DAGSTUHL REPORTS

Volume 4, Issue 4, April 2014

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
November, 2014

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.

Digital Object Identifier: 10.4230/DagRep.4.4.i

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.

Contact
Schloss Dagstuhl – Leibniz-Zentrum für Informatik
Dagstuhl Reports, Editorial Office
Oktavie-Allee, 66687 Wadern, Germany
reports@dagstuhl.de
http://www.dagstuhl.de/dagrep

Report from Dagstuhl Seminar 14171

Evaluating Software Verification Systems: Benchmarks and Competitions

Edited by

Dirk Beyer[1], Marieke Huisman[2], Vladimir Klebanov[3], and Rosemary Monahan[4]

1 **University of Passau, DE**
2 **University of Twente, NL**
3 **Karlsruhe Institute of Technology, DE**
4 **National University of Ireland, IE**

Abstract

This report documents the program and the outcomes of Dagstuhl Seminar 14171 "Evaluating Software Verification Systems: Benchmarks and Competitions". The seminar brought together a large group of current and future competition organizers and participants, benchmark maintainers, as well as practitioners and researchers interested in the topic. The seminar was conducted as a highly interactive event, with a wide spectrum of contributions from participants, including talks, tutorials, posters, tool demstrations, hands-on sessions, and a live competition.

Seminar April 21–25, 2014 – http://www.dagstuhl.de/14171
1998 ACM Subject Classification D.2.4 Software/Program Verification, F.3.1 Specifying and Verifying and Reasoning about Programs
Keywords and phrases Formal Verification, Deductive Verification, Automatic Verification, Theorem Proving, Model Checking, Program Analysis, Competition, Comparative Evaluation
Digital Object Identifier 10.4230/DagRep.4.4.1

1 Executive Summary

Dirk Beyer
Marieke Huisman
Vladimir Klebanov
Rosemary Monahan

The seminar aimed to advance comparative empirical evaluation of software verification systems by bringing together current and future competition organizers and participants, benchmark maintainers, as well as practitioners and researchers interested in the topic.

The objectives of the seminar were to (1) advance the technical state of comparative empirical evaluation of verification tools, (2) achieve cross-fertilization between verification communities on common/related issues such as selection of relevant problems, notions of correctness, questions of programming language semantics, etc., (3) explore common evaluation of different kinds of verification tools, its appropriate scope and techniques, (4) raise mutual awareness between verification communities concerning terminology, techniques and trends, and (5) promote comparative empirical evaluation in the larger formal methods community.

Altogether, 43 researchers and practitioners have attended the seminar. A vast majority of the attendees (almost 90%) have participated either in the SV-COMP or in the VerifyThis (and related, e. g., VSComp/VSTTE) verification competitions. For lack of better terms, we tend to refer to these communities as the "automatic verification" and "deductive verification" community respectively, though, as Section 1 discusses in more detail, these labels are based on pragmatics and history rather than on technical aspects.

The presentations, hands-on sessions, and discussions provided valuable feedback that will help competition organizers improve future installments. To continue the effort, a task force will compile a map of the—in the meantime very diverse—competition landscape to identify and promote useful evaluation techniques. It was agreed that evaluation involving both automatic and deductive verification tools would be beneficial to both communities as it would demonstrate the strengths and weaknesses of each approach. Both SV-COMP and VerifyThis will be associated with the ETAPS conference in 2015.

A call to the public: It has been reported that competition-verification challenges have been used as homework for students. The seminar organisers would appreciate feedback and experience reports from such exercises.

Seminar Structure

The seminar was structured as a highly interactive event, rather than a sequence of talks, compared to workshops or conferences. The seminar opened with a session of *lightning talks* that gave every participant two minutes to introduce themselves and mark their activities and interests in verification and in comparative evaluation in particular.

In order to give participants insight into different verification techniques and practical capabilities of some existing verification tools, the seminar featured short overviews of the state of the art in deductive verification resp. automatic verification, as well as several tutorials, hands-on sessions, and accompanying discussions. These included a longer tutorial on deductive verification with the Dafny system together with a hands-on session, as well as short mini-tutorials on the automatic verifiers CPAchecker and CBMC, and deductive verifiers KIV and VeriFast. Another hands-on session concluded the seminar.

Discussions on evaluation techniques and setups were initiated with presentations by competition organizers and benchmark collectors. The presented evaluation vehicles included: VerifyThis competition (deductive verification), SV-COMP (automatic verification), VSTTE competition (deductive verification), SMT-COMP (Satisfiability Modulo Theories), Run-time Verification competition, RERS challenge (Rigorous Examination of Reactive Systems), INTS benchmarks (Integer Numerical Transition Systems).

Since evaluation must be grounded with the requirements of current and prospective users of verification technology, the seminar incorporated contributions from industrial participants. Among them were a talk on the use of the SPARK deductive verification tool-set as a central tool for the development of high-integrity systems at Altran UK, a talk on the use of automatic verification tools in the Linux Driver Verification project, an accompanying discussion, as well as statements by other industry representatives (from GrammaTech, Galois, LLNL, and Microsoft Research).

Verification Communities: Remarks on Commonalities, Differences, and Terminology

Important goals of the seminar were to raise mutual awareness and to foster cross-fertilization between verification communities. We may habitually refer to communities as "automatic verification" or "deductive verification", but as time passes, these labels become less adequate.

A trend is apparent that different types of tools are slowly converging, both technically and pragmatically. Instances of both automatic and deductive verifiers may use symbolic execution or SMT solvers. Automatic verifiers can synthesize (potentially quantified) invariants, verify infinite-state systems, or systems that are heap-centric.

The pace of development is high and the surveys are costly (the last comprehensive survey on automatic verification appeared in 2008). As a consequence, community outsiders typically have an outdated—sometimes by decades—view on verification technology that does not reflect the state of the art. We expect publications from competitions to fill the void between more comprehensive surveys.

One of the terminological pitfalls concerns the use of the attribute "automatic". For instance, model checking and data-flow analysis are typically advertised as "automatic". This is indeed true in the sense that the model-checking user does not have to supply proof hints such as loop invariants to the tool. On the other hand, using a model checker in practical situations may as well require user interaction for the purpose of creating test drivers, choosing parameters, tuning abstractions, or interpreting error paths (which can be quite complex). These aspects are typically abstracted away during evaluation of automatic verifiers, which allows better standardization but does not capture all aspects that are relevant in practice.

The situation is further confused by the fact that some deductive verifiers are also advertised as "automatic", even though all established deductive verification systems require user interaction and the amount of interaction that is needed with different tools is not radically different. The main meaningful differences are rather

1. whether user interaction happens only at the beginning of a single proof attempt or whether the user can/has to intervene during proof construction, and
2. whether user interaction happens in a purely textual manner or whether non-textual interaction is possible/required.

The seminar has confirmed the need for improved terminology, as well as made an attempt to eliminate misconceptions and communication pitfalls. Unfortunately, there is still no widely-accepted and usable terminology to communicate these distinctions.

