implement a corresponding program. Asking participants to work in pairs could

enable capturing their thoughts better because they would need to communicate with each other. For comparison purposes, the study might have five functional teams, five object-oriented teams, and five procedural teams.

The design could include multiple problems that seem more biased towards one type of language or the other, enabling analysis regarding the effect of the language paradigm. Further research questions include:

- How can we predict the task/language fit for a given task and language pair?
- What does it mean to pick the right language for the task? Particularly, if a language supports more than one paradigm, how do developers choose a specific paradigm?

References

- 1 Amjad Altadmri and Neil C.C. Brown. 2015. 37 Million Compilations: Investigating Novice Programming Mistakes in Large-Scale Student Data. In Proceedings of the 46th ACM Technical Symposium on Computer Science Education (SIGCSE '15). ACM, New York, NY, USA, 522-527. DOI: <http://dx.doi.org/10.1145/2676723.2677258>

4.3.2 Concurrency and parallelism

Research Question What are appropriate language representations for concurrency and/or parallelism?

Description How do people think about these kinds of programs? Perhaps understanding human cognition would help develop a more effective way of writing concurrent and/or parallel code.

4.3.3 Creativity vs. Utility

Research Question What are the best language features to support creativity and how do they differ from those that provide other kinds of benefits?

4.3.4 Making the average programmer much better

Research Question Perhaps current languages are still too low-level. How could we make the average programmer 10x better? Some ideas for language approaches include:

- Search-based
- Machine learning
- Programming by example and imitation
- Programming by question-answer
- Natural language

In databases, users moved from databases to analysis of big data. How would that look like for programming? We should not only think about languages, but also about tools/IDEs and processes.

4.4 Novices and learning

Programming languages are used at all levels, from Kindergarten in many U.S. schools to seasoned professionals working in the field. While the spectrum is broad, some participants were interested in learnability or ease of use by those earlier in a lifetime. These questions thus focused around issues they thought might be relevant.

4.4.1 Learning performance of CS concepts

Research Question How does language design affect performance when students are studying various topics in computer science?

4.4.2 Learnability

Research Question What factors affect learnability of individual language features?

4.4.3 Attributes of authenticity

Authenticity typically concerns the extent to which users of educational programming tools (such as block-based editors) feel that they are learning “authentic” programming (as opposed to learning a different set of skills that do not count as “real” programming). However, there is an argument that authenticity applies to professionals as well.

Research Question What are the principal attributes of authenticity in programming languages?

Description For example, how do we prevent authenticity questions from stifling innovation?

4.4.4 Accents: implications of training on cross-language use

Research Question What is the impact of one’s first programming language on subsequent programming behavior? What is the impact on language tools on programmer behavior across language environments? For example, at code.org, users are given the blocks in advance, so users do not practice the skill of finding a block. There was a position paper about this at VL/HCC 2015: <http://www.felienne.com/archives/4352>

4.4.5 Error handling and learning

Research Question Does graceful error handling improve the experience of learning programming for novices?

4.4.6 Novice vs. Expert compiler usage

Research Question How do published novice programmer compilation behaviors compare to that of industry developers?

Description Replicate a set of Blackbox [1] studies against Google developer logs and quantitatively compare distribution of error rates and other statistics. This serves as a bridging question between academia and industry to move the field forward. What can we learn from novices to help experts? What can we learn from experts to help novices?

4.4.7 Error messages

Research Question How do users read error messages and how is it affected by different error message content or display choices?

4.4.8 Error-proneness

Research Question What is error-proneness?

Description What leads to incorrect thinking / logic errors? Come up with a formal definition that would allow this to be checked. All PL designers want to have low values for error-proneness. Ideally come up with something that could be objectively measured.

4.5 Cognition and affect

A variety of questions were asked that were related to users thinking or feeling about programming. Such attributes could inform us about reasons for adoption or other factors of interest.

4.5.1 Thoughts about program behavior

Research Question How do people think about what a program does?

Description Try to evaluate program understanding. Is there an analogy to “code-switching” from linguistics? If so, is code switching good or bad? What models should we have of programming and how do we apply them to writing code? How do models change during programming? What’s the relationship to cognitive load and working memory?

4.5.2 Language Love

Research Question What makes people love a language?

Description Evaluate what factors influence people’s feelings about languages. Is loving programming not at all related to loving a programming language? Maybe culture and the ecosystem matter too.

4.5.3 Programmer descriptions of structures

Research Question In what ways do programmers describe the structure of their systems? And how do such descriptions help with programming tasks?

Description *Study 1:* Interview developers to see how they describe system structures. Ask them to describe the system in several different ways; how do those descriptions differ, and what order are they described in? Watch them drawing diagrams and talking through them in a think-aloud study.

Study 2: How do these descriptions prime a programmer for completing tasks? Prompt half of the participants to describe a system in one way prior to completing a task. The other half of the participants would just get the task without having to describe the system first. Measure productivity and interaction (possibly using eye tracking).

4.5.4 Mental spatial structures

Research Question Learn what mental spatial structures people develop regarding code (and what affects this). For example, how do people learn where (in a given file) particular pieces of code are? How does this differ between people? Is this different for people with various disabilities?

5 Working groups

The participants divided into working groups to focus on some of the research questions above that attracted the most interest and had potential for significant contributions.

5.1 Novices: polyglot questions

Johannes Bechberger (Karlsruher Institut für Technologie, DE), Scott Fleming (University of Memphis – Memphis, US), Briana B. Morrison (University of Nebraska – Omaha, US), Bonita Sharif (Youngstown State University – Youngstown, US), Andreas Stefik (University of Nevada, Las Vegas – Las Vegas, US)

This working group focused on questions on *polyglot* programming, which regards working in multiple programming languages at once. What happens when a programmer learns multiple languages? To what extent does knowledge transfer between languages? Is there an impact on productivity for domains, like the web, where programmers are forced to use multiple languages at once?

Study participants could be evaluated at several times for a longitudinal study: for example, after two semesters, after the end of the semester in which they learn their second language, and at graduation. At each time, they might be given code for various tasks in multiple languages and ask them to explain the code, even though they haven't necessarily seen those languages before. They could also be asked to trace the execution with various inputs. Eye tracking might provide insight into the participants' approach to the problem and help identify whether the participants recognize beacons [1], which help programmers understand the structure and behavior of programs. It might be helpful to do the study on pairs of participants rather than individual participants so that experimenters can observe the conversation. Regarding language learning, the group also discussed the question of whether it is better to learn one language deeply or learn multiple languages at once.

References

- 1 Susan Wiedenbeck. Beacons in computer program comprehension. *International Journal of Man-Machine Studies* 25.6 (1986): 697-709.

5.2 Context Switching and Cognitive Load

Ameer Armaly (University of Notre Dame – Notre Dame, US), Andrew Begel (Microsoft Research – Redmond, US), Igor Crk (Southern Illinois University – Edwardsville, US), Tanja Blascheck (INRIA Saclay – Orsay, FR), John Daughtry (Google Inc. – Seattle, US), Rob DeLine (Microsoft Corporation – Redmond, US), Ciera Jaspan (Google Inc. – Mountain View, US), Philip Merlin Uesbeck (University of Nevada, Las Vegas – Las Vegas, US)

This working group focused on enumerating interesting research questions. Part of the conversation overlapped with the *Novices* group, with a shared interest in polyglot programming. The idea is that a polyglot programming scenario might lead to higher cognitive load, as programmers must remember information pertaining to multiple (potentially inconsistent) programming methodologies. Language designs that facilitate polyglot programming might involve addressing language inconsistencies, such as in loop constructs; missing constructs (one could compare languages in which one has a construct that the other lacks); and embeddings, in which programmers could embed code in one language inside code written in another language.

The group discussed the need for better methodologies to measure cognitive load; to be usable for a programming study, the methods must be non-destructive and non-interfering. Low cost is also important for practical reasons. The group was interested in how programming language designs and program design decisions correlate with cognitive load. Do particular

programming language constructs correlate with cognitive load? The group hypothesized that the number of types in use correlates with cognitive load, and that cognitive load might correlate with typos or stutter. The current results regarding cognitive load seem to try to categorize users as being overloaded or not rather than trying to assess cognitive load in a quantitative way. It might be interesting to consider the units of cognitive load so that it can be correlated more directly.

5.3 Language features and error-proneness

Jonathan Aldrich (Carnegie Mellon University – Pittsburgh, US), Michael Coblenz (Carnegie Mellon University – Pittsburgh, US), Andrew J. Ko (University of Washington – Seattle, US), Thomas LaToza (George Mason University – Fairfax, US), Sibylle Schupp (TU Hamburg-Harburg, DE), Walter Tichy (KIT – Karlsruher Institut für Technologie, DE)

Programming language designers tend to think a lot about user errors (bugs) because a lot of the techniques used in language design are intended to exclude particular classes of bugs. As a result, the working group on *language features* focused on creating a *grand theory of error-proneness of language features*, specifically regarding errors that emerge through reasoning, not perceptual slips or conflicts with prior knowledge.

Language features are abstractions. As such, they hide certain aspects of execution for the benefit of simpler reasoning. For example, consider integer division, which hides remainders, integer truncation, and division by zero. If, in hiding that complexity, the language feature allows the developer to never have to reason about that internal complexity, error-proneness is low. However, if the language feature still occasionally forces the developer to have to reason about that internal complexity, that reasoning will be even harder because the complexity is hidden, and therefore error-proneness may be higher. For example, $10/3$ requires a programmer to reason about integers versus floating-point numbers. $10/0$ requires programmers to reason about runtime errors.

The group hypothesized that the mechanism of error production is that developers need to be able to reason correctly about the behavior of an abstraction. If they do not have access (e.g. via training) to a correct model of that behavior, they will make mistakes. Abstractions occasionally hide behavior developers must reason about. Furthermore, error-proneness may arise by composition of language features: language features may interact with each other in ways that are error-prone.

5.3.1 Examples

- The division operator in most languages fully encapsulates the complexities of division, but not in the case of 0 or floating-point. Those nuances are not visible in the `/` operator in most languages, reducing the visibility of the need to handle those cases, increasing the likelihood of errors.
- Constraints make it easier to express declarative properties between values, but one must understand the hidden semantics of constraint satisfaction algorithms to avoid unintended side effects of cycles.
- Memory-safe languages allow programmers to think in terms of abstract objects and fields instead of the linear memory on which those objects are imposed. This abstraction eliminates many non-local interactions (or alternatively, “safety rules”) that programmers have to consider in unsafe languages. The abstraction rarely breaks down in terms of

correctness; the main cases where it does are interaction with code written in unsafe languages (e.g. the native code interface in Java). It does break down in cases where an application's performance or memory-usage needs are strict enough to be affected by the costs of the garbage collection and dynamic checks necessary for memory safety.

- Monads (are these error-prone or hard to use? Are those different?)

5.3.2 Operationalization

Error-proneness is defined as semantics of an abstraction that are hidden but a developer must reason about correctly in order to be used correctly. Error-proneness may have a trade off with expressiveness because increased expressiveness requires abstraction, which hides complexity. If one must reason about that abstraction, an error might occur if that abstraction is hard to understand or use. This is related to the idea of leaky abstractions, but that work focuses more on architectural consequences and not language features or not error-proneness.

The theory might be used as a thinking tool in a language design process, and as a way to generate hypotheses to test. Doing so might validate the theory: if it effectively predicts parts of designs that are error-prone, it is a useful theory. APIs might also be analyzable in the same framework. The group also considered perceptual sources of errors from syntax; these are more akin to *slips* than to *mistakes*, depending on whether one considers these terms to be substantively different, but they are errors nonetheless.

5.3.3 Evaluation

Researchers might work to test the theory by building a predictive error model and showing that it makes useful predictions for particular language features. Future work should figure out criteria for thresholds: how much complexity is enough to cause error-proneness? What does this depend on: experience, etc...? In the future, researchers should compare fine-grained differences in what an abstraction hides and compare developers' defect production.

5.4 Why do people love programming languages?

Igor Crk (Southern Illinois Univ. – Edwardsville, US), Andrew Duchowski (Clemson University, USA), Matthias Hauswirth (University of Lugano, Switzerland), Brad A. Myers (Carnegie Mellon University – Pittsburgh, US), Craig Anslow (Victoria University of Wellington), Brian Dorn (University of Nebraska, Omaha), Lutz Prechelt (FU Berlin, Germany), Antti-Juhani Kaijanaho (University of Jyväskylä, Finland)

This working group focused on understanding *affect*: what is it about languages that causes people to love or hate them? Is it primarily due to technical attributes (e.g. features) or is personal relevance more important? For example, people might love the first language they used, or their feelings might be driven by their prior positive (or negative) experiences using languages on particular projects.

The group designed an interview study in which participants would be asked:

- Is there a language you love, and why?
- Tell me about a time when a language feature helped you achieve a goal.
- Tell me about a time when a language feature stood in the way of a goal.
- Tell me about a time when you abandoned a language in favor of another one.

In the discussion, the participants noted that it is easy to conflate love with adoption. Sometimes people don't have choices because their language selection is dictated by their employer or other practical considerations. Any study of this would need to separate the questions of adoption from the questions of emotional attachment, and separate the influences of the various factors. The study might leverage standard business or psychological measures of affect.

The group was interested in what qualities of people cause them to be drawn to particular languages. One might ask: "How would one characterize people who love language *X*?" The group was also interested in the progression of feelings over time as people learn more languages (for example, at a university). By asking senior programmers, teachers, and students at various stages, one could study how language learning affects feelings about languages. It might also be interesting to compare adoption and use to what people love. Other study techniques might involve asking programmers who know at least two languages well for comparisons and explanations of what languages they do not like. It is important to include social aspects in this discussion; perhaps people like or dislike languages in part because of the people in the language communities. Perhaps some people love programming languages in general rather than loving specific languages. Another question the group identified pertained to global adoption: is there a map of which languages are in use in each part of the world? If not, is it worth creating?

5.5 Representativeness of subjects in studies

Felienne Hermans (TU Delft, NL), Johannes Hofmeister (Universität Passau, DE), Amelia A. McNamara (Smith College – Northampton, US), Lea Verou (MIT – Cambridge, US)

This working group discussed a practical problem that the community has encountered when running studies: most studies so far have been on university students at high-quality universities who have studied some programming. This biases the sample toward white, male, able-bodied, high socio-economic status students. The group identified three mechanisms to mitigate the impact of this problem:

1. Studies should be reporting gender, SES, ethnicity. However, the European Union has restrictions on what data can be collected, which may make tracking ethnicity difficult for researchers in the EU.
2. Run studies on other groups to assess impact on alternative demographics. Medicine has had this problem too: men were used largely in the 1940s because of the connections to the military in early controlled experiments, for example.
3. Make sure we understand the results from psychology and HCI in this area [1].

References

- 1 Ari Schlesinger, Keith Edwards, Rebecca E. Grinter. (2017). Intersectional HCI: Engaging Identity through Gender, Race, and Class. 5412-5427.
<https://doi.org/10.1145/3025453.3025766>

6 Open problems

6.1 Evidence standards

The participants had a plenary discussion about the need for an evidence standard in human factors research in computer science. There seemed to be general consensus that an evidence standard would clarify to authors and reviewers how to report on their research to improve replication and to clarify the limitations of particular studies. The group discussed the details of the CONSORT evidence standard [1]. Some participants expressed concern at having to comply with a standard, but the strong majority seemed to find the idea reasonable. The group also had broad consensus that the types of information that the CONSORT standard requires in its reporting are a reasonable place to start when proposing an evidence standard.

6.1.1 Adoption of evidence standards

Some ideas that were proposed to assist **adoption** include:

- Send a draft of an evidence standard for publication. Where to send such work was unclear.
- Two approaches were considered for adoption of such a standard. One way was to use a top-down approach where we attempt to convince a publication venue to require its use for some papers. Another way would be to propagate the standard bottom-up by submitting papers that follow it, citing in each work in a consistent way. Both approaches have pros and cons.
- It is important to start with the right community. The IEEE VL/HCC conference (<http://www.vlhcc.org>) might be a good place to start because lots of the VL/HCC leadership are among the meeting participants. As such, perhaps a recommendation could be attached to the VL/HCC call for papers. In comparison, CHI has an informal evidence standard based on psychology papers. Alternatively, the computing education conferences and journals already use empirical data heavily, perhaps especially TOCE and ICER, and as such might be good targets.
- An evidence-based track at a conference might be a lightweight way to further adoption. Perhaps eventually the whole conference would be in this track.
- Make reviewers feel like they are the “last ones to know” about the evidence standard so that they feel like they should adopt the new standards of the “in” group.
- Change is not an instant process and working toward a standard is a marathon, not a sprint.
- The Open Science Framework is an existing mechanism to manage scientific workflows, including study registration. Study registration is a well-accepted technique in science to reduce a number of potential problems during the publication process (e.g., Type I error, fraud, “surprise factor”). Perhaps leveraging existing mechanisms could reduce the cost and difficulty of adoption.
- While CONSORT, the What Works Clearinghouse, APA, and other standards exist, it was unclear what evidence standard would work best for computer science. Given other disciplines already have such standards, we may be able to learn from them in designing ours, while potentially improving them to avoid historical mistakes.
- If an evidence standard were to be adopted at particular conferences or journals, or across conferences and journals, training will be needed to help scholars use and understand it, both at the reviewing and at the writing stages.

6.1.2 Criteria

Participants discussed some criteria for good evidence standards:

- Evidence standards should be clear so that authors know what the expectations are, but the standard should not unnecessarily make papers longer since page limits are of concern.
- Care should be taken to avoid unnecessarily rejecting papers to which the standards don't align well.
- The standard should include guidelines **both** for authors and reviewers.
- It needs to be possible to cite the evidence guidelines. Also it would be nice to be able to cite study guidelines.
- It should be clear that the evidence standard does not apply to all kinds of research. For example, there was a backlash in the UIST community when it became expected that authors include an A/B study of their work because some authors felt that this was inappropriate for their work.

References

- 1 Schulz, K.F., Altman, D.G., Moher, D., for the CONSORT Group. CONSORT 2010 Statement: updated guidelines for reporting parallel group randomised trials. *Annals of Internal Medicine*. 2010;152. Epub 24 March. <http://annals.org/article.aspx?articleid=745807>

Participants

- Jonathan Aldrich
Carnegie Mellon University –
Pittsburgh, US
- Craig Anslow
Victoria University –
Wellington, NZ
- Ameer Armaly
University of Notre Dame, US
- Johannes Bechberger
KIT – Karlsruher Institut für
Technologie, DE
- Brett A. Becker
University College Dublin, IE
- Andrew Begel
Microsoft Research –
Redmond, US
- Tanja Blascheck
INRIA Saclay – Orsay, FR
- Neil C. C. Brown
King's College London, GB
- Michael Coblenz
Carnegie Mellon University –
Pittsburgh, US
- Igor Crk
Southern Illinois Univ. –
Edwardsville, US
- John M. Daughtry
Google – Seattle, US
- Fabian Deitelhoff
Fachhochschule Dortmund, DE
- Rob DeLine
Microsoft Corporation –
Redmond, US
- Brian Dorn
University of Nebraska –
Omaha, US
- Andrew Duchowski
Clemson University, US
- Scott Fleming
University of Memphis, US
- Baker Franke
Code.org – Seattle, US
- Reiner Hähnle
TU Darmstadt, DE
- Matthias Hauswirth
University of Lugano, CH
- Felienne Hermans
TU Delft, NL
- Johannes Hofmeister
Universität Passau, DE
- Ciera Jaspán
Google Inc. –
Mountain View, US
- Antti-Juhani Kaijanaho
University of Jyväskylä, FI
- Andrew J. Ko
University of Washington –
Seattle, US
- Thomas LaToza
George Mason University –
Fairfax, US
- Andrew Macvean
Google – Seattle, US
- Jonathan I. Maletic
Kent State University, US
- Amelia A. McNamara
Smith College –
Northampton, US
- Briana B. Morrison
University of Nebraska –
Omaha, US
- Brad A. Myers
Carnegie Mellon University –
Pittsburgh, US
- Lutz Prechelt
FU Berlin, DE
- Sibylle Schupp
TU Hamburg-Harburg, DE
- Bonita Sharif
Youngstown State University, US
- Andreas Stefik
Univ. of Nevada – Las Vegas, US
- Walter F. Tichy
KIT – Karlsruher Institut für
Technologie, DE
- Phillip Merlin Uesbeck
Univ. of Nevada – Las Vegas, US
- Lea Verou
MIT – Cambridge, US



Planning and Operations Research

Edited by

J. Christopher Beck¹, Daniele Magazzeni², Gabriele Röger³, and Willem-Jan Van Hoeve⁴

¹ University of Toronto, CA, jcb@mie.utoronto.ca

² King's College London, GB, daniele.magazzeni@kcl.ac.uk

³ Universität Basel, CH, gabriele.roeger@unibas.ch

⁴ Carnegie Mellon University – Pittsburgh, US, vanhoeve@andrew.cmu.edu

Abstract

This report documents the program and the outcomes of Dagstuhl Seminar 18071 “Planning and Operations Research”. The seminar brought together researchers in the areas of Artificial Intelligence (AI) Planning, Constraint Programming, and Operations Research. All three areas have in common that they deal with complex systems where a huge space of interacting options makes it almost impossible to humans to take optimal or even good decisions. From a historical perspective, operations research stems from the application of mathematical methods to (mostly) industrial applications while planning and constraint programming emerged as subfields of artificial intelligence where the emphasis was traditionally more on symbolic and logical search techniques for the intelligent selection and sequencing of actions to achieve a set of goals. Therefore operations research often focuses on the allocation of scarce resources such as transportation capacity, machine availability, production materials, or money, while planning focuses on the right choice of actions from a large space of possibilities. While this difference results in problems in different complexity classes, it is often possible to cast the same problem as an OR, CP, or planning problem. In this seminar, we investigated the commonalities and the overlap between the different areas to learn from each other’s expertise, bring the communities closer together, and transfer knowledge about solution techniques that can be applied in all areas.

Seminar February 11–16, 2018 – <https://www.dagstuhl.de/18071>

2012 ACM Subject Classification Applied computing → Operations research, Computing methodologies → Planning and scheduling, Theory of computation → Constraint and logic programming

Keywords and phrases Artificial Intelligence, Automated Planning and Scheduling, Constraint Programming, Dynamic Programming, Heuristic Search, Mixed Integer Programming, Operations Research, Optimization, Real-world Applications, Reasoning under Uncertainty

Digital Object Identifier 10.4230/DagRep.8.2.26

Edited in cooperation with Florian Pommerening

1 Executive Summary

Florian Pommerening (Universität Basel, CH)

License  Creative Commons BY 3.0 Unported license
© Florian Pommerening

This seminar brought together leading experts in the fields of AI planning, constraint programming and operations research. These areas historically come from different roots but are all concerned with supporting decision making in complex systems which a huge space



Except where otherwise noted, content of this report is licensed under a Creative Commons BY 3.0 Unported license

Planning and Operations Research, *Dagstuhl Reports*, Vol. 8, Issue 02, pp. 26–63

Editors: J. Christopher Beck, Daniele Magazzeni, Gabriele Röger, and Willem-Jan Van Hoeve



Dagstuhl Reports

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

of interacting options. While the approach and focus is different, some concepts have been developed in multiple areas and some solution techniques are or could be transferred. There is also a growing intersection of the areas that considers hybrid problems or uses solvers developed in one area to solve problems from a different area, for example by compiling planning problems into MIP or by using CP for subproblems in a MIP solver. Solvers of a different community are often used as black boxes and the deeper understanding of each other's area of expertise that was developed in this seminar will help to foster collaboration and transfer knowledge between the areas.

The seminar started with eleven short but intense tutorials on Monday and Tuesday morning. The tutorials on Monday conveyed the basics of AI Planning, MIP, and CP. They also already introduced the main connections between the fields by talking about compilations from planning to CP and MIP and using LPs as heuristics in planning. The tutorials on Tuesday delved deeper into areas that became the focus of discussion later in the seminar, such as non-deterministic planning, Markov decision processes, and decision diagrams. Front-loading these tutorials worked well to bring everyone up to speed and created a good basis for the rest of the seminar.

The rest of the seminar was organized into working groups that included one to three short presentations followed by a longer discussion all focused on a central topic. Three of these sessions were organized as break-out sessions where the participants split into two groups, each discussion one topic and then reconvening to present the main points discussed in each group to each other. The schedule for each day was created on the evening before which kept the topics flexible and allowed the organizers to include topics that came up during the discussion. Notes on each of the working groups and abstracts of the tutorials are included in the rest of this report.

2 Table of Contents

Executive Summary

<i>Florian Pommerening</i>	26
--------------------------------------	----

Overview of Tutorials

Introduction to Planning	
<i>Malte Helmert</i>	30
Introduction to MIP	
<i>Thorsten Koch</i>	30
Compiling Planning to Mixed Integer Linear Programming	
<i>Chiara Piacentini</i>	31
LP for Classical Planning Heuristics	
<i>Florian Pommerening</i>	31
Heuristics for Numeric Planning	
<i>Patrik Haslum</i>	32
Introduction to Constraint Programming	
<i>Pierre Schaus</i>	33
Compiling Planning to Constraint Programming	
<i>Roman Bartak</i>	33
Introduction to Non-deterministic Planning	
<i>Christian Muise</i>	34
Markov Decision Processes in Planning	
<i>Matthijs Spaan</i>	34
Decision Diagrams for Planning and Optimization	
<i>Scott Sanner</i>	35
Decision Diagrams for Optimization	
<i>Willem-Jan Van Hoeve</i>	35

Working groups

MINLP and Nonlinearities in Planning	
<i>J. Christopher Beck</i>	36
Benchmarking	
<i>Christina N. Burt</i>	37
Merge&Shrink and Decision Diagrams	
<i>Christina N. Burt and Florian Pommerening</i>	38
Teaching	
<i>Hadrien Cambazard</i>	42
Hybrids & Decomposition	
<i>Mathijs de Weerd</i>	43
Non-traditional Objective Functions for MDPs	
<i>Patrik Haslum</i>	45

Open and Challenging Problems	
<i>Patrik Haslum</i>	46
Explaining Solutions and Solution Methods	
<i>Christian Muise</i>	48
Modeling	
<i>Christian Muise</i>	49
Cross-Domain Example Problems	
<i>Florian Pommerening</i>	50
Dealing with Uncertainty	
<i>Florian Pommerening</i>	52
Connections Between our Fields	
<i>Mark Roberts</i>	53
LP Heuristics: The Details	
<i>Gabriele Röger</i>	55
Planning, Optimization, and Machine Learning	
<i>Scott Sanner</i>	56
Multiagent Planning and Optimization	
<i>Matthijs Spaan</i>	58
Exploiting Substructures	
<i>Charlotte Truchet</i>	61
Participants	63

3 Overview of Tutorials

3.1 Introduction to Planning

Malte Helmert (Universität Basel, CH)

License © Creative Commons BY 3.0 Unported license
© Malte Helmert

The talk gave a brief introduction to domain-independent automated planning. It introduced the classical (finite-domain, grounded, sequential) planning problem and generalizations to numerical state variables and to temporal planning. The talk discussed common representations used for classical planning, including finite-domain (SAS⁺) representation, transition normal form (TNF), STRIPS, and ADL. It also pointed out the connection between classical planning and automata theory, where individual state variables can be interpreted as finite automata, and the overall planning problem corresponds to language intersection/product automata. Finally, the talk discussed the planning research community in order to help locate relevant research (published primarily at the ICAPS, IJCAI, AAAI and ECAI conferences and in the JAIR and AIJ journals), to understand benchmarking practices using the PDDL representation language and the benchmark collections from the International Planning Competitions (IPC), and to understand the historical emphasis of planning research of domain-independent algorithms and “neutral” models (“Physics, not advice”).

3.2 Introduction to MIP

Thorsten Koch (Konrad-Zuse-Zentrum – Berlin, DE)

License © Creative Commons BY 3.0 Unported license
© Thorsten Koch

Joint work of The Mathematical Optimization Methods group at ZIB

Main reference Ambros Gleixner, Leon Eifler, Tristan Gally, Gerald Gamrath, Patrick Gemander, Robert Lion Gottwald, Gregor Hendel, Christopher Hojny, Thorsten Koch, Matthias Miltenberger, Benjamin Müller, Marc E. Pfetsch, Christian Puchert, Daniel Rehfeldt, Franziska Schlösser, Felipe Serrano, Yuji Shinano, Jan Merlin Viernickel, Stefan Vigerske, Dieter Weninger, Jonas T. Witt, and Jakob Witzig: “The SCIP Optimization Suite 5.0”. ZIB-Report, pp. 17–61, ISSN: 1438-0064, Zuse Institute Berlin, 2017.

URL <http://nbn-resolving.de/urn:nbn:de:0297-zib-66297>

The presentation gave an introduction to computational Mixed Integer Programming (MIP) and the algorithms behind it, in particular the Branch-and-Cut paradigm. This included notes about parallel implementations and extension to non-convex Mixed Integer Non-Linear Programming (MINLP).

3.3 Compiling Planning to Mixed Integer Linear Programming

Chiara Piacentini (University of Toronto, CA)

License © Creative Commons BY 3.0 Unported license
© Chiara Piacentini

Joint work of Chiara Piacentini, Margarita P. Castro, André A. Ciré, J. Christopher Beck

Main reference Chiara Piacentini, Margarita P. Castro, André A. Ciré, J. Christopher Beck: “Compiling Optimal Numeric Planning to Mixed Integer Linear Programming”, in Proc. of the Twenty-Eighth International Conference on Automated Planning and Scheduling, ICAPS 2018, Delft, The Netherlands, June 24-29, 2018., pp. 383–387, AAAI Press, 2018.

URL <https://aaai.org/ocs/index.php/ICAPS/ICAPS18/paper/view/17770>

Compilation techniques in planning reformulate a problem into an alternative encoding for which efficient, off-the-shelf solvers are available. In this tutorial, we presented mixed-integer linear programming (MILP) compilations for cost-optimal planning with instantaneous actions. We presented three encodings of classical planning taken from the literature and we show how they can be extended to handle problems with numeric state variables, numeric linear conditions, and numeric linear action effects.

References

- 1 Thomas Vossen, Michael O. Ball, Amnon Lotem, and Dana S. Nau. *On the Use of Integer Programming Models in AI Planning*. In Proceedings of the Sixteenth International Joint Conference on Artificial Intelligence, pp. 304–309, 1999.
- 2 Menkes van den Briel, Thomas Vossen, and Subbarao Kambhampati. *Reviving Integer Programming Approaches for AI Planning: A Branch-and-Cut Framework*. In Proceedings of the Fifteenth International Conference on Automated Planning and Scheduling, pp. 310–319, 2005.

3.4 LP for Classical Planning Heuristics

Florian Pommerening (Universität Basel, CH)

License © Creative Commons BY 3.0 Unported license
© Florian Pommerening

Main reference Florian Pommerening: “New Perspectives on Cost Partitioning for Optimal Classical Planning”. Dissertation, University of Basel, Switzerland, 2017.

Linear Programs (LPs) are used in classical planning in different ways. This talk focused around their uses while computing admissible *heuristic functions*, which compute lower bounds during a search for an optimal plan. The use of LPs in such cases can be grouped into three classes: *computation*, *combination*, and *synthesis*.

In heuristic computation, heuristic values of heuristic functions like the state-equation heuristic [1] or the delete-relaxation heuristic [2] are computed as the objective value of an LP.

Heuristic combination is specifically interesting for admissible heuristics where they solve the problem of combining several lower bounds into a single lower bound while minimizing the loss of information from all bounds. Cost partitioning [3], posthoc optimization [4], and operator counting [5] are examples of this.

Finally, LPs can be used to synthesize a whole heuristic function. For example, potential heuristics [5] use linear constraints that characterize desired heuristic properties and optimize a measure of quality to select the best potential heuristic with the desired properties.

References

- 1 Menkes van den Briel, J Benton, Subbarao Kambhampati, and Thomas Vossen. *An LP-based heuristic for optimal planning*. In Proceedings of the Thirteenth International Conference on Principles and Practice of Constraint Programming, pp. 651–665, 2007.
- 2 Tatsuya Imai and Alex Fukunaga. *On a practical, integer-linear programming model for delete-free tasks and its use as a heuristic for cost-optimal planning*. Journal of Artificial Intelligence Research 54, pp. 631–677, 2015.
- 3 Michael Katz and Carmel Domshlak. *Optimal admissible composition of abstraction heuristics*. Artificial Intelligence 174(12–13), pp. 767–798, 2010.
- 4 Florian Pommerening, Gabriele Röger, and Malte Helmert. *Getting the most out of pattern databases for classical planning*. In Proceedings of the 23rd International Joint Conference on Artificial Intelligence, pp. 2357–2364, 2013.
- 5 Florian Pommerening, Malte Helmert, Gabriele Röger, and Jendrik Seipp. *From non-negative to general operator cost partitioning*. In Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence, 3335–3341, 2015.

3.5 Heuristics for Numeric Planning

Patrik Haslum (Australian National University, AU)

License  Creative Commons BY 3.0 Unported license
© Patrik Haslum

Numeric planning extends the classical discrete state planning model with state variables whose values are rational numbers and actions whose preconditions and effects involve them. Unlike in classical setting, this makes the set of reachable states potentially non-finite, but apart from that, numeric planning keeps all the classical planning assumptions of determinism, observability and model completeness. Although semantically simple, this extension is quite powerful: the plan existence problem is semi-decidable even for very restricted fragments of numeric planning. Extending classical planning in a different way, “semantic attachments” are procedurally defined (“black box”) functions or predicates used to model aspects of a planning problem that do not fit in the classical model.

Recently, several relaxations, with associated heuristics, have been proposed both for general and restricted numeric planning and for planning with semantic attachments. This allows heuristic search methods, which have been very successful in classical planning, to address these extend models as well.

The aim of this tutorial was to give an overview of the numeric planning formalism, some of its relaxations, and draw parallels with ideas in OR.

3.6 Introduction to Constraint Programming

Pierre Schaus (UC Louvain, BE)

License © Creative Commons BY 3.0 Unported license
© Pierre Schaus

Main reference Laurent Michel, Pierre Schaus, Pascal Van Hentenryck: “Mini-CP: A Minimalist Open-Source Solver to teach Constraint Programming”. 2017.

URL <https://www.info.ucl.ac.be/~pschaus/minicp>

We introduced constraint programming and its driving mantra CP=modeling+search. Standard examples such as job-shop models were used to illustrate the CP technology. Filtering algorithms for global constraints were briefly described for cumulative scheduling constraints, table constraints, element constraints, etc. Some implementation details about CP solvers internals were given: the search and state-restoration techniques (trails, DFS, fix-point computation). The differences between CP models and MIP/SAT models were emphasized. The importance of the search heuristics was illustrated with recent black-box searches able to learn from conflicts during the search. Several hybridization techniques were mentioned briefly: lazy-clause generation, Lagrangian-based filtering, column generation, multi-decision diagrams. Large Neighborhood Search was explained as a CP and Local-Search hybridization to make CP scale well and improve any-time behavior on large size problems.

3.7 Compiling Planning to Constraint Programming

Roman Barták (Charles University – Prague, CZ)

License © Creative Commons BY 3.0 Unported license
© Roman Barták

Main reference Roman Barták: “On Constraint Models for Parallel Planning: The Novel Transition Scheme”, in Proc. of the Eleventh Scandinavian Conference on Artificial Intelligence, SCAI 2011, Trondheim, Norway, May 24th – 26th, 2011, Frontiers in Artificial Intelligence and Applications, Vol. 227, pp. 50–59, IOS Press, 2011.

URL <http://dx.doi.org/10.3233/978-1-60750-754-3-50>

The tutorial described several methods of encoding the problem of finding a plan with at most k actions as a constraint satisfaction problem. First, three constraint models were introduced, a straightforward (textbook) model, a Graphplan-based model, and a model with successor-state axioms. For each model its reformulation using tabular constraints was suggested and experimental comparison of all models was given. The best model was then extended using techniques such as lifting, symmetry breaking, and singleton consistency and efficiency improvement was experimentally demonstrated. The planner based on this final model is called SeP (sequential planner). In the second part, we presented a constraint model based on finite state automata that supports parallel actions. The obtained planner is called PaP (parallel planner). The model was then reduced to achieve even better efficiency, which was demonstrated experimentally.

3.8 Introduction to Non-deterministic Planning

Christian Muise (IBM TJ Watson Research Center – Cambridge, US)

License © Creative Commons BY 3.0 Unported license
© Christian Muise

Main reference Christian Muise, Sheila A. McIlraith, J. Christopher Beck: “Improved Nondeterministic Planning by Exploiting State Relevance”, in Proc. of the 22nd Int’l Conf. on Automated Planning and Scheduling, pp. 172–180, 2012.

URL <http://www.aaai.org/ocs/index.php/ICAPS/ICAPS12/paper/view/4718>

Non-deterministic planning [1] is a variant of planning where the action outcomes are uncertain but can be observed at execution time. Unlike classical planning, where a solution is a sequence of actions, a solution to a non-deterministic planning problem corresponds to a policy that maps the state of the world to an action for execution. In this talk, I presented the fully observable non-deterministic (FOND) planning formalism, outlined the main solution techniques that exist for FOND, discussed some of the applications of FOND planning, and concluded with some of the major open research challenges in the field.

References

- 1 Alessandro Cimatti, Marco Pistore, Marco Roveri, and Paolo Traverso. *Weak, strong, and strong cyclic planning via symbolic model checking*. Artificial Intelligence 147(1) pp. 35–84, 2003.

3.9 Markov Decision Processes in Planning

Matthijs Spaan (TU Delft, NL)

License © Creative Commons BY 3.0 Unported license
© Matthijs Spaan

Joint work of Matthijs Spaan, Erwin Walraven

Main reference Erwin Walraven, Matthijs T. J. Spaan: “Accelerated Vector Pruning for Optimal POMDP Solvers”, in Proc. of the Thirty-First AAAI Conference on Artificial Intelligence, February 4-9, 2017, San Francisco, California, USA., pp. 3672–3678, AAAI Press, 2017.

URL <http://aaai.org/ocs/index.php/AAAI/AAAI17/paper/view/14960>

Planning under uncertainty can be formalized using Markov Decision Processes (MDPs) [1]. In this talk, I first introduced the MDP model for fully observable environments and basic algorithms for computing policies such as value iteration and policy iteration. Next, I considered Partially Observable Markov Decision Processes (POMDPs), which extend MDPs to handle planning for agents that cannot perfectly sense the state of the system. Finally, I discussed recent work that builds on a Benders decomposition to speed up solving of linear programs in optimal POMDP solving.