2 Table of Contents

3 Tutorials and Hands-on Sessions

3.1 Introduction to Deductive Software Verification

Rosemary Monahan (NUI Maynooth, IE)

Deductive software verification is a program verification technique where a program and its specification are input into a tool which verifies that the given program meets its specification. A human user contributes in two ways: through formalizing an informally stated specification for a program and through providing guidance to a verification system to show formally the conformance of the program to its specification. The specification is the form of a contract, typically containing preconditions, which state the conditions under which the program should be executed; post-conditions, which will be achieved by executing the program; and frame conditions which specify what the program execution is permitted to modify. Deductive verification tools generate and prove the verification conditions necessary to verify that the given program meets its specification. The tool user often contributes to this process providing proof tips that direct the verifier, as well as lemmas and assertions that can assist the proof. This interaction is required due to the strong properties being verified and the range of different tools in the verification landscape. An overview of how these verification tools and proofs are performed in a range of state-of-the-art tools is given with reference to examples from many tools that took part in the VerifyThis 2012 verification competition at FM 2012.

3.2 Introduction to Automatic Software Verification

Dirk Beyer (University of Passau, DE)

Automatic program verification is a verification technique that takes as input a program and a (temporal) property and without user interaction constructs a witness for correctness (program invariants, proof certificate) or violation (counterexample). The talk gives an overview over techniques that are implemented in software verifiers that participate in SV-COMP. Those techniques include abstraction refinement, predicate abstraction, CEGAR, interpolation, shape analysis, bounded model checking, large-block encoding, and conditional model checking.

3.3 Dafny Crash Course and Hands-on Session

Rustan Leino (Microsoft Research, US) and Rosemary Monahan (NUI Maynooth, IE)

In this specialized mini-tutorial, features of Dafny, the programming language and interactive program verifier, are presented. Material is tailored to a program verification exercise announced later in the afternoon, allowing participants to use what they learned in the Dafny crash course. Tutorial materials and the coincidence count challenges are available on the web.[1] 33 participants took part in the tutorial and hands-on session.

3.4 VeriFast: A Mini-tutorial

Bart Jacobs (KU Leuven, BE)

During this mini-tutorial I introduced the seminar members to the VeriFast program verification tool that I have been developing with my group since 2008. It is a tool for sound modular automatic verification of safety and correctness properties of annotated single-threaded and multi-threaded C and Java programs. It requires that each function/method be annotated with a pre-condition and post-condition expressed in a variant of separation logic. It verifies each function/method separately using symbolic execution.

I started the tutorial by interactively verifying, in the VeriFast IDE, a simple example C program, illustrating modular symbolic execution, the symbolic store, and the symbolic path condition. Then I moved to an example involving dynamic memory allocation, illustrating the symbolic heap, and the production and consumption of heap chunks. Then, through the example of a modular specification and verification of the functional behavior of a stack data structure, I showed how recursive data structures can be specified and verified, through the use of inductive separation logic predicates and inductive data types. To conclude, I briefly discussed my VeriFast solution to the coincidence count challenge.

3.5 KIV: A Mini-tutorial

Gidon Ernst and Gerhard Schellhorn (University of Augsburg, DE)

The tutorial demonstrated the interactive theorem prover KIV to the seminar participants. It highlighted some of its core features:

- Sequent calculus for Higher Order Logic.
- An automatic simplification strategy based on user-defined conditional rewrite rules.
- Structured algebraic data types with lots of predefined ones in KIV's library.
- A weakest precondition calculus for imperative programs over these abstract data types. Programs can be verified by symbolic execution and induction.
- An elaborate graphical interface that graphically displays specification hierarchies ("development graphs"). Proof trees of sequent calculus, that can be inspected and replayed.
- A correctness management that takes care to invalidate only a minimal set of theorems after changes.
- Context-sensitive application of rewrite rules by just clicking on function symbols.

[1] http://www.rise4fun.com/dafny/eOCY
http://www.rise4fun.com/dafny/wJHw
http://www.rise4fun.com/dafny/hcvan

The tutorial then demonstrated how the coincidence count example introduced in the Dafny tutorial is proved in KIV. A second example was the deletion of the minimal element of a binary search tree from the VerifyThis competition at FM 2012, using KIV's separation library and a direct induction which avoids the use of complex invariants.

3.6 Bounded Software Model Checking using CBMC: A Mini-tutorial

Michael Tautschnig (Queen Mary University of London, UK)

CBMC implements bit-precise bounded model checking for C programs and has been developed and maintained for more than ten years. Only recently support for efficiently checking concurrent programs, including support for weak memory models, has been added. CBMC verifies the absence of violated assertions under a given loop unwinding bound by reducing the problem to a Boolean formula. The formula is passed to a SAT solver, which returns a model if and only if the property is violated. In the tutorial I will provide an overview of the key components of CBMC, underlining its straightforward pipeline. Then a number of examples will be presented, including floating point and concurrent programs, as well as a full SAT solver (PicoSAT).

3.7 Predicate Abstraction with CPAchecker: A Mini-tutorial

Philipp Wendler (University of Passau, DE)

Predicate abstraction is a traditional abstract domain used for model checking. In this talk we learn how it works in combination with CEGAR on a small C program. We then see how the verification framework CPAchecker analyzes the same program with predicate abstraction, and study the produced proof. An overview of CPAchecker and a tutorial on its usage conclude the talk.

3.8 Concluding Hands-on Session

The seminar concluded with another hands-on session. A verification challenge based on a real bug encountered in the Linux kernel source was chosen by the seminar organizers (description below). Participants were encouraged to build teams of up to three people, in particular mixing attendees from different communities. Some teams have applied several different tools to the problem. The challenge was as follows (abridged description):

The accompanying C file implements a doubly-linked list with integer payloads. The central function of the program sorts the list using a bubble-sort-style algorithm in ascending order of payloads. The data structure also supports nesting, but this is not used for sorting. Separation between nesting pointers and list-linkage pointers should be maintained though.
The program contains a basic correctness checker consisting of assert statements (‘inspect’ function). The checker does not check the complete specification given above.
Verify the program w.r.t. the given assertions. Should you find bugs, please fix them and proceed to verify. If your tool does not support C, we ask you to reimplement the core data structure/functionality in the language of your choice. Please try to stay as faitful to the original code as possible.

- For verifiers that do not need user-supplied invariants, checking that the assertions pass would be a starting point. Feel free to add assertions that are meaningful w.r.t. the informal specification. If you can’t verify all of the included assertions, please produce a maximal set of assertions that you can verify.
- For verifiers that use more expressive specification formalisms and user-supplied invariants, we encourage you to prove a more complete functional specification rather than just checking the assertions.

After the allocated two hours elapsed, we have received eleven submissions. The program indeed contained a bug resulting from a typo in the list initialization code.[2] The applied automatic verifiers could detect the assertion violation easily though interpreting the error path, but locating the bug required considerable effort in some cases. Unsurprisingly, proving the program correct after fixing the bug was not easy for the automatic verifiers. If at all, correctness could be shown only within a small bounded scope, with one exception. The program analyzer Predator, which is geared towards verifying heap-manipulating programs, could synthesize the invariant that is necessary to show that the assertions are never violated on lists of arbitrary length.

With deductive verifiers, the situation was more varied. Several teams succeeded in verifying parts of the code respective to a subset of assertions. Success factors were support for verifying C programs (as otherwise time was lost translating the subject program into the language supported by the verifier) and having found the bug first either by testing or automatic verification as auxiliary technique. A followup on the challenge is planned. An interesting question is whether and how the automatically synthesized safety invariant can be used in a deductive verifier.

4 Poster Presentations and Tool Demonstrations

The following posters and tool demonstrations have been presented at the seminar.

Posters:

- Annotations for All, David Cok
- OpenJML, David Cok

[2] The challenge is part of the SV-COMP database and can be found at https://svn.sosy-lab.org/software/sv-benchmarks/trunk/c/heap-manipulation/bubble_sort_linux_false-unreach-call.c.

- AutoProof, an auto-active verifier for Eiffel, Nadia Polikarpova
- Ultimate Automizer, Matthias Heizmann
- Cryptol: The Language of Cryptography, Joe Kiniry
- The Astrée Static Analyser, Antoine Miné
- KeY System 2.0, Mattias Ulbrich, Vladimir Klebanov
- IVIL, Mattias Ulbrich
- CPAchecker: The Configurable Software Verification Platform, Phillip Wendler
- SPARKSkein, Angela Wallenburg
- VERCORS, Marieke Huisman

Tool demonstrations:

- UFO, Aws Albarghouthi
- SPEEDY, David Cok
- OpenJML, David Cok
- Why3, Andrei Paskevich and Jean-Christophe Filliâtre
- AutoProof, Nadia Polikarpova
- Frankenbit, Arie Gurfinkel
- Ultimate Automizer, Matthias Heizmann
- Cryptol, Joe Kiniry
- Astrée, Antoine Miné
- SIL, Peter Müller
- GROOVE, Arend Rensink
- HSE, Andrey Rybalchenko
- KIV, Gerhard Schellhorn
- CodeThorn, Markus Schordan
- LLBMC, Carsten Sinz
- CBMC, Michael Tautschnig
- KeY/IVIL, Mattias Ulbrich
- CPAchecker, Phillip Wendler
- GRASShopper, Thomas Wies

5 Discussion on Comparative Evaluation

5.1 Deductive Verification

The following is an incomplete summary of remarks voiced during the discussion on evaluation issues by the deductive verification community. As in all good discussions a range of, sometimes conflicting, views was presented. Nineteen participants of the seminar took part in the discussion.

5.1.1 Sourcing Competition Challenges

The issue of sourcing good challenges for deductive verification tools was discussed at length. It was agreed that a call for challenges could be issued well in advance of the competition with contributions from industrial users particularly welcome. Those who submit challenges should submit a solution and scoring scheme. If they participate in the competition, they would be eliminated from that competition challenge. The ACM programming and similar competitions were also mentioned as a potential source for challenges.

5.1.2 Range of Competition Challenges

The following suggestions were made with respect to competition challenges:

- A selection of challenges could be offered with teams expected to complete a subset of challenges (e. g., offer five challenges; teams submit their best three).
- Challenges should focus on program verification rather than theorem proving. Emphasis should be put on programming language constructs such as iterators, threads, reflection, concurrency, complicated data structures, and interaction between the user and the program.
- Challenges that we cannot complete but that can be mapped to simpler problems (e. g., by mapping to the integer domain) are motivating.
- Challenge areas could be announced in advance of the competition to encourage tool developers to improve tools prior to the competition.
- On-site competitions need to be selective about the problems so that a small number of challenges with a focus on modular specification are presented.
- Challenges should progress from previous competitions and from challenge to challenge within the competition.
- Good challenges will motivate tools to evolve in different directions, with the more interesting challenges presenting a large gap between their specification and their implementation.
- Providing challenges in pseudo-code is favored over program languages, because the input to verification tools differs significantly. Time to implement the solution in the language of the tool must therefore be allocated.
- Challenges with errors should be included so that error detection as well as verification of program properties are part of the problem set.

5.1.3 Types of Competition

It was agreed that short on-site competitions, longer off-site competitions and benchmark collections (such as VACID-0) all have a role in motivating tool development and comparison. Challenges need to be self-contained, involving a maximum of two days work (to be realistic regarding the time constraints of those who will participate).

5.1.4 Relevance to Industry

Challenges should focus tool developers on building industrial-strength tools, allowing users to use features of real-world programming languages, rather than forcing users to encode problems in the way that the tool requires.

5.1.5 Solution Strategies and Variations

Grading of challenges could take solution strategies into account, with extra points for using different approaches to the problem, e. g., a maximum of X points for an iterative solution and a maximum of Y points for a recursive solution.

Another suggestion was to provide challenges with a (rough) solution. This would evaluate how easy it is to encode the solution in a particular tool and reduce influence of the user on the result. Of course, different tools might require very different solutions.

5.1.6 Motivating Competition Participation

To date, post-competition publications have been an important motivating factor for competition participation, yet the sustainability of this approach is unclear. Tool developers also benefit through comparison of tools and through advancing tools to meet the verification challenges. Other motivators to develop further include:

- Cash prizes obtained via sponsorship, which would motivate student teams.
- Prizes sponsored by professional societies, e. g., an ACM software verification prize.
- Different competition tracks, e. g., undergraduate, postgraduate, developers, users.
- Opportunities for discussion of tools and solutions, as well as exchange of tool development ideas.
- Bringing student supervisors on board so that they allow student time for competitions, integrate the challenges and competitions into case studies, and develop course exercises based on competition challenges.
- Guest presentations and registration at conferences.

5.1.7 Use of Libraries

The use of libraries in competition should be encouraged. The largest challenge here is the lack of specified libraries that are available for use. Some verification challenges could include verification of library code.

5.1.8 Tool Integration

The integration of tools and techniques is necessary to allow scalability in program verification. A combination of automatic and deductive verification techniques should be explored. A good starting point for case studies would be to take solutions from deductive verification competition challenges, inject bugs and ask automatic verifiers to locate the bugs. The main drawback at present seems to be the focus of automatic verifiers on C programs as input and their restricted use of theories. Competitions that require two tools per team would provide for tool integration, e. g., automatic invariant generation could assist many deductive verifiers at present.

5.2 Automatic Verification

The following is an incomplete summary of remarks voiced during the discussion on evaluation issues by the automatic verification community. Fourteen participants of the seminar took part in the discussion.

5.2.1 How to Grow the Benchmark Set of Verification Tasks

The discussion included some concrete plans for obtaining more benchmark verification tasks for programs involving arrays, programs involving floating-point arithmetic, generated random loop benchmarks, generated random concurrent benchmarks, and more verification tasks from Linux systems code.