References

- 1 Matthijs T. J. Spaan. *Partially Observable Markov Decision Processes*. In Reinforcement Learning: State of the Art, pp. 387–414, Springer Verlag, 2012.

3.10 Decision Diagrams for Planning and Optimization

Scott Sanner (University of Toronto, CA)

License © Creative Commons BY 3.0 Unported license
© Scott Sanner

Decision diagrams have proved to be a useful data structure for avoiding state enumeration in model checking, temporal verification, graphical model inference, classical planning, and value and policy optimization in factored MDPs and POMDPs. This tutorial covered the foundations of binary and algebraic decision diagrams (BDDs & ADDs) – their properties, their algorithms, their use in various automated planning settings. Beyond BDDs and ADDs, the tutorial also covered a variety of less well-known but important decision diagrams and their applications: Zero-suppressed DDs (ZDDs) for set representation, Affine ADDs (AADDs) for arithmetic function representation, and recent extensions of decision diagrams to continuous variables (XADDs) enabling novel exact solutions to a variety of continuous planning problems.

3.11 Decision Diagrams for Optimization

Willem-Jan Van Hoeve (Carnegie Mellon University – Pittsburgh, US)

License © Creative Commons BY 3.0 Unported license
© Willem-Jan Van Hoeve
Joint work of David Bergman, André A. Ciré, Willem-Jan van Hoeve, John N. Hooker
Main reference David Bergman, André A. Ciré, Willem-Jan van Hoeve, John N. Hooker: “Decision Diagrams for Optimization”, Springer, 2016.
URL <http://dx.doi.org/10.1007/978-3-319-42849-9>

Binary decision diagrams have a long history in computer science, where they have been widely used as an efficient data structure for representing and solving circuit design and verification problems. Decision diagrams are a relatively new tool for optimization, however, and were introduced for this purpose about 10 years ago. This presentation described the development and application of decision diagrams as a framework for generic discrete optimization [1]: Decision diagrams can be used as discrete relaxations to generate dual bounds, as restrictions to generate primal solutions, and they form the basis for a new branch-and-bound search scheme. For several classical optimization problems, including the independent set, maximum cut, and maximum 2-SAT problems, decision diagrams can be competitive with or outperform state-of-the-art integer programming solvers such as CPLEX. Moreover, decision diagrams have been successfully applied to solve disjunctive scheduling and routing problems, and were able to close several open sequential ordering problems from the well-known TSPLIB benchmark set.

References

- 1 David Bergman, André A. Ciré, Willem-Jan van Hoeve, and John N. Hooker. *Decision Diagrams for Optimization*. Springer, 2016.

4 Working groups

4.1 MINLP and Nonlinearities in Planning

J. Christopher Beck (University of Toronto, CA)

Speakers Michael Cashmore (King's College London, GB), Patrik Haslum (Australian National University, AU), and Chiara Piacentini (University of Toronto, CA)

License © Creative Commons BY 3.0 Unported license
© J. Christopher Beck

The session was devoted to discussing the ways in which nonlinear constraints had been incorporated into planning models and solution techniques with the goals of exposing the work to the MINLP present.

4.1.1 Planning vs. Compilation for Linear and Non-linear Problems

Patrik Haslum discussed his experience with planning for complex systems where the dynamics of the environment could be described with linear or non-linear constraints. Such problems arise for example in determining power flow on a network (e.g., opening and closing switches to maintain network safety, power levels, etc.). While the power flow equations are non-linear, power engineers often use a linear relaxation.

There are (at least) two competing approaches: 1) use planning technology with an optimization sub-solver called at important points (often at each node in the planning search) and 2) unroll the whole problem in to an optimization problem (“unroll[ing]” is needed because there is a need for repeated decision making over time as decisions are taken and the network state is observed).

When a linear relaxation is used:

- Planning-with-LP: 50% of instances solved in < 1800 seconds with 1-3 faults (that is for problems with 1 to 3 simulated faults)
- MIP: 100 seconds for 5+ faults

However, if the nonlinear model is solved:

- Planning-with-NLP: 50% solved in 1000 seconds
- MINLP: 30% solved in 1000 seconds

So MIP beats planning-with-LP in the linear model but planning-with-NLP beats MINLP in the nonlinear case.

Conjecture: The MIP solver has benefited from last 2 decades of intense development but MINLP solver has not. But there is also the problems of comparing commercial MIP solvers to research MINLP code: MIP solvers are commercially software engineered, MINLP are not.

From a research perspective it is a general question to understand what decisions should be pushed to what technology with the unrolled solvers being an extreme version (everything in optimization). The answer depends not only on the (software) maturity of the solvers involved but also the problem solving leverage that planning might provide.

4.1.2 SMT for Planning with Non-linear Temporal Change

Michael Cashmore presented the problem of planning with non-linear change. The problem has discrete modes and nonlinear evolution while it is in a particular mode. It is a control problem with the goal of finding the sequence of the changes to the discrete state to achieve a goal.

The approach taken is SAT Modulo theory approach using a time-indexed model. Each index corresponds to a “happening” where discrete change takes place. Between the happenings there is continuous nonlinear change on the numeric variables.

One challenge is how do we check the numeric constraints between happenings. They have created a system that combines quantifier-free non-linear real arithmetic using a computer algebra based preprocessing system to do symbolic integration to derive expressions to include in the theory solver. The time between happenings is the maximum interval such that the numeric constraints are guaranteed to be satisfied. The second happening is placed at the earliest point where the constraints could be broken so that discrete change could deal with it.

Questions:

- How could this be solved with MINLP? Can we take the SMT model and solve it directly with an MINLP solver?

4.1.3 Planning with Non-linear (But Non-temporal) Change

Chiara Piacentini presented the problem of planning with nonlinear changes to the numeric state variables that do not depend on time - that is, with some nonlinear effects. This is not a problem with the progression of the planning because the state is known and so you can model the impact of the nonlinearities on the numeric variables getting you to the next step. A challenge arises in calculating the heuristic because as you calculate the heuristic distance estimate you do not have a state in the heuristic calculation.

As an example: power flow problem based on the discrete changes in power settings at the nodes in the power flow network. Linearizing the power flow equations can result in an invalid state but this can be good enough for the heuristic.

Another approach is a benders/semantic attachment approach where the planning-level relaxation is a hand-built linearization of the nonlinear power flow equations. The problem is how to automatically generate the linearization that could be used in heuristic.

Questions:

- Are there connections with spatial branching in MINLP?
- Is the concept of semantic attachments really analogous to Benders? Refining the relaxation which Benders does would seem to require a re-evaluation of the open list.

4.2 Benchmarking

Christina N. Burt (Satalia – London, GB)

Speakers Malte Helmert (Universität Basel, CH) and Domenico Salvagnin (University of Padova, IT)

License © Creative Commons BY 3.0 Unported license

© Christina N. Burt

The benchmarking discussion focussed on commonalities and differences in both issues and solutions between the mathematical programming and automated planning communities. In both communities it was acknowledged that benchmarking pushes the state of the art and has a big impact on research.

Several issues were raised with benchmarking in general, with some proposed solutions. The first point related to comparing two algorithms, and determining what metric or point of comparison is appropriate, such as time to solve. First it must be ensured that the algorithms are comparable, such as having the same settings and parameters. When

comparing algorithms based on time, the environment should be clean (e.g. on a cluster, ensure all machines are identical and each job is run exclusively on each machine). It was also highlighted that we should encourage researchers to state which CPU they use in their experiments, as this is more meaningful than processor speed. The second issue raised was how to be sure an improvement in algorithm performance is not due to an artefact. In MIP benchmarking, multiple runs of the algorithm are made on the same instance with differing random seeds in order to evaluate if changes to the algorithm are consistent.

This led to a discussion of how to select benchmarking instances in a fair way. e.g. finding instances that are hard for both solvers/algorithms. It is biased to take all the problems that are difficult for one algorithm as the test-bed. Finally, the topic of data aggregation was discussed, in particular that the way the results are aggregated can change the outcome. Using shifted geometric mean helps prevent bias toward smaller numbers. Truncated geometric mean has the same effect. However, it was suggested that we could move away from relative metrics, such as how far this solver is away from the fastest, but instead award points for achievements, such as solving in 1 second.

4.2.1 Talk: Benchmarking in MIP

Domenico Salvagnin (University of Padova, IT)

License © Creative Commons BY 3.0 Unported license
© Domenico Salvagnin

We surveyed the major challenges and pitfalls in algorithm benchmarking, and how they are currently handled by the MIP community.

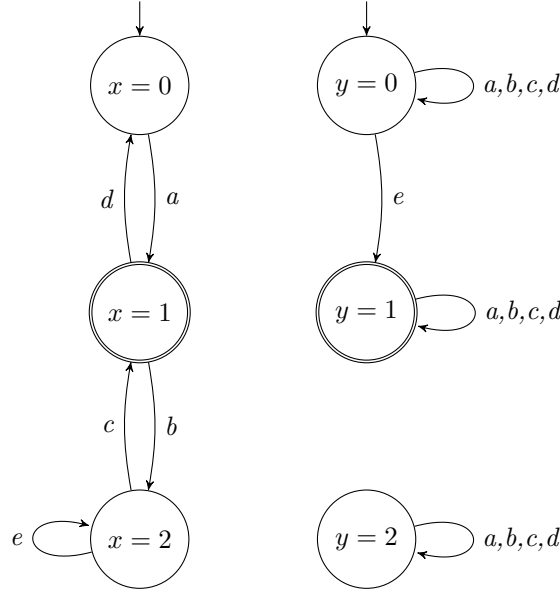
4.3 Merge&Shrink and Decision Diagrams

Christina N. Burt (Satalia – London, GB) and Florian Pommerening (Universität Basel, CH)

Speakers Malte Helmert (Universität Basel, CH) and Willem-Jan Van Hoeve (Carnegie Mellon University – Pittsburgh, US)
License © Creative Commons BY 3.0 Unported license
© Christina N. Burt and Florian Pommerening

In this discussion session moderated by Willem Van Hoeve we investigated the connections of merge&shrink [1], decision diagrams [2], and regular language constraints [3]. We started with an example planning task that has two variables x and y . Both variables can take values from $\{0, 1, 2\}$, are initially 0 and should get a value of 1 in a goal state. There are five operators a, b, c, d, e that change the variables as shown in Figure 1.

Consider only variable x for now. The behaviour of a variable in a planning task can be described with a deterministic finite automaton (DFA). The alphabet of the DFA contains all operators, and a word (i.e., an operator sequence) is in the language of the DFA iff executing the operator sequence in the projection to this planning variable leads to goal state. The state space of the projection almost directly corresponds to the transition function of the DFA: the only difference is that a DFA has exactly one outgoing transition for every symbol in every state. This can be easily satisfied by adding a dead state and route all missing transitions to this state. Other than that, the states of the DFA correspond to the value a variable has and applying an operator changes this value, so it brings the DFA into a new state.

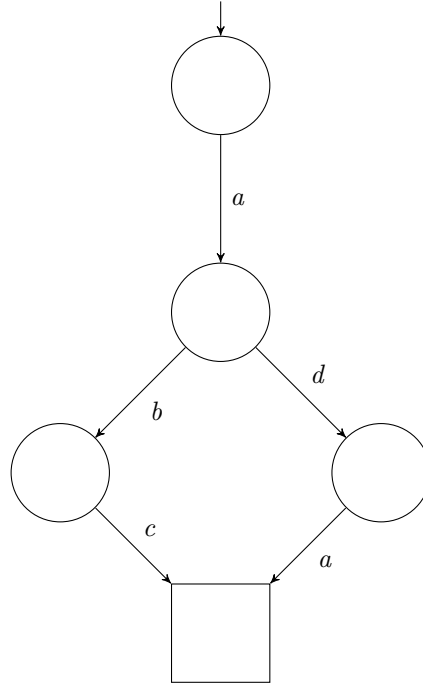


■ **Figure 1** Example planning task. Each diagram shows the projection of the task to one of the variables, i.e. the effect of all operators on the variable.

A solution to the planning task has to bring all variables to their goal value, i.e. we have to find a single plan that is a plan in all projections. With the interpretation above this means that we have to find a word that is in the intersection of several regular languages. This yields a simple compilation of planning tasks to CP using the **regular** constraint for regular languages [3]: We use a fixed time horizon of n steps and introduce a CP variable X_i for every time step $1 \leq i \leq n$. The domain of each variable is the set of operators ($\{a, b, c, d, e\}$ in our example). There is one constraint **regular**(X, ρ_j) for each planning variable j , where ρ_j is the regular expression for the DFA that describes the behavior of j . In our example task, ρ_x is the regular expression $a((da)|(be^*c))^*$ and ρ_y is $(a|b|c|d)^*e(a|b|c|d)^*$ (cf. Figure 1). By using the **cost-regular** constraint [4] instead, we can also support action costs in planning. Additional redundant constraints can be added using landmarks. For example, if a landmark analysis finds out that operator o is used at least k times, we can add the constraint $\sum_i (X_i = o) \geq k$. We could also use a global cardinality constraint [5] to express bounds on several operators: if the number of operator uses of operator o is between L_o and U_o then the global constraint **gcc**(X, L, U) expresses the constraint $\bigwedge_o L_o \leq \sum_i (X_i = o) \leq U_o$. A CP-based planner would probably require such landmarks to be efficient but they can be extracted from the planning problem.

A **regular** constraint for finite sequences can be expressed with an MDD as shown in Figure 2. By intersection of these MDDs we can solve the planning problem for a bounded plan length. The shortest plan can then be found by iteratively solving for longer sequences. The MDD can be relaxed by bounding its width so the computation remains polynomial and returns a lower bound on the cost of a cheapest plan.

It is generally not possible (unless PSPACE = NP) to have a polynomial algorithm without the bounded length restriction because planning is PSPACE-complete. However, we can unroll levels only partially and use loops in the last or first layer to maintain completeness. This was under investigation in a Masters thesis in Basel [6].



■ **Figure 2** MDD expressing the behavior of variable x in our example task for sequences up to length 3.

The merge&shrink heuristic [1] is a planning heuristic that uses a similar approach to get a lower bound on the plan cost. The heuristic computation starts with the projections to all planning variables (i.e., the DFAs). All DFAs together preserve the original behavior of the system. Replacing two DFAs with their product automaton (called “merging”) does not lose any information. Thus automata are merged while there is enough space to represent the resulting products. If a given space limit is reached, two states in one of the DFAs are combined into one (called “shrinking”). It would be possible to get a bound from each automaton but the bounds get better by combining all of them. Thus the algorithm continues merging and shrinking until there is only one automaton left. It is not so easy to keep track of which DFA state belongs to which states from the original problem after some merging and shrinking steps since we cannot keep track of all possibilities during the construction in a table. This would be as difficult as enumerating all states. Instead, trees are used to map partial states to abstract states. For the original DFAs, the tree consists of a root node labeled with the variable and one leaf for each abstract state and edges that are labeled with the variable values. The merging step combines two trees by appending one tree at each leaf state of the other tree (and then possibly simplifying the result). The shrinking step just relabels some leaf states (followed by simplifications). The representation at any time is an MDD. Since the leaves are the abstract states of all automata, its size is limited by the space limit of the heuristic.

The heuristic function of merge&shrink can be represented as an ADD by writing the heuristic values into the leaf nodes instead of the abstract states as for example done in symbolic merge&shrink heuristics. The interesting connection to the earlier discussion is that the memory bound in merge&shrink is essentially bounding the width of the MDD/ADD. While the bound is actually bounding the number of leaves, trees are combined by replacing

leaves of one tree by another tree so limiting the number of leaves in turn limits the width of the decision diagram on all intermediate layers as well.

However, there are differences between the decision diagrams created by merge&shrink and those created for a regular language constraint since the variables do not represent the same objects. In the compilation to CP, variables represent operators used at specific time steps, unrolling the state space to some depth. In the heuristic on the other hand, the decision diagram stores the information to which abstract state an original state is mapped.

Apart from the use as a lower bound (heuristic), an MDD as build by merge&shrink could be useful for several other questions. If it were not relaxed, it would represent all solutions, so it could for example be used to count the number of solutions. It could also be limited to paths with a certain cost to represent all optimal paths. A relaxed MDD can give approximate answers to these questions.

Another interesting direction would be to use a CP model based on **regular** constraints for planning heuristics that extract more information than just a heuristic value. This way, it could for example easily be combined with operator-counting heuristics. Other suggestions that came up in discussion were to combine merge&shrink with more information such as mutexes (this is possible in general but takes a lot of memory) and using MDDs for proving unsolvability. Merge&shrink has been specialized for unsolvability where more abstract states can be merged without losing information. Unrolling the DFAs of all state variables up to a certain bound could also be used to prove that plans cannot have a certain length. However self-loops in the state space are common and lead to a problem because with loops it is always possible to unroll the state space one level further.

Relaxed MDDs can also be used to represent the whole problem. Ongoing work in Toronto builds an MDD and extracts relaxed solutions from it, unrolling the MDD until an actual solution is found. The difference to the methods described above is that this does not start from the DFAs describing the behavior of each variable.

References

- 1 Malte Helmert, Patrik Haslum, Jörg Hoffmann, and Raz Nissim. *Merge-and-Shrink Abstraction: A Method for Generating Lower Bounds in Factored State Spaces*. Journal of the ACM 61 (3), pp. 16:1–63. 2014.
- 2 David Bergman, André A. Ciré, Willem-Jan van Hove, and John N. Hooker. *Decision Diagrams for Optimization*. Springer, 2016.
- 3 Gilles Pesant. *A regular membership constraint for finite sequences of variables*. In Proceedings of the Tenth Conference on Principles and Practice of Constraint Programming, pp. 482–495, 2004.
- 4 Sophie Demassey, Gilles Pesant, Louis-Martin Rousseau. *A Cost-Regular based Hybrid Column Generation Approach*. Constraints 11 (4), pp. 315–333, 2006.
- 5 Jean-Charles Régin. *Generalized Arc Consistency for Global Cardinality Constraint*. In Proceedings of the Thirteenth National Conference on Artificial Intelligence, pp. 209–215, 1996.
- 6 Philipp Oldenburg. *Time Unrolling Heuristics*. Master Thesis, University of Basel, Switzerland, 2017.

4.4 Teaching

Hadrien Cambazard (Grenoble INP, FR) and Florian Pommerening (Universität Basel, CH)

Speakers Hadrien Cambazard (Grenoble INP, FR), Emmanuel Hebrard (LAAS – Toulouse, FR), Serdar Kadioglu (Fidelity Investments – Boston, US), and Pierre Schaus (UC Louvain, BE)

License © Creative Commons BY 3.0 Unported license
© Hadrien Cambazard

The goal of this session was to discuss the use of optimization techniques for teaching as well as discussing way to teach these techniques.

Pierre Schaus started the session by presenting MiniCP [1], a CP solver dedicated to teaching CP. The solver has the same design goal as the well-known MiniSat solver [2]: to provide a simple solver that can easily be investigated and used to learn about the most important techniques in a modern CP solver. MiniSat has largely contributed to the dissemination of SAT solvers and MiniCP should do the same for CP solvers. A solver like this can be used as a tool to help people get into the CP community and teach newcomers the fundamentals of how CP solvers are implemented. A suggestion that came up in discussion was to run a MiniCP competition.

Serdar Kadioglu then reported on his experience with gamification for a course on SAT, CP, LP, and MIP. The course was driven by projects which ran over three weeks. Each project started with a high level narrative description, a defined input/output format and students then designed their own solutions. A leader board was used to give immediate feedback on how well their strategy worked and the quality of results of the group are visible to all students but are kept anonymous. An interesting observation is that once a student managed to get a first feasible solution, others succeeded usually quickly afterwards. Because students were engaged and driven to improve their approach, solution qualities tended to converge and the projects became very hard to grade. Overall the engagement of students was very strong. The discussion focused on the advantages and disadvantages of this approach for teaching and academia. For example, this way of teaching might lead to a more academic orientation of students by encouraging them to search for innovative solutions by themselves. The difficulty in grading was also discussed in more detail. This issues is shared with massive open online courses (MOOCs) where techniques like this one are also used. The discussion then turned on whether and how online classes will affect the way universities work? Serdar argued that so far, we have not seen strong effects but these new techniques are likely to change our practices.

Next, Emmanuel Hebrard showed a Python solver called PySched [3] that provides state-of-the-art filtering algorithms in CP and can output diagrams in LaTeX to visualize the reasoning performed by the solver. Christian Muise asked if teaching materials like PySched or MiniCP are collected and shared. An interesting idea would be to include a demonstration about teaching in the ICAPS demo session.

Finally, Hadrien Cambazard presented Caseine [4], a learning platform in Industrial Engineering, Mathematics and Computer science. Its aim is to stimulate students' learning and autonomy while improving the quality of the time the teacher gives them. Based on Moodle, it allows to automatically evaluate the students' computer code and mathematical models, to monitor the students' progress, and to share content among the teachers through a community of users. It is meant to support "active classrooms" or "flipped classrooms". Caseine could be an option to share teaching materials for CP as suggested by Christian Muise earlier.

References

- 1 Laurent Michel, Pierre Schaus, Pascal Van Hentenryck. *Mini-CP: A Minimalist Open-Source Solver to teach Constraint Programming*. 2017. Available from www.info.ucl.ac.be/~pschaus/minicp.
- 2 Niklas Eén and Niklas Sörensson. *An extensible SAT-solver*. In Proceedings of the Sixth International Conference on Theory and Applications of Satisfiability Testing, pp. 502–518, 2003.
- 3 Emmanuel Hebrard <https://github.com/ehebrard/PySched> Accessed May 2018.
- 4 Hadrien Cambazard. <http://caseine.org>. Accessed May 2018.

4.4.1 Talk: Mini-CP – A Minimalist Open-Source Solver to teach Constraint Programming

Pierre Schaus (UC Louvain, BE)

License © Creative Commons BY 3.0 Unported license
© Pierre Schaus

Joint work of Laurent Michel, Pierre Schaus, Pascal Van Hentenryck

Main reference Laurent Michel, Pierre Schaus, Pascal Van Hentenryck: “Mini-CP: A Minimalist Open-Source Solver to teach Constraint Programming”. 2017.

URL <https://www.info.ucl.ac.be/~pschaus/minicp>

The successful minisat solver has largely contributed to the dissemination of (CDCL) SAT solvers. Minisat has a neat and minimalist architecture that is well documented. We believe the CP community is currently missing such a solver that would permit new-comers to demystify the internals of CP technology. We introduce Mini-CP a white-box bottom-up teaching framework for CP implemented in Java. Mini-CP is voluntarily missing many features that you would find in a commercial or complete open-source solver. The implementation, although inspired by state-of-the-art solvers is not focused on efficiency but rather on readability to convey the concepts as clearly as possible. Mini-CP is small (< 1500 LOC excluding tests) and well tested.

4.5 Hybrids & Decomposition

Mathijs de Weerd (TU Delft, NL)

Speakers Jeremy D. Frank (NASA – Moffett Field, US), John N. Hooker (Carnegie Mellon University – Pittsburgh, US), and Louis-Martin Rousseau (Polytechnique Montreal, CA)

License © Creative Commons BY 3.0 Unported license
© Mathijs de Weerd

The goal of this session was to discuss commonalities of MIP, CP, and planning, and how ideas such as for efficient representations of the solution space, heuristics, upper and lower bounds, and decomposition can be more easily combined.

We had three main speakers in this session. John Hooker started with explaining (logic-based) Benders decomposition and the relation to similar ideas in MIP, CP, SAT, SMT, and planning. Jeremy Frank then talked about an architecture to allow combining many different techniques in solving real-life problems. Finally, Louis-Martin Rousseau explained how a combination of column generation and dynamic programming can be used to solve a complex vehicle routing problem (VRP).

4.5.1 Benders decomposition

Logic-based Benders decomposition is a method to facilitate communication between subproblems via a master problem. John Hooker explained this using two concrete examples. Here I give one.

Suppose the problem is to assign jobs with deadlines and processing times to agents that each have resource constraints (cumulative scheduling) and each agent may have different costs for executing a job. The master problem in this case is minimizing the allocation costs of assigning jobs to agents, and the subproblems are for each agent its resource-constrained scheduling problem. If an agent cannot fit all its allocated jobs, a constraint (Benders cut) is added to the master problem, which is subsequently resolved. A good cut would be a small subset of allocated jobs that together lead to infeasibility. This can be found using a dual formulation of the subproblem.

This approach has strong similarities to well-known successful approaches in various fields. In SAT solving, conflict clauses are exactly Benders cuts. In SMT the theory part is in fact a subproblem. Semantic attachments also represent a subproblem. This type of conflict analysis in MIP is called branch and check. In planning a relaxed graph or MDD can be seen as the master problem, and updating this structure can then be seen as adding Bender's cuts.

4.5.2 Architecture for hybrid approaches

Jeremy Frank talked about a planning/constraint reasoning architecture consisting of

- A temporal network
- Causal link and resources over timelines
- SAT/CSP/SMTs
- MILP
- Semantic attachments: physics of battery, power, orbit

This planning/constraint reasoning architecture has been used by several automated planners developed for applications with many different types of constraints, ranging from the usual causal constraints, to time and simple resources, to physical simulations.

The architecture splits constraints into categories for which efficient algorithms exist (e.g. temporal constraints). Doing this, however, requires both extracting the 'homogeneous' constraints from the planning model, transporting information between components that reason about a specific type of constraint, and transporting information to components that perform functions like heuristic evaluation. The architecture resembles the MILP architecture presented earlier by Thorsten Koch, but is somewhat more complex.

Generally, search in this architecture is incremental (commit to a single action or constraint at a time). This approach emphasizes the need for constraint reasoning and propagation, and efficient rollback; it facilitates many types of search (progression, regression, local search, and mixed initiative search). Unlike traditional constraint satisfaction and MILP techniques, the whole problem is not available for reasoning and heuristic decision making.

The discussion continued with the use of semantic attachments. If these are (costly) simulator runs, can we still extract useful information from this, such as the Bender cuts from John Hooker's talk? How to analyze the results from such semantic attachments to generate useful information, whether constraints or heuristic decisions, is an open question.

4.5.3 Column generation and DP for VRP

Louis-Martin Rousseau explained how at Hanalog they use a hybrid approach for a complex VRP problem with extra constraints such as on capacity and truck driver regulations, and continuity of care in the context of Homecare.

He first explained the approach of column generation in general: if a solution can be seen as a combination of solutions to subproblems, it may be more efficient to enumerate such solutions and repeatedly try to find a good combination of the ones constructed thus far. This works well for problems with some underlying structure, for example for Vehicle routing, Airline Scheduling, Shift Scheduling and Jobshop Scheduling.

For VRP the MIP constraints can lead to bad relaxations and column generation works quite well. In this case, the subproblem is finding a good route through a set of customers. For the problem at hand, there are multiple relevant resource constraints (such as the time already spent at other customers). This can however, be reasonably efficiently be expressed by a (multi-label) dynamic program. Perhaps some links can be established with the techniques used in planning here as well?

4.5.4 Discussion

We can indeed conclude that there are strong similarities between algorithms in the different paradigms (like Benders decomposition), and also concrete examples where hybrids are more successful than formulating the whole problem in one of the paradigms. It is decided that in a next session we will discuss also other similarities (e.g. with decision diagrams).

4.6 Non-traditional Objective Functions for MDPs

Patrik Haslum (Australian National University, AU)

Speakers Sven Koenig (USC – Los Angeles, US)
License  Creative Commons BY 3.0 Unported license
 © Patrik Haslum

The MDP model is used and solved both in OR (where it originated) and in AI. Sven posed two questions:

1. what has AI contributed to solving MDPs?
2. what kind of objective functions (for MDPs) can be solved, and which ones do we want to use?

The main answer to the first question was that AI contributed structured representations like probabilistic planning, factored MDPs, heuristic search, macro actions, decision diagram representations of value functions, and so on.

Concerning the second question, most AI methods work (well) with problems that are Markovian including objectives like expected (discounted) reward (for MDPs), end-state goals and sum of costs (in deterministic planning).

But utility functions, in particular for one-shot high-stakes decisions, are non-linear functions of accumulated reward (e.g., in economic decisions, attitude to risk is dependent on the amount at stake relative to total wealth), i.e., non-Markovian. Temporally extended goals (as expressed in LTL, CTL) are also non-Markovian. We can solve these by encoding enough history into the state to make the problem Markovian again, then apply (efficient) AI methods. On the other hand, encodings of planning (deterministic or not) into SAT, CP,

MIP, etc, provide a representation of the space of (bounded-length) solution plans, where non-Markovian constraints (i.e., goals) and objectives can potentially be expressed directly.

There is an interesting link between this and multi-objective problems, in efficient (symbolic) representations of the Pareto front.

4.6.1 Talk: Non-Traditional Objective Functions for MDPs

Sven Koenig (USC – Los Angeles, US) and Yaxin Liu

License © Creative Commons BY 3.0 Unported license
© Sven Koenig and Yaxin Liu

Joint work of Yaxin Liu, Sven Koenig

Main reference Yaxin Liu, Sven Koenig: “Functional Value Iteration for Decision-Theoretic Planning with General Utility Functions”, in Proceedings of the Twenty-First National Conference on Artificial Intelligence, July 16-20, 2006, Boston, Massachusetts, USA, pp. 1186–1193, AAAI Press, 2006.

URL <http://www.aaai.org/Library/AAAI/2006/aaai06-186.php>

We argue in this historical perspective on joint research with Yaxin Liu that research on probabilistic planning with Markov Decision Processes (MDPs) so far has been more interested in exploiting their structure for efficient planning for simple objective functions than in studying more realistic and thus also more complex objective functions. We try to understand the reasons for it, outline both some early and current AI research on non-traditional objective functions and explain why future AI research should focus more on realistic objective functions to make probabilistic planning more attractive for actual applications.

As an example, we discuss how to find plans that maximize the expected total utility for a given MDP, a planning objective that is important for decision making in high-stakes domains. The optimal actions can now depend on the total reward that has been accumulated so far in addition to the current state. We present our research from more than 10 years ago that extends the functional value iteration framework from one-switch utility functions to all utility functions that can be approximated with piecewise linear utility functions (with and without exponential tails) by using functional value iteration to find a plan that maximizes the expected total utility for the approximate utility function. Functional value iteration does not maintain a value for every state but a value function that maps the total reward that has been accumulated so far into a value. We describe how functional value iteration represents these value functions in finite form, how it performs dynamic programming by manipulating these representations and what kinds of approximation guarantees it is able to make.

See <http://idm-lab.org/project-d.html> for more information and published papers.

4.7 Open and Challenging Problems

Patrik Haslum (Australian National University, AU)

Speakers Malte Helmert (Universität Basel, CH), Thorsten Koch (Konrad-Zuse-Zentrum – Berlin, DE), Andrea Micheli (Bruno Kessler Foundation – Trento, IT), and Hana Rudová (Masaryk University – Brno, CZ)

License © Creative Commons BY 3.0 Unported license
© Patrik Haslum

Four speakers outlined challenging problems: Hanna Rudova, Andrea Micheli and Torsten Koch discussed application problems. Malte Helmert described the linear programs that arise in cost-partitioning planning heuristics, and raised the question what ways there are to exploit the particular structure of these LPs to solve them faster.

Hanna Rudová described the university course timetabling problem. Timetabling is a long-studied problem, but its practical application usually have additional complex constraints and/or objectives. In the university course timetabling context, these can include scheduling of linked small group activities (such as tutorials and labs) and large group lectures, with precedences (e.g., small group activities in a given week must occur either all before or all after a lecture). Problems are typically over-constrained, and objectives include maximising satisfaction of participants time and room preferences, and minimising student conflicts and other “soft” constraint violations.

Andrea Micheli described a production scheduling problem in an electroplating plant. The problem is similar to the known “hoist scheduling problem”, but has additional constraints. Hoists move parts between different different chemical baths, and there are minimum and maximum constraints on the time that each part spends in as well as between stages. Several hoists move on the same track, so one cannot bypass another; moreover, there are no “parking” spots where parts can be temporarily stored or handed over from one hoist to another. The problem is not only an operational one, but also solved as part of the plant design, for example to determine what is the optimal number of hoists for a production line.

Torsten Koch described operational problems in the control of energy systems, with specific focus on gas networks. Pressure in gas networks is controlled with a range of regulators, compressors, and valves (so it is a switching problem). The main constraint is to maintain a minimum pressure at network exits and a maximum pressure in the pipes. Optimisation problems arise at many scales, in network size (from a single building to continent-wide), level of abstraction, time horizon, and precision. European gas networks are getting old and subject to increasing use: as a result, they are operated closer to the boundaries, and more often parts are out of service. Thus, an optimisation formulation that treats all constraints as hard is more and more often not feasible.

4.7.1 Talk: The MAIS P&S Problem: Hoist Scheduling Problem in the wild

Andrea Micheli (Bruno Kessler Foundation – Trento, IT)

License © Creative Commons BY 3.0 Unported license
© Andrea Micheli

Joint work of Andrea Micheli, Alessandro Cimatti, Marco Roveri, Enrico Scala

This talk presents the characteristics of the planning and scheduling problem we are currently working on. The problem is a generalization of the Hoist Scheduling Problem for electroplating factories. In particular, we need to synthesize the movement plans for a number of hoists that need to respect a given “recipe” for production, maximizing throughput. Differently from most of the literature, we are not aiming at cyclic plans for a fixed kind of production, but we want to generate plans for a given order of production. We have a solution technique that is able to quickly generate feasible plans, but we now need to focus on (sub-)optimality.

4.7.2 Talk: Complex University Course Timetabling

Hana Rudová (Masaryk University – Brno, CZ)

License © Creative Commons BY 3.0 Unported license
© Hana Rudová

Joint work of Hana Rudová, Tomáš Müller, Zuzana Müllerová

Datasets of complex university course timetabling problems will be collected for the International Timetabling Competition 2019. Datasets are available thanks to the UniTime

timetabling system which is applied for timetabling at many institutions worldwide. Our goal is the creation of rich real-world datasets with diverse characteristics which will stimulate further research in timetabling and scheduling. The aim is to find an optimal assignment of times, rooms, and students to events related to the set of courses. Various hard constraints ensure the feasibility of the timetable and soft constraints define optimization criteria and the quality of the timetable. The key novelty lies in the combination of student sectioning together with standard time and room assignment of events in courses. We encourage scheduling and planning community to consider the study of our complex real-life problems. More details will be provided at the International Conference on the Practice and Theory of Timetabling 2018.

4.8 Explaining Solutions and Solution Methods

Christian Muise (IBM TJ Watson Research Center – Cambridge, US)

Speakers Christina N. Burt (Satalia – London, GB), Jeremy D. Frank (NASA – Moffett Field, US), Thorsten Koch (Konrad-Zuse-Zentrum – Berlin, DE), and Daniele Magazzeni (King's College London, GB)
License © Creative Commons BY 3.0 Unported license
 © Christian Muise

This session focused on issues surrounding the need to explain solutions and solution methods, and primarily focused on explanations geared towards those that need to work with the solutions (as opposed to solver developers producing the solutions).

Dan Magazzeni discussed key challenges and techniques for explaining planning solutions; an increasingly important issue with new EU regulations being enacted that will require companies to explain why certain decisions based on AI have been made. Ultimately, the building blocks for planning (e.g., causal link inference) provide a natural means for solution explanation.

Christina Burt described her experience working for Satalia and the need to explain and provide solutions to clients. Typically they do not want just a single solution, but a variety of good solutions to compare. The approach they plan on taking is to solicit a set of soft constraints from the user, and iteratively relax them until a solution is found (which simultaneously provides an explanation of what is feasible).

Jeremy Frank detailed the challenges that arise in the human-interaction interface for the space exploration setting. Typically, interaction must occur between several subject matter experts, and insights drawn across the entire spectrum of expertise. This makes the task of providing (tailored) explanations to the humans working with the systems particularly difficult.

Finally, Thorsten Koch described some of the techniques for explainable infeasibility in MIPs. Pre-trained infeasibility can be readily explained, but the explanations for irreducible subsets of constraints can be unwieldy to understand. A popular alternative is to introduce slack variables into the problem (making it feasible) and then optimizing. At times, a natural explanation may not exist, as we are trying (and failing) to solve problems that require super-human capability.

A common thread from the workshop is that we should strive to find ways to model the task of explanation explicitly into the problem representation itself. Doing so allows the strengths of modern solver technology to be leveraged both for computing solutions, and explaining them as well.

4.8.1 Talk: Explainable Planning

Daniele Magazzeni (King's College London, GB)

License © Creative Commons BY 3.0 Unported license
© Daniele Magazzeni

Main reference Maria Fox, Derek Long, and Daniele Magazzeni. *Explainable Planning*. IJCAI Workshop on XAI, 2017.

URL <https://arxiv.org/abs/1709.10256>

As AI is increasingly being adopted into application solutions, the challenge of supporting interaction with humans is becoming more apparent. Partly this is to support integrated working styles, in which humans and intelligent systems cooperate in problem-solving, but also it is a necessary step in the process of building trust as humans migrate greater responsibility to such systems. The challenge is to find effective ways to communicate the foundations of AI-driven behaviour, when the algorithms that drive it are far from transparent to humans. In this talk we consider the opportunities that arise in AI planning, exploiting the model-based representations that form a familiar and common basis for communication with users, while acknowledging the gap between planning algorithms and human problem-solving.

4.9 Modeling

Christian Muise (IBM TJ Watson Research Center – Cambridge, US)

Speakers Roman Bartak (Charles University – Prague, CZ), Thorsten Koch (Konrad-Zuse-Zentrum – Berlin, DE), Christian Muise (IBM TJ Watson Research Center – Cambridge, US), and Scott Sanner (University of Toronto, CA)

License © Creative Commons BY 3.0 Unported license
© Christian Muise

This workshop focused on the issues in planning and OR when it comes to modeling: what representation to use, what level of abstraction to use, how does the solver technology influence the model, etc. The discussions leaders were Christian Muise, Roman Bartak, Thorsten Koch, and Scott Sanner.

First, we discussed the hierarchy of complexity that appears in both fields: e.g., moving from deterministic to probabilistic planning or moving from MILP to MINLP. The general advice is that even if the model is less clear to the user, using a less rich language can often lead to substantial improvements in the scalability of the solver technology. Knowing how to model problems effectively should thus include recognizing the appropriate level of expressivity and abstraction that can capture the problem.

We then discussed how the solving technology can influence the modeling itself. This is reflected most prominently in OR where the inclusion of additional constraints can drastically improve the performance of certain solvers. Additionally, not all solvers have the same expressivity in terms of constraints (particularly in the area of CP), and so solver technology will largely influence the modeling choices that are made.

The discussion then moved towards the parallel between problem modeling and programming languages. Just like the levels abstraction that we see when moving from assembly to C to Java or Python, we have a similar hierarchy in fields such as CP where lower-level constraints are succinctly represented by higher level ones. The longer-term hope is that we will need to rely less on modeling ingenuity and domain-specific hand-tailored heuristics, and rely more on sophisticated solvers recognizing the techniques that should be applied in the

same sense as we currently rely on the sophistication in modern compilers for programming languages.

Finally, we discussed some of the key issues that exist in traditional probabilistic planning settings, and a modern proposal to use a different style of problem modeling. Rather than use the traditional focus of planning for state change to be action-centric (modeling the impact individual actions can have), the newly introduced language focuses on what makes up the state of the world and how that may change based on the agent's choice of control. This opens the door to modeling far more complex phenomenon where the possible successors of a state may be on the order of the entire state space itself.

4.9.1 Talk: Is Modeling Important for Efficiency of Problem Solving?

Roman Bartak (Charles University – Prague, CZ)

License © Creative Commons BY 3.0 Unported license
© Roman Bartak

Joint work of Roman Barták, Lukáš Chrpá, Agostino Dovier, Jindřich Vondrážka, Neng-Fa Zhou
Main reference Roman Barták, Lukáš Chrpá, Agostino Dovier, Jindřich Vondrážka, Neng-Fa Zhou: “Modeling and solving planning problems in tabled logic programming: Experience from the Cave Diving domain”, *Sci. Comput. Program.*, Vol. 147, pp. 54–77, 2017.
URL <http://dx.doi.org/10.1016/j.scico.2017.04.007>

The talk argues for importance of problem modeling on efficiency of problem solving. By showing several examples from different areas, I argue that the formal model plays a big role in problem solving. I used the Golomb-ruler problem to demonstrate influence of symmetry breaking, redundant constraints, and search strategy on efficiency when using constraint programming. I also compared efficiency of several constraint models of classical planning problems. Next, I presented the role of state representation, heuristics, and control knowledge when solving planning problems using search (iterative deepening and branch-and-bound) in the Picat planner. Finally, I presented a comparison of various solving approaches, namely the Picat system, pure SAT-based solver, ASP, and search-based solver for multi-agent path finding. The talk was concluded by a list of questions that might serve as a future research program. Do we agree that modeling is important for efficiency of solving? Do we know what the good practices to model problems are? Are these practices available to users/modelers? Can we formalize these practices such that we can automatically reformulate models? Can we learn the models automatically?

4.10 Cross-Domain Example Problems

Florian Pommerening (Universität Basel, CH)

Speakers Michael Cashmore (King's College London, GB), Thorsten Koch (Konrad-Zuse-Zentrum – Berlin, DE), and Louis-Martin Rousseau (Polytechnique Montreal, CA)
License © Creative Commons BY 3.0 Unported license
© Florian Pommerening

In a panel discussion with Michael Cashmore, Thorsten Koch, and Louis-Martin Rousseau we talked about the differences and commonalities of AI planning, Constraint Programming (CP), and Mixed Integer Programming (MIP). A main goal of the discussion was to find properties in the structure of problems that make them specifically suitable for one of these approaches.

One of the features that is specific to planning is its focus on sequential aspects. A* search can find sequential solution without knowing a limit on the horizon beforehand, while MIP

and CP often require unrolling or resolving with different horizons for sequential solutions. Planning problems can also handle state-specific constraints or costs more naturally than the other approaches. In contrast to scheduling with MIP or CP which *orders* activities, planning *chooses and orders* activities. This means that the solution of a planning problem is a path, not just the goal state at the end of that path. Therefore, search strategies can be used that do not have a match in the MIP or CP world, like searching backwards from a solution. Puzzles like the 4-peg towers of Hanoi, Freecell, or Rubik's cube are simple examples that exhibit strong sequential aspects and state-dependent constraints. As more real-world examples, intrusion detection, software verification, and the tactical level of protocol verification were proposed as examples.

MIP and CP on the other hand offer a global view on the problem. In contrast to planning where the focus is on local effects along a transition from one state to another, the focus for MIP and CP is on the whole problem with global constraints. One key difference between the MIP and CP communities is how constraints are used to exclude invalid assignments: while MIP algorithms approach the solution space from “outside” (e.g., using cutting planes), CP algorithms can also exclude assignments from the “inside” of the space (e.g., by removing values from a domain).

Another difference between the three areas that was brought up is the different value of domain-independent solutions. While planning research almost exclusively focuses on domain-independent techniques – to the degree that solving a specific problem is not considered planning – MIP models typically contain a lot of problem-specific optimizations. Likewise, finding a good representation for a given problem has a low priority in planning research, whereas it is a main focus of CP and MIP. Using domain-independent solutions can slow down planning on specific problems but allows rapid prototyping and transfer of solution techniques. A final implementation can then be a specialization of such a prototype that is a lot faster but with a parallel performance curve. On the MIP side, an out-of-the-box implementation of general algorithms (e.g., Benders decomposition) is missing and algorithms have to be specialized for each new problem.

In the discussion about typical problems for CP, most suggestions were highly discrete and disjunctive problems like n -queens, Crossword puzzles, or Sudoku. But also complicated scheduling problems were mentioned where state-dependent constraints can be more easily handled by CP than by MIP. In contrast to planning, CP seems better suited for problems without a temporal aspect (or with a fixed horizon) like graph coloring. Interestingly, Sudoku can be solved quite efficiently by MIP and the reason for this is not perfectly clear. MIP solvers have an advantage here because they use CP solvers in their pre-solve step which will handle all cases that are well-suited for CP and leave only the hard combinatorial part for the core MIP solver. It is common that a large part of the work is already done in this pre-solve step.

In contrast to the mostly discrete suggestions for problems fitting CP and planning, MIP easily deals with continuous problems. Min-cost flow problems, for example, would likely be best solved with MIP. In fact only LP is needed to solve them. However, adding state-dependent restrictions can make a min-flow problem harder for MIP again and might require a combination of techniques.


In real-world applications like many robotics applications, the problem is often a mix of many subproblems that might require different solvers. A common approach is to decompose such problems and solve each subproblem with a specialized solver. Finding the right tool for each subproblem is currently still a task that requires expert knowledge so so knowing more about the strengths and weaknesses of each tool would be a big help.

One common theme in all areas is that they solve problems by exploiting structure contained in the problem and each solution method exploits a different kind of structure. Instead of finding a typical planning, MIP, or CP problem we can thus try to describe what kind of structure each method can exploit. A question that was brought up was if we can build a hierarchy of structures that can be added to or removed from problems where the best solution method changes with each added kind of structure. This would require modeling a problem (e.g., vehicle routing) with many variants in different frameworks. Compiling between the frameworks would also work but may lead to bad models. However, something could be learned from comparing the naively compiled models to hand-crafted models for the same problem as well.

4.11 Dealing with Uncertainty

Florian Pommerening (Universität Basel, CH)

Speakers Michele Lombardi (University of Bologna, IT), Matthijs Spaan (TU Delft, NL), and Scott Sanner (University of Toronto, CA)

License  Creative Commons BY 3.0 Unported license
© Florian Pommerening

We discussed ways of dealing with uncertainty in planning and optimization. Michele Lombardi presented work that handles uncertainty by creating robust partial order schedules. This was followed by a presentation of Matthijs Spaan and a implementation technique suggested by Scott Sanner.

4.11.1 Talk: On the Robustness of Partial Order Schedules

Michele Lombardi

License © Creative Commons BY 3.0 Unported license
© Michele Lombardi

Joint work of Michele Lombardi, Alessio Bonfietti, Michela Milano

Main reference Michele Lombardi, Alessio Bonfietti, Michela Milano: “Deterministic Estimation of the Expected Makespan of a POS Under Duration Uncertainty”, in Proc. of the Principles and Practice of Constraint Programming – 21st Int’l Conference, CP 2015, Cork, Ireland, August 31 – September 4, 2015, Proceedings, Lecture Notes in Computer Science, Vol. 9255, pp. 279–294, Springer, 2015.

URL http://dx.doi.org/10.1007/978-3-319-23219-5_20

Partial Order Schedules (POSs) are a convenient method for representing solutions to scheduling problems with end-to-start precedence constraints and limited resources. Specifically, a POS is a directed acyclic graph that contains the original precedence relations, augmented with extra arcs that prevent the occurrence of resource conflicts. POSs are naturally robust against duration uncertainty, since they allow recomputing a feasible schedule in polynomial time. A POS can be obtained very efficiently from a classical, fixed-start, schedule. In this talk, we present empirical results that show how, for a large variety of POSs obtained from multiple scheduling problems, there exists a very strong linear correlation between the expected value of the makespan and the makespan obtained by fixing all durations to their expected value. The correlation is strong enough that optimizing the makespan with fixed durations is approximately equivalent to optimizing the expected makespan. The result has immediate practical applications, since the former problem is computationally much simpler than the latter – assuming that expected value optimization is appropriate (e.g. for activity sets that must be repeated multiple times). In case of statistically independent durations, the behavior is due to the fact that interference from multiple paths and variance compensation on sequences of activities tend to cancel each other. In case of statistically dependent durations, formal arguments show that the correlation can be much weaker, although this may require quite unrealistic probability distributions. When more realistic scenarios are empirically evaluated, interestingly, the correlation with statistically dependent durations is often at least as strong as with independent durations. As a secondary result, the talk contains a proof sketch that a tight upper bound on the expected makespan can always be obtained by taking into account at most $n + 1$ scenarios in the sampling space, where n is the number of activities.

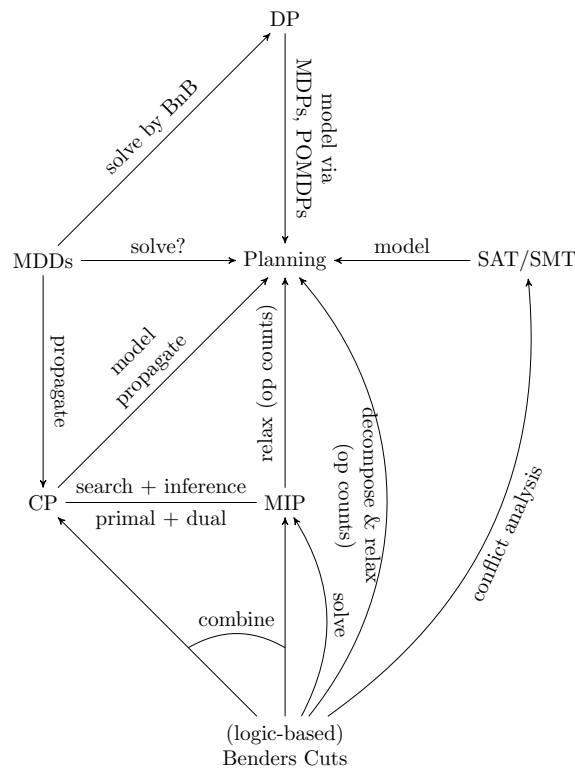
4.12 Connections Between our Fields

Mark Roberts (Naval Research – Washington, US)

Speakers Christina N. Burt (Satalia – London, GB), John N. Hooker (Carnegie Mellon University – Pittsburgh, US), Christian Muise (IBM TJ Watson Research Center – Cambridge, US), and Florian Pommerening (Universität Basel, CH)

License © Creative Commons BY 3.0 Unported license
© Mark Roberts

During this session, several speakers spoke about interrelationships between CP, Planning, and Optimization. John Hooker started with a diagram that highlighted the key linkages discussed during the week. Figure 3 illustrates the key connections. CP can be used in planning for modeling and propagation. MIP can be used for operator counts in planning; MIP and CP have relationships relating to search and inference. Bender’s cuts can be used in many applications for solving, conflict analysis, and operator counts. Bender’s cuts can

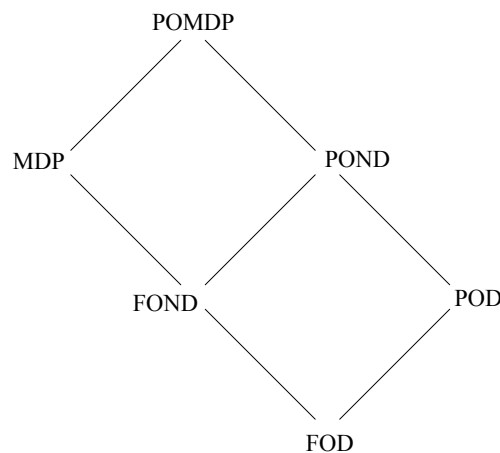


■ **Figure 3** Interrelationships between CP, Planning, and Optimization.

also be used to decompose or relax planning problems. SAT and SMT can be used to model and solve planning problems. MDDs can be used to solve planning as well as for propagation in CP. MDDs can also be used to solve, via Branch and Bound, Dynamic Programming problems. DP can be used, via MDPs and POMDPs, to model and solve planning problems. Finally, planning can provide factored or structured representations for other techniques. During discussion, several issues were raised. One person noted that a central question related to pruning techniques, bounding, and solution set sizes. Another person noted the role of lower bounds under relaxations.

Florian Pommerening spoke about the the use of action or fact landmarks in planning. These serve a similar function to redundant constraints in CP. In terms of complexity, there can be exponentially many of these, and verifying any arbitrary one is as hard as planning. Some ideas that surfaced during the discussion related to the use of dead-end detection and optimization-based landmarks. It was also noted how landmarks relate to reductions in MIP. Specifically, knowing that an operator (or a fact) is a landmark is similar to knowing that a certain LP variable must take a certain value and can be fixed. For disjunctive action landmarks, the exact value of an LP variable may not be known, but knowing some information about it corresponds to a cut.

Christian Muise spoke about the relationship of determinism and observability in planning to various topics from the week. Figure 4 illustrated the relationship of POMDP, MDP, FOND, POND, POD, FOD. Along one axis, observability can be fully observable (FO) or partially observable (PO). On the other axis, determinism is varied from deterministic (D), non-deterministic (ND), and the Markovian Decision Problem (MDP). These types of planning relate to topics from the week, which included irreducible infeasible subsets in



■ **Figure 4** Relationship between different planning formalisms.

MIP, landmarks, strengthening benders cuts in CP, quick explain in CP, and multi-agent planning under uncertainty. Importantly, landmarks can play a central role in several of these techniques.

4.13 LP Heuristics: The Details

Gabriele Röger (Universität Basel, CH)

Speakers Florian Pommerening (Universität Basel, CH) and Malte Helmert (Universität Basel, CH)

License © Creative Commons BY 3.0 Unported license

© Gabriele Röger

The aim of this session was to understand LP-based cost-partitioning from an Operations Research perspective and to identify ways of improving the technique with common OR wisdom.

The core idea of cost partitioning in planning (using an additive cost function) is to distribute the action costs among several copies of the planning instance so that the sum of lower bounds on solution costs still gives a lower bound for the original problem. An optimal cost partitioning can be computed with linear programming. Florian Pommerening showed the structure of the LPs in a primal and a dual formulation. These show that reasoning about plans in the operator counting framework can also be cast as reasoning about operator costs (if allowing negative costs). The constraints in the operator counting framework have a declarative nature, describing necessary properties of all plans. We briefly spoke about how these constraints can be derived, which lead to the question whether it would be possible to discover landmarks with constraint optimization techniques and whether there is an analogy between landmarks and backbones. Also, is it possible to use linear approximations of heuristics that cannot directly be expressed as an LP?

We also wondered about the role of negative costs. Why are they possible, why are they necessary? Negative cost are used for extracting value of aspects that another relaxation ignores. From an intuitive point of view this seems to be similar to amortized cost analysis.

It seems that Lagrangian decomposition could improve the heuristic estimates. Would it also be possible to identify independent macro actions to get a faster computation and better values with flow constraints?

How does all this relate to things that go on in MIP/CP? How large is the gap to the (unrelaxed) MIP values? For landmark constraints, in the cases where we can solve the IP, the gap is often 0. It is also low for other heuristics that were tried.

Potential heuristic avoid re-computation for every state. To avoid an extreme point (as one gets it by simplex), one can use the interior point method. Writing out the properties of the splitting might help.

We also discussed the questions whether the actual partitioning is important or only the value, and what other information can we derive apart from lower bounds.


Some sets of constraints formulate shortest-path problems. How exactly do these look like? The combination is a minimax problem (maximizing the sum of minimal cost paths), formulated in one LP. To see this as a MIP we need to consider its dual.

Overall, we spend a significant amount of time discussing what we were actually talking about because the two communities do not share the same notion for highly related concepts. There seem to be close relationships to some MIP techniques but more detailed discussion is needed to match them. The most promising direction seems to be exploiting substructures and the cost of these substructures.

Although we did not find final answers to many of the questions, the session identified many interesting connecting factors for future research.

4.14 Planning, Optimization, and Machine Learning

Scott Sanner (University of Toronto, CA)

Speakers Serdar Kadioglu (Fidelity Investments – Boston, US), Michele Lombardi (University of Bologna, IT), Christian Muiße (IBM TJ Watson Research Center – Cambridge, US), and Scott Sanner (University of Toronto, CA)
License  Creative Commons BY 3.0 Unported license
 © Scott Sanner

Michele Lombardi began the session with a discussion of how to optimize traffic light placement. He proposed learning a model $f(x)$ from data and then embedding the constraint $y = f(x)$ in the optimization model. The key question though is how to embed the constraint $y = f(x)$ in optimization models, when $f(x)$ is not in a convenient form such as a linear function but rather a complex empirically learned model like a neural network.

Michele mentioned various solutions to this problem depending on the type of optimization model. For the case of Constraint Programming, Michele suggested that intervals could be propagated through the neural net in both directions, or alternately a Lagrangian relaxation could be used to add the neural network constraints in the optimization objective [1].

However, if one is only verifying a property (constraint satisfaction) of the neural network for all inputs and the neural network uses only linear and rectified linear unit (ReLU) activation functions, one can use a modification of the simplex algorithm for ReLU activations termed ReLUpex [2]. There is also related work on verifying properties of neural networks by finding counterexamples via SMT [3].

Beyond embedding neural networks in optimization models, Michele mentioned that decision trees are fairly easily directly embedded in a variety of models.

Some final discussion centered on the problem of the “optimizer’s curse” when embedding learned models in optimization models, namely the problem that if the learned model makes significant errors (often in input regions where the data for learning is sparse), the optimizer will exploit these errors (e.g., large extrapolations that lead to large objective values when maximizing). That is, when using the learned model for prediction, one can expect the

prediction errors to be randomly distributed, but due to the optimizer's curse, the optimizer will zero in on precisely the prediction errors that lead to erroneous, but favorable objective values that are practically unachievable.

Scott Sanner presented next on the topic of planning in learned models. He started with an example of using neural networks to learn a deterministic transition model from data. If one then wants to plan in this model using a Mixed Integer Linear Program (MILP) with neural network constraints or objective and the neural network is trained with rectified linear unit (ReLU) activation functions, the neural network definition can be encoded directly as MILP constraints with additional redundant constraints to strengthen the LP relaxation [4]. However, Scott also pointed out that if we can learn neural networks with rectified linear unit (ReLU) activation functions then freezing the weights and optimizing the network activations themselves is a symmetric problem to learning, and furthermore the optimization problem for control is often only constrained with simple action bound constraints. Hence Scott suggested that Tensorflow with projected gradient descent (to project all action values to their nearest bound) could be used as the optimizer of actions for such control problems. He also remarked that similarly to the Tensorflow planning model of Wu et al. [5] using RMSProp as the optimizer with deep net learned transition models performed very well in these highly non-convex control optimization problems.

Finally, Serdar Kadioglu presented on the topic of learning for SAT solver selection. He started with the question of what the best solver for every instance is, and whether it always is the same. The answer was that the best solver per instance is almost never the best solver overall! In this case he proposed the problem of how to learn which solver to use on each instance. To address this problem, he suggested following standard methodology in machine learning to derive features of problem instances followed by a PCA projection of these features. He then suggested classifying or clustering instances according to their features should yield classifications or clusters of instances with similar solver behavior. Further, Michele suggested that beyond simple instance-specific selection of problem solvers, one could embed learning into different stages of a backtracking solver as done by Di Liberto et al. [6].

Serdar then went on to discuss machine learning for optimization and specifically for selecting rules. An example of rule learning was given for the purpose of scheduling jobs on a single machine. For example, Serdar mentioned that one could use learned rules to select different scheduling methods on the first job and remaining jobs.

Christian Muise spoke last to follow up on the first topic from Serdar's talk. Christian discussed the issue that occurs when no features are available in branching optimizers (such as for SAT). Chris Beck and colleagues worked on learning search rules and techniques that were dependent on the search depth with some very positive results [7].

References

- 1 Andrea Bartolini, Michele Lombardi, Michela Milano, and Luca Benini. *Neuron constraints to model complex real-world problems*. In Proceedings of the Seventeenth International Conference on Principles and Practice of Constraint Programming, pp. 115–129, 2011.
- 2 Guy Katz, Clark Barrett, David Dill, Kyle Julian, and Mykel Kochenderfer. *Reluplex: An efficient SMT solver for verifying deep neural networks*. In Proceedings of the Twenty-Ninth International Conference on Computer Aided Verification, pp. 97–117, 2017.
- 3 Xiaowei Huang, Marta Kwiatkowska, Sen Wang, and Min Wu. *Safety verification of deep neural networks*. In Proceedings of the Twenty-Ninth International Conference on Computer Aided Verification, pp. 3–29, 2017.

- 4 Buser Say, Ga Wu, Yu Qing Zhou, and Scott Sanner. *Nonlinear hybrid planning with deep net learned transition models and mixed-integer linear programming*. In Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence, pp. 750–756, 2017.
- 5 Ga Wu, Buser Say, and Scott Sanner. *Scalable Planning with Tensorflow for Hybrid Non-linear Domains*. In Advances in Neural Information Processing Systems 30, pp. 6276–6286, 2017.
- 6 Giovanni Di Liberto, Serdar Kadioglu, Kevin Leo, and Yuri Malitsky. *DASH: Dynamic approach for switching heuristics*. European Journal of Operational Research 248(3), pp. 943–953, 2016.
- 7 Tom Carchrae and J. Christopher Beck. *Applying Machine Learning to Low-Knowledge Control of Optimization Algorithms*. Journal of Computational Intelligence 21(4), pp. 372–387, 2005.

4.14.1 Talk: Empirical Model Learning

Michele Lombardi

License © Creative Commons BY 3.0 Unported license

© Michele Lombardi

Joint work of Michele Lombardi, Andrea Micheli, Alessio Bonfietti, Andrea Bartolini, Tias Guns, Pascal Van Hentenryck, Michela Milano

Main reference Michele Lombardi, Michela Milano, Andrea Bartolini: “Empirical decision model learning”, Artif. Intell., Vol. 244, pp. 343–367, 2017.

URL <http://dx.doi.org/10.1016/j.artint.2016.01.005>

This talk provided an overview of Empirical Model Learning (EML), a method to tackle optimization problems defined over complex systems by extracting an approximate model via Machine Learning, and then embedding it into a classical combinatorial optimization model. The talk focused on the case of embedding Neural Networks and Decision Trees, respectively in Constraint Programming and SAT Modulo Theories. As a case study, we considered a workload allocation problem on a many-core CPU, for which a thermal model cannot be compactly expressed in a declarative fashion. The talk presented the basic ideas of EML, and highlighted open questions.

4.15 Multiagent Planning and Optimization

Matthijs Spaan (TU Delft, NL)

Speakers Roman Bartak (Charles University – Prague, CZ), Adi Botea (IBM Research – Dublin, IE), and Mathijs de Weerd (TU Delft, NL)

License © Creative Commons BY 3.0 Unported license

© Matthijs Spaan

During this session, we discussed particular aspects of planning and optimization that arise in the context of multiagent systems.

First, Mathijs de Weerd spoke about multi-party optimization, which refers to the problem of optimizing the decision making of multiple stakeholders whose objectives are not aligned. For instance, we can consider the interactions of multiple companies or different departments within a single company. Striving for economic efficiency either leads to mergers of companies or to optimization across the boundaries of organizations. Three specific challenges were introduced. First, it is important to deal with preferences, which relates to social choice theory. A way forward could be to study voting or aggregation rules in the context of optimization. Second, participants might demonstrate strategic behaviour,

which can be addressed by using Groves mechanisms. Third, it is not yet clear how to deal with incomplete information and local autonomy. Here, decomposition techniques could be promising.

Next, Adi Botea presented recent work on multiagent path planning for strongly biconnected directed graphs. In this problem, a set of agents have to move from an initial configuration to a goal configuration. For the particular type of graphs, an open ear decomposition is used. If the graph satisfies certain conditions, then every instance with at least two blanks has a solution. For the latter case, a complete algorithm is presented. Future work is to generalize to other classes of directed graphs and to explore optimal solving with an optimization-based approach.

Finally, Roman Bartak spoke about modeling and solving the multiagent path finding problem in the Picat language. The problem can be solved using state-space or conflict-based search techniques, but the talk focused on compilation approaches. The abstract model is based on a layered graph describing the agent positions at each time step and the objective function can be either the makespan or the sum of costs (minimize the sum of end times). The problem can be represented in the Picat language, which allows you to solve it using SAT, MIP or CP. Picat makes it easy to swap solvers or adapt the model and for this problem, the efficiency is comparable to the state of the art.

4.15.1 Talk: Multi-Party Optimization

Mathijs de Weerd (TU Delft, NL)

License © Creative Commons BY 3.0 Unported license
© Mathijs de Weerd

This talk introduced and motivated the importance of optimization problems across organizational boundaries. This brings interesting new challenges for optimization. A few known optimization techniques (like decomposition) that may be useful in this context were indicated, and some relevant concepts from game theory highlighted.

References

- 1 Frits de Nijs, Matthijs T. J. Spaan, and Mathijs de Weerd. *Best-Response Planning of Thermostatically Controlled Loads under Power Constraints*. In Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence. 2015.
- 2 Frits de Nijs, Erwin Walraven, Mathijs de Weerd, and Matthijs T. J. Spaan. *Bounding the Probability of Resource Constraint Violations in Multi-Agent MDPs*. In Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence. 2017.
- 3 Mathijs de Weerd. *Plan Merging in Multi-Agent Systems*. PhD Thesis, Delft University of Technology, Delft, The Netherlands. 2003.
- 4 Mathijs de Weerd, Sebastian Stein, Enrico H. Gerding, Valentin Robu, and Nicholas R. Jennings. *Intention-Aware Routing of Electric Vehicles*. IEEE Transactions on Intelligent Transportation Systems, 17(5) pp.1472–1482, 2016.
- 5 Joris Scharpf, Matthijs T. J. Spaan, Leentje Volker, and Mathijs de Weerd. *Planning under Uncertainty for Coordinating Infrastructural Maintenance*. In Proceedings of the Twenty-Third International Conference on Automated Planning and Scheduling, pp. 425–433, 2013.

4.15.2 Talk: Multi-agent path planning on strongly biconnected digraphs

Adi Botea (IBM Research – Dublin, IE)

License © Creative Commons BY 3.0 Unported license
© Adi Botea

Joint work of Adi Botea, Davide Bonusi, Pavel Surynek

Main reference Adi Botea, Pavel Surynek: “Multi-Agent Path Finding on Strongly Biconnected Digraphs”, in Proc. of the Twenty-Ninth AAAI Conference on Artificial Intelligence, January 25-30, 2015, Austin, Texas, USA., pp. 2024–2030, AAAI Press, 2015.

URL <http://www.aaai.org/ocs/index.php/AAAI/AAAI15/paper/view/9653>

In a common formulation of the problem of multi-agent path planning, a set of mobile agents share an environment represented as a graph. The time is discretized. At a given time, each agent is located at one graph vertex, under the constraint that two or more agents cannot share the same vertex at the same time. An agent located at a vertex a at the time t can move to an adjacent vertex b at time the $t + 1$, provided that no other agent will be at the vertex b at the time $t + 1$. An additional common condition is that the vertex b is free at time t as well. Several agents can move in parallel at a given time. The task is to compute a set of moves that will take the agents from their initial configuration to a goal configuration.

Many sub-optimal, polynomial-time algorithms are designed for undirected graphs, where the movement is allowed in both directions along an edge. Yet, directional constraints (e.g., allowing moves only along one direction of an edge) are relevant in several domains, including games and asymmetric communication networks.

We presented an approach to multi-agent path planning on strongly biconnected directed graphs. A directed graph is strongly biconnected if: it is strongly connected; and the undirected digraph obtained by ignoring the directions of the edges is biconnected.

In our work, we show that all instances with at least two unoccupied positions have a solution, except for a particular, degenerate sub-class where the graph has a cyclic shape. Our algorithm, called diBOX, runs in polynomial time, computes suboptimal solutions and is complete for instances on strongly biconnected digraphs with at least two unoccupied positions.

To our knowledge, our work is the first study of multi-agent path finding focused on directed graphs.

4.15.3 Talk: Modeling and Solving the Multi-Agent Pathfinding Problem in Picat

Roman Barták (Charles University – Prague, CZ)

License © Creative Commons BY 3.0 Unported license
© Roman Barták

Joint work of Roman Barták, Neng-Fa Zhou, Roni Stern, Eli Boyarski, Pavel Surynek

Main reference Roman Barták, Neng-Fa Zhou, Roni Stern, Eli Boyarski, Pavel Surynek: “Modeling and Solving the Multi-agent Pathfinding Problem in Picat”, in Proc. of the 29th IEEE Int’l Conference on Tools with Artificial Intelligence, ICTAI 2017, Boston, MA, USA, November 6-8, 2017, pp. 959–966, IEEE Computer Society, 2017.

URL <http://dx.doi.org/10.1109/ICTAI.2017.00147>

The multi-agent pathfinding (MAPF) problem has attracted considerable attention because of its relation to practical applications. We presented a constraint-based declarative model for MAPF, together with its implementation in Picat, a logic-based programming language. We showed experimentally that our Picat-based implementation is highly competitive and sometimes outperforms previous approaches. Importantly, the proposed Picat implementation is very versatile. We demonstrated this by showing how it can be easily adapted to optimize different MAPF objectives, such as minimizing makespan or minimizing the sum of costs, and

for a range of MAPF variants. Moreover, a Picat-based model can be automatically compiled to several general-purpose solvers such as SAT solvers and Mixed Integer Programming solvers (MIP). This is particularly important for MAPF because some MAPF variants are solved more efficiently when compiled to SAT while other variants are solved more efficiently when compiled to MIP. We analyzed these differences and the impact of different declarative models and encodings on empirical performance.

4.16 Exploiting Substructures

Charlotte Truchet (University of Nantes, FR)

Speakers Malte Helmert (Universität Basel, CH), and Gilles Pesant (École Polytechnique – Montréal, CA),
Domenico Salvagnin (University of Padova, IT)
License © Creative Commons BY 3.0 Unported license
© Charlotte Truchet

This session had three presentations: Gilles Pesant on “Counting in individual substructures”, Domenico Salvagnin on “Global structures in MIP” and Malte Helmert.

The first presentation introduced different techniques for solution counting on global constraints, which represent a specific substructure of a combinatorial problem: based on a compact representation (e.g. regular constraint), based on sampling (e.g. all different constraint), based on domain relaxation (e.g. knapsack constraint) or based on theoretical results. There were several questions, with a discussion focused on ways to deal with the #P complexity of these problems in general.

The second presentation reviewed different ways of handling global structures in MIP problems, where the flat model is based on very simple primitives (linear and integrity constraints). The need for higher level representations was identified, either by extending the language expressivity (as in CP) or by doing reverse engineering on a flat model. Several global substructures like LP relaxation, symmetry groups, and graphs were presented. The audience had many questions and the discussions focused on automatic modeling, reified constraints, efficiency of global constraints, and the question whether the user should chose the substructures or they should be discovered by an educated guess by the solver.

The third presentation detailed an example in planning where global structures needed to be identified, based on two state variables and an implicit transition graph that is induced by domain transition graphs for each variable. What can be done on this representation was first presented by the speaker then discussed with the audience: apart from solving the the problem, properties of solutions can be extracted (e.g. mandatory actions); bounds on the number of times an operator is used in a solution can be used to extract information from the domain transition graphs; marginals of the actions in one of the automata can be computed; and landmarks can be discovered automatically.

4.16.1 Talk: Counting in Individual Substructures

Gilles Pesant (École Polytechnique – Montréal, CA)

License © Creative Commons BY 3.0 Unported license
© Gilles Pesant

Main reference Gilles Pesant: “Getting More Out of the Exposed Structure in Constraint Programming Models of Combinatorial Problems”, in Proc. of the Thirty-First AAAI Conference on Artificial Intelligence, February 4-9, 2017, San Francisco, California, USA., pp. 4846–4851, AAAI Press, 2017.

URL <http://aaai.org/ocs/index.php/AAAI/AAAI17/paper/view/14982>

To solve combinatorial problems, Constraint Programming builds high-level models that expose much of the structure of the problem. The distinctive driving force of Constraint Programming has been this direct access to problem structure. This has been key to the design of powerful inference algorithms and branching heuristics. In particular, the ability to evaluate the number of solutions in such structures helps reduce the combinatorial search space, guide its exploration, but also has applications in model counting, uniform sampling, structure elicitation, and possibly planning.

This talk presented some of the algorithmic techniques used so far in order to perform solution counting on several important combinatorial substructures.

4.16.2 Talk: Global Structures in MIP

Domenico Salvagnin (University of Padova, IT)

License © Creative Commons BY 3.0 Unported license
© Domenico Salvagnin

We surveyed the global structures that are automatically constructed and exploited by MIP solvers to improve performance.

Participants

- Roman Bartak
Charles University – Prague, CZ
- J. Christopher Beck
University of Toronto, CA
- Adi Botea
IBM Research – Dublin, IE
- Christina N. Burt
Satalia – London, GB
- Hadrien Cambazard
Grenoble INP, FR
- Michael Cashmore
King's College London, GB
- Alessandro Cimatti
Bruno Kessler Foundation – Trento, IT
- Mathijs de Weerd
TU Delft, NL
- Jeremy D. Frank
NASA – Moffett Field, US
- Patrik Haslum
Australian National University, AU
- Emmanuel Hebrard
LAAS – Toulouse, FR
- Malte Helmert
Universität Basel, CH
- John N. Hooker
Carnegie Mellon University – Pittsburgh, US
- Serdar Kadioglu
Fidelity Investments – Boston, US
- Michael Katz
IBM TJ Watson Research Center – Yorktown Heights, US
- Thorsten Koch
Konrad-Zuse-Zentrum – Berlin, DE
- Sven Koenig
USC – Los Angeles, US
- Michele Lombardi
University of Bologna, IT
- Daniele Magazzeni
King's College London, GB
- Andrea Micheli
Bruno Kessler Foundation – Trento, IT
- Christian Muise
IBM TJ Watson Research Center – Cambridge, US
- Eva Onaandia
Technical University of Valencia, ES
- Gilles Pesant
École Polytechnique – Montréal, CA
- Chiara Piacentini
University of Toronto, CA
- Nicola Policella
Solenix – Darmstadt, DE
- Florian Pommerening
Universität Basel, CH
- Mark Roberts
Naval Research – Washington, US
- Gabriele Röger
Universität Basel, CH
- Louis-Martin Rousseau
Polytechnique Montreal, CA
- Hana Rudová
Masaryk University – Brno, CZ
- Domenico Salvagnin
University of Padova, IT
- Scott Sanner
University of Toronto, CA
- Pierre Schaus
UC Louvain, BE
- Matthijs Spaan
TU Delft, NL
- Charlotte Truchet
University of Nantes, FR
- Willem-Jan Van Hoeve
Carnegie Mellon University – Pittsburgh, US
- Parisa Zehtabi
King's College London, GB



Designing and Implementing Algorithms for Mixed-Integer Nonlinear Optimization

Edited by

Pierre Bonami¹, Ambros M. Gleixner², Jeff Linderoth³, and
Ruth Misener⁴

- 1 IBM Spain – Madrid, ES, pierre.bonami@es.ibm.com
- 2 Konrad-Zuse-Zentrum – Berlin, DE, gleixner@zib.de
- 3 University of Wisconsin – Madison, US, linderoth@wisc.edu
- 4 Imperial College London, GB, r.misener@imperial.ac.uk

Abstract

Mathematical models for optimal decisions often require both nonlinear and discrete components. These mixed-integer nonlinear programs (MINLP) may be used to optimize the energy use of large industrial plants, integrate renewable sources into energy networks, design biological and biomedical systems, and address numerous other applications of societal importance. The first MINLP algorithms and software were designed by application engineers. While these efforts initially proved useful, scientists, engineers, and practitioners have realized that a transformational shift in technology will be required for MINLP to achieve its full potential. MINLP has transitioned to a forefront position in computer science, with researchers actively developing MINLP theory, algorithms, and implementations. Even with their concerted effort, algorithms and available software are often unable to solve practically-sized instances of these important models. Current obstacles include characterizing the computability boundary, effectively exploiting known optimization technologies for specialized classes of MINLP, and effectively using logical formulas holistically throughout algorithms.

Seminar February 18–23, 2018 – <https://www.dagstuhl.de/18081>

2012 ACM Subject Classification Information systems → Data structures

Keywords and phrases Complexity, Mathematical optimization, Mathematical software, Mixed-integer optimization, Nonlinear optimization, Numerical issues, Optimization algorithms

Digital Object Identifier 10.4230/DagRep.8.2.64

Edited in cooperation with Radu Baltean-Lugojan


1 Executive Summary

Pierre Bonami (IBM Spain – Madrid, ES)

Ambros M. Gleixner (Konrad-Zuse-Zentrum – Berlin, DE)

Jeff Linderoth (University of Wisconsin – Madison, US)

Ruth Misener (Imperial College London, GB)

License  Creative Commons BY 3.0 Unported license

© Pierre Bonami, Ambros M. Gleixner, Jeff Linderoth and Ruth Misener

This workshop aimed to address this mismatch between natural optimization models for important scientific problems and practical optimization solvers for their solution. By bringing together experts in both theory and implementation, this workshop energized efforts making MINLP as ubiquitous a paradigm for both modeling and solving important decision problems



Except where otherwise noted, content of this report is licensed under a Creative Commons BY 3.0 Unported license

Designing and Implementing Algorithms for Mixed-Integer Nonlinear Optimization, *Dagstuhl Reports*, Vol. 8, Issue 02, pp. 64–87

Editors: Pierre Bonami, Ambros M. Gleixner, Jeff Linderoth, and Ruth Misener



Dagstuhl Reports

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

as mixed-integer linear programming (MIP) and nonlinear programming (NLP) have become in recent years. In particular, we highlighted:

- **MINLP Solver Software** Early in the workshop, the main developers of MINLP software packages outlined the current state of their software. This served as a needs analysis for the community to identify crucial areas for future development. We also dedicated a break-out session discussing best practices for conducting scientifically-meaningful computational experiments in MINLP.
- **Intersecting Mixed-Integer & Nonlinear Programming** MINLP is a superset of both mixed integer linear optimization and nonlinear optimization, so we leveraged the best methods from both by incorporating both sets of experts.
- **Driving Applications** Applications experts, e.g. in petrochemicals, manufacturing, and gas networks, offered their perspectives on what practitioners need from MINLP solvers. We dedicated an entire break-out session to energy applications and explored what are the needs for MINLP within the energy domain. During the open problem session, several other applications experts outlined other open problems in engineering.
- **Connections between MINLP and machine learning** Many machine learning challenges can be formulated as MINLP. Also, machine learning can significantly improve MINLP solver software. We explored these connections at length in a break-out session.