5.2.2 New Category on Bug Finding

The participants at the SV-COMP community meeting have agreed on introducing a demonstration category on bug finding. The goal in this category is to find as many bugs as possible within a global time limit of 4 h. More precisely, a participating verifier is given the full set of verification tasks of SV-COMP 2015 and a computing resource for 4 h (CPU time), and the verifier reports all bugs that it can find. The ranking is based on the number of correctly identified bugs. The bugs are accompanied by a witness path that is checked, and the bug report is counted if the witness is successfully validated.

5.2.3 Directory of Verifiers

A proposal was made to create an "electronic book of verification tools". A common pattern for the description of the tools is applied, such that users/readers quickly get an overview over technologies, features, and applicability of the verification tool to certain use cases.

5.2.4 Verification Results with Confidence Levels?

A discussion regarding the evaluation of challenge solutions centered around confidence levels. Is a solution's confidence level simply true or false, or is there a more fine-grained level, such as a confidence level from 0 to 9? It was agreed that numeric confidence levels are helpful only for a fixed set of examples. Confidence may be applicable more to humans rather than to verification tools. How should results be compared if confidence levels are used? If used in practice of competitions, both a numeric scheme and a non-numeric scheme is needed. Every tool should then include a description of what is achieved, feedback to the user, perhaps counterexamples. However, this would make the competition rules quite complicated, and simple rules are important to not confuse readers/users while interpreting the results. Therefore, it was voiced that the rules should be kept as simple as possible. No matter how complicated and detailed an evaluation would be made—there are always different criteria for evaluation by different audiences—winning a competition does not automatically mean that the winner is the best tool (from the consumers' point of view).

6 Competition Overviews

6.1 The VerifyThis Verification Competition

Vladimir Klebanov (Karlsruhe Institute of Technology, DE)

Joint work of Huisman, Marieke; Klebanov, Vladimir; Monahan, Rosemary
Main reference M. Huisman, V. Klebanov, R. Monahan, "On the Organization of Program Verification Competitions," in Proc. of the 1st Int'l Workshop on Comparative Empirical Evaluation of Reasoning Systems (COMPARE'12), CEUR-WS, Vol. 873, pp. 50–59, 2012.
URL http://ceur-ws.org/Vol-873/papers/paper_2.pdf

We discuss the challenges that have to be addressed when organizing program verification competitions. Our focus is on competitions for verification systems where the participants both formalize an informally stated requirement and (typically) provide some guidance for the tool to show it. We draw insights from our experiences with organizing VerifyThis, a program verification competition aiming (1) to bring together those interested in formal verification, and to provide an engaging, hands-on, and fun opportunity for discussion; (2) to evaluate the usability of logic-based program verification tools in a controlled experiment that could be easily repeated by others. We discuss in particular the following aspects: challenge selection, on-site versus online organization, team composition and judging.

6.2 SV-COMP: Competition on Software Verification

Dirk Beyer (University of Passau, DE)

Main reference D. Beyer, "Status Report on Software Verification," in Proc. of the 20th Int'l Conf. on Tools and Algorithms for the Construction and Analysis of Systems (TACAS'14), LNCS, Vol. 8413, pp. 373–388, Springer, 2014.
URL http://dx.doi.org/10.1007/978-3-642-54862-8_25

SV-COMP is a thorough evaluation of verification tools that take as input software programs and run a fully automatic verification of a given property. The overview describes the definitions, rules, setup, and procedure of SV-COMP. The verification tasks of the competition consist of nine categories containing a total of 2 868 C programs, covering bit-vector operations, concurrent execution, control-flow and integer data-flow, device-drivers, heap data structures, memory manipulation via pointers, recursive functions, and sequentialized concurrency. The specifications include reachability of program labels and memory safety. The most recent (3rd) edition of the competition had 15 participants. http://sv-comp.sosy-lab.org/

6.3 RERS Challenge and Property-Driven Benchmark Generation

Bernhard Steffen (TU Dortmund, DE)

Joint work of Steffen, Bernhard; Isberner, Malte; Naujokat, Stefan; Margaria, Tiziana; Geske, Maren
Main reference B. Steffen, M. Isberner, S. Naujokat, T. Margaria, M. Geske, "Property-Driven Benchmark Generation," in Proc. of the 20th Int'l Symp. on Model Checking Software (SPIN'13), LNCS, Vol. 7976, pp. 341–357, Springer, 2013.
URL http://dx.doi.org/10.1007/978-3-642-39176-7_21

We present a systematic approach to the automatic generation of platform-independent benchmarks of realistic structure and tailored complexity for evaluating verification tools for reactive systems. The idea is to mimic a systematic constraint-driven software development process by automatically transforming randomly generated temporal-logic-based requirement specifications on the basis of a sequence of property-preserving, randomly generated structural design decisions into executable source code of a chosen target language or platform. Our automated transformation process steps through dedicated representations in terms of Büchi automata, Mealy machines, decision diagram models, and code models. It comprises LTL synthesis, model checking, property-oriented expansion, path condition extraction, theorem proving, SAT solving, and code motion. This setup allows us to address different communities via a growing set of programming languages, tailored sets of programming constructs, different notions of observation, and the full variety of LTL properties—ranging from mere reachability over general safety properties to arbitrary liveness properties. The paper illustrates the corresponding tool chain along accompanying examples, emphasizes the current state of development, and sketches the envisioned potential and impact of our approach.

6.4 The 2nd Verified Software Competition

Jean-Christophe Filliâtre (University Paris South, FR)

Joint work of Filliâtre, Jean-Christophe; Paskevich, Andrei; Stump, Aaron
Main reference J.-C. Filliâtre, A. Paskevich, A. Stump, "The 2nd Verified Software Competition: Experience Report," in Proc. of the 1st Int'l Workshop on Comparative Empirical Evaluation of Reasoning Systems (COMPARE'12), CEUR-WS, Vol. 873, pp. 36–49, 2012.
URL http://ceur-ws.org/Vol-873/papers/paper_6.pdf

We report on the second verified software competition. It was organized by the J.-C. Filliâtre, A. Paskevich, and A. Stump on a 48 hours period on November 8–10, 2011. We describe the competition, present the five problems that were proposed to the participants, and give an overview of the solutions sent by the 29 teams that entered the competition.

6.5 First International Competition of Software for Runtime Verification

Ezio Bartocci (TU Wien, AT)

Joint work of Bartocci, Ezio; Bonakdarpour, Borzoo; Falcone, Yliès

We report the description of the procedures, the participating teams, the submitted benchmarks, the evaluation process and the results of the 1st International Competition of Software for Run-time Verification. This competition is held as a satellite event of the 14th International Conference on Run-time Verification (RV) and is organized in three main tracks: offline monitoring, online monitoring of C programs and online monitoring of Java programs.