This seminar brought together an assortment of computer scientists with expertise in mathematical optimization. Many of the presentations were more theoretical and suggested new technologies that the solver software could incorporate. Other presentations were more practical and discussed building solver software or applying that software to specific domain applications.

As a result of this seminar, we are planning a special issue in the journal “Optimization & Engineering”. We are also working to turn the notes from our open problem session into a larger document that will start a conversation with the entire mathematical optimisation community. Participants broadly expressed that this week at Dagstuhl helped them workshop their papers, so several academic papers will explicitly mention the Dagstuhl seminar. Finally, a new set of metrics for comparing MINLP solvers were developed at this meeting and will greatly aid future solver testing.

2 Table of Contents

Executive Summary

Pierre Bonami, Ambros M. Gleixner, Jeff Linderoth and Ruth Misener 64

Overview of Talks

A deterministic global optimisation algorithm for mixed-integer nonlinear bilevel programs <i>Claire Adjiman</i>	69
Strengthened Semidefinite Relaxations for Quadratic Optimization with Switching Variables <i>Kurt M. Anstreicher</i>	69
Online generation via offline selection of strong linear cuts from QP SDP relaxation <i>Radu Baltean-Lugojan and Ruth Misener</i>	69
Finding feasible solutions in MINLP problems <i>Pietro Belotti</i>	70
Incorporating Quadratic Approximations in the Outer-Approximation Method for Convex MINLP <i>David Bernal Neira and Ignacio Grossmann</i>	70
Numerical Challenges in MINLP solvers <i>Timo Berthold</i>	71
Optimization using both models and input-output data <i>Fani Boukouvala</i>	71
SMT-based mixed non-linear optimization <i>Andrea Callia D'Iddio</i>	72
On some hard quadratic unconstrained boolean optimization problems <i>Sanjeeb Dash</i>	72
Subset selection with sparse matrices <i>Alberto Del Pia, Santanu Dey, and Robert Weismantel</i>	73
Robust Treatment of Non-Convex Optimization Problems with Application to Gas Networks <i>Denis Aßmann (FAU), Frauke Liers (FAU), Juan Vera (Tilburg), and Michael Stingl (FAU)</i>	73
New SOCP relaxation and branching rule for bipartite bilinear programs <i>Santanu Dey</i>	74
Deep Learning and Mixed Integer Optimization <i>Matteo Fischetti</i>	74
Improved quadratic cuts for convex mixed-integer nonlinear programs <i>Ignacio Grossmann</i>	75
Binary Extended Formulations for Mixed-Integer Linear Programs <i>Oktay Gunluk, Sanjeeb Dash, and Robert Hildebrand</i>	75
Disjunctive cuts and extended formulations for bilinear functions <i>Akshay Gupta</i>	76

Semidefinite Programming Cuts in Gravity <i>Hassan Hijazi</i>	76
Stronger polyhedral relaxations for polynomial optimization problems <i>Aida Khajavirad and Alberto Del Pia</i>	77
Mixed-Integer Nonlinear Programming Applications: Pyomo Tools for Tailored Strategies <i>Carl Damon Laird</i>	77
Global solutions of MIQCPs <i>Amélie Lambert</i>	77
Virtuous smoothing and more virtuous smoothing <i>Jon Lee, Daphne Skipper, and Luze Xu</i>	78
Mixed-Integer Derivative-Free Optimization <i>Sven Leyffer</i>	78
LP and SDP for kissing numbers <i>Leo Liberti</i>	79
External Intersection Cuts <i>James Luedtke and Eli Towle</i>	79
Automatic reformulations of convex MINLPs in Minotaur <i>Ashutosh Mahajan</i>	79
Optimising with Gradient-Boosted Trees and Risk Control <i>Miten Mistry, Dimitrios Letsios, Gerhard Krennrich, Ruth Misener, and Robert M. Lee</i>	80
Using 2D Projections for Separation and Propagation of Bilinear Terms <i>Benjamin Müller, Ambros M. Gleixner, and Felipe Serrano</i>	80
CIA Decomposition <i>Sebastian Sager</i>	80
ALAMO: Machine learning from data and first principles <i>Nikolaos V. Sahinidis</i>	81
Separating over the convex hull of MINL constraints <i>Felipe Serrano</i>	81
Product convexification: A new relaxation framework for nonconvex programs <i>Mohit Tawarmalani</i>	82
Mixed-integer convex representability <i>Juan Pablo Vielma</i>	82
Experimentation with MINLP solver software <i>Stefan Vigerske and Ambros M. Gleixner</i>	83
Integer Optimal Solutions are Sparse <i>Robert Weismantel</i>	83
Combining ADAL with Factorizing the Dual to Solve SDP <i>Angelika Wiegele</i>	83
Mixed-integer conic optimization and MOSEK <i>Sven Wiese</i>	84

Working groups	
Applications in Energy	84
Sound experimentation with MINLP software	84
Connections between MINLP and Machine Learning	85
Open problems	
Mathematical Foundations	86
Computing Implications	86
Engineering Applications	86
Participants	87

3 Overview of Talks

3.1 A deterministic global optimisation algorithm for mixed-integer nonlinear bilevel programs

Claire Adjiman (Imperial College London, GB)

License © Creative Commons BY 3.0 Unported license
© Claire Adjiman

Joint work of Claire Adjiman, Remigijus Paulavicius, Polyxenia-M. Kleniati

Main reference Polyxeni-Margarita Kleniati, Claire S. Adjiman: “A generalization of the Branch-and-Sandwich algorithm: From continuous to mixed-integer nonlinear bilevel problems”, *Computers & Chemical Engineering*, Vol. 72, pp. 373–386, 2015.

URL <http://dx.doi.org/10.1016/j.compchemeng.2014.06.004>

We present Branch-and-Sandwich, an algorithm for the global solution of mixed-integer nonlinear bilevel programs, based on a branch-and-bound framework in which branching is permitted on outer and inner variables. We discuss the tree management and bounding strategies, present some examples and introduce ongoing implementation efforts and a test library.

3.2 Strengthened Semidefinite Relaxations for Quadratic Optimization with Switching Variables

Kurt M. Anstreicher (University of Iowa, US)

License © Creative Commons BY 3.0 Unported license
© Kurt M. Anstreicher

We consider indefinite quadratic optimization problems that include continuous and discrete variables of the form $F = \{(x, y) \mid 0 \leq x \leq y, y \in \{0, 1\}^n\}$. The binary y variables are often referred to as “switching” or “indicator” variables and occur frequently in applications. Our focus is to represent, or approximate, the convex hull of $\{(x, xx', y) \mid (x, y) \in F\}$ using constraints that include semidefiniteness conditions. We obtain an exact convex hull representation for $n = 2$ that also provides valid constraints that can be used to tighten semidefinite relaxations for higher n .

3.3 Online generation via offline selection of strong linear cuts from QP SDP relaxation

Radu Baltean-Lugojan (Imperial College London, GB) and Ruth Misener (Imperial College London, GB)

License © Creative Commons BY 3.0 Unported license
© Radu Baltean-Lugojan and Ruth Misener

Joint work of Radu Baltean-Lugojan, Ruth Misener, Pierre Bonami, Andrea Tramontani

Convex and in particular semidefinite relaxations (SDP) for non-convex continuous quadratic optimization can provide tighter bounds than traditional linear relaxations. However, using SDP relaxations directly in Branch&Cut is impeded by lack of warm starting and inefficiency when combined with other cut classes, i.e. the reformulation-linearization technique. We present a general framework based on machine learning for a strong linear outer-approximation

that can retain most tightness of such SDP relaxations, in the form of few strong low dimensional linear cuts selected offline. The cut selection complexity is taken offline by using a neural network estimator (trained before installing solver software) as a selection device for the strongest cuts.

3.4 Finding feasible solutions in MINLP problems

Pietro Belotti (Fair Isaac, GB)

License © Creative Commons BY 3.0 Unported license
© Pietro Belotti

Finding feasible solutions in MINLPs is helpful beyond providing a sensible result to the solve process. In MINLP, especially the nonconvex variant, a good cutoff is of double usefulness in finding a global optimum as it reduces the tree search but also, most importantly, reduces variables bounds, which in turn results in a better linearization and hence a tighter lower bound.

We present two heuristics for MINLP: one is an ancient MINLP heuristic implemented in the very first version of Couenne, which is a probing-based rounding algorithm. The second is a heuristic to find a solution to a NLP problem that is currently used in the Xpress-SLP solver.

3.5 Incorporating Quadratic Approximations in the Outer-Approximation Method for Convex MINLP

David Bernal Neira (Carnegie Mellon University – Pittsburgh, US) and Ignacio Grossmann (Carnegie Mellon University – Pittsburgh, US)

License © Creative Commons BY 3.0 Unported license
© David Bernal Neira and Ignacio Grossmann

Joint work of David Bernal Neira, Ignacio E. Grossmann, Lijie Su, Lixin Tang, Jan Kronqvist

Main reference Lijie Su, Lixin Tang, David E. Bernal, Ignacio E. Grossmann: “Improved quadratic cuts for convex mixed-integer nonlinear programs”, *Computers & Chemical Engineering*, Vol. 109, pp. 77–95, 2018.

URL <http://dx.doi.org/10.1016/j.compchemeng.2017.10.011>

We present two new methods for solving convex mixed-integer nonlinear programming problems based on the outer approximation method. The first method is inspired by the level method and uses a regularization technique to reduce the step size when choosing new integer combinations. The second method combines ideas from both the level method and the sequential quadratic programming technique and uses a second order approximation of the Lagrangian when choosing the new integer combinations. The main idea behind the methods is to choose the integer combination more carefully in each iteration, in order to obtain the optimal solution in fewer iterations compared to the original outer-approximation method. We prove rigorously that both methods will find and verify the optimal solution in a finite number of iterations. Furthermore, we present a numerical comparison of the methods based on 109 test problems, which illustrates the benefits of the proposed methods.

3.6 Numerical Challenges in MINLP solvers

Timo Berthold (FICO – Berlin, DE)

License  Creative Commons BY 3.0 Unported license
© Timo Berthold

While numerics are already challenging us in MILP, it is my impression that in MINLP it is even more demanding to build a robust model and to develop a numerically stable solver. In MILP and MINLP solvers, reproducibility and reliability are key features to success. All major MILP solvers are deterministic, even in parallel mode, numerical tolerances are carefully chosen and various procedures and safety measures are implemented aiming for avoiding numerical issues, e.g. in presolving or cutting plane generation. In LP and MILP, there are approaches to solve problems to “true”, exact optimality. One possibility to extend those is the use of hybrid approaches like iterative refinement for LP and QP ([1]) or a hierarchy of dual bounding procedures for MILP ([2]). Then, in MILP models, challenges occur that are not present in MINLP, like super-linear error propagation, boundary conditions, and singular points. Moreover, some of the most-used solution methods are particularly prone to getting solutions which are “slightly off”. Take outer approximation as an example. Since we are approximating from the outside by cutting, we typically stop when the approximate solution is within the feasibility tolerances for the constraints, or differently speaking, violating them by the maximal amount that we are willing to tolerate. Also, vanishing eigenvectors are a problem in MINLP/OA which does not have a pendant in MILP.

I would like to kick off a discussion on:

- How are MINLP models different from MILP models?
- How can we measure the numerical conditioning of a MINLP model?
- Why are some of our standard solution approaches asking for numerical troubles and how can we address this?
- What would be an “exact” standard to compare floating-point, numerical-tolerance solvers against?
- Is numerical robustness a blocker for MINLP becoming an out-of-the-box tool like MILP?

References

- 1 Gleixner, Ambros M., Daniel E. Steffy, and Kati Wolter. “Iterative refinement for linear programming.” *INFORMS Journal on Computing* 28.3 (2016): 449–464.
- 2 Cook, William, et al. “A hybrid branch-and-bound approach for exact rational mixed-integer programming.” *Mathematical Programming Computation* 5.3 (2013): 305–344.

3.7 Optimization using both models and input-output data

Fani Boukouvala (Georgia Institute of Technology – Atlanta, US)

License  Creative Commons BY 3.0 Unported license
© Fani Boukouvala


Joint work of Fani Boukouvala, Jianyuan Zhai, Sun Hye Kim

In many engineering fields, there is a continuously increasing interest in coupling equation-based first-principle modeling with information that comes in the form of input-output data, in order to be able to optimize systems incorporating very detailed information regarding the material, flow, geometry, physical properties and chemistry of the systems. This need has given rise in developments of optimization methods that can optimize systems without

equations or derivatives, but simply with the exchange of input-output data streams. The typical applications of data-driven optimization, consider the system that needs to be optimized entirely as a “black-box”, however, many applications exist (i.e., process synthesis and design of modular manufacturing systems) which can be formulated as hybrid mixed integer nonlinear optimization problems, comprised of both explicitly known equations and black-box, or data-dependent equations. This talk highlights the need for MINLP solvers to be able to incorporate black-box components, and proposes ways of enabling this capability. Specifically, we propose several ideas for developing adaptive, flexible and tractable surrogate parametric functions to represent the input-output data and show their effectiveness in optimizing a library of benchmark problems, which are treated as black-box functions. In order to develop efficient and useful surrogate models, we borrow ideas from machine learning (i.e., feature selection), numerical integration (i.e., sparse grid sampling and polynomial interpolation) and optimization under uncertainty.

3.8 SMT-based mixed non-linear optimization


Andrea Callia D’Iddio (Imperial College London, GB)

License  Creative Commons BY 3.0 Unported license
© Andrea Callia D’Iddio

I would like to introduce ManyOpt, an optimization tool based on Satisfiability Modulo Theories (SMT). SMT solving can provide methods for feasibility checking whose main benefits are incrementality and deductive reasoning. Thanks to these benefits, it is possible to have a “warm start” in which an initial optimization problem is solved, and to take advantage of the learned information to solve extensions or modifications of that problem. This would be especially useful for what-if scenarios and for an interactive use of the tool. Experimental results with benchmarks from the MINLPLIB2 library show the effectiveness of the approach.

3.9 On some hard quadratic unconstrained boolean optimization problems

Sanjeeb Dash (IBM TJ Watson Research Center – Yorktown Heights, US)

License  Creative Commons BY 3.0 Unported license
© Sanjeeb Dash

In this talk we discuss the solution of hard quadratic unconstrained optimization problems arising from Ising model problems defined on Chimera graphs.

3.10 Subset selection with sparse matrices

Alberto Del Pia (University of Wisconsin – Madison, US), Santanu Dey (Georgia Institute of Technology – Atlanta, US), and Robert Weismantel (ETH Zürich, CH)

License © Creative Commons BY 3.0 Unported license
© Alberto Del Pia, Santanu Dey, and Robert Weismantel

In subset selection we search for the best linear predictor that involves a small subset of variables. Due to the vast applicability of this model, many approaches have been proposed by different communities, including enumeration, greedy algorithms, branch and bound, and convex relaxations. Our point of departure is to understand the problem from a computational complexity viewpoint. Using mainly tools from discrete geometry, we show that the problem can be solved in polynomial time if the associated data matrix is obtained by adding a fixed number of columns to a block diagonal matrix. This is joint work with Santanu S. Dey and Robert Weismantel.

3.11 Robust Treatment of Non-Convex Optimization Problems with Application to Gas Networks

Denis Aßmann (FAU), Frauke Liers (FAU), Juan Vera (Tilburg), and Michael Stingl (FAU)

License © Creative Commons BY 3.0 Unported license
© Denis Aßmann (FAU), Frauke Liers (FAU), Juan Vera (Tilburg), and Michael Stingl (FAU)
Main reference D. Aßmann, F. Liers, M. Stingl, J. Vera: “Deciding Robust Feasibility and Infeasibility Using a Set Containment Approach: An Application to Stationary Passive Gas Network Operation”, 2017
URL <https://opus4.kobv.de/opus4-trr154/frontdoor/index/index/docId/136>
Main reference D. Aßmann, F. Liers, M. Stingl: “Decomposable Robust Two-Stage Optimization: An Application to Gas Network Operations Under Uncertainty”, 2017
URL <https://opus4.kobv.de/opus4-trr154/frontdoor/index/index/docId/209>

In this talk, we will study uncertain stationary gas network problems. We will first explain the concept that can also be applied in other contexts and will then make it concrete for the application. For passive networks, the task of deciding robust feasibility of the corresponding nonlinear two-stage fully adjustable problem is equivalent to deciding set containment of a projection of the feasible region and the uncertainty set. For answering this question, two polynomial optimization problems – one for showing feasibility and one for showing infeasibility – are developed.

For the case of networks with active elements such as compressors, we reformulate the robust two-stage problem and then apply a piecewise linear relaxation of the non-convex functions. It is shown that comparably large realistic instances can be solved in practice, with only a mild increase in conservatism due to the used piecewise linear relaxation.

This is joint work with Denis Aßmann, Michael Stingl (both FAU), and Juan Vera (Tilburg).

3.12 New SOCP relaxation and branching rule for bipartite bilinear programs

Santanu Dey (Georgia Institute of Technology – Atlanta, US)

License © Creative Commons BY 3.0 Unported license

© Santanu Dey

Joint work of Santanu Dey, Asteroide Santana, Yang Wang

A bipartite bilinear program (BBP) is a quadratically constrained quadratic optimization problem where the variables can be partitioned into two sets such that fixing the variables in any one of the sets results in a linear program. We propose a new second order cone representable (SOCP) relaxation for BBP, which we show is stronger than the standard SDP relaxation intersected with the boolean quadratic polytope. We then propose a new branching rule inspired by the construction of the SOCP relaxation. We describe a new application of BBP called as the finite element model updating problem, which is a fundamental problem in structural engineering. Our computational experiments on this problem class show that the new branching rule together with an polyhedral outer approximation of the SOCP relaxation outperforms a state-of-the-art commercial global solver in obtaining dual bounds.

3.13 Deep Learning and Mixed Integer Optimization

Matteo Fischetti (University of Padova, IT)

License © Creative Commons BY 3.0 Unported license

© Matteo Fischetti

Joint work of Matteo Fischetti, Jason Jo

Main reference Matteo Fischetti, Jason Jo: “Deep Neural Networks as 0-1 Mixed Integer Linear Programs: A Feasibility Study”, CoRR, Vol. abs/1712.06174, 2017.

URL <http://arxiv.org/abs/1712.06174>

Deep Neural Networks (DNNs) are very popular these days, and are the subject of a very intense investigation. A DNN is made up of layers of internal units (or neurons), each of which computes an affine combination of the output of the units in the previous layer, applies a nonlinear operator, and outputs the corresponding value (also known as activation). A commonly-used nonlinear operator is the so-called rectified linear unit (ReLU), whose output is just the maximum between its input value and zero. In this (and other similar cases like max pooling, where the max operation involves more than one input value), *for fixed parameters* one can model the DNN as a 0-1 Mixed Integer Linear Program (0-1 MILP) where the continuous variables correspond to the output values of each unit, and a binary variable is associated with each ReLU to model its yes/no nature. In this talk we discuss the peculiarity of this kind of 0-1 MILP models, and describe an effective bound-tightening technique intended to ease its solution. We also present possible applications of the 0-1 MILP model arising in feature visualization and in the construction of adversarial examples. Computational results are reported, aimed at investigating (on small DNNs) the computational performance of a state-of-the-art MILP solver when applied to a known test case, namely, hand-written digit recognition.

3.14 Improved quadratic cuts for convex mixed-integer nonlinear programs

Ignacio Grossmann (Carnegie Mellon University – Pittsburgh, US)

License © Creative Commons BY 3.0 Unported license

© Ignacio Grossmann

Joint work of Ignacio Grossmann, Lijie Su, Lixin Tang, David E. Bernal

Main reference Marco A. Duran, Ignacio E. Grossmann: “An outer-approximation algorithm for a class of mixed-integer nonlinear programs”, *Math. Program.*, Vol. 36(3), pp. 307–339, 1986.

URL <http://dx.doi.org/10.1007/BF02592064>

This presentation presents scaled quadratic cuts based on scaling second-order Taylor expansion terms for decomposition methods, Outer Approximation (OA) and Partial Surrogate Cuts (PSC) for convex Mixed Integer Nonlinear Programming (MINLP). The scaled quadratic cut is proven to be a stricter and tighter underestimation for the convex nonlinear functions than the classical supporting hyperplanes. The scaled quadratic cut can accelerate the convergence of the MINLP methods. We integrate the presented strategies of the scaled quadratic cuts, multi-generation cuts with OA and PSC, and develop six types of MINLP solution methods with scaled quadratic cuts. We also discuss the computational implementation of the Mixed Integer Quadratically Constrained Programming (MIQCP) master problem that makes use of the Quadratically Constrained Programming (QCP) solution methods. Numerical results of benchmark MINLP problems demonstrate the effectiveness of the proposed MINLP solution methods with scaled quadratic cuts. In particular, numerical experiments show that OA and PSC with scaled quadratic cuts and multigeneration cuts can solve all the tested MINLP benchmark problems, and need few iterations and CPU solution times, especially for difficult problems.

3.15 Binary Extended Formulations for Mixed-Integer Linear Programs

Oktay Gunluk (IBM TJ Watson Research Center – Yorktown Heights, US), Sanjeeb Dash (IBM TJ Watson Research Center – Yorktown Heights, US), and Robert Hildebrand

License © Creative Commons BY 3.0 Unported license

© Oktay Gunluk, Sanjeeb Dash, and Robert Hildebrand

Main reference Sanjeeb Dash, Oktay Gunluk, Robert Hildebrand: “Binary Extended Formulations of Polyhedral Mixed-integer Sets”, *R. Math. Program.*, Vol. 170(1), pp. 207–236, Springer, 2018.

URL <https://doi.org/10.1007/s10107-018-1294-0>

URL http://www.optimization-online.org/DB_HTML/2018/01/6403.html

We analyze different ways of constructing binary extended formulations of polyhedral mixed-integer sets with bounded integer variables and compare their relative strength with respect to split cuts. We show that among all binary extended formulations where each bounded integer variable is represented by a distinct collection of binary variables, what we call “unimodular” extended formulations are the strongest. We also compare the strength of some binary extended formulations from the literature. Finally, we study the behavior of branch-and-bound on such extended formulations and show that branching on the new binary variables leads to significantly smaller enumeration trees in some cases.

3.16 Disjunctive cuts and extended formulations for bilinear functions

Akshay Gupte (Clemson University, US)

License © Creative Commons BY 3.0 Unported license
© Akshay Gupte

Joint work of Akshay Gupte, Thomas Kalinowski, Fabian Rigterink, Hamish Waterer

Main reference Akshay Gupte, Thomas Kalinowski, Fabian Rigterink, Hamish Waterer: “Extended formulations for convex hulls of some bilinear functions”, CoRR, Vol. abs/1702.04813, 2017.

URL <https://arxiv.org/abs/1702.04813>

We consider the problem of convexifying a bilinear function over some bounded polyhedral domain. When the incidence graph of this function is bipartite, the convex hull is known to be polyhedral. Special structure on one set of variables in the partition can be used to characterize this convex hull. If the structure is box constraints, then we note that a sequential convexification procedure converges to the convex hull. If the structure is a simplicial polytope, i.e., a polytope whose every facet is a simplex, then we use disjunctive programming to derive a small (polynomial)-sized extended formulation of the convex hull. Our second set of results is for the non-bipartite case of the incidence graph. Here, we characterize several graphs for which a small (linear) number of inequalities that are valid for the Boolean Quadric Polytope are sufficient to obtain a minimal extended formulation of the convex hull of the bilinear function. Our proof technique uses a new measure-theoretic characterization of combinatorial polytopes that we simplify from literature and establish for graphs of nonlinear functions over boxes.

3.17 Semidefinite Programming Cuts in Gravity

Hassan Hijazi (Los Alamos National Laboratory, US)

License © Creative Commons BY 3.0 Unported license
© Hassan Hijazi

Joint work of Ksenia Bestuzheva, Hassan Hijazi

URL <https://www.allinsights.io/gravity>

Gravity is an open source, scalable, memory efficient modeling language for solving mathematical models in Optimization and Machine Learning. It exploits structure to reduce function evaluation time including Jacobian and Hessian computation. Gravity is implemented in c++ with a flexible interface allowing the user to specify the numerical accuracy of variables and parameters. It is also designed to handle iterative model solving, convexity detection, distributed algorithms, and constraint generation approaches. When compared to state-of-the-art modeling languages such as Jump, Gravity is 5 times faster in terms of function evaluation and up to 60 times more memory efficient. It also dominates commercial languages such as Ampl on structured models including quadratically-constrained and polynomial programs. In this talk, we will give a brief overview of the language and present preliminary results on generating linear cuts that capture the strength of semidefinite programming relaxations and their implementation in Gravity.

3.18 Stronger polyhedral relaxations for polynomial optimization problems

Aida Khajavirad (Carnegie Mellon University – Pittsburgh, US) and Alberto Del Pia (University of Wisconsin – Madison, US)

License © Creative Commons BY 3.0 Unported license
© Aida Khajavirad and Alberto Del Pia

We consider the Multilinear set defined by a collection of multilinear terms over the unit hypercube. Such sets appear in factorable reformulations of many types of mixed-integer nonlinear programs including polynomial optimization problems. Utilizing an equivalent hypergraph representation for the Multilinear set, we derive various types of facet defining inequalities for its polyhedral convex hull and present a number of tightness results based on the acyclicity degree of the underlying hypergraph. Finally, we discuss the complexity of corresponding separation problems.

3.19 Mixed-Integer Nonlinear Programming Applications: Pyomo Tools for Tailored Strategies

Carl Damon Laird (Sandia National Labs – Albuquerque, US)

License © Creative Commons BY 3.0 Unported license
© Carl Damon Laird

There are a number of applications in safety and critical infrastructure protection that are best represented as mixed-integer nonlinear programming problems (MINLP), having nonlinear models of the key physics and discrete decisions. These applications push the limits of general off-the-shelf solvers due to large size induced by network structure or discretization due to uncertainty or time. In this presentation, I will discuss two MINLP applications (gas detector placement in chemical process facilities and global solution of ACOF problems) and our multi-tree solution strategies based on problem tailored relaxations and progressive refinement. We are working on a toolset within Pyomo to provide support for rapid development of these tailored solution strategies and testing of new relaxations and cuts.

3.20 Global solutions of MIQCPs

Amélie Lambert (CNAM – Paris, FR)

License © Creative Commons BY 3.0 Unported license
© Amélie Lambert

We present an algorithm for solving MIQCPs. For this, we develop a B&B based on a quadratic convex relaxation of the initial problem. This relaxation is built from the solution of a semidefinite programming relaxation and captures its strength. Computational experiences show that our general method is competitive with standard solvers, on many instances. This is a dummy text.

3.21 Virtuous smoothing and more virtuous smoothing

Jon Lee (University of Michigan – Ann Arbor, US), Daphne Skipper, and Luze Xu

License © Creative Commons BY 3.0 Unported license
© Jon Lee, Daphne Skipper, and Luze Xu

Joint work of Luze Xu, Daphne Skipper, Jon Lee

I am going to talk about how to smooth a univariate increasing concave function that is nice except at 0, where the derivative maybe be intolerably large (or infinite, which is definitely computationally intolerable). The killer application is power functions $f(w) := w^p$, with $0 < p < 1$, on $[0, \infty]$; some advocate these types of functions for inducing sparsity. In the context of MINLP (where we care about being nice to NLP solvers and in producing bounds), I will explain how to do this in a nice way, with a particular increasing concave smooth piecewise-defined function that is better than a simple shift. Our results apply under very general conditions, which I won't fully reveal in a 15 minute talk. Facility to handle our methodology was introduced in SCIP. We (Luze Xu, Daphne Skipper, J. Lee) now have a paper on arXiv (see [1]).

References

- 1 Luze Xu, Daphne Skipper, Jon Lee. More virtuous smoothing.
arXiv: <https://arxiv.org/abs/1802.09112>

3.22 Mixed-Integer Derivative-Free Optimization

Sven Leyffer (Argonne National Laboratory – Lemont, US)

License © Creative Commons BY 3.0 Unported license
© Sven Leyffer

Many design and engineering applications result in optimization problems that involve so-called black-box functions as well as integer variables, resulting in mixed-integer derivative-free optimization problems (MIDFOs). MIDFOs are characterized by the fact that a single function evaluation is often computationally expensive (requiring a simulation run for example) and that derivatives of the problem functions cannot be computed or estimated efficiently. In addition, many problems involve integer variables that are non-relaxable, meaning that we cannot evaluate the problem functions at non-integer points.

We present a new method for non-relaxable MIDFO that enables us to prove global convergence under idealistic convexity assumptions. To the best of our knowledge this is the first globally convergent method for non-relaxable MIDFO apart from complete enumeration. Our method constructs hyperplanes that interpolate the objective function at previously evaluated points. We show that in certain portions of the domain, these hyperplanes are valid underestimators of the objective, resulting in a set of conditional cuts. The union of these conditional cuts provide a nonconvex underestimator of the objective. We show that these nonconvex cuts can be modeled as a standard mixed-integer linear program (MILP). We provide some early numerical experience with our new method.

3.23 LP and SDP for kissing numbers

Leo Liberti (CNRS & Ecole Polytechnique – Palaiseau)

License © Creative Commons BY 3.0 Unported license
© Leo Liberti

Joint work of Jon Lee, Leo Liberti

Main reference Jon Lee, Leo Liberti: “On an SDP relaxation for kissing number”, Optimization Letters, pp. 1–6, Springer, 2018.

URL <https://doi.org/10.1007/s11590-018-1239-9>

Main reference Leo Liberti: “Mathematical programming bounds for kissing numbers, in Proc. of AIRO/ODS 2017, Proceedings in Mathematics and Statistics, Vol. 217., pp. 213–222, Springer, 2017.

URL https://doi.org/10.1007/978-3-319-67308-0_22

The “natural” MINLP formulation for the Kissing Number Problem (KNP) yields an SDP formulation for which we prove a uselessness theorem. We also look at some practical issues arising from the well known Delsarte LP bound.

3.24 External Intersection Cuts

James Luedtke (University of Wisconsin – Madison, US) and Eli Towle

License © Creative Commons BY 3.0 Unported license
© James Luedtke and Eli Towle

Intersection cuts are a general framework for deriving valid inequalities for nonconvex optimization problems. Specifically, given an open convex set C that does not contain any feasible solution and a basic solution that lies inside C , the intersection cut is derived by intersecting the rays of the cone K defined by the basic solution with the boundary of C , and using these points to define the separating hyperplane. Intersection cuts can be derived from both feasible and infeasible bases of a polyhedral relaxation of the problem. We investigate a new approach for deriving cuts from infeasible bases for which the basic solution does not lie in the set C (i.e., external points). Surprisingly, we find that the intersection cut framework can be used to derive valid disjunctions from such solutions, which can in turn be used to derive valid inequalities via Balas’ disjunctive cut framework. We present examples that illustrate when this framework can identify cuts that are not implied by standard intersection cuts.

3.25 Automatic reformulations of convex MINLPs in Minotaur

Ashutosh Mahajan (Indian Institute of Technology – Mumbai, IN)

License © Creative Commons BY 3.0 Unported license
© Ashutosh Mahajan

Joint work of Ashutosh Mahajan, Sharma, Meenarli

We consider two structures that may be exploited for solving convex MINLPs faster: separability in the nonlinear functions and perspective reformulations. We have implemented algorithms to identify these structures and reformulate the convex relaxation automatically. We will describe these methods and show how they were implemented in the Minotaur framework. Our experiments show that automatic reformulation can significantly reduce the solution time for benchmark instances when these structures are detected.

3.26 Optimising with Gradient-Boosted Trees and Risk Control

Miten Mistry (Imperial College London, GB), Dimitrios Letsios, Gerhard Krennrich, Ruth Misener (Imperial College London, GB), and Robert M. Lee

License © Creative Commons BY 3.0 Unported license
© Miten Mistry, Dimitrios Letsios, Gerhard Krennrich, Ruth Misener, and Robert M. Lee

Decision trees usefully represent the sparse, high dimensional and noisy nature of chemical data from experiments. Having learned a function from this data, we may want to thereafter optimise the function, e.g. for picking the best catalyst for a chemical process. This work studies a mixed-integer nonlinear optimisation problem involving:

- gradient boosted trees modelling catalyst behaviour
- penalty functions mitigating risk
- penalties enforcing chemical composition constraints.

We develop several heuristic methods to find feasible solutions, and an exact, branch and bound algorithm that leverages structural properties of the gradient boost trees and penalty functions. We computationally test our methods on an industrial instance from BASF.

3.27 Using 2D Projections for Separation and Propagation of Bilinear Terms

Benjamin Müller (Konrad-Zuse-Zentrum – Berlin, DE), Ambros M. Gleixner (Konrad-Zuse-Zentrum – Berlin, DE), and Felipe Serrano (Konrad-Zuse-Zentrum – Berlin, DE)

License © Creative Commons BY 3.0 Unported license
© Benjamin Müller, Ambros M. Gleixner, and Felipe Serrano

One of the most fundamental ingredients in mixed-integer nonlinear programming solvers is the well-known McCormick relaxation for a bilinear product of two variables x and y over a box-constrained domain. The starting point of this talk is the fact that these may be far from tight if the feasible region and its convexification projected in the x - y -space is a strict subset of the box. We develop an algorithm that solves a sequence of linear programs in order to compute globally valid inequalities on x and y in a similar fashion as optimization-based bound tightening. These valid inequalities allow us to exploit polyhedral results from the literature in order to tighten the classical McCormick relaxation. As a consequence we obtain a convexification procedure that can exploit objective cutoff information during branch-and-bound. We use the MINLP solver SCIP to analyze the impact of the tighter relaxations on instances of the MINLPLib2.

3.28 CIA Decomposition

Sebastian Sager (Universität Magdeburg, DE)

License © Creative Commons BY 3.0 Unported license
© Sebastian Sager

Solving mixed-integer nonlinear programs (MINLPs) is hard in theory and practice. Decomposing the nonlinear and the integer part seems promising from a computational point of view. In general, however, no bounds on the objective value gap can be guaranteed and

iterative procedures with potentially many subproblems are necessary. The situation is different for mixed-integer optimal control problems with binary choices that switch over time and space. Here, a priori bounds were derived for a decomposition into one continuous nonlinear control problem and one mixed-integer linear program, the combinatorial integral approximation (CIA) problem.

We generalize and extend the decomposition idea. The extension is also transferable in a straightforward way to recently suggested variants for certain partial differential equations, for algebraic equations, for additional combinatorial constraints, and for discrete time problems.

All algorithms and subproblems were implemented in AMPL for proof of concept. Numerical results show the improvement compared to standard CIA decomposition with respect to objective function value and compared to the general purpose MINLP solver Bonmin with respect to runtime.

3.29 ALAMO: Machine learning from data and first principles

Nikolaos V. Sahinidis (Carnegie Mellon University – Pittsburgh, US)

License © Creative Commons BY 3.0 Unported license
© Nikolaos V. Sahinidis

Joint work of Alison Cozad, David Miller, Zachary Wilson

Main reference Zachary T. Wilson, Nikolaos V. Sahinidis: “The ALAMO approach to machine learning”, Computers & Chemical Engineering, Vol. 106, pp. 785–795, 2017.

URL <http://dx.doi.org/10.1016/j.compchemeng.2017.02.010>

We have developed the ALAMO methodology with the aim of producing a tool capable of using data to learn algebraic models that are accurate and as simple as possible. ALAMO relies on integer nonlinear optimization, derivative-free optimization, and global optimization to build and optimize models. We present the methodology behind ALAMO and comparisons with a variety of learning techniques, including the lasso.

3.30 Separating over the convex hull of MINL constraints

Felipe Serrano (Konrad-Zuse-Zentrum – Berlin, DE)

License © Creative Commons BY 3.0 Unported license
© Felipe Serrano

Techniques for computing facets of arbitrary mixed-integer sets have had many applications. We apply the methodology to optimize over the relaxation obtained by intersecting the convex hull of every 1-row MINL relaxations of an MINLP.

3.31 Product convexification: A new relaxation framework for nonconvex programs

Mohit Tawarmalani (Purdue University – West Lafayette, US)

License © Creative Commons BY 3.0 Unported license

© Mohit Tawarmalani

Joint work of Mohit Tawarmalani, Taotao He

We develop a new relaxation that exploits function structure while convexifying a product of n functions. The function structure is encapsulated using at most d over and underestimators. We convexify the function product in the space of estimators. The separation procedure generates facet-defining inequalities in time polynomial in d for a fixed n . If the functions are non-negative, the concave envelope can be separated in $O(nd \log(d))$. Then, we extend our construction to infinite families of under and overestimators. Our relaxation procedure can be interpreted as a two-step procedure where we first express the product as a telescoping sum and in the second step apply a simple relaxation strategy. This interpretation admits various generalizations that yield various valid inequalities for nonconvex programs. We conclude by discussing techniques to generate the over and underestimators and various ways in which the proposed techniques improve and/or generalize current relaxation schemes for factorable programs.

3.32 Mixed-integer convex representability

Juan Pablo Vielma (MIT – Cambridge, US)

License © Creative Commons BY 3.0 Unported license

© Juan Pablo Vielma

Joint work of Miles Lubin, Ilias Zadik, Juan Pablo Vielma

Main reference Miles Lubin, Ilias Zadik, Juan Pablo Vielma: “Mixed-Integer Convex Representability”, in Proc. of the Integer Programming and Combinatorial Optimization – 19th Int’l Conference, IPCO 2017, Waterloo, ON, Canada, June 26-28, 2017, Proceedings, Lecture Notes in Computer Science, Vol. 10328, pp. 392–404, Springer, 2017.

URL http://dx.doi.org/10.1007/978-3-319-59250-3_32

Main reference Miles Lubin, Ilias Zadik, Juan Pablo Vielma: “Regularity in mixed-integer convex representability”, CoRR, Vol. abs/1706.05135, 2017.