6.6 SMT-COMP: The SMT Competition

Alberto Griggio (Bruno Kessler Foundation – Trento, IT)

Joint work of Cok, David R.; Griggio, Alberto; Bruttomesso, Roberto; Deters, Morgan
Main reference D. R. Cok, A. Griggio, R. Bruttomesso, M. Deters, "The 2012 SMT Competition," in Proc. of the 10th Int'l Workshop on Satisfiability Modulo Theories (SMT'12), EPiC Series, Vol. 20, pp. 131–142, EasyChair, 2012.
URL http://www.easychair.org/publications/?page=527924520

The talk presents the lessons learned from participating in the annual SMT solvers competition SMT-COMP, both as a competitor and as an organizer. I describe the organization of the competition, highlight its positive impacts on the community, but also discuss some of its current limitations, concluding with some suggestions for possible future improvements.

6.7 Numerical Transition Systems

Philipp Rümmer (Uppsala University, SE)

Joint work of Hojjat, Hossein; Konecný, Filip; Garnier, Florent; Iosif, Radu; Kuncak, Viktor; Rümmer, Philipp
Main reference H. Hojjat, F. Konecný, F. Garnier, R. Iosif, V. Kuncak, P. Rümmer, "A Verification Toolkit for Numerical Transition Systems," in Proc. of the 18th Int'l Symp. on Formal Methods (FM'12), LNCS, Vol. 7436, pp. 247–251, Springer, 2012.
URL http://dx.doi.org/10.1007/978-3-642-32759-9_21

Verification systems accept software programs or hardware designs as input in a wide variety of formats, some of which are intricate or partly underspecified. This leads to challenges for the development of verification systems, as well as the collection of verification benchmarks and competitions. This talk introduces a standardized format for verification problems, Numerical Transition Systems, which is applicable both for the representation of benchmarks and as a simple yet expressive intermediate verification language. Numerical Transition Systems support a variety of data types, including arithmetic and arrays, and provide features such as procedures and parallelism.

7 Industrial Applications

7.1 SPARK 2014 – Beyond Case Studies

Angela Wallenburg (Altran, UK)

Formal software verification has been successfully applied and demonstrated to scale to industrial projects. While many case studies have been successful, few formal methods have reached the take-up and maturity level where industrial non-experts continue to use the method for project after project, and where this formal method is a permanent part of the business of industrial software development. However, there are some notable exceptions: for example the SPARK language and tool-set for static verification has been applied for many years in on-board aircraft systems, control systems, cryptographic systems, and rail systems. SPARK has been developed and used at Altran UK (formerly Praxis) for almost 30 years. The grand challenge of building a verifying compiler for static formal verification of programs aims at bringing formal verification to non-expert users of powerful programming languages. This challenge has nurtured competition and collaboration among verification tool builders such as the participants in this Dagstuhl seminar. In this talk I describe our approach to popularizing formal verification in the design of the SPARK 2014 language and the associated formal verification tool GNATprove (co-developed by Altran UK and AdaCore). In particular, I will describe our solution to combining tests and proofs, which provides a cost-competitive way to develop software to standards such as DO-178. At the heart of our technique are executable contracts, and the ability to both test and prove those. I will also report on experiences in evaluation of verification tools from an industrial perspective.

7.2 The Experience of Linux Driver Verification

Vadim Mutilin and Alexey Khoroshilov (Russian Academy of Sciences, Moscow, RU)

The talk presents experience of Linux Verification Center in verification of Linux Device Drivers. For that purpose we use software model checkers which solve reachability problems, i. e. they prove that labeled error locations in the program can not be reached by any execution starting from an entry point. But Linux device drivers have neither entry point no error location. That is why the driver needs to be prepared for the verification. The driver is essentially asynchronous. On driver loading, the kernel core invokes the initialization function of the driver which registers callback functions. Then these functions are called by the kernel core on receiving events from user space and from hardware. So for the verification we need to prepare the model environment with explicit calls of driver callbacks. The model environment should reproduce the same scenarios of interactions with the driver as in the real kernel and at the same time it should be simple enough to be analyzed by existing static verification tools. For that purpose we proposed a method of environment modeling based on pi-calculus where the environment is represented as a set of communicating processes interacting with driver via messages. From the pi model we generate a C program which being combined with the driver becomes valid input for software model checkers with an entry point emulating all feasible paths in driver's code. But this code still have no error locations representing erroneous behavior of the driver. So the second task is to prepare specification on driver-kernel interfaces to be checked by software model checkers. The specification is weaved into the driver source code with the help of C Instrumentation Framework (CIF) and it becomes a source of error locations.

In the talk we discussed the architecture of Linux Driver Verification (LDV) Tools and showed its analytical and error trace visualization features. The comparison of CPAchecker with BLAST on the Linux drivers showed that while the total number of found bugs is the same the tools find different bugs. Thus it is worth to use the tools together. For now, LDV Tools helped to find more than 150 bugs which were approved and fixed in the latest Linux kernels. In the talk we discussed lessons learned. We found that it is important that the software model checker supports a full set of language features and could parse it. Moreover, the tool should not fail if it does not support some feature. If it cannot prove correctness it still may, for example, continue the search for unsafe trace if possible. We found that for verification tools it is even more important to ignore thousands irrelevant transitions than efficiently handle relevant ones. Also, the engineering efforts can help to get significantly better results and we shared experience in speeding up BLAST from 8 times on small-sized drivers and to 30 times on medium-sized drivers. On the base of the lessons learnt we make several conclusions how to improve Software Verification Competition (SV-COMP). First of all, there are two different use cases for software model checkers. The first one targets to prove correctness of code under analysis regarding some properties, the second one targets to find as much bugs as possible. Currently, rules and scoring scheme of SV-COMP evaluate tools from the first point of view, while it would be useful to evaluate the tools from the second point of view as well. Another conclusion is that benchmarks we produced so far target current generation of software model checkers and these benchmarks can not help to evaluate next generation tools. But fore verification of device drivers we need much more features, for example, better pointer analysis support, specifications in terms of sets and maps, verification of parallel programs, data race detection, support for function pointers.

We should produce benchmarks which target new functionality required for device driver verification, not only existing one.

The effective analysis of error traces produced by software model checkers in case of specification violation is crucial for the industrial use. At least, it should be possible to analyze the errors traces without knowing implementation details of the software model checker. A common representation of error traces in the competition would ease usage of the tools for driver verification as far as a single trace converter could be developed for Error Trace Visualizer component inside LDV Tools which is used for trace analysis. Users waits for verification results in terms of wall time, not the CPU time. The use of CPU time for the competition does not encourage the developers to utilize the available resources for parallelization. For example, the tools may use the CPU cores available on the machine instead of using a single one, thus reducing the wall time. It would be good if competition rules would encourage to reduce wall time of verification as well.