URL <https://arxiv.org/abs/1706.05135>

We consider the question of which nonconvex sets can be represented exactly as the feasible sets of mixed-integer convex optimization problems (MICP). We first show a complete characterization for the case when the number of possible integer assignments is finite. We then further study the characterization for the more general case of unbounded integer variables and introduce a simple necessary condition for representability. This condition can be used to show that the set of prime numbers is not MICP representable, even though it can be represented using polynomial equations and integrality constraints. While the result for the prime numbers suggests certain regularity of MICP representable sets, we show that even for subsets of the natural numbers, MICP representable sets can be significantly more irregular than rational mixed integer linear programming representable sets. Inspired by these irregular MICP representable sets we introduce a notion of rational MICP representability and show how this notion imposes regularity to MICP representable subsets of the natural numbers, for compact convex sets and the graphs and epigraphs of certain functions. Finally, we study other notions of regularity associated to infinite unions of convex sets with the same volume. This is joint work with Miles Lubin and Ilias Zadik.

3.33 Experimentation with MINLP solver software

Stefan Vigerske (GAMS Software GmbH, DE) and Ambros M. Gleixner (Konrad-Zuse-Zentrum – Berlin, DE)

License © Creative Commons BY 3.0 Unported license
© Stefan Vigerske and Ambros M. Gleixner

We briefly mention some issues that frequently come up when benchmarking software that solves MINLPs. In particular, we discuss different interpretations of primal feasibility, dependence of performance on the choice of the optimality tolerance, and difficulty in defining the optimal value of a problem. By applying permutations to the order of variables and constraints, we exploit the extend of performance variability, including its dependence on the choice of optimality tolerance and whether the primal-dual integral is less prone to variability than solving time.

3.34 Integer Optimal Solutions are Sparse

Robert Weismantel (ETH Zürich, CH)

License © Creative Commons BY 3.0 Unported license
© Robert Weismantel
Joint work of Iskander Aliev, Jesus De Loera, Fritz Eisenbrand, Timm Oertel, Robert Weismantel

Given a system of m equations in n variables subject to nonnegativity, how sparse is an optimal integer solution? We prove that an optimal integer solution is independent on n and in the order of $m \log(ma)$ where a is the largest absolute value of an entry of the constraint matrix. This bound is asymptotically optimal.

3.35 Combining ADAL with Factorizing the Dual to Solve SDP

Angelika Wiegele (Alpen-Adria-Universität Klagenfurt, AT)


License © Creative Commons BY 3.0 Unported license
© Angelika Wiegele
Joint work of Marianna De Santis, Franz Rendl, Angelika Wiegele
Main reference Marianna De Santis, Franz Rendl, Angelika Wiegele: “Using a Factored Dual in Augmented Lagrangian Methods for Semidefinite Programming”, CoRR, Vol. abs/1710.04869, 2017.
URL <https://arxiv.org/abs/1710.04869>

Using semidefinite programming has become a promising method for solving or approximating various combinatorial optimization problems. However, solving semidefinite programs is challenging due to either the large size of the matrices involved or a huge number of constraints. The most prominent methods, interior point methods, run out of memory for many practical applications.

Other algorithms for solving semidefinite problems are based on augmented Lagrangian methods using various ways of dealing with the semidefiniteness constraint. We developed such an augmented Lagrangian algorithm where we replace the semidefiniteness constraint of the dual problem by a factorization $Z = VV^t$. Using this factorization, we end up with an unconstrained (non-convex) problem in V . We then perform updates of this matrix V towards the optimal solution of Z following an alternating direction method or in the fashion of the boundary point method. We will present results for computing the theta number of a graph and computing bounds on the quadratic linear ordering problem.

3.36 Mixed-integer conic optimization and MOSEK

Sven Wiese (MOSEK ApS – Copenhagen, DK)

License  Creative Commons BY 3.0 Unported license
© Sven Wiese

Recently Lubin et al. showed that all convex instances of the nonlinear mixed-integer benchmark library MINLPLIB2 can be reformulated as conic optimization problems using 5 different cone types. These are the linear, the quadratic, the semidefinite, the exponential and the power cones. The former three cones belong to the class of symmetric cones, whereas the latter two are non-symmetric.

We call modeling with affine expressions and the five previously mentioned cone types extremely disciplined modeling. Based on Lubin et al., and on the experience at MOSEK, we claim that almost all practical convex optimization problems can be expressed using extremely disciplined modeling, making it a quite general framework. Now it is much easier to build optimization algorithms and software for extremely disciplined optimization models rather than for general (less structured) convex problems due to the limited and explicit structure. This fact is exploited in the software package MOSEK that we will discuss.

MOSEK has for many years been able to solve conic optimization problems over the symmetric cones, but in the upcoming version 9 MOSEK can also handle the two non-symmetric cones i.e. the exponential and the power cones. In this presentation we will discuss the continuous and mixed-integer conic optimizer in MOSEK. In addition, computational results, that illustrate the performance of MOSEK on problems including non-symmetric cones, are presented.

4 Working groups

4.1 Applications in Energy

The working group on applications in the energy sector, led by Alexander Martin and Frauke Liers from the University of Erlangen-Nuremberg, addressed both structural topics at the interface between engineering and optimization as well as algorithmic bottlenecks. It was agreed that mixed-integer nonlinear programming is a key paradigm for modeling and solving engineering applications both in the energy sector and beyond. However, structurally, it became clear that the engineering community puts a stronger focus on the modeling aspect, while the applied mathematics community rather values algorithmic research, a divide that is also reflected in the publication system. Algorithmically, the participants identified that the mathematical MINLP community potentially overemphasizes research on proving optimality. However, success of MINLP in engineering applications rather requires near-optimal answers quickly and robustly. The participants formulated the recommendation to adapt solver benchmarks to these needs in order to set the right incentives for solver development according to requirements in practice.

4.2 Sound experimentation with MINLP software

This working group channeled discussions on several questions that lie at the core of the seminar. Two technical presentations on Monday prepared the session thematically and summarized central questions of solver benchmarking to the entire seminar. Stefan Vigerske

from GAMS Software presented results from a series of experiments to compare how different solvers react to various experimental parameters including tolerances for feasibility and representations of models. Timo Berthold from the solver FICO Xpress highlighted particular difficulties regarding numerical stability. During the session these observations were analyzed in more detail. The participation of core developers from the MIP and MINLP solvers ANTIGONE, BARON, Couenne, Gurobi, Minotaur, SCIP and FICO Xpress resulted in a discussion on a technical level that is generally difficult to reach in a broader audience.

The main conclusions of the session were:

- the need to incorporate experimentation on permuted models as has become standard in the MIP community;
- the need to develop more informative solver output beyond optimal solutions (including sensitivity information);
- the potential usefulness of a MINLP analogue to the „MIP kappa“ measure for an a posteriori evaluation of numerical safety;
- the affirmation that conducting fair benchmarks between different solvers remains inherently difficult and users of solver software should not rely on simplistic comparisons.

Since our working group, Timo Berthold and the FICO Xpress team are already working to move the solver software discussions into practice (see [1]).

References

- 1 <https://community.fico.com/community/articles/blog/2018/05/01/numerical-challenge-accepted>

4.3 Connections between MINLP and Machine Learning

The machine learning working group was organized and led by Andrea Lodi. Dr. Lodi started the session with a presentation on the two views of machine learning and optimization/mixed integer nonlinear programming.

- *What can MINLP do for machine learning?* One great example is improved algorithm performance on classification problems where the loss function is not convex.
- *What can machine learning do for MINLP?* He described one interesting example where classical ML techniques were used to dynamically tune parameters for the commercial MINLP software CPLEX. The discussion led to other areas where ML could be used to help MINLP algorithm performance, for example in variable branching.

Matteo Fischetti then described an interesting application of mixed integer linear programming to machine learning. It is a simple but interesting observation that for a very common type of neural net activation function – the so-called ReLU – the mapping from input to output performed by a (deep) neural network can be modeled as the solution of a mixed integer program. Given a fixed network topology and weights, this gives the opportunity of optimizing over inputs to find, for example, an input to the network that is “close” to other inputs of a given class, but for which the output of the network classifies it differently. This “adversarial” approach to machine learning is important for creating robust training sets. There followed discussion on other classes of problems where MINLP could be used in ML. First, MINLP is unlikely to be the appropriate tool for one of the most important and canonical optimization problems in ML – the training of neural networks. Second, a lot of interest was generated around finding other places where one could make use of the equivalence of MIP and the neural network encoding.

To finish the working group, Dr. Oktay Gunluk from IBM described an application in symbolic regression. The algebraic operations in a grammar of valid mathematical expressions can be modeled using integer variables, so the problem of finding the optimal mathematical form of a model that best matches data, the so-called symbolic regression problem – can be modeled as a MINLP. This is an extremely interesting and challenging class of problems. To date, the MINLP technology can solve only small-sized problems.

5 Open problems

During the *Open Problem Solving Session*, we identified several areas where more research is required. We are currently working on a more extensive document to be published in the mathematical optimization community. This paper will give complete background for these questions and invite further comments from the community.

5.1 Mathematical Foundations

- *Column generation* - If we have many columns but a convex objective function, is there a notion of column generation?
- *Second-Order Cone Representation* - When is the convex hull of a semi-algebraic set second-order cone representable? Alternatively, what is the minimum number of extra variables needed for second-order cone representability?

5.2 Computing Implications

Here we focussed on the need to make MINLP solvers (and the relevant sub solvers) more reliable, by discussing:

- improving semidefinite optimization solvers, nonlinear optimization solvers working with perspective functions, and nonlinear optimization solvers in general;
- borrowing an assortment of tools from other communities, e.g. high performance computing, multiscale and multigrid methods, and logic-based methods;
- extended formulations of mathematical optimization problems and why these are so difficult to solve.

5.3 Engineering Applications

The applications where we see significant growth potential are machine learning, optimal power flow, and the pooling problem. We discussed the current barriers to being able to solve large-scale instantiations of these problems with MINLP.

Participants

- Tobias Achterberg
Konrad-Zuse-Zentrum –
Berlin, DE
- Claire Adjiman
Imperial College London, GB
- Shabbir Ahmed
Georgia Institute of Technology –
Atlanta, US
- Kurt M. Anstreicher
University of Iowa, US
- Radu Baltean-Lugojan
Imperial College London, GB
- Pietro Belotti
Fair Isaac, GB
- David Bernal Neira
Carnegie Mellon University –
Pittsburgh, US
- Timo Berthold
FICO – Berlin, DE
- Christian Blik
IBM – Valbonne, FR
- Pierre Bonami
IBM Spain – Madrid, ES
- Fani Boukouvala
Georgia Institute of Technology –
Atlanta, US
- Andrea Callia D'Iddio
Imperial College London, GB
- Sanjeeb Dash
IBM TJ Watson Research Center
– Yorktown Heights, US
- Alberto Del Pia
University of Wisconsin –
Madison, US
- Santanu Dey
Georgia Institute of Technology –
Atlanta, US
- Matteo Fischetti
University of Padova, IT
- Ambros M. Gleixner
Konrad-Zuse-Zentrum –
Berlin, DE
- Ignacio Grossmann
Carnegie Mellon University –
Pittsburgh, US
- Andreas Grothey
University of Edinburgh, GB
- Oktay Gunluk
IBM TJ Watson Research Center
– Yorktown Heights, US
- Akshay Gupte
Clemson University, US
- Hassan Hijazi
Los Alamos National
Laboratory, US
- Aida Khajavirad
Carnegie Mellon University –
Pittsburgh, US
- Carl Damon Laird
Sandia National Labs –
Albuquerque, US
- Amélie Lambert
CNAM – Paris, FR
- Jon Lee
University of Michigan –
Ann Arbor, US
- Sven Leyffer
Argonne National Laboratory –
Lemont, US
- Leo Liberti
CNRS & Ecole Polytechnique –
Palaiseau
- Frauke Liers
Universität Erlangen-Nürnberg,
DE
- Jeff Linderoth
University of Wisconsin –
Madison, US
- Andrea Lodi
Polytechnique Montreal, CA
- James Luedtke
University of Wisconsin –
Madison, US
- Ashutosh Mahajan
Indian Institute of Technology –
Mumbai, IN
- Alexander Martin
Universität Erlangen-
Nürnberg, DE
- Ruth Misener
Imperial College London, GB
- Miten Mistry
Imperial College London, GB
- Benjamin Müller
Konrad-Zuse-Zentrum –
Berlin, DE
- Sebastian Sager
Universität Magdeburg, DE
- Nikolaos V. Sahinidis
Carnegie Mellon University –
Pittsburgh, US
- Felipe Serrano
Konrad-Zuse-Zentrum –
Berlin, DE
- Mohit Tawarmalani
Purdue University –
West Lafayette, US
- Juan Pablo Vielma
MIT – Cambridge, US
- Stefan Vigerske
GAMS Software GmbH, DE
- Robert Weismantel
ETH Zürich, CH
- Angelika Wiegele
Alpen-Adria-Universität
Klagenfurt, AT
- Sven Wiese
MOSEK ApS – Copenhagen, DK



Formal Methods for the Synthesis of Biomolecular Circuits

Edited by

Yaakov Benenson¹, Neil Dalchau², Heinz Koeppl³, and
Oded Maler⁴

¹ ETH Zürich – Basel, CH, kobi.benenson@bsse.ethz.ch

² Microsoft Research UK – Cambridge, GB, ndalchau@microsoft.com

³ TU Darmstadt, DE, heinz.koeppl@bcs.tu-darmstadt.de

⁴ VERIMAG – Grenoble, FR, oded.maler@imag.fr

Abstract

This report documents the program and the outcomes of Dagstuhl Seminar 18082 “Formal Methods for the Synthesis of Biomolecular Circuits”. Synthetic biology aims for the rational bottom-up engineering of new biological functionalities. Recent years have witnessed an increase in the degree of “rationality” in the design of synthetic biomolecular circuits. With it, fewer design-build-test cycles were necessary to achieve a desired circuit performance. Most of these success stories reported the realization of logic circuits, typically operating via regulation of gene expression and/or direct manipulation of DNA sequences with recombinases, executing combinatorial and sometimes sequential logic. This was often achieved with the help of two ingredients, a library of previously well-characterized parts and some computational modeling. Hence, although circuits in synthetic biology are still by far less understood and characterized than electronic circuits, the opportunity for the formal synthesis of circuit designs with respect to a behavioral specification starts to emerge in synthetic biology.

Seminar February 18–23, 2018 – <https://www.dagstuhl.de/18082>

2012 ACM Subject Classification Theory of computation → Verification by model checking, Theory of computation → Verification by model checking, Theory of computation → Concurrency, Computing methodologies → Machine learning, Mathematics of computing → Probability and statistics, Software and its engineering → Specification languages

Keywords and phrases Synthetic biology, Electronic design automation, Program synthesis and verification

Digital Object Identifier 10.4230/DagRep.8.2.88

1 Executive Summary

Heinz Koeppl

License  Creative Commons BY 3.0 Unported license
© Heinz Koeppl

The seminar brought together experts in formal methods for the verification and synthesis of hardware and software with wet-lab and dry-lab synthetic biologists to (1) achieve a common understanding of the current state of design methodology in synthetic biology; (2) to identify the limitations of current approaches and (3) to investigate dedicated solutions to the synthesis problem in synthetic biology. Some of these methods are based on leveraging experience and methods from electronic design automation (EDA) and from program synthesis and verification. In addition, ideas for entirely new methodologies specifically tailored for synthetic biology are likely to emerge. For example, features that are not apparent in



Except where otherwise noted, content of this report is licensed under a Creative Commons BY 3.0 Unported license

Formal Methods for the Synthesis of Biomolecular Circuits, *Dagstuhl Reports*, Vol. 8, Issue 02, pp. 88–100

Editors: Yaakov Benenson, Neil Dalchau, Heinz Koeppl, and Oded Maler



Dagstuhl Reports

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

electronic circuits such as heterogeneity and variability between the cells and between the circuits embedded in different cells, were addressed.

Apart from talk by participants, the seminar also featured break out session that were well received by the participants. In particular, we had sessions on “Modeling context-dependency of synthetic circuits” on “Metrology in Synthetic Biology” and on “Formal Specification for Biological Circuit Synthesis”.

2 Table of Contents

Executive Summary

<i>Heinz Koeppl</i>	88
-------------------------------	----

Overview of Talks

How AI can help synthetic biology <i>Aron Adler</i>	92
Bayesian approaches to engineering synthetic biological systems <i>Chris Barnes</i>	92
Foundational Metrology for Engineering Biomolecular Circuits <i>Jacob Beal</i>	92
Syntax-Guided Optimal Synthesis for Chemical Reaction Networks <i>Milan Ceska</i>	93
Programming DNA circuits <i>Neil Dalchau</i>	93
Automated Reasoning in Stem Cell Biology <i>Sara-Jane Dunn</i>	94
A compiler of computable real-valued functions in abstract biochemical reaction networks <i>François Fages</i>	94
A critical view on Synthetic Biology (by an outsider!) <i>Eric Fanchon</i>	94
Exploring and verifying complex genetic circuit designs and design spaces using deep-sequencing <i>Thomas Gorochowski</i>	95
Three synthetic biology design challenges we face, and how we are approaching them <i>Nathan Hillson</i>	95
CloneFlow – Computer-aided Planning of DNA Assembly Reactions and Experimental Workflows <i>Johannes Kabisch</i>	95
Towards flexible, and data-driven construction and analysis of dynamical models for synthetic biology <i>Gareth Molyneux</i>	96
An executable software model of tetracycline-aptamer-mediated translation in yeast <i>Radu Muschevici</i>	96
Design of Asynchronous Genetic Circuits <i>Chris Myers</i>	97
Model-based multiobjective optimization framework for automated design in Synthetic Biology <i>Irene Otero-Muras</i>	97

Visual Representations for CRN Synthesis
James Scott-Brown 98

Synthesis of Biomolecular Circuits with Z3
Boyan Yordanov 98

SMT-based Set Synthesis for Biological Models
Paolo Zuliani 98

Working groups 99

Participants 100

3 Overview of Talks

3.1 How AI can help synthetic biology

Aron Adler (BBN Technologies – Cambridge, US)

License  Creative Commons BY 3.0 Unported license
© Aron Adler

A wide variety of Artificial Intelligence (AI) techniques, from expert systems to machine learning to robotics, are needed in the field of synthetic biology. This talk will look back at progress in recent years and highlight places where AI has already helped and the ongoing opportunities for applying the lessons from decades of AI research to problems in synthetic biology.

3.2 Bayesian approaches to engineering synthetic biological systems

Chris Barnes (University College London, London, UK)

License  Creative Commons BY 3.0 Unported license
© Chris Barnes

An aim of synthetic biology is to apply engineering tools and principles to the design and construction of novel biological systems. There is huge potential for clinical applications but for such advanced therapeutics to be implemented, we must first be able to design and build systems that can function reliably in complex and changing environments. Using the example of engineering a two-species bacterial community, I will describe how Bayesian statistics and dynamical modelling can be used at different points within the design-build-test cycle.

3.3 Foundational Metrology for Engineering Biomolecular Circuits

Jacob Beal (BBN Technologies – Cambridge, US)

License  Creative Commons BY 3.0 Unported license
© Jacob Beal

People have been running experiments to characterize biological circuit components ever since these notions were first conceived. Such data, however, is very rarely actually able to be used effectively for circuit design by anyone besides its creators and often not even by them. I will argue that this is largely due to inadequacies in how circuits and components are typically measured. This discussion will include process vs. experimental controls, statistical analysis of biomolecular circuits, reliable unit calibration, and precision requirements. Motivating examples and evidence will be drawn from my experience with precision prediction of biological circuit behavior, development of high-performance biological computing devices, and large scale interlaboratory studies with iGEM and the NIST Synthetic Biology Standards Consortium.

3.4 Syntax-Guided Optimal Synthesis for Chemical Reaction Networks

Milan Ceska (Brno University of Technology, CZ)

License  Creative Commons BY 3.0 Unported license
© Milan Ceska

We study the problem of optimal syntax-guided synthesis of stochastic Chemical Reaction Networks (CRNs) that plays a fundamental role in design automation of molecular devices and in the construction of predictive biochemical models. We propose a sketching language for CRNs that concisely captures syntactic constraints on the network topology and allows its under-specification. Given a sketch, a correctness specification, and a cost function defined over the CRN syntax, our goal is to find a CRN that simultaneously meets the constraints, satisfies the specification and minimizes the cost function. To ensure computational feasibility of the synthesis process, we employ the Linear Noise Approximation allowing us to encode the synthesis problem as a satisfiability modulo theories problem over a set of parametric Ordinary Differential Equations (ODEs). We design and implement a novel algorithm for the optimal synthesis of CRNs that employs almost complete refutation procedure for SMT over reals and ODEs, and exploits a meta-sketching abstraction controlling the search strategy. Through relevant case studies we demonstrate that our approach significantly improves the capability of existing methods for synthesis of biochemical systems and paves the way towards their automated and provably-correct design.

3.5 Programming DNA circuits


Neil Dalchau (Microsoft Research UK – Cambridge, GB)

License  Creative Commons BY 3.0 Unported license
© Neil Dalchau

Biological organisms use complex molecular networks to navigate their environment and regulate their internal state. The development of synthetic systems with similar capabilities could lead to applications such as smart therapeutics or fabrication methods based on self-organization. To achieve this, molecular control circuits need to be engineered to perform integrated sensing, computation and actuation. In this talk, I will describe an approach based on DNA hybridization and strand displacement to implement the computational core of such control circuits. We use domain-specific programming languages to specify the sequence-level circuit design, which compile to chemical reaction networks, a well-established formalism for describing and simulating chemistry. Furthermore, we have integrated parameter inference techniques into this design platform, which facilitates design-build-test-learn cycles via model-based characterization and circuit prediction. A first example will introduce how we designed and constructed a DNA implementation of the approximate majority algorithm, which seeks to establish consensus in a population of agents (molecules). A second example will illustrate how DNA circuits can be considerably accelerated by tethering DNA hairpin molecules to a fixed template, overcoming molecular diffusion as a rate-limiting step.

3.6 Automated Reasoning in Stem Cell Biology

Sara-Jane Dunn (Microsoft Research UK – Cambridge, GB)

License  Creative Commons BY 3.0 Unported license
© Sara-Jane Dunn

A fundamental goal in developmental biology is to understand the logic of cellular decision-making at the molecular level. However, the complexity of biological processes presents a major challenge when trying to delineate fate decisions, which are typically influenced by a multiplicity of extrinsic and intrinsic regulators displaying non-intuitive interactions. Against this backdrop, automated reasoning is a powerful methodology that can allow researchers to navigate biological complexity and derive explanations of behaviour that are provably consistent with experimental evidence. In this talk, I will discuss the synthesis and analysis of dynamic networks of biological components that govern decision-making in embryonic stem cells. I will demonstrate how this approach can be used to derive a predictive explanation of cellular behaviour, generating counterintuitive and informing future experiments.

3.7 A compiler of computable real-valued functions in abstract biochemical reaction networks

François Fages (INRIA Saclay – Île-de-France, FR)

License  Creative Commons BY 3.0 Unported license
© François Fages

I shall present a compiler of real-valued functions in continuous CRNs based on a recent proof of their Turing completeness. More specifically, our compiler, implemented in BIOCHAM v4, takes as input a polynomial differential equation system and produces as output a finite reaction network which implements it with at most binary reactions and a fixed set of molecular species with positive concentration values. Then I will discuss the issues of error control, robustness measure a posteriori, robustness control a priori, and concrete implementations with real enzymes in DNA-free vesicles in partnership with Franck Molina's lab Sys2diag.

3.8 A critical view on Synthetic Biology (by an outsider!)

Eric Fanchon (TINC-IMAG Lab – La Tronche, FR)

License  Creative Commons BY 3.0 Unported license
© Eric Fanchon


This talk is mainly about sharing some thoughts on synthetic biology and asking questions...

I will first discuss some conceptual/philosophical issues which influence the way the biological engineering problems are approached: the genome as a book or a program; the cell as a machine.

I will then mention some practical issues: the lack of data needed to build dynamical models, the dense web of molecular interactions which makes the composition of modules hard to realize, the large variability of biological systems. Regarding Formal Methods the biggest issue seems to be related to the interaction of the engineered system with the host cell. In view of the many unexpected interactions that might occur, how to prove safety?

3.9 Exploring and verifying complex genetic circuit designs and design spaces using deep-sequencing


Thomas Gorochowski (University of Bristol, GB)

License  Creative Commons BY 3.0 Unported license
© Thomas Gorochowski

As the size and complexity of genetic circuits grows (as well as our ambitions) their assembly and functional verification becomes increasingly difficult due to the sheer number of parts involved. I will show how we have been using deep-sequencing and one-pot pooled assembly methods to construct subsets of design spaces to make exploration of large design spaces quicker and easier. I will also introduce our efforts to move away from fluorescent reports as a means of probing internal circuit states and propose RNA-sequencing as a viable alternative that enables the entire transcriptional state of all components to be observed at a point in time. I will end by giving my personal perspective on how these threads might come together with new design methods to better understand the underlying principles for effective genetic circuit synthesis.

3.10 Three synthetic biology design challenges we face, and how we are approaching them

Nathan Hillson (JBEI – Emeryville, US)

License  Creative Commons BY 3.0 Unported license
© Nathan Hillson

After a brief introduction to the Agile BioFoundry and to the Joint Genome Institute (for context), three synthetic biology design challenges that we face, and how we are currently approaching them, will be presented. These challenges include: 1) how to reliably/predictably control protein levels, in context of statistical modelling design of experiments; 2) the design of (and Learning from) operon structure variants; and 3) informatic integration/continuity across layers of a hierarchical Build process.

3.11 CloneFlow – Computer-aided Planning of DNA Assembly Reactions and Experimental Workflows

Johannes Kabisch (TU Darmstadt, DE)

License  Creative Commons BY 3.0 Unported license
© Johannes Kabisch

Progress in synthetic biology can be facilitated by rapid prototyping approaches to test a wide variety of, e.g., genetic circuits. In the last decades, many methods for assemblies were used to build DNA constructs (e.g., Gibson assembly, Golden Gate assembly). Among these methods ligase cycling reaction (LCR) fulfills many prerequisites for an easy application and automation like the usage of unmodified DNA parts and that the assembly order is only determined by single-stranded oligonucleotides building a bridge (so called bridging oligonucleotide, BO) between adjacent parts. In order to enable an easy access for scientist to apply LCR we are developing the web-application CloneFlow. CloneFlow offers to create LCR

workflows based on the input of DNA sequences in a multi-fasta format and experimental parameters. Different output formats are available including csv-files enabling easy downstream processing e.g. for automation with liquid-handling systems. In our case we feed such files into a nanoliter dispenser. As a unique feature, all oligonucleotides are secondary structure optimized, i. e. our service finds the oligos which have low free energy and thus a small contribution to secondary structures. Our work presents a synergistic collaboration between experimental expertise and computational know-how.

3.12 Towards flexible, and data-driven construction and analysis of dynamical models for synthetic biology

Gareth Molyneux (Oxford University, Oxford, UK)

License  Creative Commons BY 3.0 Unported license
© Gareth Molyneux

We propose a technique that integrates Bayesian learning and model verification to quantify the likelihood that the underlying data-generating biological system satisfies a given dynamical property of interest. We extend an approach developed for diverse models and systems and adapt it to biological circuits, and specifically to continuous-time Markov Chain models of Chemical Reaction Networks. We argue that the approach is flexible and adaptable to time varying data, and that it extracts more information from gathered data than standard statistical techniques. From the perspective of synthesis, it can be used to build parts of the model of a genetic circuit, as well as to generate optimal experiments for model learning.

3.13 An executable software model of tetracycline-aptamer-mediated translation in yeast


Radu Muschevici (TU Darmstadt, DE)

License  Creative Commons BY 3.0 Unported license
© Radu Muschevici

Gene expression in yeast cells can be controlled by mediating translation through insertion of tetracyclin aptamers into the 5' UTR of the mRNA molecule. This mechanism has been shown to work well in practice, but its exact mode of action and the regulatory parameters needed to fine-tune the rate of protein synthesis are not fully understood. We formalize the underlying bio-chemical processes in a concurrent, executable modelling language. Using object orientation and asynchronous communication yields a model that captures bio-chemical concepts in a natural manner at the level of domain experts. It can be made more fine-grained incrementally as needed. Our model is fully executable and, after calibration, can precisely simulate in vivo experiments. It can be used to predict and gain a deeper understanding of the effect of different assumptions about the working mechanisms during tetracycline-aptamer-mediated translation, such as the aptamer structure, the quantity of inserted aptamers and their position, tetracycline concentration, etc. Hence, the model can be used to design targeted experiments that test new theories. Simulations can be executed in a manner of minutes and it is easy to obtain different kinds of visualizations.

3.14 Design of Asynchronous Genetic Circuits

Chris Myers (University of Utah, US)

License  Creative Commons BY 3.0 Unported license
© Chris Myers

Researchers are now able to engineer synthetic genetic circuits for a range of applications in the environmental, medical, and energy domains. Crucial to the success of these efforts is the development of methods and tools for genetic design automation (GDA). While inspiration can be drawn from experiences with electronic design automation (EDA), design with a genetic material poses several challenges. In particular, genetic circuits are composed of very noisy components making their behavior more asynchronous, analog, and stochastic in nature. This talk presents our research in the development of the GDA tool, iBioSim, which leverages our past experiences in asynchronous circuit synthesis and formal verification to address these challenges. The iBioSim tool enables the synthetic biologist to construct models in a familiar graphical form, analyze them using a variety of methods that leverage efficient abstractions, visualize their analysis results using an intuitive interface, and ultimately synthesize a genetic implementation from a library of genetic parts. Each step of this design process utilizes standard data representation formats enabling the ready exchange of results.

3.15 Model-based multiobjective optimization framework for automated design in Synthetic Biology

Irene Otero-Muras (Spanish National Council for Scientific Research – Vigo, ES)

License  Creative Commons BY 3.0 Unported license
© Irene Otero-Muras

One of the challenges of Synthetic Biology is building genetic circuits with higher complexity, not only in terms of the number of regulatory regions involved, but also in the kind of tasks that these circuits can accomplish. I present a framework for automated design of biocircuits starting from libraries of standard parts that takes the following aspects into account: i) optimality, ii) several design criteria, iii) high computational efficiency. The method is based on optimization, and any target behaviour (adaptation, change fold detection, oscillations, pattern formation, bistability...) can be encoded in the set of objective functions. In this way can design, for example, synthetic circuits with optimal performance with respect to a given criterion, while keeping the protein cost at minimum, or oscillators with optimal tunability without compromising the stability of the limit cycle (i.e. with optimal robustness with respect to molecular noise). Our strategy exploits hybrid Mixed Integer Nonlinear Optimization Programming solvers, allowing to search simultaneously topology and parameter spaces. At present, the method relies on a deterministic description of the gene regulation dynamics. In a future work I plan to incorporate stochastic simulations exploiting a recently developed method for efficient simulation of stochastic gene regulatory networks, based on a Partial Integro-Differential Model approximation of the Chemical Master Equation.

3.16 Visual Representations for CRN Synthesis

James Scott-Brown (University of Oxford, GB)

License  Creative Commons BY 3.0 Unported license
© James Scott-Brown

In this talk I introduce TimeRails, a diagrammatic representation for formal specifications expressed in Signal Temporal Logic. This represents specifications using the visual metaphor of rectangles and rails, along which other rails or rectangles can slide. I describe how this is integrated into a tool for the automated synthesis and verification of Chemical Reaction Networks (CRNSynth). This provides a graphical interface for the user to express constraints on the network structure of a parametric CRN and specifications describing its behaviour, which are translated into a set of ordinary differential equations and constraints. Together these form a satisfaction problem modulo the theory of ODEs that can be passed to an SMT-ODE solver such as iSAT-ODE or dReach.

3.17 Synthesis of Biomolecular Circuits with Z3

Boyan Yordanov (Microsoft Research UK – Cambridge, UK)

License  Creative Commons BY 3.0 Unported license
© Boyan Yordanov

Z3 is an automated theorem prover that integrates specialized solvers for domains relevant to program analysis, testing and verification. At Microsoft, Z3 powers a variety of tools that reason about program states and transformations to improve the quality and security of software and services. Z3 also provides a powerful tool for reasoning about biological programs and biomolecular circuits, enabling the development of analysis and synthesis methods. In this talk, I will introduce our approach to encoding biological queries as Satisfiability Modulo Theories (SMT) problems that are solved using Z3. I will illustrate the approach with results from two projects involving (i) synthesizing biological interaction networks from prior knowledge and experimental data, and (ii) synthesizing chemical reaction networks (CRNs) from input/output specifications of desired computations.

3.18 SMT-based Set Synthesis for Biological Models

Paolo Zuliani (University of Newcastle, GB)

License  Creative Commons BY 3.0 Unported license
© Paolo Zuliani

for which a given system model satisfies a desired behaviour. In this talk we present BioPSy, a tool that performs guaranteed parameter set synthesis for ordinary differential equation (ODE) biological models expressed in the Systems Biology Markup Language given a desired behaviour expressed by time-series data. Three key features of BioPSy are: 1) BioPSy computes parameter intervals, not just single values; 2) for the identified intervals the model is formally guaranteed to satisfy the desired behaviour; and 3) BioPSy can handle virtually any Lipschitz-continuous ODEs, including nonlinear ones. BioPSy is able to achieve guaranteed synthesis by utilising Satisfiability Modulo Theory (SMT) solvers to determine

acceptable parameter intervals. Furthermore, BioPSy can formally check parameter estimates generated by other (non-formal) methods. We have successfully applied our tool to several biological models including a prostate cancer therapy model, a human starvation model, and a cell cycle model.

4 Working groups

After initial presentations from Monday to Wednesday, we identified in a discussion session which topics are most relevant and pressing to advance the field of synthesis methods in synthetic biology. We finally voted on proposed topics and decided in the following three: (1) “Modeling Context-Dependency of Synthetic Circuits”, (2) “Metrology in Synthetic Biology” and (3) “Formal Specification for Biological Circuit Synthesis”. We then split up in working groups and discussed the respective topics in details. In order to consolidate the findings every group was asked to prepare a short presentation in the plenum on Friday morning. In the following we briefly summarize the findings.

For topic (1), the group converged after an initial discussion on the problem of weak termination as it was identified as a major source of context-dependency. In particular, the group investigated how terminators can structurally be optimized and whether biophysical folding models and machine learning can be combined to generate new terminator libraries that can be used in design automation.

For topic (2), a smaller working group considered the concrete problem of converting fluorescence data from flow cytometers into absolute copy number of fluorescent proteins by using commercially available reference beads. They used a reference dataset provided by participant Jake Beal.

For topic (3), the working group identified different classes of specifications for circuits, e.g., static versus dynamic properties, Boolean versus real-valued or whether a cost model is specified or not. Accordingly, three main dimensions for a specification were determined: From functional to temporal, from qualitative to quantitative, from robustness not specified to robustness specified. Furthermore a synthesis challenge was conceived and programming was performed throughout the whole evening. One had to design a two-input reaction system that oscillates between a NOR gate and OR gate in a pre-specified time period. The informal challenge was won by Francois Fages using his tool BIOCHAMP. A reaction system with 13 reactions was found semi-automatically that realized this function.

Participants

- Aaron Adler
BBN Technologies –
Cambridge, US
- Ben Barak
Tel Aviv, IL
- Chris Barnes
University College London, GB
- Jacob Beal
BBN Technologies –
Cambridge, US
- Yaakov Benenson
ETH Zürich – Basel, CH
- Milan Ceska
Brno University of
Technology, CZ
- Neil Dalchau
Microsoft Research UK –
Cambridge, GB
- Sara-Jane Dunn
Microsoft Research UK –
Cambridge, GB
- François Fages
INRIA Saclay –
Île-de-France, FR
- Eric Fanchon
TINC-IMAG Lab –
La Tronche, FR
- Thomas Gorochowski
University of Bristol, GB
- Maleen Hanst
TU Darmstadt, DE
- Nathan Hillson
jibe – Emeryville, US
- Johannes Kabisch
TU Darmstadt, DE
- Heinz Koepl
TU Darmstadt, DE
- Jan Madsen
Technical University of
Denmark – Lyngby, DK
- Oded Maler
VERIMAG – Grenoble, FR
- Gareth Molyneux
University of Oxford, GB
- Radu Muschevici
TU Darmstadt, DE
- Chris J. Myers
University of Utah, US
- Irene Otero-Muras
CSIC – Vigo, ES
- James Scott-Brown
University of Oxford, GB
- Boyan Yordanov
Microsoft Research UK –
Cambridge, GB
- Paolo Zuliani
University of Newcastle, GB



Data Consistency in Distributed Systems: Algorithms, Programs, and Databases

Edited by

Annette Bieniusa¹, Alexey Gotsman², Bettina Kemme³, and Marc Shapiro⁴

1 TU Kaiserslautern, DE, bieniusa@cs.uni-kl.de

2 IMDEA Software – Madrid, ES, alexey.gotsman@imdea.org

3 McGill University – Montreal, CA, kemme@cs.mcgill.ca

4 Sorbonne-Université – LIP6 & Inria – Paris, FR, marc.shapiro@acm.org

Abstract

For decades distributed computing has been mainly an academic subject. Today, it has become mainstream: our connected world demands applications that are inherently distributed, and the usage of shared, distributed, peer-to-peer or cloud-computing infrastructures are increasingly common. However, writing distributed applications that are both correct and well distributed (e.g., highly available) is extremely challenging.

In fact, there exists a fundamental trade-off between data consistency, availability, and the ability to tolerate failures. This trade-off has implications on the design of the entire distributed computing infrastructure, including storage systems, compilers and runtimes, application development frameworks and programming languages. Unfortunately, this also has significant implications on the programming model exposed to the designers and developers of applications. We need to enable programmers who are not experts in these subtle aspects to build distributed applications that remain correct in the presence of concurrency, failures, churn, replication, dynamically-changing and partial information, high load, absence of a single line of time, etc.

This Dagstuhl Seminar proposes to bring together researchers and practitioners in the areas of distributed systems, programming languages, verifications, and databases. We would like to understand the lessons learnt in building scalable and correct distributed systems, the design patterns that have emerged, and explore opportunities for distilling these into programming methodologies, programming tools, and languages to make distributed computing easier and more accessible.