Participants

- Aws Albarghouthi
University of Toronto, CA
- Ezio Bartocci
TU Wien, AT
- Bernhard Beckert
KIT – Karlsruher Institut für Technologie, DE
- Dirk Beyer
Universität Passau, DE
- Marc Brockschmidt
Microsoft Research UK – Cambridge, GB
- David Cok
GrammaTech Inc. – Ithaca, US
- Gidon Ernst
Universität Augsburg, DE
- Marie Farrell
NUI Maynooth, IE
- Jean-Christophe Filliâtre
University Paris South, FR
- Bernd Fischer
University of Stellenbosch, ZA
- Alberto Griggio
Bruno Kessler Foundation – Trento, IT
- Radu Grigore
University of Oxford, GB
- Arie Gurfinkel
Carnegie Mellon University, US
- Matthias Heizmann
Universität Freiburg, DE
- Marieke Huisman
University of Twente, NL
- Bart Jacobs
KU Leuven, BE
- Alexey Khoroshilov
Russian Academy of Sciences – Moscow, RU
- Joseph Roland Kiniry
Galois Inc. – Portland, US
- Vladimir Klebanov
KIT – Karlsruher Institut für Technologie, DE
- K. Rustan M. Leino
Microsoft Res. – Redmond, US
- Stefan Löwe
Universität Passau, DE
- Antoine Miné
ENS – Paris, FR
- Rosemary Monahan
NUI Maynooth, IE
- Wojciech Mostowski
University of Twente, NL
- Peter Müller
ETH Zürich, CH
- Petr Müller
Brno Univ. of Technology, CZ
- Vadim Mutilin
Russian Academy of Sciences – Moscow, RU
- Andrei Paskevich
University Paris South, FR
- Nadia Polikarpova
ETH Zürich, CH
- Arend Rensink
University of Twente, NL
- Philipp Rümmer
Uppsala University, SE
- Andrey Rybalchenko
Microsoft Research UK – Cambridge, GB
- Gerhard Schellhorn
Universität Augsburg, DE
- Markus Schordan
Lawrence Livermore National Laboratory, US
- Carsten Sinz
KIT – Karlsruher Institut für Technologie, DE
- Bernhard Steffen
TU Dortmund, DE
- Jan Strejcek
Masaryk University – Brno, CZ
- Michael Tautschnig
Queen Mary University of London, GB
- Mattias Ulbrich
KIT – Karlsruher Institut für Technologie, DE
- Jaco van de Pol
University of Twente, NL
- Angela Wallenburg
Altran UK – Bath, GB
- Philipp Wendler
Universität Passau, DE
- Thomas Wies
New York University, US

Report from Dagstuhl Seminar 14172

Unifying Product and Software Configuration

Edited by

Krzysztof Czarnecki[1], Arnaud Hubaux[2], Ethan Jackson[3], Dietmar Jannach[1], and Tomi Männistö[5]

1 **University of Waterloo, CA,** kczarnec@gsd.uwaterloo.ca
2 **ASML – Veldhoven, NL,** contact@ahubaux.com
3 **Microsoft Research – Redmond, US**
4 **TU Dortmund, Germany,** dietmar.jannach@tu-dortmund.de
5 **University of Helsinki, FI**

Abstract

Research on computer-supported configuration of customizable products and services is currently carried out in two main communities: one community is mainly focused on the configuration of hardware artifacts, the other one is interested in configurable software systems and software product lines. Despite the significant overlap in research interests, the fields have mainly evolved in isolation in different fields such as Artificial Intelligence, Constraint Programming and Software Engineering. Yet, the communities have produced results that are applicable across the communities. The trend of products becoming increasingly heterogeneous, i. e., consisting of hardware, software and services, is furthermore increasingly blurring the line between the configuration domains in practice.

This report documents the program and the outcomes of Dagstuhl Seminar 14172 "Unifying Product and Software Configuration". The seminar gathered researchers and practitioners working on configuration problems. The seminar consisted of invited presentations and working group sessions covering various topics of software and product configuration including knowledge representation issues, automated reasoning and configuration management and had a particular focus on the industry perspective.

Seminar April 21–24, 2014 – http://www.dagstuhl.de/14172
1998 ACM Subject Classification I.2.4 Knowledge Representation Formalisms and Methods, D.2.13 Reusable Software, F.4.1 Mathematical Logic, Logic and constraint programming
Keywords and phrases Product Configuration, Software Product Lines, Configuration Management
Digital Object Identifier 10.4230/DagRep.4.4.20

1 Executive Summary

Krzysztof Czarnecki
Arnaud Hubaux
Ethan Jackson
Dietmar Jannach
Tomi Männistö

Customizable products are an integral part of most Business-to-Business (B2B) and Business-to-Consumer (B2C) markets. The fast-growing demand for mass-customization affects both tangible products (e. g., cars and mobile phones) and intangible products like software (e. g., operating systems, Enterprise Resource Planning systems and mobile phones). To this end, companies use software *configurators* that provide automated support to tailor products to the requirements of specific customers or market segments. These configurators have been developed essentially in two threads of research: *Product Configuration* (PC) and *Software Configuration* (SC).

PC is the umbrella activity of assembling and customizing physical artefacts (e. g., cars or muesli) or services (e. g., insurances). Due to the inherent complexity of configuration problems, PC was one of the first large-scale application fields of artificial intelligence (AI), as it required both powerful knowledge-representation formalisms and efficient reasoning methods. The particular challenges of knowledge representation and reasoning in PC even led to the development of new AI techniques. Today, PC can be seen as one of the major fields in which AI-based technology found its way into industrial practice and is part of many industrial configuration systems.

Mostly independent of PC, the software engineering community was confronted with challenging configuration problems. A typical challenge is the design and implementation of software components that can be adapted and parameterized according to customer requirements and business or technical constraints. As in PC approaches, the goal is to save costs by assembling individualized systems from reusable components. These challenges are dealt with in different strands of software engineering, e. g. software product line engineering or self-adaptive systems.

Questions of knowledge representation and types of reasoning support have been investigated for many years in PC and SC. Interestingly, research in these two fields has been carried out so far mostly independently. Except in rare cases, researchers in both fields are often unaware of approaches that have been developed in the other community.

This fragmentation is observable in two particular dimensions: *knowledge representation* and *configuration reasoning*. Knowledge representation is concerned with the question of how to encode the domain knowledge, e. g., about the compatibility of different features of a configurable product, in a formal or machine processible way. Configuration reasoning covers various aspects of how to make inferences given a knowledge base (configuration model), specific user requirements or an existing configuration. Typical tasks include the automatic completion of a partial configuration or checking the consistency of a given configuration.

The seminar was organized around the following research questions:

- (RQ1) What classes of configuration problems exist?
- (RQ2) How are these problems modelled?
- (RQ3) What automated tasks are supported?
- (RQ4) How are these automated tasks implemented?

The seminar was structured into three main blocks: *Problem characteristics*, *Knowledge representation* and *Reasoning and tools*. Each block consisted of a number of introductory presentations on the topic, which were given by researchers from different subfields and the seminar participants from industry. These talks then served as a basis for discussions on commonalities, differences and possible synergies. These discussions were made in small working groups in break-out sessions and the results then synthesized in plenary meetings. To make these break-out sessions more effective, the seminar participants were asked to fill out a detailed questionnaire before the seminar.

Overall, the seminar featured more than a dozen introductory talks from academia and from industry. In general, the interest from industry was particularly encouraging and the seminar was attended by representatives and speakers, e. g., from IBM, SAP, Microsoft, Siemens and BigLever. The evening sessions were used by several seminar participants to give additional “lightning” talks, to share recent research results and dive deeper into technical aspects.

2 Table of Contents

3 Overview of Talks

3.1 Selected knowledge representation aspects

Michel Aldanondo (University of Toulouse, France)

Main reference M. Aldanondo, E. Vareilles, "Configuration for mass customization: how to extend product configuration towards requirements and process configuration," Journal of Intelligent Manufacturing, 19(5):521–535, 2008.

The talk deals first with the elements that need to be modeled. First, three views (functional, physical and process) are shown; then the multi-level modeling idea is introduced. The need of using both discrete and continuous types of variables and constraints is then discussed. In some situations, the need to add some location aspects (ports, location constraints, ...) to the traditional bill-of-material (physical view) is described. The fact that the model of the problem can change during configuration (adding variables or variables set) is discussed and also the need for distributed modelling. Then the talk presents two key modeling ideas: either modeling the problem or modelling the solution space. Classical constraint based approaches are recalled as a problem modelling solution while less known solutions automata illustrates the solution space modelling idea.

References

1 P. Pitiot, M. Aldanondo, E. Vareilles, Concurrent product configuration and process planning : Some optimization experimental results. Computers in Industry, Vol. 65, pp. 610–621, 2014
2 A. Felfernig, G. E. Friedrich, D. Jannach, UML as domain specific language for the construction of knowledge-based configuration systems. International Journal of Software Engineering and knowledge Engineering, 10(4), 449–469, 2000.

3.2 Configuration reasoning is hard in general, but can be made efficient by exploiting the hierarchical structure of configuration problems

Conrad Drescher (SAP AG – Walldorf, Germany)

In my talk I give an overview of reasoning problems arising in configuration and the most common solution approaches. Problems discussed include:

- checking the consistency of a configuration
- computing valid domains for user choices in interactive configuration; explaining why some option is not available
- finding a valid / optimal / updated configuration
- proving properties about configuration models (model equivalence, . . .)

Except for the first one all of the problems are computationally hard in general, an observation that has important implications for scalability of configuration reasoning. I also discuss key differences between the most important approaches to reasoning such as constraint propagation solvers, conflict-driven clause learning solvers for Boolean problems, compilation of problems to finite graphs (BDDs/MDDs), mathematical programming and local search.

I then discuss two state-of-the-art approaches for calculating valid domains. One is based on storing a maximally condensed version of the complete search tree of the configuration problem (i. e., as an MDD) [1]. The other is based by incrementally exploring portions of the search tree on the fly [2].

Finally I argue that configuration problems as typically encountered in practice differ from general constraint satisfaction problems in that they have a low tree-width. This fact can be exploited in both approaches for computing valid domains as CSP with low tree-width are known to admit search trees and hence also BDDs/MDDs of polynomial size [3].

More generally, for the above mentioned reasoning problems low tree-width can in many cases be exploited to devise efficient algorithms.

References

1 J. Amilhastre, H. Fargier, P. Marquis, "Consistency Restoration and Explanations in Dynamic CSP – Application to Configuration", Artificial Intelligence, 2002

2 C. Bessiere, H. Fargier, C. Lecoutre, "Global Inverse Consistency for Interactive Constraint Satisfaction", Proceedings of CP, 2013

3 P. Jegou and C. Terrioux, "Hybrid Backtracking Bounded by Tree-Decomposition of Constraint Networks", Artificial Intelligence, 2003

3.3 Problem Characteristics of Industrial Product Configuration

Andreas Falkner (Siemens AG, Austria), Albert Haag (SAP AG – Walldorf, Germany)

A "product" can be anything a company offers for sale, both tangible (manufacturable) goods and intangible ones such as services, software, or projects. The product can also be an upgrade of something a customer already has. A configurable product is one that is not specified solely by its product designation. Product configuration is a step in a business process to establish a complete and correct specification of a configurable product. A predominant business process is sales, including associated manufacturing and/or assembly. Sales configuration is not meaningful without being able to give and guarantee both a price and an availability date. It also affects the entire logistics supply chain. For example, production pre-planning will also need to be based on the history of sold configurations. After a sale is completed it may be necessary to configure the specific product instance sold to make it operational (such as an airplane or computer server). This after-sales configuration poses distinct challenges and will tend to be product specific. Various terms are in use to characterize different ways of dealing with sales configuration. Some examples are Pick-to-Order, Assemble-to-Order, Make-to-Order, and Engineer-to-Order. These aim mainly to distinguish two business relevant aspects:

- to what degree the product is assembled at the company's site and the customer's site respectively
- to what degree the product has been standardized. Standardization enables a streamlined fulfillment process. In the worst case, manual intervention by engineers is necessary in non-standard cases. (However, note that product design is considered outside the scope of configuration.)

Sales configuration poses three somewhat different tasks:

- High-level configuration is the interactive configuration dialog with the customer/sales person. Besides needing an underlying sales model for the product, this will depend on sales organization data and an availability date.

- Completion derives additional properties/components needed for manufacturing and at the same time ensures that the configuration can be manufactured as ordered for the given date and at what cost. This is usually non-interactive and will depend on a manufacturing (engineering) model of the product, and the particular manufacturing plant(s) as well as the effective date of manufacture.
- Low-level configuration (or Bill-of-Materials (BoM) explosion) is selecting the actual parts (BoMs) and operations (route sheets/routings) needed for manufacturing and reserving corresponding resources. Again, this process depends on plant and effective date.

It is a current topic of research whether all three tasks might be handled using a uniform approach (such as SAT-solving). Practical experience in the last decades suggests separation. This is also vindicated by the fact that different parts of the organization have responsibility for the different tasks. The first task is the responsibility of the sales organization. They will want to influence the model to some degree. The last task is fulfillment and mainly the responsibility of the manufacturing engineers. The completion task joins the two.

Often high-level configuration must primarily provide decision support rather than just solving under given constraints. A sales configuration will typically be under-constrained with respect to actual hard constraints, but poses an embedded multi-criteria optimization problem in determining a best configuration. Optimality is not expressed explicitly, but in the form of soft constraints or desirable properties that should be fulfilled in a good solution. When adding these soft constraints the problem becomes over-constrained. Which properties to forego is ultimately a decision of the user. The requirements for a user interface (UI) depend on the type of user. In a B2C scenario the user is non-expert and requires very explicit visualization of alternatives/aid in resolving inconsistencies. In a B2B or in-house scenario the users are more expert and it may be sufficient to simply alert them to inconsistencies and incompleteness.