Main issues in discussion:

Application writers are constantly making trade-offs between consistency and availability. What kinds of tools and methodologies can we provide to simplify this decision making? How does one understand the implications of a design choice? Available systems are hard to design, test and debug. Do existing testing and debugging tools suffice for identifying and isolating bugs due to weak consistency? How can these problems be identified in production using live monitoring? Can we formalize commonly desired (generic) correctness (or performance) properties? How can we teach programmers about these formalisms and make them accessible to a wide audience? Can we build verification or testing tools to check that systems have these desired correctness properties? How do applications achieve the required properties, while ensuring adequate performance, in practice? What design patterns and idioms work well? To what degree can these properties be guaranteed by the platform (programming language, libraries, and runtime system)? What are the responsibilities of the application developer, and what tools and information does she have?

Seminar February 25–March 2, 2018 – <https://www.dagstuhl.de/18091>

2012 ACM Subject Classification Information systems → Database design and models, Information systems → Data structures, Information systems → Storage replication, Computer



Except where otherwise noted, content of this report is licensed under a Creative Commons BY 3.0 Unported license

Data Consistency in Distributed Systems: Algorithms, Programs, and Databases, *Dagstuhl Reports*, Vol. 8, Issue 02, pp. 101–121

Editors: Annette Bieniusa, Alexey Gotsman, Bettina Kemme, and Marc Shapiro



Dagstuhl Reports

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

systems organization → Parallel architectures, Computer systems organization → Distributed architectures

Keywords and phrases consistency, CRDTs, Distributed Algorithms, distributed computing, Distributed Systems, partitioning, replication, Strong Consistency, transactions, Weak Consistency

Digital Object Identifier 10.4230/DagRep.8.2.101

Edited in cooperation with Manuel Bravo

1 Executive Summary

Annette Bieniusa (TU Kaiserslautern, DE)

Alexey Gotsman (IMDEA Software – Madrid, ES)

Bettina Kemme (McGill University – Montreal, CA)

Marc Shapiro (Sorbonne-Université – LIP6 & Inria – Paris, FR)

License © Creative Commons BY 3.0 Unported license

© Annette Bieniusa, Alexey Gotsman, Bettina Kemme, and Marc Shapiro

Large-scale distributed systems have become ubiquitous, and there are a variety of options to develop, deploy, and operate such applications. Typically, this type of application is data-centric: it retrieves, stores, modifies, forwards, and processes data from different sources. However, guaranteeing availability, preventing data loss, and providing efficient storage solutions are still major challenges that a growing number of programmers are facing when developing large-scale distributed systems. In our seminar, we brought together academic and industrial researchers and practitioners to discuss the status quo of data consistency in distributed systems. As result of talks and discussions, we identified several topics of interest that can be grouped into the following four areas.

Theoretical foundations: The seminar included a tutorial on specification of consistency guarantees provided by distributed systems and talks on comparing different styles of specification and expressing replicated data type semantics in Datalog. Different specification styles are suitable for different purposes and more work is needed to identify the most appropriate ones. The seminar also included talks on formally reasoning about which consistency levels are enough to satisfy correctness properties of applications. The talks demonstrated that formal verification is a promising approach to cope with the challenge of selecting appropriate consistency levels.

Distributed systems and database technologies: With the growing number of replicated data stores, the two fields of distributed systems and databases are moving closer together. The communities should be made more aware of each others results. A common concern in agreement, i.e., ensuring that database copies are updated correctly. Traditionally, the distributed systems community has based many of their approaches on classical consensus algorithms or looked at weaker consistency models. In contrast, database systems focused most work on 2-phase commit protocols and eager update protocols. At the same time, the database community also considered other ACID aspects that required to combine commit protocols with concurrency control protocols and recovery schemes. In the last decade however, and in particular with practical implementations of the Paxos consensus algorithms, and the use of file replication in storage systems for availability, work of the two communities has come closer together. A challenge in this context is that work that emerges from the

different communities still makes slightly different assumptions about failure and correctness models. They can often be quite subtle so that the differences are not obvious, even to the experts. And they can lead to very different approaches to find solutions. Bridging this gap in terms of understanding each other, and the implications of correctness and failure models remains a challenging task. As an example, the separation of the concepts of atomicity, isolation and durability in the database world offers many opportunities for optimization, but includes extra complexity when analyzing which algorithms are appropriate in which situations.

Conflict-handling in highly-scalable systems: In the last years, conflict-free replicated data types (CRDTs) have been adopted by an ever-growing number of products and companies to deal with high-availability requirements under concurrent modifications of data. Recent advances in related techniques for collaborative editing might make it possible that hundreds of people work together on a shared document or data item with limited performance impact. Several talks presented programming guidelines, static analyses, and related tools for safe usage of CRDTs in situations where eventual consistency is not enough to maintain application invariants.

Programming models for distributed systems: Micro-services have become a standard approach for constructing large-scale distributed systems, though microservice composition and scalability raises a lot of questions. Some presentations discussed current work on actor-based and data-flow programming. Design for testability and test frameworks are crucial for providing reliable services, but they currently require a lot of experience as of today. We believe that future progress on programming models and new results in theoretical foundations will help to simplify this challenging task and support programmers in building safe systems.

2 Table of Contents

Executive Summary

Annette Bieniusa, Alexey Gotsman, Bettina Kemme, and Marc Shapiro 102

Overview of Talks

Does your fault-tolerant distributed system tolerate faults?	
<i>Peter Alvaro</i>	106
The FuzzyLog Approach to Building Distributed Services	
<i>Mahesh Balakrishnan</i>	106
Highly available applications done correctly	
<i>Annette Bieniusa</i>	107
Towards Affordable Externally Consistent Guarantees for Geo-Replicated Systems	
<i>Manuel Bravo and Luis Rodrigues</i>	107
A Tutorial on Specifications for Distributed Services	
<i>Sebastian Burckhardt</i>	108
Building Elastic Micro-Services with Orleans, now Geo-Distributed	
<i>Sebastian Burckhardt</i>	108
Comparing Specification Styles for Transactional Consistency Models	
<i>Andrea Cerone</i>	108
Low Latency vs Strong Semantics in Causal Consistency: Protocols and trade-offs	
<i>Diego Didona</i>	109
Paxos on the Edge	
<i>Amr El Abbadi, Divyakant Agrawal, and Faisal Nawab</i>	110
Compositional Reasoning and Inference for Weak Isolation	
<i>Suresh Jagannathan</i>	110
Consistency Compromises at the Coalface	
<i>Brad King</i>	111
Jepsen 9: A Fsyncing Feeling	
<i>Kyle Kingsbury</i>	111
Data structures as queries: Expressing CRDTs using Datalog	
<i>Martin Kleppmann</i>	111
Staying in Sync: From Transactions to Streams	
<i>Martin Kleppmann</i>	112
Homomorphic Computation for Distributed Computing	
<i>Christopher Meiklejohn</i>	112
Massive Collaborative Editing	
<i>Pascal Molli</i>	113
External Consistency in Partial Replication without TrueTime API	
<i>Roberto Palmieri, Masoomah Javidi Kishi, and Sebastiano Peluso</i>	113
Programming Scalable Cloud Services	
<i>Gustavo Petri, Patrick Eugster, Srivatsan Ravi, Masoud Saeida Ardekani, and Bo Sang</i>	114

Enforcing SQL constraints in Weakly Consistent Databases <i>Nuno Preguica</i>	114
Isolation Level Analysis <i>Sebastian Schweizer, Annette Bieniusa, Keijo Heljanko, Roland Meyer, and Arnd Poetzsch-Heffter</i>	115
Just-Right Consistency: As available as possible, consistent when necessary, correct by design <i>Marc Shapiro, Annette Bieniusa, Christopher Meiklejohn, Nuno Preguica, and Valter Balesgas</i>	115
Fast State-Machine Replication via Monotonic Generic Broadcast <i>Pierre Sutra</i>	116
Robust (Parallel) Snapshot Isolation <i>Viktor Vafeiadis</i>	116
Elements of a unified semantics for synchronization-free programming based on Lasp and Antidote <i>Peter Van Roy</i>	117
Working groups	
“Theory and Practice” working group report <i>Carlos Baquero and Carla Ferreira</i>	117
Theory vs Practice: are we developing the right models or systems? <i>Khuzaima Daudjee</i>	118
Where Do We Go Next <i>Kyle Kingsbury</i>	119
Participants	121

3 Overview of Talks

3.1 Does your fault-tolerant distributed system tolerate faults?

Peter Alvaro (University of California – Santa Cruz, US)

License © Creative Commons BY 3.0 Unported license
© Peter Alvaro

Joint work of Peter Alvaro, Josh Rosen, Joseph M. Hellerstein, Kolton Andrus

Main reference Peter Alvaro, Kolton Andrus, Chris Sanden, Casey Rosenthal, Ali Basiri, Lorin Hochstein: “Automating Failure Testing Research at Internet Scale”, in Proc. of the Seventh ACM Symposium on Cloud Computing, Santa Clara, CA, USA, October 5-7, 2016, pp. 17–28, ACM, 2016.

URL <http://dx.doi.org/10.1145/2987550.2987555>

Large-scale distributed systems must be built to anticipate and mitigate a variety of hardware and software failures. In order to build confidence that fault-tolerant systems are correctly implemented, an increasing number of large-scale sites regularly run failure drills in which faults are deliberately injected in production or staging systems. While fault injection infrastructures are becoming relatively mature, existing approaches either explore the combinatorial space of potential failures randomly or exploit the “hunches” of domain experts to guide the search. Random strategies waste resources testing “uninteresting” faults, while programmer-guided approaches are only as good as the intuition of a programmer and only scale with human effort.

In this talk, I will present intuition, experience and research directions related to lineage-driven fault injection (LDFI), a novel approach to automating failure testing. LDFI utilizes existing tracing or logging infrastructures to work backwards from good outcomes, identifying redundant computations that allow it to aggressively prune the space of faults that must be explored via fault injection. I will describe LDFI’s theoretical roots in the database research notion of provenance, present early results from the field, and present opportunities for near- and long-term future research.

3.2 The FuzzyLog Approach to Building Distributed Services

Mahesh Balakrishnan (Yale University – New Haven, US)

License © Creative Commons BY 3.0 Unported license
© Mahesh Balakrishnan

Control plane applications such as coordination services, SDN controllers, filesystem namespaces, and big data schedulers have strong requirements for consistency as well as performance. Building such applications is currently a black art, requiring a slew of complex distributed protocols that are inefficient when layered and difficult to combine. The shared log approach achieves simplicity for distributed applications by replacing complex protocols with a single shared log; however, it does so by introducing a global ordering over all updates in the system, which can be expensive, unnecessary, and sometimes impossible. We propose the FuzzyLog abstraction, which provides applications the simplicity of a shared log without its drawbacks. The FuzzyLog allows applications to construct and access a durable, iterable partial order of updates in the system. FuzzyLog applications retain the simplicity of their shared log counterparts while extracting parallelism, providing a range of consistency guarantees and tolerating network partitions. In effect, the FuzzyLog is a democratizing abstraction for building scalable, robust distributed systems.

3.3 Highly available applications done correctly

Annette Bieniusa (*TU Kaiserslautern, DE*)

License © Creative Commons BY 3.0 Unported license
© Annette Bieniusa

Joint work of Mathias Weber, Peter Zeller, Arnd Poetzsch-Heffter

Main reference Mathias Weber, Annette Bieniusa, Arnd Poetzsch-Heffter: “EPTL – A Temporal Logic for Weakly Consistent Systems (Short Paper)”, in Proc. of the Formal Techniques for Distributed Objects, Components, and Systems – 37th IFIP WG 6.1 International Conference, FORTE 2017, Held as Part of the 12th International Federated Conference on Distributed Computing Techniques, DisCoTec 2017, Neuchâtel, Switzerland, June 19–22, 2017, Proceedings, Lecture Notes in Computer Science, Vol. 10321, pp. 236–242, Springer, 2017.

URL http://dx.doi.org/10.1007/978-3-319-60225-7_17

The construction of highly available applications poses challenges to developers and software architects. Reasoning about the correctness of such systems requires special care when it comes to security aspects.

In our talk, we discuss different aspects that arise in the practise of developing highly available systems in the context of the AntidoteDB, a highly-available transactional CRDT data store. We will show how programmers can use the Repliss tool for specifying the semantics of their programs and check what type of consistency is required for maintaining invariants in their application. Further, we introduce a novel temporal, event-based parallel temporal logic (EPTL) that allows to specify weakly-consistent systems. In contrast to temporal logics like LTL or CTL, EPTL can model semantics of components that are truly concurrent while abstracting from implementation and communication details such as causality tracking mechanisms. As a third contribution, we present an access control mechanism for providing secure access to data items under causal consistency together with its specification EPTL.

3.4 Towards Affordable Externally Consistent Guarantees for Geo-Replicated Systems

Manuel Bravo (*INESC-ID – Lisbon, PT*) and Luis Rodrigues (*INESC-ID – Lisbon, PT*)

License © Creative Commons BY 3.0 Unported license
© Manuel Bravo and Luis Rodrigues

Cloud services’s designers are faced with a dilemma: either favor low latency adopting weaker consistency models such as eventual and causal consistency; or favor strong consistency imposing higher latency responses. A promising approach to alleviate the tension between semantics and performance consists in allowing multiple consistency levels to coexist.

We propose a novel consistency model, namely external causality, that takes causal consistency and spice it up with affordable externally consistent guarantees. The idea behind external causality is that most operations, namely internal operations, are executed locally (in a single site) and asynchronously replicated. Nevertheless, stronger operations called external operations, which provide externally consistent guarantees, coexist with internal operations. An external operation is ordered after any other operation—both internal and externals—already successfully installed in the system as of the time the external operation began. External operations allow developers to make stronger assumptions. Our hope is that external causality can potentially simplify the development of applications without compromising performance.

3.5 A Tutorial on Specifications for Distributed Services

Sebastian Burckhardt (Microsoft Research – Redmond, US)

License © Creative Commons BY 3.0 Unported license
© Sebastian Burckhardt

URL <https://1drv.ms/f/s!AgoBH3oDy8d-ilh86SHxOpzKAbRdfg>

Applications are increasingly developed as a composition of services, with advanced distributed protocols hidden beneath simple service APIs. Any service (whether it is cloud storage, an advanced CRDTs, or an application-defined microservice) must however somehow specify a semantics under concurrent and/or distributed accesses, which is nontrivial in the presence of consistency relaxations, such as lazy replication and asynchronous update propagation. In this tutorial, I give an introduction to an advanced specification methodology for service semantics, how it can be used to specify the behavior of typical CRDTs and collaborative editing, and how it has helped us to clarify the terminology and prove correctness and optimality of implementations.

3.6 Building Elastic Micro-Services with Orleans, now Geo-Distributed

Sebastian Burckhardt (Microsoft Research – Redmond, US)

License © Creative Commons BY 3.0 Unported license
© Sebastian Burckhardt

Joint work of Philip A. Bernstein, Sebastian Burckhardt, Sergey Bykov, Natacha Crooks, Jose M. Faleiro, Gabriel Klot, Alok Kumbhare, Muntasir Raihan Rahman

Main reference Philip A. Bernstein, Sebastian Burckhardt, Sergey Bykov, Natacha Crooks, Jose M. Faleiro, Gabriel Klot, Alok Kumbhare, Muntasir Raihan Rahman, Vivek Shah, Adriana Szekeres, Jorgen Thelin: “Geo-distribution of actor-based services”, PACMPL, Vol. 1(OOPSLA), pp. 107:1–107:26, 2017.

URL <http://dx.doi.org/10.1145/3133931>

Virtual actor frameworks, such as the Orleans system, have proven quite useful to build elastically scalable micro-services. However, it is not a priori clear how to use them in a geo-distributed setting with high communication latency. To this end, we have developed 2 extensions to the model, one with and one without actor replication. The replicated version, which supports reading and updating with a choice of linearizable and eventual consistency. Our evaluation on several workloads shows the advantage of offering varying configuration choices: for example, replication can provide fast, always-available reads and updates globally, while batching of linearizable storage accesses at a single location can boost the throughput of an order processing workload by 7x.

3.7 Comparing Specification Styles for Transactional Consistency Models

Andrea Cerone (Imperial College London, GB)

License © Creative Commons BY 3.0 Unported license
© Andrea Cerone

We compare three different frameworks for specifying weak consistency models of databases whose transactions enjoy atomic visibility.

The first framework allows for declarative specifications of consistency models by placing constraints, or axioms, over abstract executions. Abstract executions were originally introduced by Burckhardt et al. [1].

The second framework is based on Adya’s dependency graphs [3]: consistency models are specified by considering only those dependency graphs that forbid cycles of a certain form. I show that, for a particular class of specifications of consistency models given in terms of abstract executions, it is possible to automatically infer an acyclicity condition that captures the same consistency models using dependency graphs; such an acyclicity condition is encoded as an irreflexive relation in a recursive variant of Tarki’s calculus of binary relations, with transaction dependencies as ground terms. Complete details of this result are given in [2]. I also conjecture that, in the general case, Tarki’s calculus of binary relations is not expressive enough to capture consistency models that can be specified using axioms over abstract executions.

The third framework is based on a novel notion of history heaps, which I recently developed together with P. Gardner and S. Xiong. History heaps record, for each object in the database, the whole list of versions that have been written by transactions for such an object; versions also contain the meta-data corresponding to the transactions that accessed such a version. Consistency models are specified in an operational way: history heaps are used to encode an abstract view of the state of the database accessed by transactions, while a transition relation between history heaps describes how the system may evolve when executing a transaction. I show that specifications of consistency models using dependency graphs can be easily converted into equivalent specifications given in terms of history heaps.

References

- 1 S. Burckhardt, D. Leijen, M. Fähndrich, M. Sagiv. *Eventually Consistent Transactions*. ESOP, 2012.
- 2 A. Cerone, A. Gotsman, H. Yang. *Algebraic Laws for Weak Consistency*. CONCUR, 2017.
- 3 A. Adya. *Weak Consistency: A Generalized Theory and Optimistic Implementations for Distributed Transactions*. Ph.D. Thesis, MIT 1999.

3.8 Low Latency vs Strong Semantics in Causal Consistency: Protocols and trade-offs

Diego Didona (EPFL – Lausanne, CH)

License © Creative Commons BY 3.0 Unported license
© Diego Didona

Joint work of Willy Zwaenepoel, Jingjing Wang, Rachid Guerraoui

Main reference Diego Didona, Rachid Guerraoui, Jingjing Wang, Willy Zwaenepoel: “Causal Consistency and Latency Optimality: Friend or Foe?”, CoRR, Vol. abs/1803.04237, 2018.

URL <http://arxiv.org/abs/1803.04237>

Causal consistency is appealing because it is among the strongest consistency levels compatible with availability, and avoids the performance penalties incurred by strongly consistent systems. Yet existing causal consistency designs either sacrifice scalability or low latency to support stronger semantics, e.g., generic read-write transactions or read-only transactions.

In this talk we will present scalable approaches to achieve low latency by means of nonblocking read operations. These approaches apply to systems that support generic and read-only transactions. Then, we will analyze so called “latency optimal” read-only transaction designs. We find that, surprisingly, latency-optimal designs can perform worse

than non-optimal ones. To explain this result, we will present a theorem that shows that latency-optimal read-only transactions incur an unavoidable overhead that grows with the number of clients, thus reducing the overall system efficiency.

3.9 Paxos on the Edge

Amr El Abbadi (University of California – Santa Barbara, US), Divyakant Agrawal, and Faisal Nawab

License © Creative Commons BY 3.0 Unported license

© Amr El Abbadi, Divyakant Agrawal, and Faisal Nawab

Joint work of Faisal Nawab, Divyakant Agrawal, Amr El Abbadi

Main reference Faisal Nawab, Divyakant Agrawal, Amr El Abbadi: “DPaxos: Managing Data Closer to Users for Low-Latency and Mobile Applications”, in Proc. of the 2018 International Conference on Management of Data, SIGMOD Conference 2018, Houston, TX, USA, June 10-15, 2018, pp. 1221–1236, ACM, 2018.

URL <http://dx.doi.org/10.1145/3183713.3196928>

The utilization of edge nodes is inevitable for the success and growth of many emerging low latency and mobile applications. In this talk, we will explore a novel Paxos-based consensus protocol that manages access to partitioned data across globally-distributed datacenters and edge nodes. The main objective is to reduce the latency of serving user requests, while ensuring fault-tolerance and adapting gracefully to mobility. These goals are achieved by proposing changes to the traditional Paxos protocol that reduce the size of quorums needed to serve requests and to react to failures and mobility.

3.10 Compositional Reasoning and Inference for Weak Isolation

Suresh Jagannathan (Purdue University – West Lafayette, US)

License © Creative Commons BY 3.0 Unported license

© Suresh Jagannathan

Joint work of Gowtham Kaki, Kartik Nagar, Mahsa Najafzadeh

Main reference Gowtham Kaki, Kartik Nagar, Mahsa Najafzadeh, Suresh Jagannathan: “Alone together: compositional reasoning and inference for weak isolation”, PACMPL, Vol. 2(POPL), pp. 27:1–27:34, 2018.

URL <http://dx.doi.org/10.1145/3158115>

Serializability is a desirable correctness property that simplifies reasoning about concurrently executing transactions. But, on weakly consistent distributed stores, serializability cannot be achieved without sacrificing availability, an unpalatable trade-off for many applications. Consequently, applications typically choose to weaken the strong isolation guarantees afforded by serializability in favour of weaker, albeit more available, variants. In this talk, I’ll present some recent work on a verification methodology for reasoning about weakly-isolated transactions, and an inference procedure that determines the weakest isolation level that can be ascribed to transactions without violating an application’s high-level invariants. The key to effective inference is the observation that weakly-isolated transactions can be viewed as functional (monadic) computations over an abstract database state, allowing us to treat their operations as state transformers over the database. This interpretation enables automated verification using off-the-shelf SMT solvers.

References

- 1 Gowtham Kaki, Kartik Nagar, Mahsa Najafzadeh, Suresh Jagannathan, “Alone Together: Compositional Reasoning and Inference for Weak Isolation”. PACMPL 2(POPL): 27:1-27:34 (2018).

3.11 Consistency Compromises at the Coalface

Brad King (Scality – Paris, FR)

License © Creative Commons BY 3.0 Unported license
© Brad King

Consistency is always desirable but comes at a cost. The path taken to find acceptable consistency compromises for a multi-petabyte scale storage platform will be discussed. The Scality storage platform uses an appealing shared-nothing architecture which has excellent scaling characteristics, but cannot reliably handle many workloads without some form of coordination to provide consistency guarantees. The basic architecture, the challenges faced, the tools chosen and ongoing work will be presented. The current platform has several different approaches used in combination including: flat group quorums, Paxos, Raft and Totem based protocols. The constraints of working in production environments with mission critical applications presents challenges in combining performance, correctness and reliability while continuing to evolve the technology will be considered. Among other challenges, sufficient testing of the possible degraded and partitioned situations can become an intractable problem.

References

- 1 Bradley King *Consistency Compromises at the Coalface*. Scality, 11 rue Tronchet, 75008 Paris France

3.12 Jepsen 9: A Fsyncing Feeling

Kyle Kingsbury (San Francisco, US)

License © Creative Commons BY 3.0 Unported license
© Kyle Kingsbury

Distributed systems often claim to save our data durably, to provide isolated transactions, to make writes visible to reads. Jepsen is a distributed systems testing harness, which applies property-based testing to databases to verify their correctness claims during common failure modes: network partitions, process crashes, and clock skew. In this talk, we discuss anomalies in Tendermint, Hazelcast, and Aerospike.

3.13 Data structures as queries: Expressing CRDTs using Datalog

Martin Kleppmann (University of Cambridge, GB)

License © Creative Commons BY 3.0 Unported license
© Martin Kleppmann

Joint work of Martin Kleppmann, Victor B. F. Gomes, Dominic P. Mulligan, Alastair R. Beresford

Currently there are two conventional formulations of CRDTs: state-based (where we prove that our merge function is commutative, associative, and idempotent) or operation-based (where we prove that the functions that apply operations to the local state are commutative). I propose a third formulation in which the CRDT is expressed as a query over a monotonically growing set of operations. The merge function for the set of operations is just the set union, which is trivially commutative, associative, and idempotent. By expressing the desired data

structure as a deterministic query over that set we get convergence automatically. I will discuss how we can use the Datalog language to express such queries, how this query-based approach can help us better understand existing CRDTs, and how it facilitates designing new ones.

3.14 Staying in Sync: From Transactions to Streams

Martin Kleppmann (University of Cambridge, GB)

License © Creative Commons BY 3.0 Unported license
© Martin Kleppmann

Main reference Martin Kleppmann: “Designing Data-Intensive Applications: The Big Ideas Behind Reliable, Scalable, and Maintainable Systems”, O’Reilly, 2016.

URL <http://shop.oreilly.com/product/0636920032175.do>

For the very simplest applications, a single database is sufficient, and then life is pretty good. But as your application needs to do more, you often find that no single technology can do everything you need to do with your data. And so you end up having to combine several databases, caches, search indexes, message queues, analytics tools, machine learning systems, and so on, into a heterogeneous infrastructure. . .

Now you have a new problem: your data is stored in several different places, and if it changes in one place, you have to keep it in sync in the other places, too. It’s not too bad if all your systems are up and running smoothly, but what if some parts of your systems have failed, some are running slow, and some are running buggy code that was deployed by accident?

It’s not easy to keep data in sync across different systems in the face of failure. Distributed transactions and 2-phase commit have long been seen as the “correct” solution, but they are slow and have operational problems, and so many systems can’t afford to use them.

In this talk we’ll explore using event streams and Kafka for keeping data in sync across heterogeneous systems, and compare this approach to distributed transactions: what consistency guarantees can it offer, and how does it fare in the face of failure?

3.15 Homomorphic Computation for Distributed Computing

Christopher Meiklejohn (UC Louvain, BE)

License © Creative Commons BY 3.0 Unported license
© Christopher Meiklejohn

State-of-the-art programming models for building coordination-free distributed applications typically rely on a combination of lattice-based programming with monotonic application logic. As the CALM result has demonstrated, these programs guarantee convergence in the face of various network anomalies such as message reordering and message duplication. However, two of the systems that represent the state-of-the-art, Lasp [1] and BloomL [2], each place the onus on the developer of a.) modeling their application state as join-semilattices, and b.) ensuring that computations in application code are monotone. Furthermore, these programming models can take advantage of homomorphisms, a special case of monotone programming where function application distributes over the join, to provide incremental computing: key to applications that are geographically distributed.

In this talk, we present a work-in-progress result on writing correct monotone programs with join-semilattices. This framework generalizes the representation for lattice-based data types, provides a type system approach to ensuring monotonicity, and provides a mechanism for automatically lifting monotone functions to homomorphic functions between lattices. We present an operational semantics for an incremental evaluation model, that generalizes the execution models of both Lasp and Bloom^L and demonstrate how the existing systems fit into our framework.

References

- 1 MEIKLEJOHN, C., AND VAN ROY, P. Lasp: A language for distributed, coordination-free programming. In *Proceedings of the 17th International Symposium on Principles and Practice of Declarative Programming* (2015), ACM, pp. 184–195.
- 2 Logic and lattices for distributed programming. In *Proceedings of the Third ACM Symposium on Cloud Computing*. ACM.

3.16 Massive Collaborative Editing

Pascal Molli (University of Nantes, FR)

License © Creative Commons BY 3.0 Unported license
© Pascal Molli

Distributed real-time editors made real-time editing easy for millions of users. However, main stream editors rely on Cloud services to mediate sessions raising privacy and scalability issues. Decentralized editors tackle privacy issues, but scalability issues remain. We aim to build a decentralized editor that allows real-time editing anytime, anywhere, whatever is the number of participants. In this study, we propose an approach based on a massively replicated sequence data structure that represents the shared document. We establish an original trade-off on communication, time, and space complexity to maintain this sequence over a network of browsers. We prove a sublinear upper bound on communication complexity while preserving an affordable time and space complexity. To validate this trade-off, we built a full working editor and measured its performance on large-scale experiments involving up to 600 participants. As expected, the results show a traffic increasing as $O((\log I)^2 \ln R)$ where I is the number of insertions in the document, and R the number of participants.

3.17 External Consistency in Partial Replication without TrueTime API

Roberto Palmieri (Lehigh University – Bethlehem, US), Masoomeh Javidi Kishi, and Sebastiano Peluso

License © Creative Commons BY 3.0 Unported license
© Roberto Palmieri, Masoomeh Javidi Kishi, and Sebastiano Peluso


This paper speaks about challenges of guaranteeing external consistency in a partially replicated system without any centralized synchronization component and where read-only transactions are never abort. Google Spanner establishes external consistency by leveraging the TrueTime API; in this work we replace it with a combination of vector and scalar clocks to achieve similar guarantees.

In our system, which we name SSS, write transactions commit by leveraging two-phase commit. Read-only transactions implement non-blocking execution by leaving a trace of their execution on accessed replicas so that write transactions can detect the presence of a write-after-read conflict, which forces write transaction to hold their response to client until the read-only is completed. Internally in the system, although write transaction waits for read-only transactions, their written values are already exposed to other concurrent transactions, therefore system throughput is not affected by the above delay.

Interestingly, read-only transactions notify concurrent and conflicting write transactions upon completion so that they can proceed providing the response (put on hold previously) to clients. This notification sent by read-only transactions also serves as garbage collection message to discard any left trace by a read-only transaction in the system.

3.18 Programming Scalable Cloud Services

Gustavo Petri (University Paris-Diderot, FR), Patrick Eugster, Srivatsan Ravi, Masoud Saeida Ardekani (Samsung Research – Mountain View, US), and Bo Sang


License  Creative Commons BY 3.0 Unported license

© Gustavo Petri, Patrick Eugster, Srivatsan Ravi, Masoud Saeida Ardekani, and Bo Sang

In this talk we will introduce a programming model for elastic cloud applications based on actors. Our model leverages a native notion of ownership to structure the actors at runtime. By means of this ownership, we can deliver atomic cross-actor transactions, while retaining scalability and elasticity. After presenting the programming model, we will conclude with open problems and some future directions.

3.19 Enforcing SQL constraints in Weakly Consistent Databases

Nuno Preguica (New University of Lisbon, PT)

License  Creative Commons BY 3.0 Unported license

© Nuno Preguica

Joint work of João Sousa, Valter Balegas, Sérgio Duarte, Carla Ferreira, Rodrigo Rodrigues, Subhajit Sidhanta

Weak consistency is popular in the design of geo-replicated databases. When compared with strong consistency, this approach has the advantage of allowing low latency and high availability, as operations can execute in any replica without the need to coordinate with other replicas. For working correctly, some applications need to enforce application-specific constraints, which is challenging in weakly consistent databases. In this talk, we discuss to which extent it is possible to enforce SQL constraints in such settings.

3.20 Isolation Level Analysis

Sebastian Schweizer (TU Braunschweig, DE), Annette Bieniusa (TU Kaiserslautern, DE), Keijo Heljanko, Roland Meyer, and Arnd Poetzsch-Heffter

License © Creative Commons BY 3.0 Unported license
© Sebastian Schweizer, Annette Bieniusa, Keijo Heljanko, Roland Meyer, and Arnd Poetzsch-Heffter

Modern database systems offer different isolation levels. The isolation level defines what synchronization guarantees a programmer can rely on. The choice is a trade-off between performance (weak isolation) and strong guarantees (strong isolation).

Isolation Level Analysis is an approach to compare the behavior of a database program in different isolation levels. It allows to automatically find the isolation level that is best for a specific application, i.e. it is strong enough to avoid synchronization issues but weak enough to provide good throughput. Our technique takes as input a database program and two isolation levels. It then checks whether there is an execution that is possible in the weak but not in the strong isolation level. If no such execution is found, then the database operator can safely switch to the weaker level without adding additional behavior.

3.21 Just-Right Consistency: As available as possible, consistent when necessary, correct by design

Marc Shapiro (Sorbonne-Université – LIP6 & Inria – Paris, FR), Annette Bieniusa (TU Kaiserslautern, DE), Christopher Meiklejohn (UC Louvain, BE), Nuno Preguica (New University of Lisbon, PT), and Valter Balegas

License © Creative Commons BY 3.0 Unported license
© Marc Shapiro, Annette Bieniusa, Christopher Meiklejohn, Nuno Preguica, and Valter Balegas
Main reference Marc Shapiro, Annette Bieniusa, Nuno M. Preguica, Valter Balegas, Christopher Meiklejohn: “Just-Right Consistency: reconciling availability and safety”, CoRR, Vol. abs/1801.06340, 2018.
URL <http://arxiv.org/abs/1801.06340>

In a distributed data store, the CAP theorem forces a choice between strong consistency (CP) and availability and responsiveness (AP) when the network can partition. To address this issue, we take an application-driven approach, Just-Right Consistency (JRC). JRC defines a consistency model that is sufficient to maintain the application invariants, and otherwise remaining as available as possible.

JRC leverages knowledge of the application. Two invariant-maintaining patterns, ordered updates and atomic grouping, are compatible with concurrent and asynchronous updates, orthogonally to CAP. In contrast, checking a data precondition on partitioned state is CAP-sensitive. However, if two updates do not negate each other’s precondition, they may legally execute concurrently. Updates must synchronise only if one negates the precondition of the other.

The JRC approach is supported: by the CRDT data model that ensures that concurrent updates converge; by Antidote, a cloud-scale CRDT data store that guarantees transactional causal consistency; and by developer tools (static analysers and domain-specific languages) that help guarantee invariants. This research is supported in part by FP7 SyncFree, H2020 LightKone, and by ANR project RainbowFS.

3.22 Fast State-Machine Replication via Monotonic Generic Broadcast

Pierre Sutra (Télécom SudParis – Évry, FR)

License © Creative Commons BY 3.0 Unported license
© Pierre Sutra

Joint work of Vitor Enes, Tuanir França Rezende, Alexey Gotsman, Matthieu Perrin, Pierre Sutra

This talk introduces Monotonic Generic Broadcast (MG-broadcast), a new group communication primitive enabling efficient state-machine replication. Like generic broadcast, MG-broadcast does not require ordering commutative state-machine commands. In addition, it allows a replicated state machine to serve reads from a local replica while preserving sequential consistency.

We present a protocol implementing MG-broadcast that is leaderless: commands do not have to be ordered by a single leader node, which results in better scalability and availability. Furthermore, the latency of our protocol is optimal when one failure may occur at a time, in which case an update command contacts a simple majority of processes and always completes in one round trip. This makes our protocol especially appropriate for geo-distribution.

We close this talk by presenting several empirical results that evaluate MG-broadcast in a geo-distributed setting, using 3 to 11 geographical locations. We show that, under a range of workloads, our protocol outperforms prior replicated state machine solutions.

3.23 Robust (Parallel) Snapshot Isolation

Viktor Vafeiadis (MPI-SWS – Kaiserslautern, DE)

License © Creative Commons BY 3.0 Unported license
© Viktor Vafeiadis

Joint work of Azalea Raad, Ori Lahav, Viktor Vafeiadis

Main reference Azalea Raad, Ori Lahav, Viktor Vafeiadis: “On Parallel Snapshot Isolation and Release/Acquire Consistency”, in Proc. of the Programming Languages and Systems – 27th European Symposium on Programming, ESOP 2018, Held as Part of the European Joint Conferences on Theory and Practice of Software, ETAPS 2018, Thessaloniki, Greece, April 14–20, 2018, Proceedings, Lecture Notes in Computer Science, Vol. 10801, pp. 940–967, Springer, 2018.

URL http://dx.doi.org/10.1007/978-3-319-89884-1_33

Snapshot isolation (SI) and *parallel snapshot isolation* (PSI) are two standard transactional consistency models that is used in databases and distributed systems. Since they provide much better performance than serializability, it makes sense to adopt them as a transactional models for STMs in programming languages.

In the programming language setting, however, one must crucially allow the interaction of transactional and non-transactional code. In a recent paper [1], we constructed RPSI, a robust version of PSI that is better suited for the setting. We have built a simple lock-based reference implementation of RPSI over the release-acquire fragment of the C/C++ concurrency model [2], and have proved that our implementation can exhibit exactly the same behaviour as allowed by RPSI’s declarative specification.

In ongoing work, we are looking to achieve a similar result for SI.

References

- 1 Azalea Raad, Ori Lahav, and Viktor Vafeiadis. On parallel snapshot isolation and release/acquire consistency. In ESOP 2018: 27th European Symposium on Programming, Springer, 2018.

- 2 Ori Lahav, Nick Giannarakis, and Viktor Vafeiadis. Taming release-acquire consistency. In POPL 2016: 43rd Annual ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages, pages 649-662, ACM, 2016.

3.24 Elements of a unified semantics for synchronization-free programming based on Lasp and Antidote

Peter Van Roy (UC Louvain, BE)

License © Creative Commons BY 3.0 Unported license
© Peter Van Roy

Joint work of Peter Zeller, Annette Bieniusa, Mathias Weber, Christopher Meiklejohn, Peter Van Roy, Nuno Preguiça, Carla Ferreira

Main reference LightKone: Lightweight Computations at the Edge. H2020 Project, Jan. 2017 – Dec. 2019, see lightkone.eu.

URL <https://www.lightkone.eu>

We outline a unified semantics for Lasp and Antidote based on Burckhardt’s abstract execution formalism. This semantics is the foundation for an edge computing platform that we are building in the LightKone H2020 project. The platform provides a functional programming style with an efficient implementation on large dynamic networks with unreliable nodes and communication. Lasp and Antidote are both synchronization-free programming systems, i.e., they are based on CRDTs, which are distributed data structures that provide consistency between replicas using a weak synchronization model, namely eventual replica-to-replica communication. Lasp and Antidote are situated in two very different areas of the synchronization-free design space. Lasp is a distributed dataflow system implemented on large dynamic networks. Antidote is a causally consistent transactional database implemented on georeplicated data centers. The unified semantics brings together two communities and will help us make progress in synchronization-free programming.

4 Working groups

4.1 “Theory and Practice” working group report

Carlos Baquero (University of Minho – Braga, PT) and Carla Ferreira (New University of Lisbon, PT)

License © Creative Commons BY 3.0 Unported license
© Carlos Baquero and Carla Ferreira

On February 27 we split into two groups to discuss the interplay among theory and practice in distributed systems. Our group had twelve participants and the initial discussion was sparked by examples on highly available editable sequences, where there is a contrast between known lower bounds on metadata size and, in contrast, the practical need to continue research on efficient solutions for average case scenarios. We looked for more examples that reflected this tension, and discussed how the FLP results lead, for a while, to a decrease in research on asynchronous consensus algorithms, until Paxos finally re-surfaced. The discussion then evolved into how difficult it can be for practitioners to navigate the spectrum of system models and algorithms for solving specific tasks. Finally we came up with the following two take home messages, resulting from the discussion:

- Lower bounds and impossibility results are important navigation tools, but if taken too generally they can limit research on contexts that might seem covered by those results. E.g. multi-master sequence editing, FLP vs Paxos algorithms, vector clocks size lower bounds and scalable causality research.
- Consistency models have a complex taxonomy, it is hard to expect users to navigate that correctly. A flowchart or guided navigation, a wiki taxonomy, could help users choose the best tools/algorithms for the aimed setting. The end effect of uncharted complexity can lead practitioners to chose stronger consistency than needed or move to the other end of the spectrum and choose basic key-value stores.

4.2 Theory vs Practice: are we developing the right models or systems?

Khuzaima Daudjee (University of Waterloo, CA)

License  Creative Commons BY 3.0 Unported license
© Khuzaima Daudjee

The group of participants ranged from researchers working on aspects related to verification and formalizing notions of consistency to managing data to building industrial systems.

Several questions/issues were raised including how is the theory relevant to people developing systems. Some views that were shared:

- systems people need to build systems that work
- practitioners are quite interested in understanding how things work and correctness of protocols; in fact there is a demand for these

Researchers expressed concern about what stops theoretical notions from working that included:

- strong assumptions
- relevant papers are badly written
- terminology is inconsistent
- protocols are not efficient when implemented
- need to mathematically verify guarantees that papers propose

There was concern over jargon and imprecision in the use of technical terms from researchers from different communities. Some expressed hope that “equivalence classes” will develop and terms will solidify and converge. Often, programmers do not understand the differences between the different terms and the consistency levels they offer.

It can really help when someone implements a theory to make its use and understanding concrete. Oracle delivered SI as serializable isolation; this is a good example of how practice accelerated acceptance into theory. Well-specified protocols help gain acceptance, e.g., Lamport’s Paxos.

It takes time for adoption of ideas and concepts into tools. Maybe what we need are a few consistency models that are used as gold standards.

4.3 Where Do We Go Next

Kyle Kingsbury (San Francisco, US)

License  Creative Commons BY 3.0 Unported license
© Kyle Kingsbury

We discussed the future of distributed consistency—what research directions appear most promising? What has failed, or gone under-explored? And more importantly, where do we **need** to be?

We start by recalling that the **last** Next Big Thing was going to be dataflow. This did not quite come to pass—we believe, in part, because our field has abandoned monolithic approaches in favor of heterogenous systems. However, we remain **unhappy** with system building. The systems approach has come to dominate industry, but the compositional **rules** for those system components are poorly understood.

In addition, layering has sometimes proven **weaker** than integrated wholes. TAPIR, for instance, suggests that we can build faster transactional systems by giving **up** ordering at lower layers—in exchange for a more complex transaction system on top. That said, it’s nice to know what the layers **are** before we start breaking them.

We suffer from a lack of uniform abstractions for consistency problems, both in describing safety models, but also the **performance** of systems, and their compositional rules. We would like to see a language with which one could describe a given system’s abstract behavior, and prove how composing it with a second system would provide, or fail to provide, important invariants such as isolation and fault tolerance. This specification language would need to be usable by (at least some) engineers.

What would be the semantics of the interfaces between systems, in such a language?

Perhaps a focus on request-response patterns and their relationships would be useful? This cannot be the whole story, because many problems, like streaming or materialized view maintenance, cannot be represented easily in terms of RPC. Perhaps (temporally qualified) relations between system states might prove useful. A cache, for instance, should reflect, at some time, a state of the database it draws from. Perhaps a process algebra would be more useful?

Moreover, we don’t just want to compose. We want to impose **restrictions**, like access control. We want to **hide** things behind abstraction boundaries. Can a language encode these things?

Given the success of weakly safe systems glued together from heterogenous components, we suspect that building “cathedral-style” languages or databases, mean to encompass everything programmers might need to do in a distributed system, is likely doomed; there are a host of technical and social reasons that drive adoption, and monolithic designs are resistant to change and difficult to adapt to new contexts. Perhaps what we need are **models** or **patterns** for building distributed computation: MapReduce, for example, has been a successful model implemented in several ways across industry.

Another such model which has **not** been broadly adopted might be metadata for causality tracking. Vector clocks, causal tokens, idempotence tokens, request IDs in distributed tracing like Zipkin, and causally consistent timestamps: these seem like patterns that could be formalized and shared between components. It would be nice if, say, a Riak vector clock could be passed as a causal token into some other database to ensure a query includes data reflective of that vclock.

There are, of course, lots of ways to implement causality. We could use explicit vector clocks vs implicit session or global orders. We could leave this unspecified, or offer extension points. As new types of causal tokens are identified, the language should grow to accommodate them.

Some may claim that we have never reaped the promised benefits of model or standards re-use. However, some successful ideas *have* proven remarkably successful. Libraries and programming languages are widely re-used. Unix pipes and the concept of files remain ubiquitous. Proxies, caches, and load balancers are well-understood patterns now, and all cooperate beautifully in the case of HTTP. HTTP (and REST) itself has proven successful through its use of standardized *and* extensible headers, providing a language for heterogenous components to cooperate. Perhaps we can learn from their example.

Participants

- Peter Alvaro
University of California –
Santa Cruz, US
- Mahesh Balakrishnan
Yale University – New Haven, US
- Carlos Baquero
University of Minho – Braga, PT
- Annette Bieniusa
TU Kaiserslautern, DE
- Ahmed Bouajjani
University Paris-Diderot, FR
- Manuel Bravo
INESC-ID – Lisbon, PT
- Sebastian Burckhardt
Microsoft Research –
Redmond, US
- Andrea Cerone
Imperial College London, GB
- Gregory Chockler
Royal Holloway, University of
London, GB
- Khuzaima Daudjee
University of Waterloo, CA
- Diego Didona
EPFL – Lausanne, CH
- Amr El Abbadi
University of California –
Santa Barbara, US
- Carla Ferreira
New University of Lisbon, PT
- Alexey Gotsman
IMDEA Software – Madrid, ES
- Suresh Jagannathan
Purdue University –
West Lafayette, US
- Bettina Kemme
McGill University –
Montreal, CA
- Brad King
Scality – Paris, FR
- Kyle Kingsbury
San Francisco, US
- Martin Kleppmann
University of Cambridge, GB
- Christopher Meiklejohn
UC Louvain, BE
- Roland Meyer
TU Braunschweig, DE
- Maged M. Michael
Facebook – New York, US
- Pascal Molli
University of Nantes, FR
- Roberto Palmieri
Lehigh University –
Bethlehem, US
- Matthieu Perrin
University of Nantes, FR
- Gustavo Petri
University Paris-Diderot, FR
- Nuno Preguica
New University of Lisbon, PT
- Luis Rodrigues
INESC-ID – Lisbon, PT
- Rodrigo Rodrigues
INESC-ID – Lisbon, PT
- Masoud Saeida Ardekani
Samsung Research – Mountain
View, US
- Sebastian Schweizer
TU Braunschweig, DE
- Marc Shapiro
University Pierre & Marie Curie –
Paris, FR
- Pierre Sutra
Télécom SudParis – Évry, FR
- Viktor Vafeiadis
MPI-SWS – Kaiserslautern, DE
- Peter Van Roy
UC Louvain, BE



The Logical Execution Time Paradigm: New Perspectives for Multicore Systems

Edited by

Rolf Ernst¹, Stefan Kuntz², Sophie Quinton³, and Martin Simons⁴

1 TU Braunschweig, DE, ernst@ida.ing.tu-bs.de

2 Continental Automotive GmbH - Regensburg, DE,
stefan.kuntz@continental-corporation.com

3 INRIA - Grenoble, FR, sophie.quinton@inria.fr

4 Daimler Research - Stuttgart, DE, martin.simons@daimler.com

Abstract

This report documents the program and the outcomes of Dagstuhl Seminar 18092 “The Logical Execution Time Paradigm: New Perspectives for Multicore Systems”. The seminar brought together academic and industrial researchers working on challenges related to the Logical Execution Time Paradigm (LET). The main purpose was to promote a closer interaction between the sub-communities involved in the application of LET to multicore systems, with a particular emphasis on the automotive domain.

Seminar February 25–28, 2018 – <https://www.dagstuhl.de/18092>

2012 ACM Subject Classification Computer systems organization → Embedded systems, Computer systems organization → Real-time systems, Software and its engineering → Scheduling

Keywords and phrases Automotive domain, logical execution time, multicore architectures, real-time systems

Digital Object Identifier 10.4230/DagRep.8.2.122

Edited in cooperation with Borislav Nikolić

1 Executive Summary

Rolf Ernst (TU Braunschweig, DE)

Stefan Kuntz (Continental Automotive GmbH - Regensburg, DE)

Martin Simons (Daimler Research - Stuttgart, DE)

Borislav Nikolić (TU Braunschweig, DE)

Sophie Quinton (INRIA - Grenoble, FR)

Hermann von Hasseln (Daimler Research - Stuttgart, DE)

License © Creative Commons BY 3.0 Unported license

© Rolf Ernst, Stefan Kuntz, Martin Simons, Borislav Nikolić, Sophie Quinton, and Hermann von Hasseln

The Logical Execution Time (LET) abstraction, which was originally introduced as a real-time programming paradigm, has gained traction recently in the automotive industry with the shift to multicore architectures. The objective of this Dagstuhl Seminar was to investigate new opportunities and challenges raised by the use of LET as a basis for implementing parallel execution of control software.

LET abstracts from the actual timing behavior of real-time tasks on the physical platform: Independent of when a task executes, the time interval between its reading input and writing output is fixed by the LET. This introduces a separation between functionality on the one



Except where otherwise noted, content of this report is licensed under a Creative Commons BY 3.0 Unported license

The Logical Execution Time Paradigm: New Perspectives for Multicore Systems, *Dagstuhl Reports*, Vol. 8, Issue 02, pp. 122–149

Editors: Rolf Ernst, Stefan Kuntz, Sophie Quinton and Martin Simons



Dagstuhl Reports

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

hand, and mapping and scheduling on the other hand. It also provides a clean interface between the timing model used by the control engineer and that of the software engineer.

The LET paradigm was considered until recently by the automotive industry as not efficient enough in terms of buffer space and timing performance. The shift to embedded multicore processors has represented a game changer: The design and verification of multicore systems is a challenging area of research that is still very much in progress. Predictability clearly is a crucial issue which cannot be tackled without changes in the design process. Several OEMs and suppliers have come to the conclusion that LET might be a key enabler and a standardization effort is already under way in the automotive community to integrate LET into AUTOSAR.

The seminar brought together researchers and practitioners from different backgrounds to discuss and sketch solutions to the problems raised by the use of LET in multicore systems, with a focus on the automotive domain. The program was structured around the following topics: (i) Implementations of LET; (ii) LET and related paradigms; (iii) LET and control; (iv) Future directions of LET. The fruitful discussions covered the following issues:

- LET was designed as a programming paradigm but is now being used as a mechanism for predictable communication. How can the principles of LET be adapted accordingly? How should LET values be chosen?
- LETs act as deadlines for tasks, which means that they must be dimensioned for the worst-case response time of tasks. This may be too inefficient in practice. Alternatives exist where a bounded number of deadline misses may be tolerated. How should LET exceptions (violations of the specified LET) be handled then? How can deadline miss patterns which still guarantee functional correctness (e.g., system stability) be established?
- How should the LET constructs be integrated into AUTOSAR? More generally, how should the design and verification process in the automotive industry be modified to integrate the LET paradigm?
- How does the use of the LET paradigm for multicore systems fit into the more general context of achieving predictability of multicore systems?

This seminar provided a unique opportunity for participants from the automotive industry to get feedback from academia on their effort to adopt the LET paradigm. At the same time, it allowed other participants to confront their own models and/or solutions with industrial reality and identify new research challenges. This seminar furthermore brought together research communities which do not so often interact with each other, e.g. the synchronous, control and real-time communities.

Organization of the seminar

The seminar took place from 25th to 28th February 2018. The first day started with an introduction by the organizers, followed by a talk from one of the co-founders of the LET paradigm – Christoph Kirsch. The following two sessions included talks providing an industrial view on the challenges of implementing LET in the multi-core automotive setting. The first day continued with a session comprised of talks presenting the academic view on LET-related challenges, and concluded with breakout sessions (detailed below). The second day of the seminar started with two sessions in which LET was compared to related paradigms, such as the synchronous model. The afternoon talks focused on the connection between LET and control as well as on a possible application of the LET approach to the domain of graphical processing units. The second day concluded with another set of breakout sessions. The third day included talks exploring future directions of LET, and a final set of breakout sessions.

Breakout sessions led to very interesting and fruitful discussions, and covered, among others, the following aspects:

- **Dimensioning of LET intervals:** The main focus was on how to efficiently dimension LET intervals to fit specific applications, which is currently a very pragmatic and experience based activity. Moreover, the two uses of LET in the automotive setting were identified: (i) Functional LET and (ii) Implementation LET.
- **Buffer optimization within LET:** The main focus was on the management of buffers in a LET-based implementation. The following topics were identified as relevant and thus discussed: minimizing the number of used buffers, strategies to handle memory contentions when accessing buffers, location of buffers in the memory hierarchy of hardware platforms and locality affinities between buffers, impact of spatial partitioning or periodicity of LET frames (harmonic or not) the buffers.
- **The synchronous approach vs LET:** The focus was on the comparison between the synchronous and LET models, with a discussion of their advantages and limitations, and their positioning in the context of the needs of the automotive industry, with a special emphasis on a transition from a singlecore to a multicore setting.
- **Control and LET:** The main focus was on the use of the LET paradigm to implement controllers. The following topics were identified as relevant and thus discussed: Is LET the correct paradigm for controller implementation? What is a viable period choice? How are potential deadline misses handled? Can a proper fault model be conveniently incorporated into the LET methodology? Can LET lead to new contributions in the control research domain?

More details on breakout sessions are available in a dedicated section of this document, after the overview of the talks given during the seminar.

Outcome of the seminar

The seminar has already enabled several collaborations: (i) a white paper on the topic is under preparation; (ii) a special session at EMSOFT'18 will be proposed. In addition, since participants expressed very positive opinions about the seminar and were in favor of reproducing the experience, a follow-up seminar will be considered.

Finally, as organizers, we would like to thank all of the participants for their strong interaction, interesting talks, fruitful group discussions, and work on open problems.

2 Table of Contents

Executive Summary

Rolf Ernst, Stefan Kuntz, Martin Simons, Borislav Nikolić, Sophie Quinton, and Hermann von Hasseln 122

Overview of Talks

Integration of the Logical-Execution-Time Paradigm in the Automotive E/E Architecture	
<i>Matthias Beckert, Leonie Ahrendts, Rolf Ernst, and Borislav Nikolić</i>	127
Achieving Predictable Multicore Execution of Automotive Applications Using the LET Paradigm	
<i>Alessandro Biondi and Marco Di Natale</i>	127
Beyond the LET and back to Synchronous Models	
<i>Marco Di Natale</i>	130
A Time-Triggered execution model for the automotive field and some perspectives	
<i>Mathieu Jan</i>	132
On Event- and Time-triggered Communication in Networked Control Systems	
<i>Karl Henrik Johansson</i>	132
Parallelization of Automotive Control Software	
<i>Sebastian Kehr</i>	133
From Logical Execution Time to Principled Systems Engineering	
<i>Christoph M. Kirsch</i>	133
From Physical Timing Requirements to Certifiable Real-Time Systems: How to Capture Requirements and Generate Correct Real-Time Programs?	
<i>Florence Maraninchi</i>	134
End-To-End Latency with Logical Execution Time	
<i>Jorge Luis Martínez García</i>	134
LET for Legacy and Model-based Applications	
<i>Andreas Naderlinger and Stefan Resmerita</i>	135
Embedding multi/many-core COTS in the avionics domain	
<i>Claire Pagetti</i>	138
Reconciling the Original LET Paradigm with its Current Use in the Automotive Industry	
<i>Sophie Quinton</i>	139
Logical Execution Time - An Industrial Perspective	
<i>Hermann von Hasseln and Martin Simons</i>	139
Migration of Legacy Embedded Control Software from Singlecore to Multicore Controllers	
<i>Hermann von Hasseln</i>	140
LET as an interface between control engineers and SW integrators?	
<i>Dirk Ziegenbein</i>	140

Working groups

Buffer optimization within LET
Mathieu Jan, Alessandro Biondi, and Sylvain Cotard 141

Control and LET
Martina Maggio and Rolf Ernst 143

The synchronous approach vs LET
Florence Maraninchi and Alain Girault 145

Dimensioning of LET Intervals
Dirk Ziegenbein and Hermann von Hasseln 148

Participants 149

3 Overview of Talks

3.1 Integration of the Logical-Execution-Time Paradigm in the Automotive E/E Architecture

Matthias Beckert (TU Braunschweig, DE), Leonie Ahrendts (TU Braunschweig, DE), Rolf Ernst (TU Braunschweig, DE), and Borislav Nikolić (TU Braunschweig, DE)

License © Creative Commons BY 3.0 Unported license
© Matthias Beckert, Leonie Ahrendts, Rolf Ernst, and Borislav Nikolić

More often the logical execution time (LET) paradigm is considered to ensure synchronization among multiple cores. In theory LET introduces an zero-time communication model, which can be used to provide a consistent core-to-core communication at fixed points in time. This contribution to the Dagstuhl Seminar 18092 provides a possible implementation of the LTE paradigm on a multicore ECU, as a test framework for future research. As first topics the handling of LET misses as well as the integration of LET into the in-vehicle network are discussed.

3.2 Achieving Predictable Multicore Execution of Automotive Applications Using the LET Paradigm

Alessandro Biondi (Sant'Anna School of Advanced Studies - Pisa, IT) and Marco Di Natale (Sant'Anna School of Advanced Studies - Pisa, IT)

License © Creative Commons BY 3.0 Unported license
© Alessandro Biondi and Marco Di Natale

3.2.1 Introduction

This document is an extended abstract in support of a talk proposal for the Dagstuhl Seminar on the Logical Execution Time (LET) paradigm (February 25-28 2018). The abstract is a short summary of a work that has recently been submitted by the same authors to the 24th IEEE RTAS conference.

3.2.2 Realizing LET with GIOTTO semantic on multicores

The GIOTTO LET semantic

At a high level, the LET paradigm assumes that the input and output operations of a periodic task τ happen in zero time at the beginning and the end of each periodic instance of τ , respectively. In practice, the actual input/output operations must be scheduled for execution, and can also take place at various time instants (i.e., not necessarily in a strictly periodic fashion) provided that the order of their execution preserves the desired logical semantic.

Note that the order with which they are executed influences the timing properties of the system, especially when flow preservation along communication chains is required. To ensure time determinism, the GIOTTO programming paradigm [1] specifies an order of execution for the communication operations, with a particular focus on those that should happen at the same logical time (see GIOTTO micro steps in [1]).

LET as an opportunity to control memory contention

Due to the contention of architectural shared resources in the memory hierarchy (such as levels of caches and shared memories), real-time applications that are executed upon multicore platforms may experience several delays that are difficult to predict, hence making the timing analysis of the system arduous. That is, without a proper synchronization mechanism, in the worst-case the memory accesses issued by one core can interfere with the other, and viceversa, leaving room for pathological scenarios that inevitably affect the tasks' response times.

Several works in the literature (e.g., see [2], [3]) addressed this problem by proposing clever solutions to improve the predictability of memory accesses. Nevertheless, leveraging the periodic access to the communication variables, the adoption of the LET paradigm brings the potential to improve time determinism by design.

In fact, although the LET paradigm can be realized by scheduling data write and read operations at various time instants, scheduling the communication phases at the beginning of the periodic instances of tasks carries considerable benefits in controlling the memory traffic. Specifically, this approach allows localizing the memory accesses within precise time windows that are determined by the task periods, which are hence known off-line. This rationale enables the possibility of realizing an explicit arbitration of the accesses to shared memories that become under the control of the system designer.

LET tasks with inter-core synchronization

As a reference abstract platform, the presented work assumed that the tasks execute upon $m > 1$ processors each disposing of a local scratchpad memory. The platform also includes a global memory and a crossbar switch that enables point-to-point communication between each core and each memory. All the memories are accessible from all the cores. For instance, this model matches the popular AURIX Tricore platform by Infineon, which is widely adopted in the automotive domain.

Tasks work on local copies of the communication variables that are managed under the LET paradigm, which are stored in local memories. Global (i.e., shared) copies of the communication variables are allocated to the global memory.

The proposed approach to realize LET communication is based on the following design principles:

- Synchronous activation of all the tasks in the system (i.e., all the tasks on all the cores are synchronously released at the system startup).
- Definition of a LET task for each processor that moves data from the local copies to the global copies (write operations), and viceversa (read operations). Such tasks run at the highest priority.
- Adoption of an inter-core synchronization protocol to arbitrate the accesses to the global memory performed by the LET tasks.

Note that since tasks work only on local copies, their execution is not affected by memory contention. Conversely, the accesses to the global copies performed by the LET tasks are subject to contention.

Thanks to a timing analysis of LET communications, it has been identified that, as a function of the task periods, a producer does not need to always update the shared copies of the accessed variables at every periodic instance. A dual conclusion has been reached for consumer tasks. Overall, given a task set, the subset of memory accesses that are required to safely realize the LET paradigm can be analytically characterized. This fact has been

leveraged to realize the LET tasks, which resulted to require a variable behavior job by job, but with a cyclically repeating order of the job behaviors. For this reason, LET tasks can be modeled and analyzed as generalized multi-frame (GMF) tasks [4].

To match principle (iii), a simple synchronization protocol has been implemented. The protocol is based on baton passing and enforces an order with which the processors access the global memory. The order can change frame by frame. Spinbased busy-waiting has been adopted.

3.2.3 Implementation and Evaluation

The proposed approach has been implemented on the popular Aurix Tricore platform produced by Infineon and by building upon the ERIKA open-source real-time operating system [5], which is certified OSEK/VDX and implements most of the AUTOSAR OS requirements.

The synchronization of the tasks' periods among the cores has been realized by exploiting the remote procedure call (RPC) features that are available in ERIKA. The resulting design consists in the first core that is in charge of activating all the tasks in the system as a function of a common time reference (a hardware timer).

The realization of the LET tasks required facing a memory vs. time trade-off. A literal implementation of the approach may require the definition of a table that stores the frames of the LET tasks up to the hyperperiod of all the tasks in the system. While this choice would have a limited impact in terms of runtime overhead, it is memory eager for realistic applications. To contain the memory footprint, the solution adopted in our implementation is based on providing counters for each pair of communicating tasks. Such counters can be used to identify the time instants in which the LET communications for a pair of tasks must be performed.

Finally, thanks to the characteristics of the Aurix Tricore, it was possible to devise a lightweight implementation of the inter-core synchronization mechanism. For each processor, two spin variables allocated to the corresponding local memory are provided: one to wait for write operations, and another to wait for read operations. Notification of a LET task that is spinning is then performed by simply updating one of its spin variables from a remote core, i.e., passing the baton.

A case study

An experimental evaluation has been conducted to assess the feasibility of the proposed approach and its impact in terms of timing performance. The proposed LET implementation has been adopted for a synthetic application that has been automatically generated from a model provided by Bosch for the WATERS 2017 challenge [6], which is claimed as representative for a realistic engine control application. The tests have been performed on an Infineon TriBoard v2 equipped with an Aurix TC275 microcontroller running at 200MHz and connected to a Lauterbach PowerTrace to perform debugging and tracing.

The WATERS 2017 challenge came with a model of an engine control application consisting of 1250 runnables grouped into 21 tasks/ISRs that access 10000 labels. The model specifies the labels accessed by each runnable, the type of access (read or write), and the number of accesses. Execution times are also provided together with the tasks' periods.

Based on set of assumptions, a code generator has been developed. The generator inputs the XML file that encodes the system model and generates C code for each runnable where execution segments are realized with for loops including a nop operation in the body. The generator is also in charge of producing (i) the definition of all the communication variables

accessed by the tasks (both the local and the global copies), (ii) the corresponding accesses within the runnable code, (iii) the tasks' code (to call a sequence of runnables), (iv) the configuration for the operating system, and (v) the code to setup the OSEK alarms to periodically activate the tasks.

Furthermore, the generator is in charge of generating the code of the LET tasks starting from the information available in the challenge model (i.e., communication relationships between tasks and task periods).

The collected results demonstrated that LET communication – with all the benefits that it brings in terms of predictability of the timing of control outputs and end-to-end latencies – can be realized without significantly harming the timing of the application with respect to the case of direct accesses to the global memory, which by definition lacks of the benefit provided by LET.

The major impact of the realization of LET has been found in terms of memory footprint, which increased by the 7.5% (about 40KB) with respect to the case of explicit communication (i.e., without LET).


By looking at the collected measurements, evident benefits in terms of reduced memory contention have not been observed. Although this is mainly attributed to the fact that the tested application was not sufficiently memory-intensive, note that, in general, the usage of LET does not bring average-case improvements, but rather it allows avoiding worst-case pitfalls and simplifying (by removing pessimism) worst-case analysis. More investigation is required to compare the worst-case performance of LET with the case of explicit communication : a detailed theoretical analysis is in the research agenda of the authors.

References

- 1 T. A. Henzinger, B. Horowitz, and C. M. Kirsch. *Giotto: a time-triggered language for embedded programming*.. Proceedings of the IEEE, vol. 91, no. 1, pp. 84–99, Jan 2003.
- 2 H. Yun, G. Yao, R. Pellizzoni, M. Caccamo, and L. Sha. *Memguard: Memory bandwidth reservation system for efficient performance isolation in multi-core platforms*.. in 19th IEEE Real-Time and Embedded Technology and Applications Symposium (RTAS), 2013, pp. 55–64.
- 3 H. Yun, R. Mancuso, Z. P. Wu, and R. Pellizzoni. *PALLOCC: DRAM bankaware memory allocator for performance isolation on multicore platforms*.. in 19th IEEE Real-Time and Embedded Technology and Applications Symposium (RTAS), April 2014.
- 4 S. Baruah, D. Chen, S. Gorinsky, and A. Mok. *Generalized multiframe tasks*.. Real-Time Systems, vol. 17, no. 1, pp. 5–22, Jul 1999.
- 5 ERIKA Enterprise. *ERIKA Enterprise: Open-source RTOS OSEK/VDX kernel*.. Available: <http://erika.tuxfamily.org>
- 6 A. Hamann, D. Dasari, S. Kramer, M. Pressler, F. Wurst, and D. Ziegenbein. *WATERS Industrial Challenge 2017*.. Available: <https://waters2017.inria.fr/challenge/#Challenge17>

3.3 Beyond the LET and back to Synchronous Models

Marco Di Natale (*Sant'Anna School of Advanced Studies - Pisa, IT*)

License  Creative Commons BY 3.0 Unported license
© Marco Di Natale

Considering the objectives for which LET is planned for use in the automotive industry (determinism and enforcing causality), there is a clearly strong relationship between the

LET model and the general class of synchronous (reactive, or SR) models of computation [1, 2, 3, 4, 5]. The LET model has attracted significant attention for its capability of providing causality and time determinism. However, it can be considered a restriction of the SR class of systems (in which a constant delay is applied at the end of each computation step), which provide the same properties (flow preservation and determinism), but with a much greater choice of the possible delays to be applied at each input/output stage. This connection can be leveraged by reusing several results from the research on SR systems that define how to provide efficient or even optimal implementations of tasks and communication primitives [6, 7, 8, 9, 10, 11, 12].

References

- 1 D. Potop-Butucaru, R. De Simone, J. P. Talpin, “The Synchronous Hypothesis and Synchronous Languages,” in R. Zurawski, ed., *The Embedded Systems Handbook*, CRC Press, 2005.
- 2 A. Benveniste, P. Caspi, S. Edwards, N. Halbwachs, P. Guernic, and R. de Simone. “The synchronous languages 12 years later,” in *Proceedings of the IEEE*, 91, January 2003.
- 3 G. Berry and G. Gonthier, “The Esterel synchronous programming language: Design, semantics, implementation,” in *Sci. Comput. Program*, vol. 19, pp. 87–152, Nov. 1992.
- 4 F. Boussinot and R. de Simone, “The Esterel language,” in *Proceedings of the IEEE*, vol. 79, pp. 1293–1304, Sept. 1991.
- 5 P. Caspi, D. Pilaud, N. Halbwachs, and J. A. Plaice, “LUSTRE: A declarative language for programming synchronous systems,” in *ACM Symp. Principles Program. Lang. (POPL)*, Munich, Germany, 1987, pp. 178–188.
- 6 P. Caspi, N. Scaife, C. Sofronis, and S. Tripakis. “Semantics-preserving multitask implementation of synchronous programs,” in *ACM Trans. Embed. Comput. Syst.*, 7(2):1–40, January 2008.
- 7 J. Forget, F. Boniol, D. Lesens, and C. Pagetti. “A multiperiodic synchronous data-flow language”. In *11th IEEE High Assurance Systems Engineering Symposium (HASE’08)*, Nanjing, China, Dec. 2008.
- 8 J. Forget, “A Synchronous Language for Critical Embedded Systems with Multiple Real-Time Constraints”, Ph.D. Thesis, University of Toulouse, 2009.
- 9 C. Sofronis, S. Tripakis, and P. Caspi. “A memory-optimal buffering protocol for preservation of synchronous semantics under preemptive scheduling.” In *Proc. 6th ACM International Conference on Embedded Software*, 2006.
- 10 Guoqiang Wang, Marco Di Natale, and Alberto L. Sangiovanni-Vincentelli. “Optimal synthesis of communication procedures in real-time synchronous reactive models.” in *IEEE Trans. Industrial Informatics*, 6(4): 729–743, 2010.
- 11 Guoqiang Wang, Marco Di Natale, and Alberto L. Sangiovanni-Vincentelli. Improving the size of communication buffers in synchronous models with time constraints, in *IEEE Trans. Industrial Informatics*, , Volume 5, Number 3, August 2009.
- 12 Haibo Zeng and Marco Di Natale. “Mechanisms for Guaranteeing Data Consistency and Time Determinism in AUTOSAR Software on Multi-core Platforms.” In *Proceedings of the 6th IEEE Symposium on Industrial Embedded Systems (SIES)*, June 2011.

3.4 A Time-Triggered execution model for the automotive field and some perspectives

Mathieu Jan (CEA LIST - Gif-sur-Yvette, FR)

License © Creative Commons BY 3.0 Unported license

© Mathieu Jan

Main reference Damien Chabrol, Didier Roux, Vincent David, Mathieu Jan, Moha Ait Hmid, Patrice Oudin, Gilles Zeppa: “Time- and angle-triggered real-time kernel”, in Proc. of the Design, Automation and Test in Europe, DATE 13, Grenoble, France, March 18-22, 2013, pp. 1060–1062, EDA Consortium San Jose, CA, USA / ACM DL, 2013.

URL <http://dx.doi.org/10.7873/DATE.2013.223>

CEA LIST ended in 2013 the technology transfer to a spin-off company of a Real-Time Operating System (RTOS), successively called OASIS and then PharOS. This was the end of 18 years of work in this area, the initial idea behind this RTOS being stated in 1995. In the PharOS RTOS, CEA LIST introduced in 2012 a new paradigm in order to address the shortcomings of this TT approach when applied two case studies. These two case studies were an automotive powertrain system that mixes angular and physical time scales and an protection relay, which uses an electrical sample time scale.

In this talk, we will briefly describe the steps of these 18 years of work done by CEA LIST on the subject of TT. Besides, we will shortly describe the custom flavor of the TT paradigm that was initially developed at CEA and compare it against the LET paradigm. The main focus of the talk is then to present why our xT extension was needed to fit automotive requirements and show its design within the two aforementioned case studies

3.5 On Event- and Time-triggered Communication in Networked Control Systems

Karl Henrik Johansson (KTH Royal Institute of Technology - Stockholm, SE)

License © Creative Commons BY 3.0 Unported license

© Karl Henrik Johansson

Main reference Bart Besselink, Valerio Turri, Sebastian H. van de Hoef, Kuo-Yun Liang, Assad Al Alam, Jonas Mårtensson, Karl Henrik Johansson: “Cyber-Physical Control of Road Freight Transport”, Proceedings of the IEEE, Vol. 104(5), pp. 1128–1141, 2016.

URL <http://dx.doi.org/10.1109/JPROC.2015.2511446>

In an event-triggered control loop the sensing, communication, computation, or actuation takes place only when needed. This paradigm has been developed to reduce the need for feedback while guaranteeing performance. In a large networked control system, control loops are often either event- or time-triggered. In this lecture, we discuss a few such control loops in emerging cooperative road freight transport systems based on heavy-duty vehicle platooning. It is shown how safety-critical loops, such as regulating the distance between vehicles, are triggered periodically, while other controls, like fuel-optimising the platoon velocity, are triggered whenever needed. Some open problems on how implementation can be formalised using logical execution time and other paradigms are briefly discussed.

3.6 Parallelization of Automotive Control Software

Sebastian Kehr (Denso Automotive - Echting, DE)

License © Creative Commons BY 3.0 Unported license
© Sebastian Kehr

Joint work of Bert Böddeker, Eduardo Quiñones, Günter Schäfer, Dominik Langen, Miloš Panic, Jorge Alberto Becerril Sandoval, Jaume Abella, Francisco J. Cazorla

Main reference Sebastian Kehr: “Parallelization of automotive control software”, PhD thesis, Technische Universität Ilmenau, Germany, 2016.

URL <https://dblp.org/rec/bib/phd/dnb/Kehr16>

The purpose of this talk is to present methods for the parallelization of automotive control software that use logical execution times (LETs).

Automotive control software is developed according to the AUTomotive Open System ARchitecture (AUTOSAR) standard. High development costs require the re-use of existing software when the hardware platform changes from a single-core to a multicore electronic control unit (ECU).

This talk focuses on the migration of AUTOSAR legacy software to a multicore ECU. Different parallelization methods are proposed and evaluated; RunPar and Supertasks on runnable-level, timed implicit communication on task-level, and the parallel schedule quality metric for quantification of combinations. The methods respect data dependencies and still enable parallel execution, they exploit the energy-saving potential of the processor, they guarantee latency constraints, and they reproduce the reference data-flow.

3.7 From Logical Execution Time to Principled Systems Engineering

Christoph M. Kirsch (Universität Salzburg, AT)

License © Creative Commons BY 3.0 Unported license
© Christoph M. Kirsch

Joint work of Thomas A. Henzinger, Benjamin Horowitz, Christoph M. Kirsch

Main reference Thomas A. Henzinger, Benjamin Horowitz, Christoph M. Kirsch: “Giotto: A Time-Triggered Language for Embedded Programming”, in Proc. of the IEEE, 91(1):84–99, 2003.

URL <http://www.cs.uni-salzburg.at/~ck/content/publications/journals/ProcIEEE03-Giotto.pdf>

The idea of Logical Execution Time was developed at UC Berkeley starting in the year 2000. In the beginning we pretty much knew what we wanted but we did not expect how controversial the idea would be seen by the scientific community around real-time and embedded systems. Just stating how long something takes to execute seemed to be inconceivable. In this talk, I am going to share the story of how the idea evolved over a number of years at Berkeley and elsewhere and how it finally found its place among the other real-time programming models. The LET story is an excellent example of how combining ideas from fields as diverse as programming languages, real-time and embedded systems, and formal verification can help to solve hard engineering problems. At the end of the talk, I am going to mention the selfie project as another example that we are currently working on. Selfie combines a self-compiling compiler of a tiny subset of C, a self-executing RISC-V emulator targeted by the compiler, and a self-hosting hypervisor that virtualizes the emulator. Selfie can compile, execute, and virtualize itself any number of times. The selfie project would not exist without the LET experience and, to our own surprise, has already received quite a bit of attention and caused some controversy. Will it be the topic of a Dagstuhl seminar in fifteen years? This is probably too much to ask for. Thanks a lot to the organizers of this seminar for inviting me!

3.8 From Physical Timing Requirements to Certifiable Real-Time Systems: How to Capture Requirements and Generate Correct Real-Time Programs?

Florence Maraninchi (VERIMAG - Grenoble, FR)

License  Creative Commons BY 3.0 Unported license
© Florence Maraninchi


Working on various industrial case-studies in the past decade, we realized that, given the very quick evolution of hardware platforms, and the growing complexity of embedded software, it is, more than ever, necessary to start with a very general point of view. On one hand we need to understand the physical timing constraints and tolerances as determined by control engineers for a given application (sampling frequencies and jitters, end-to-end latencies, etc.), without mentioning software entities like tasks or scheduling. On the other hand we need to characterize precisely the computation and communication performances of a given execution platform, and assess their predictability.

Designing the implementation means finding space and time allocations for the application functional parts, in such a way that the physical constraints are met by construction, and in a provable way for certification authorities. The LET paradigm, or the much older “Bulk synchronous parallel” model, are elements in this broad picture, but not the only ones.

We will use several examples (with the Kalray MPPA, or a simple Arduino platform), to illustrate the nature and specification of the timing constraints, the implementation schemes, and how the constraints are met.

3.9 End-To-End Latency with Logical Execution Time

Jorge Luis Martinez Garcia (Robert Bosch GmbH - Stuttgart, DE)

License  Creative Commons BY 3.0 Unported license
© Jorge Luis Martinez Garcia

Joint work of Ignacio Sañudo, Marko Bertogna, Jorge Luis Martinez Garcia

Modern automotive embedded systems are composed of multiple real-time tasks communicating by means of shared variables. The effect of an initial event is typically propagated to an actuation signal through sequences of tasks writing/reading shared variables, creating an effect chain. The responsiveness, performance and stability of the control algorithms of an automotive application typically depend on the propagation delays of selected effect chains. Indeed, task jitter can have a negative impact on the system potentially leading to instability. The Logical Execution Time (LET) model has been recently adopted by the automotive industry as a way of reducing jitter and improving the determinism of the system.

In this talk, a formal analysis of the LET model for real-time systems composed of periodic tasks with harmonic and non-harmonic periods is provided, analytically characterizing the control performance of LET effect chains. It is also shown that by introducing tasks offsets, the real-time performance of non-harmonic tasks may improve, getting closer to the constant end-to-end latency experienced in the harmonic case. The introduction of offsets not only may reduce response times and end-to-end latencies, but it also allows decreasing the jitter of important control parameters.

3.10 LET for Legacy and Model-based Applications

Andreas Naderlinger (Universität Salzburg, AT) and Stefan Resmerita (Universität Salzburg, AT)

License © Creative Commons BY 3.0 Unported license
 © Andreas Naderlinger and Stefan Resmerita
Joint work of Andreas Naderlinger, Wolfgang Pree, Stefan Resmerita
Main reference Stefan Resmerita, Andreas Naderlinger, Stefan Lukesch: “Efficient realization of logical execution times in legacy embedded software”, in Proc. of the 15th ACM-IEEE International Conference on Formal Methods and Models for System Design, MEMOCODE 2017, Vienna, Austria, September 29 - October 02, 2017, pp. 36–45, ACM, 2017.
URL <http://dx.doi.org/10.1145/3127041.3127054>

Abstract

The Logical Execution Time (LET) paradigm has recently been recognized as a promising candidate to facilitate the migration to multi-core architectures in automotive real-time software systems. We outline several findings regarding the application of the LET paradigm that corroborate this perception. Our work in this respect deals with LET for legacy systems and LET in the context of model-based development (e.g., in MATLAB/Simulink). Furthermore, we present open issues and highlight implications on the development process when using LET as a synchronization mechanism.

3.10.1 Introduction

Since its initial introduction in the Giotto project [1] almost two decades ago, several research groups have been working on the Logical Execution Time (LET) paradigm which has by now been the foundation for several programming languages and run-time systems [2]. While the promised advantages, such as time- and value-determinism, do sound desirable for safety-critical real-time systems, the approach has long been met with skepticism. Also, with a few exceptions (e.g., [3], [4]), industry has been reluctant in the trial, let alone the adoption of LET. With the emergence of multi-core architectures in automotive system this seems to change as LET could play a key role for obtaining predictable behavior when parallelizing control software. As a consequence, it recently experienced an increase in attention from both academia and industry (e.g., [5]). Amongst other benefits, LET shall provide deterministic inter-task communication across multiple cores on automotive multi-core architectures. Central questions that need to be dealt with involve, for example, how to reconcile performance-dominated requirements of control systems with the additional memory and computational costs that come with LET, how to apply this primarily top-down and correct-by-construction approach to legacy systems that may not satisfy all the initial assumptions, and also how and where to best introduce the LET concept in a development process that is no longer centered around code but on models specified, e.g., in MATLAB/Simulink. This abstract presents two active lines of work in our group dealing with these questions: (1) LET applied to legacy automotive systems including multi-core architectures, and (2) LET in the context of a model-based development with simulation in MATLAB/Simulink.

3.10.2 LET for Legacy Systems

A substantial amount of legacy code is used in many embedded system domains, in particular in the automotive industry. When carried over to a new hardware platform, data consistency issues arise and provisions must be made to maintain proper behavior along cause-effect

chains. Our first work on applying LET to an industrial engine controller reaches back to 2010 [3], where the imposed restriction of limiting code changes to top-level functions, lead to a considerable increase in memory requirements (both RAM and ROM). Abandoning this restriction leads to a drastic reduction in run-time and memory overhead [6]. Both dimensions of overhead are largely dependent on the particular application and are depending also on the degree of freedom for choosing the exact LET [7], especially for multi-core targets. There is an enormous potential for optimizations when migrating to multi-cores using LET. Naturally, different optimizations are difficult to harmonize. For example, a strategy that reduces buffers and leads to less total run-time overhead could still lead to bulky and unacceptable copy-operations at a particular LET boundary. Also, the question is how far optimality of a certain setting (in whichever respect) impacts extensibility or changeability of the software and the potential validation effort that goes along with it. In [8], we propose a transformation process from single-core legacy software to LET-based versions that can be safely run on a multi-core. It is a process that can be applied incrementally and that is centered around a static buffer requirement analysis, which can be applied at different levels of abstraction. The most abstract level determines a minimal set of buffers for a given LET specification that is independent of the underlying platform configuration (including task priorities, scheduling, and function-to-core mapping). Being a minimal upper bound, this set can be further reduced in more refined abstraction layers where restrictions and details are incorporated into the analysis. For example, we describe an optimal buffering strategy w.r.t. the number of required buffers for a known multi-core platform configuration under fixed-priority preemptive scheduling.

At run-time, automotive applications change functionality to adapt computational demands according to the crank angle in order to avoid system overload, for example, at high engine speeds. This variation in physical execution times must be reflected also in the logical timing domain, e.g. using a multimodal specification (as already supported by Giotto). So far, to the best of our knowledge, support for multiple modes in the context of LET and multi-core has not been addressed. It is unclear how this will increase the complexity of the analysis and the run-time system that ensures the LET semantics, and it is also unclear what the exact semantics should even be in the case of a mode switch and how this goes together with AUTOSAR modes.

3.10.3 LET in Model-based Development

Model-based design has become an established development approach in the field of embedded real-time systems. Clearly, LET should be an established fixture already in the modeling and simulation phase of the development. The predominant environment for modeling and simulating automotive control systems is MATLAB/Simulink, which is based on the synchronous block diagram (SBD) formalism. Being built on the synchronous reactive programming paradigm, SBD is also suited to realize LET behavior. However as we outlined in [9], in the presence of cyclic data-dependencies as found between AUTOSAR runnables, for example, care must be taken to comply with limitations implied by the simulation engine such that a valid execution order of the blocks can be found. In [10], we present a Simulink implementation of a run-time system (E-machine) for a multi-mode multi-rate LET specification involving potentially cyclic data dependencies. The simulated control algorithms may be implemented as Simulink/Stateflow blocks or in the programming language C.

Originating from a purely control-engineering oriented view, since at least the introduction of AUTOSAR support, Simulink models realign to more and more software-centric perspectives. It is not clear how a clean transition from platform-independent to platform-dependent

models that support push-button code generation can be achieved. In any case, for obtaining highly optimized code with a minimal number of additional LET buffer variables, for example, the code generation for a particular runnable must not be considered in isolation. Timing and data-flow dependencies of the whole application must be taken into account.

The need for considering data-flow dependencies is not only an issue of optimization. In a classic LET-based specification, the LET interval of an individual task (or function) was mainly driven by physical requirements (expressed in the period) and inevitably by properties of the hardware/system (implied by worst-case execution/reaction times). Since in the multi-core setting LET is used as a synchronization mechanism, LET intervals must be harmonized across multiple cores and thus cannot be decided individually on task/function-level. This has implications on the whole development process (and also on the mode-switch issue discussed in the previous section). Despite this, the development process might benefit from using LET as a design contract between control and embedded software engineers as outlined in [11].

In the standard LET model, where a task's LET equals the period, end-to-end latencies are a major concern. However, when the LET is contained in the period, this issue is considerably relaxed [12].

An open issue, for example, is robustness w.r.t. the impact of a model change (e.g., adding a new data-dependency) on the generated code and how the attempt to minimize code changes relates to the resulting run-time efficiency.

3.10.4 Conclusion

This abstract touched on aspects of LET related to its application to automotive software systems, especially for a single- to multi-core migration. We hereby covered legacy and model-based applications and gave examples of open issues in this respect.

References

- 1 T. A. Henzinger, B. Horowitz, and C. Kirsch. *Giotto: A time-triggered language for embedded programming*. Proceedings of the IEEE, vol. 91, pp. 84–99, January 2003.
- 2 C. M. Kirsch and A. Sokolova. *The logical execution time paradigm*. in Advances in Real-Time Systems, S. Chakraborty and J. Eberspacher, Eds. Springer, 2012, pp. 103–120.
- 3 S. Resmerita, K. Butts, P. Derler, A. Naderlinger, and W. Pree. *Migration of legacy software towards correct-by-construction timing behavior*. in Monterey Workshop, 2010, pp. 55–76.
- 4 V. Belau, H. von Hasseln, and M. Simons. *Coordinating AUTOSAR runnable entities using giotto - first concepts*. 2012, poster presented at DEPCP 2012, Dresden, Germany.
- 5 A. Hamann, D. Dasari, S. Kramer, M. Pressler, F. Wurst, and D. Ziegenbein. *Waters industrial challenge 2017*. 8th International Workshop on Analysis Tools and Methodologies for Embedded Realtime Systems, WATERS 2017.
- 6 S. Resmerita, A. Naderlinger, M. Huber, K. Butts, and W. Pree. *Applying real-time programming to legacy embedded control software*. in 2015 IEEE 18th International Symposium on Real-Time Distributed Computing, April 2015, pp. 1–8.
- 7 J. Hennig, H. von Hasseln, H. Mohammad, S. Resmerita, S. Lukesch, and A. Naderlinger. *Towards parallelizing legacy embedded control software using the LET programming paradigm*. in Proc. of WiP Papers of the 22nd IEEE Real-Time and Embedded Technology and Applications Symposium, ser. RTAS'16, 2016.
- 8 S. Resmerita, A. Naderlinger, and S. Lukesch. *Efficient realization of logical execution times in legacy embedded software*. in Proceedings of the 15th ACM-IEEE International Conference on Formal Methods and Models for System Design, MEMOCODE 2017, Vienna, Austria, September 29 - October 02, 2017, 2017, pp. 36–45.

- 9 A. Naderlinger, J. Templ, and W. Pree. *Simulating real-time software components based on logical execution time*.. in Proceedings of the 2009 Summer Computer Simulation Conference, ser. SCSC'09. Vista, CA: Society for Modeling & Simulation International, 2009, pp. 148–155.
- 10 J. Templ, A. Naderlinger, P. Derler, P. Hintenaus, W. Pree, and S. Resmerita. *Real-Time Simulation Technologies: Principles, Methodologies, and Applications (Computational Analysis, Synthesis, and Design of Dynamic Systems)*.. CRC Press, April 2016, ch. Modeling and Simulation of Timing Behavior with the Timing Definition Language, pp. 159–178.
- 11 P. Derler, E. A. Lee, M. Tornngren, and S. Tripakis. *Cyber-physical system design contracts*.. in 2013 ACM/IEEE International Conference on Cyber-Physical Systems (ICCPS), April 2013, pp. 109–118.
- 12 W. Pree, J. Templ, P. Hintenaus, A. Naderlinger, and J. Pletzer. *TDL - steps beyond giotto: A case for automated software construction*.. Int. J. Software and Informatics, vol. 5, no. 1-2, pp. 335–354, 2011.

3.11 Embedding multi/many-core COTS in the avionics domain

Claire Pagetti (ONERA - Toulouse, FR)

License © Creative Commons BY 3.0 Unported license
© Claire Pagetti

Main reference Wolfgang Puffitsch, Eric Noulard, Claire Pagetti: “Off-line mapping of multi-rate dependent task sets to many-core platforms”, Real-Time Systems, Vol. 51(5), pp. 526–565, 2015.

URL <http://dx.doi.org/10.1007/s11241-015-9232-1>

The last decade has seen the emergence of multi-core processors and many-core architectures. Although these architectures may allow a huge gain in terms of performance, they also face important challenges to their integration in safety critical environments, in particular aeronautics. As an example, due to the intensive resource sharing and lack of documentation, it is very difficult to ensure time predictability, one of the key elements of certification expectation.

A solution to tackle this last issue for COTS architectures is to rely on software-enforced predictability solutions. In this talk, we will review several execution models that have been developed in collaboration with aeronautical industrial partners. An execution model is a set of rules to be followed by the designer to remove or at least reduce drastically the temporal interferences. Most of them rely on the spatial and temporal static allocation of data, applications and communication. Thanks to these kinds of solutions, there is a way to masterize complex COTS architectures and prepare their embedding in the next generation of aircrafts. We then briefly present a recent project named PHYLOG. This project aims at offering a model-based and software-aided certification framework for aeronautics systems based on multi/many-core architectures in order to reduce as much as possible the amount of textual documentation and replace it with model(s) and promote automatic analysis and replace part of the testing with formal methods, as accepted by the DO333.

3.12 Reconciling the Original LET Paradigm with its Current Use in the Automotive Industry

Sophie Quinton (INRIA - Grenoble, FR)

License © Creative Commons BY 3.0 Unported license
© Sophie Quinton

In this brief talk, I discuss the differences between the original LET paradigm and its current implementation in the automotive industry. In particular, the original LET concept assumes that LETs are specified by the application designer. In contrast, in recent work about applying the LET concept to legacy code in automotive, LETs are based on tasks' WCETs. Additionally, the original paradigm has LET tasks as basic software blocks. In the automotive context, runnables are the smallest unit of software. One therefore has to decide how to map runnables to LET tasks, which was not considered before. Finally, not all tasks are implemented following the LET paradigm.

In the presentation, I try to reconcile the original LET paradigm with the current use of LET in the automotive industry. I also discuss a possible definition of what a correct implementation of the LET paradigm should be.

3.13 Logical Execution Time - An Industrial Perspective

Hermann von Hasseln (Daimler Research - Stuttgart, DE) and Martin Simons

License © Creative Commons BY 3.0 Unported license
© Hermann von Hasseln and Martin Simons
Joint work of Stefan Kuntz, Hermann von Hassel, Martin Simons

In this talk we highlight the Industrial Automotive System Development with focus on control systems like engine control, battery control, or brake control. These systems are essentially characterized by mainly cyclic real time computations under tight memory and run-time constraints. It is shown how this design process suffers from severe complexity through a highly distributed development process, which include different stakeholders on different levels of abstractions, going down from system level to control unit levels. It is argued that therefore only simple design patterns are useful.

The need for multi-core processors with its multiple challenges has only intensified this complexity. The need for new methods, new tools and new development processes, together with the need of migration of big packages of legacy code to multi-core architectures posed real trouble. It became clear that predictability and timing have to become crucial elements in the whole software design process. And it turned out that most implementation patterns used by providers seem to fit the LET paradigm. Indeed, LET was on the radar in the Automotive Industry a long time as a recurring pattern, not only because LET is attractive because it supports determinism, but also because it defines a clear technical organizing principle for the coordination of software components that are developed independently by different stakeholders. Beside, the re-factoring of big legacy code packages and the re-definition of LET intervals offered a striking opportunity for the migration to multi-core architectures.

3.14 Migration of Legacy Embedded Control Software from Singlecore to Multicore Controllers

Hermann von Hasseln (Daimler Research - Stuttgart, DE)

License © Creative Commons BY 3.0 Unported license
© Hermann von Hasseln

There are continuing efforts to migrate legacy software of automotive applications, which has been developed for singlecore controllers to multicore platforms. The steadily growing demand for computing power can only be satisfied by embedded multicore controllers.

The process of migration has to be geared with existing highly agile and highly distributed processes of software development for the automotive applications. These demands put heavy restrictions on the use of existing parallelization schemes.

While the OEM is faced with the issues of parallelizing the software and specifying the requirements to the ECU supplier, the latter has to deal with implementing the required parallelization within the integrated system.

The Logical Execution Time (LET) paradigm addresses these concerns in a clear conceptual framework. We present here ongoing efforts for applying the LET paradigm in this respect: (1) Parallelization of legacy embedded control software, by exploiting existing inherent parallelism. The application software remains unchanged, as adaptations are only made to the middleware. (2) Using the LET programming model to ensure that the parallelized software has a correct functional and temporal behavior.

In this talk we want to report on these efforts, and show how the application of the LET paradigm helps to achieve the goal of parallelization by separation of concerns, helps on the path of re-designing legacy software more suitable for multicore platforms, and helps ECU-suppliers to seamlessly integrate OEM-software into their frameworks.

References

- 1 Hennig, J., von Hasseln, H., Mohammad, H., Resemerita, S., Lukesch, S., Naderlinger, A.: Towards Parallelizing Legacy Embedded Control Software Using the LET Programming Paradigm.
- 2 Lowonski, M., Ziegenbein, D., Glesner, S.: Splitting Tasks for Migrating Real-Time Automotive Applications to Multi-Core E.
- 3 Resmerita, S., Naderlinger, A., Lukesch, S.: Efficient Realization of Logical Execution Times in Legacy Embedded Software.
- 4 Hu, T. C.: Parallel Sequencing and Assembly Line Problems.
- 5 Keller, J., Gerhards, R.: PEELSCHED: a Simple and Parallel Scheduling Algorithm for Static Taskgraphs.

3.15 LET as an interface between control engineers and SW integrators?

Dirk Ziegenbein (Robert Bosch GmbH - Stuttgart, DE)

License © Creative Commons BY 3.0 Unported license
© Dirk Ziegenbein
Joint work of Arne Hamann, Eckart Mayer-John, Dirk Ziegenbein

The Logical Execution Time (LET) paradigm gained traction in the automotive industry and also at Bosch initially as a mean to master the transition to multi-core microprocessors.

The principal properties of the LET paradigm that are of interest in this use case are deployment-independent behavior (time and value determinism of cause effect chains) and resource-efficient implementations.

This talk proposes Logical Execution Time (LET) as the basis of a portable and composable specification of control functions. In this sense, LET serves as an interface and supports a separation of concerns between the primary tasks of control engineers and software integrators. The talk outlines the basic requirements on such an interface, discusses trade-offs and open challenges for research.

4 Working groups

4.1 Buffer optimization within LET

Mathieu Jan (CEA LIST - Gif-sur-Yvette, FR), Alessandro Biondi (Sant'Anna School of Advanced Studies - Pisa, IT), and Sylvain Cotard (Krono Safe - Orsay, FR)

License © Creative Commons BY 3.0 Unported license
© Mathieu Jan, Alessandro Biondi, and Sylvain Cotard

The discussion within this breakout session was mainly centered around defining a list of issues associated to the management of buffers when implementing the Logical Execution Timing (LET) model. Buffers are indeed used to perform data exchange between LET frames. We ended the session with the following list of issues: minimizing the number of buffers used, implement strategies to handle memory contention when accessing to these buffers, location of buffers in the memory hierarchy of hardware platforms and locality affinities between buffers, management of these buffers and impact of spatial partitioning or periodicity of LET frames (harmonic or not) on this management. In the remainder of this summary, we briefly describe some of these issues. We first describe general impact/requirements on buffers due to the use of the LET model.

4.1.1 LET impact/requirement on buffers: overview

The LET model enforces isolation and synchronization between readers and writers. At the buffer level, the consequences are:

- producers and consumers never access to the same version of data at any point of time in single or even in multi-core;
- no locks (e.g. HW semaphores, spinlocks) are used enabling wait-free / lock-free implementation for managing these buffers;
- data become visible atomically to consumers at a given point of time by a single pointer switch done by a single instruction;

Buffers implies the use of multiple copies of same data. The basic implementation requires the use of at least two versions. When additional constraints have to be taken into account, more copies may be needed, for instance when spatial protection is a requirement. In this case, intermediate copies have to be done at user and/or at kernel level.

4.1.2 Number of buffers and implementation options

Automotive OEM and Tiers have to implement robust, efficient, but also low-cost systems. Industrial applications suffer from a lack of everything: a lack of computing power - we have

to compute more and more and average CPU load become huge, and a lack of memory - hundreds of software components lead to thousands of data that have to be manipulated. If all software components were allocated to dedicated LET frames, the amount of memory needs to implement buffered communication would be a blocking point. That is why, the design of an application with LET has to be used carefully and the minimization of buffers is a critical point. The solution is twofold:

- From software point of view, the developers and researchers have to propose implementation that minimize the number of buffers;
- From engineering process point of view, even if LET is an efficient solution to ease implementation, a fine grain analysis of the application still has to be done in order to logically group together part of software that can share the same rights and does not need to be isolated. That means a LET-based implementation is an output of higher level tools and methods used all along the development process.

The implementation of the LET paradigm requires decoupling the data accessed within the execution of application tasks from the shared data that are published to the other tasks. Possible approaches to realize this decoupling are (i) the use of multi-buffering techniques or (ii) the introduction of local copies.

The former is based on a swapping mechanism: similarly to the realization of some wait-free algorithms, the tasks write and read from buffers that will be released for being accessed by other tasks at specific points in time (e.g., the end of the task execution). The actual number of buffers that is required depend on the parameters (e.g., the periods) of the communicating tasks. Multi-buffering is generally lightweight to implement as it does not require data copies. However, it may necessitate of specific arrangements of data in memory (e.g., wrapped into data structures) and, if no proper strategies are adopted, it may require multiple de-referentiation of memory pointers.

Conversely, when adopting local copies, tasks always write on private variables that are copied from and to shared copies by a communication stack. This approach tends to increase the system overhead due to the data copies, but it is typically simpler to implement and it allows tasks to directly access variables. Furthermore, it may increase the application footprint with respect to the case of multi-buffering.

4.1.3 Memory contentions when accessing buffers

A major issue in executing real-time applications upon multicore platforms is the contention of architectural shared resources in the memory hierarchy (e.g., levels of caches and global memories). In the worst-case, the memory accesses issued by one core can interfere with the other, and viceversa, leaving room for pathological scenarios that inevitably affect the tasks' response times. When looking at memory contention, the adoption of local copies to realize LET communication can provide considerable benefits that increase the software predictability.

Indeed, note that the multi-buffering approach implies that two communicating tasks will access the same memory areas, thus not providing any control on the way tasks access memories. Conversely, the adoption of local copies allows controlling the memory contention in scratchpad-based multicore platforms, where local copies are allocated to private scratchpads and shared copies to the global memory. By scheduling the LET communications at the beginning of the periodic instances of tasks, the accesses to the global memory can be localized within precise time windows that are determined by the task periods, which can host an explicit arbitration protocol that restores predictability of the memory traffic. Memory accesses simply comprise copy-in and copy-out phases from and to the global memory.

4.1.4 Location of buffers in a memory hierarchy

Under stringent memory space constraints, or to improve the tasks' worst-case response-time, the placement of variables into memory should be considered as part of the design space. In fact, it is common that multiprocessor platforms have different access times for different memories (e.g., global vs. local memories) and also the contention delays are strictly dependent on the frequency with which tasks access such memories. An optimization of the data placement is therefore a desiderata in the design flow of real-time applications.

In the presence of LET communication, this optimization phase should explicitly consider the additional buffers (or local copies) mentioned in previous sections. In the case local copies are used to implement LET, note that to enable a contention-free execution of tasks the local copies must be mapped to private memories (e.g., local scratchpads).

4.1.5 Spatial partitioning of buffers

When spatial partitioning is a requirement, hardware memory protection units must be used to provide clear and explicit segmentation of binaries. Access rights associated to sections of a binary are linked the execution modes supported by core and are (for some of them) dynamically changed upon task switch to allow appropriate buffer access for the enabled task. Sections of a binary can also be aggregated to reduce the run-time overhead of dynamically updating access rights. How this aggregation is performed depends on the memory protection abilities provided by the hardware unit (number of descriptors available, granularity of protection, etc.). Access performance of the memory support, location of memories being used to store buffers between cores when multi-core architecture is targeted and similarities in the required memory access rights are also considered. In order to implement a strict write only within the current execution context or read from the other execution contexts (to perform data copy) policy, intermediate buffers are then needed.

4.2 Control and LET

Martina Maggio (Lund University, SE) and Rolf Ernst (TU Braunschweig, DE)

License © Creative Commons BY 3.0 Unported license
© Martina Maggio and Rolf Ernst

Joint work of The entire working group on Control and LET (The summary has been only checked by Rolf Ernst and Martina Maggio)

The discussion has been centered around the use of the LET paradigm to implement controllers. Generally speaking, in this discussion, a controller is a piece of code that should run periodically on a platform. Periodicity and predictability are both crucial for the correct system behavior. A few key issues have been discussed.

- Is LET the correct paradigm for controller implementation? We have been discussing two different LET realizations. In the first one, the sensor data acquisition is followed by the LET frame, and then actuation happens at the end of the period. In the second one, the LET frame is short and the actuation occurs before the end of the period, but the controller task is triggered again at the end of the period. The first one is closer to the implementation of delayed actuation controllers, in which the controller is *designed* knowing that the actuation will happen only at the end of the period and this concern is taken into account in the control synthesis. The second alternative is closer to the most common implementation of control strategies, in which the code to compute a control

signal is as fast as possible and additional computation time is needed after the actuation for the controller state update, estimation, and housekeeping operations.

- What is a viable period choice? In both the realizations mentioned above, the LET frame period is an important parameter for control performance (usually measured as a function of the system error, the difference between the desired system state and the current one). In controller implementations, the period choice is often dictated by physical considerations on the plant to be controlled. Can these physical considerations be extended to handle hardware and software constraints that derive from the concurrent execution of many different tasks? Is this helped by LET?
- Assuming the LET paradigm is used for the implementation, how can we handle potential deadline misses? From the control perspective, how can this be: (i) analysed, (ii) predicted, (iii) factored in the control design. Some research work uses the weakly hard real-time systems model to encode deadline misses and design stabilizing controllers for a given plant, with the deadline misses constraints in mind [1]. The question of whether the weakly hard model is the correct model to use for control design is a very open research question. One of the points that was raised is that missing a deadline when the system is around its equilibrium (close to the desired behavior) is very different with respect to missing a deadline when the system is in its transient phase and should still reach a fixed point.
- Can we incorporate a proper fault model in the LET design and methodology? To match safety requirements, hardware and software faults must be included. Hardware faults are generally represented by probabilistic fault models and are traditionally addressed by hardware redundancy. In many applications, transient faults can be tolerated if they have no permanent impact on the state of a system. In control, such effects can often be treated like deadline misses. Permanent hardware faults and degradation can sometimes be treated by redundancy in time, e.g. by load redistribution. Should the LET model already include such errors (very conservative) or should this involve a mode change with a new LET model? What is the effect on control design? Software and conceptual errors are much more difficult because of their various potential effects. Can control be tolerant to software errors? How do we prove that and can LET help here? Diversification is the typical approach to safeguard against software errors. Can LET help structuring the effect of diversification on timing and function?
- What can LET do for control research? More in particular, can new type of controllers be synthesized *because of* the subsequent implementation with the LET paradigm? For example, in Model Predictive Control there are optimizations [2, 3] to shorten the controller code computation, that degrade the quality of the control signal but ensure faster termination of the code execution.

References

- 1 S. Linszenmayer and F. Allgower. Stabilization of networked control systems with weakly hard real-time dropout description. In *2017 IEEE 56th Annual Conference on Decision and Control (CDC)*, pages 4765–4770, Dec 2017.
- 2 S. Richter, C. N. Jones, and M. Morari. Computational complexity certification for real-time mpc with input constraints based on the fast gradient method. *IEEE Transactions on Automatic Control*, 57(6):1391–1403, June 2012.
- 3 J. L. Jerez, E. C. Kerrigan, and G. A. Constantinides. A condensed and sparse qp formulation for predictive control. In *2011 50th IEEE Conference on Decision and Control and European Control Conference*, pages 5217–5222, Dec 2011.

4.3 The synchronous approach vs LET

Florence Maraninchi (VERIMAG - Grenoble, FR) and Alain Girault (INRIA - Grenoble, FR)

License © Creative Commons BY 3.0 Unported license
© Florence Maraninchi and Alain Girault

Abstract

This document summarizes the discussions that started with the breakout session entitled “synchronous vs LET”, and continued until the end of the workshop. We first recall the main principles of both approaches, and then list the points that were discussed, and the points on which the participants reached an agreement. We focused on the aspects of one or the other that can help the transition between single-core and multi-core implementations of hard real-time systems as presented by the automotive industry.

4.3.1 Common Motivations and Overview of the Approaches

Both approaches are dedicated to the implementation of hard real-time systems, on top of various execution platforms, i.e., hardware with or without an operating system. Both provide general principles that help guarantee that the implementation indeed meets the hard real-time deadlines required by the application. Both accept to trade performance for predictability and safety. They impose strong constraints on the structure of the implementations, so that guaranteeing deadlines is feasible. In particular, they insist on input/output determinism: the same sequence of input samples at the same times always produce the same sequence of output samples at the same time.

Those hard real-time applications come from control engineering problems. They impose physical timing constraints like: sampling frequency for inputs from sensors, refreshment frequency for outputs to actuators, end-to-end maximum latency between inputs and the outputs they influence along a cause-and-effect chain.

The Synchronous Approach, Principles, Languages and Tools

The synchronous approach, first introduced in the 80's with languages like Esterel [2] or Lustre [1] and Signal[5], can be summarized as follows:

- It is designed to help reason on systems by decoupling logical and physical times as long as possible in the design flow; reasoning in logical time means defining variables as mathematical series indexed by N , to represent discrete time.
- Several programming languages have been proposed to write such mathematical series in a structured way. Esterel proposes an imperative style, while Lustre or Signal adopt a functional and declarative dataflow style.
- All the synchronous languages initially come with compilers into sequential code (the design concurrency being statically scheduled); a lot of approaches have been proposed to produce dynamically-scheduled code, or parallel code on multi- and many-core architectures, or even distributed systems. This is where logical time meets physical time. For instance, when a program is compiled into a single-loop sequential code (of the form: init memory; while (1) read inputs; compute outputs and update memory; write outputs;), it comes with a proof obligation: the actual WCET of the loop body on the chosen hardware should be sufficiently small w.r.t. the physical timing constraints on the inputs and outputs.

A lot of existing industrial tools belong to the synchronous family, the main example being Simulink, and the tool SCADE based on Lustre. Some of the synchronous languages and tools can be used as system-level languages, or Architecture-Description-Languages (ADLs), thanks to their dataflow style. This is for example the case of Prelude.

The LET (Logical Execution Time) Approach

The LET approach has been proposed in [4]. It is based on a simple solution to the problem of matching initial physical timing constraints of the application, and actual physical execution times of the implementation.

If $y = f(x)$, f will correspond to a piece of code, and the time it takes to execute this code will vary (because it depends on inputs, or because of other sources of variations caused by the execution platform). When a real-time system is implemented as a set of independent tasks running on top of a real-time operating system, each task i being one such function f_i , these variations mean that the order in which the tasks execute may vary, hence modifying the data exchanged between them and making the overall execution time more difficult to deduce from local execution times.

The essential idea of the LET principle is to reason as if the code of each function f_i always took exactly its WCET; and then to guarantee that the implementation is consistent with this view. Thanks to this design choice, the functional and temporal semantics are deterministic.

4.3.2 Discussion

We focused on the aspects of the two approaches that can help the transition between single-core and multi-core implementations of hard real-time systems as presented by the automotive industry, including the existence of legacy C code. The theoretical question of whether the LET principle is a subset of the synchronous approach quickly appeared as irrelevant for this matter (at least as a concept in programming language research, see for instance [3]).

In the sequel we will use SYNC when referring to the synchronous approach.

Needs of the Automotive Industry

The initial functional intention of a control engineer is to design a number of functions that define outputs in terms of inputs, with associated timing constraints (see the description of functions and tasks in Section 1). The legacy code is made of Simulink diagrams (for some applications), and a large number of lines of C code, called runnables, that can be assembled to get the implementation of one functionality. Each runnable should be executed at a given rate (i.e., they have a period expressed in terms of the physical time); runnables may exchange values by reading, or writing to, shared names (shared variables). The data dependency graph between runnables, as expressed by the read and write accesses to these names is known (or can be computed), in principle.

Building an application amounts to defining the periodic and sporadic tasks scheduled by a real-time operating system as sequences of runnables (i.e., designing their functionality f_i and their period or minimal inter-arrival time), in such a way that the runnables are run at the appropriate rate, and the data dependencies are respected. Once this design is completed, it is crucial to ensure the I/O timing determinism.

Current Practice, Pros and Cons of SYNC and LET

The main question remains at the end of the discussion: when, in the design flow from the control problem to the actual implementation on a particular execution platform, can/should we freeze the decision on the splitting of the application into components (tasks), and the periods associated with them?

With SYNC, especially with dataflow formalisms, the whole application is a dataflow diagram. It can be reorganized freely (without modifying its I/O mathematical semantics) in order to split or group components, before freezing the structure that will serve as a guide for the implementation. The implementation principles are independent of this view.

Once the execution platform, and the implementation principles are defined (for instance, we can decide to map one runnable onto one core, or one real-time task...), we start to get some information on the actual communication and execution times of the components that are indeed feasible. We can then re-import these information in the dataflow diagram (possibly modifying the semantics, e.g., by introducing delays). This approach can be used as a way to keep the model and the code “in sync”. It also allows to test/debug/verify the impact of the constraints imposed by the execution platform, on the high-level model.

With LET, the situation is a bit more constrained. It is an implementation scheme, rather than a programming language construct. The structuring of the application into “components” is considered to be given, and will not be put in question again. For each of these components, a “LET frame” has to be chosen. The size of a LET frame is, by default, set to the period of the component that is assigned to it. Thanks to this choice and to the LET semantics, communications from one component to another one are deterministic.

4.3.3 Conclusions

The problem faced by the automotive industry is the transition to multi-core architectures, in the presence of a large amount of legacy sequential code that was designed for single-core platforms. The current practice relies on a very strict definition of the allowed single-core implementations, which guarantees determinism and correctness. The ideal objective would be to reuse the legacy code in new strict implementation schemes to be defined for multi-core platforms. Problems arise because real-time operating systems on single-core platforms are very particular; they have strong synchronization properties, which will become invalid with multi-core implementations.

Some degree of re-engineering seems unavoidable, in order to make the intrinsic timing and dependencies constraints explicit. This means two things: (1) deciding which of the synchronization/order phenomena observed with the implementation (e.g., a READ of a shared variable that always comes after a WRITE) were indeed required by the application constraints, and which are just artefacts (the former have to hold also on multi-core platforms, while the latter can safely be forgotten); (2) conversely, understanding how each application constraint is guaranteed by the implementation; in case it happens to be true by chance, thanks to the particular behavior of the single-core platform, new explicit mechanisms will have to be defined for those constraints to hold on multi-core platforms. This is especially crucial for causality constraints implied by the cause-and-effect chains of the application.

A generalized version of the LET principle might well be the appropriate choice for the definition of new mechanisms and strict implementation principles on multi-core platforms; it will result in sub-optimal performances, but this is not necessarily a problem if it brings determinism and clarity. However this choice alone will not help in revealing the intrinsic timing and dependency constraints of the application. Here the general ideas and tools of

the synchronous approach can help re-engineer the legacy models and the legacy sequential code, in order to answer questions (1) and (2) above.

References

- 1 J-L. Bergerand, P. Caspi, N. Halbwachs, D. Pilaud, and E. Pilaud. *Outline of a real time data-flow language*.. In Real Time Systems Symposium, San Diego, September 1985.
- 2 G. Berry and G. Gonthier.. *The Esterel synchronous programming language: Design, semantics, implementation*. Science Of Computer Programming, 19(2):87–152, 1992.
- 3 Matthias Felleisen. *On the expressive power of programming languages*.. Science of Computer Programming, 17(1-3):35–75, December 1991.
- 4 Thomas A. Henzinger, Benjamin Horowitz, and Christoph Meyer Kirsch. *Giotto: A timetriggered language for embedded programming*.. In Embedded Software, pages 166–184, Berlin, Heidelberg, 2001. Springer Berlin Heidelberg.
- 5 P. LeGuernic, T. Gautier, M. LeBorgne, and C. LeMaire. *Programming real time applications with signal*.. Proceedings of the IEEE, 79(9):1321–1336, September 1991.

4.4 Dimensioning of LET Intervals

Dirk Ziegenbein (Robert Bosch GmbH - Stuttgart, DE) and Hermann von Hasseln (Daimler Research - Stuttgart, DE)

License © Creative Commons BY 3.0 Unported license
© Dirk Ziegenbein and Hermann von Hasseln

Joint work of the breakout session on dimensioning LET intervals

In a breakout session, seminar participants discussed approaches how the LET intervals can be defined for particular applications. The discussion showed that the current practice is currently very pragmatic and experience-based with little procedural guidance. Furthermore, the different use-cases of LET in the automotive industry are imposing different defining parameters for the dimensioning of LET intervals.

A general guideline is to dimension the LET intervals as large as possible in order to put the least constraints on platform integration and portability. The larger the ratio of physical execution time and logical execution time is, the smaller the degrees of freedom for the software integrator get, e.g. for integrating several LET workloads on the same HW platform. Of course, the size of the LET intervals is bounded by the application requirements such as cause and effect chain latencies.

In the following, the two major use cases for LET in the automotive industry and their respective guidelines for dimensioning the LET intervals are described

- “Functional LET” - LET is used as an abstraction level and interface between function developers (e.g. control engineers) and software integrators. In this use case, LET interval shall only be functionally motivated, e.g. due to a certain latency requirement of a control algorithm. The reason behind this guideline is to establish a deterministic implementation-independent behavior and leave the maximum remaining freedom for the implementation.
- “Implementation LET” - LET is used as a mean to efficiently implement a deterministic behavior in a dedicated HW/SW platform, e.g. the parallelization of a software application on multi-core processors. Here, the dimensioning of LET intervals depends on the inherent parallelism of the software application as well as on the targeted load distribution between cores and the physical execution times of SW entities.

In summary, there are some guidelines or best practices to dimension LET intervals but in general the task is not well-formalized and has been seen as a field for future research.

Participants

- Leonie Ahrendts
TU Braunschweig, DE
- James H. Anderson
University of North Carolina at Chapel Hill, US
- Matthias Beckert
TU Braunschweig, DE
- Alessandro Biondi
Sant'Anna School of Advanced Studies – Pisa, IT
- Bert Boeddeker
Denso Automotive – Eching, DE
- Björn B. Brandenburg
MPI-SWS – Kaiserslautern, DE
- Sylvain Cotard
Krono Safe – Orsay, FR
- Marco Di Natale
Sant'Anna School of Advanced Studies – Pisa, IT
- Benoît Dupont de Dinechin
Kalray – Orsay, FR
- Rolf Ernst
TU Braunschweig, DE
- Glenn Farrall
Infineon – Bristol, GB
- Gerhard Fohler
TU Kaiserslautern, DE
- Alain Girault
INRIA – Grenoble, FR
- Mathieu Jan
CEA LIST – Gif-sur-Yvette, FR
- Karl Henrik Johansson
KTH Royal Institute of Technology – Stockholm, SE
- Sebastian Kehr
Denso Automotive – Eching, DE
- Christoph M. Kirsch
Universität Salzburg, AT
- Stefan Kuntz
Continental Automotive GmbH – Regensburg, DE
- Ralph Mader
Continental Automotive GmbH – Regensburg, DE
- Martina Maggio
Lund University, SE
- Florence Maraninchi
VERIMAG – Grenoble, FR
- Jorge Luis Martinez Garcia
Robert Bosch GmbH – Stuttgart, DE
- Andreas Naderlinger
Universität Salzburg, AT
- Moritz Neukirchner
Elektrobit Automotive – Erlangen, DE
- Borislav Nikolic
TU Braunschweig, DE
- Nathan Otterness
University of North Carolina at Chapel Hill, US
- Claire Pagetti
ONERA – Toulouse, FR
- Paolo Pazzaglia
Sant'Anna School of Advanced Studies – Pisa, IT
- Christophe Prévot
INRIA – Grenoble, FR
- Sophie Quinton
INRIA – Grenoble, FR
- Stefan Resmerita
Universität Salzburg, AT
- Hermann von Hasseln
Daimler Research – Stuttgart, DE
- Eugene Yip
Universität Bamberg, DE
- Dirk Ziegenbein
Robert Bosch GmbH – Stuttgart, DE