Problems can occur in the context of upgrades or after-sales configuration that may require dedicated CSP or SAT-solving approaches. The complexity of the first two sales configuration tasks is determined mainly by how the product is represented at that level. The easiest way would be as a single component with a manageable number of specifiable characteristics (attributes). An example of a product presented this way is a car (50-70 characteristics presented in the high-level configuration, 150-300 in completion, tens of thousands of components in low-level configuration). At the other end of the spectrum is a complex multi-level system with an a priori indeterminate number of sub-components. Examples of this would be elevators, busses, or computer servers.

Since a sales contract is legally binding, care must be taken that a configuration that is accepted is complete and correct and can be delivered at the promised date and price. Thus tools for model verification, testing and debugging are very important. Limiting the complexity of the product is essential in achieving this at reasonable cost. An exact methodology for measuring the complexity of a configurable product project from the business perspective is still lacking.

3.4 Towards combining performance optimisation and constraint satisfaction in software configuration

Holger H. Hoos (University of British Columbia – Vancouver, Canada)

Joint work of Hoos, Holger H.; Hutter, Frank; Leyton-Brown, Kevin
Main reference H. H. Hoos, "Programming by Optimization," Communications of the ACM, 55(2):70–80, February 2012.
URL http://dx.doi.org/10.1145/2076450.2076469

My group works on automatically configuring software for the purpose of performance optimisation. This is very widely applicable in industry (e. g., mixed integer programming – CPLEX, scheduling, SAT-based hardware and software verification, machine learning) and academia. There are several classes of techniques available for carrying out these configuration tasks, the best of which have been demonstrated to work on configuration problems involving up to about 750 parameters. I believe that sequential model-based techniques, like our own SMAC procedure, are particularly promising, especially in cases where performance evaluations are costly and therefore few can be completed over the course of the configuration process. I see very significant potential for these kinds of techniques to fundamentally change the way performance-critical software will be designed, towards much more configurable systems than used currently. This is at root of the Programming by Optimisation (PbO) software design paradigm developed and promoted by my group (see main reference provided above).

There is a different notion of configuration problems where the focus is on finding configurations that satisfy potentially complex constraints. These can be tackled with SAT and CSP solvers, which should be automatically configured using the previously mentioned techniques to perform well on the specific configuration problems they are being used on.

I see interesting potential in combining the two aspects of configuration mentioned above: performance optimisation within a potentially large space of configurations satisfying given constraints.

3.5 Configuration in Industrial Product Families

Lothar Hotz (HITeC e. V. / Universität Hamburg, Germany)

The software product line (SPL) approach provides a general reference process for supporting reuse of software components. This process is divided into domain engineering and application engineering. In domain engineering reusable components are developed and implemented that can be used in multiple applications. During application engineering these components are selected, configured, and composed to form a particular application. However, SPL provides a general schema, how the engineering subtasks can be resolved is matter of research.

Knowledge-based configuration as a field of Artificial Intelligence provides modeling languages and reasoning tools that enable the task of composing a system from components [1]. As such, knowledge-based configuration supplies technologies that support the task of application engineering.

The ConIPF methodology [2] demonstrates a successful application of knowledge-based configuration technologies for solving the application engineering task of SPL. The ConIPF methodology enhances the reference process by configuration activities such as development of a configuration model and running the configuration process. As a difference to hardware configuration, the ConIPF methodology adds activities to the process that create (compile, link, test) configured software (sub-)systems during the configuration process. The ConIPF methodology was applied to construct software-intensive systems in the field of car manufacturing.

References

1 Felfernig, A., Hotz, L., Bagley, C., Tiihonen, J. (Eds.), Knowledge-based Configuration – From Research to Business Cases. Morgan Kaufmann Publishers, 2014.

2 Hotz, L., Wolter, K., Krebs, T., Deelstra, S., Sinnema, M., Nijhuis, J., MacGregor, J., 2006. Configuration in Industrial Product Families – The ConIPF Methodology. IOS Press, Berlin.

3.6 High-Level Languages for Configuration Modeling and Analysis

Eunsuk Kang (MIT – Cambridge, USA)

In this talk, I will introduce two modeling languages, Alloy (developed at MIT) and Formula (Microsoft Research), and describe how they can be used to model and analyze a variety of configuration problems.

Alloy is an expressive modeling language based on first-order relational logic [1]. Originally designed for describing complex structures that arise in software systems, it has been applied to a variety of applications, including requirements analysis, program verification, policy modeling, and security protocols. It has also been used to solve different types of configuration problems, including product lines, feature models, multi-objective optimization, and configuration synthesis. The analysis in Alloy is done by translating an original FOL formula to an equisatisfiable SAT formula, which is then handed off to a third-party SAT solver. When the solver finds an instance, it is translated back to a high-level representation in the original model. If no instance is found, a minimal unsatisfiable core is generated to highlight the parts of the model are contradictory; this feature can be used to produce an explanation for configuration problems.

FORMULA is a modeling language developed at Microsoft Research for model-driven architecture development [2]. Based on logic programming (stratified horn clauses), FORMULA provides expressive constructs for modeling and composing domain abstractions. Its analysis is done by translation to the Z3 SMT solver, which is capable of handling a variety of theories (arithmetic, arrays, etc.). FORMULA has been used in a number of applications, including exploring the design space of automobile architectures [3].

References

1 Official page for Alloy (http://alloy.mit.edu)

2 Official page for FORMULA (http://research.microsoft.com/en-us/projects/formula)

3 Eunsuk Kang, Ethan K. Jackson, Wolfram Schulte: An Approach for Effective Design Space Exploration. Monterey Workshop 2010:33–54.

3.7 Product Line Engineering Meets Product Line Operations

Charles Krueger (BigLever, Austin, USA)

The complexity of managing the variability for a family of similar products or systems is not limited to product line engineering (PLE) organizations. Other organizations that can spend inordinate amounts of time and effort dealing with product feature diversity include manufacturing and supply chains in automotive, certification and compliance documentation in aerospace and defense, portfolio planning in highly competitive markets, web system deployments in e-commerce, and sales automation for complex configurable systems.

Although it became clear to many successful PLE organizations that alignment of PLE with their existing business operations was crucial, the idea of consolidating the variant management and configuration disciplines across engineering and operations groups is an emerging idea at the edge of the applied research envelope. Some of the industry's most innovative product line enterprises are now leveraging their PLE competence to create highly efficient Product Line Operations. We refer to this convergence as Product Line Engineering and Operations, or PLE&O.

PLE&O is more than just a new approach for aligning PLE with business operations, it is a generational step forward in the evolution of product line paradigms. PLE&O extends PLE with fundamental new perspective and methodology, with consolidated Feature Ontology and configuration automation.

3.8 Some Verification Problems in Automotive Configuration

Wolfgang Küchlin (Universität Tübingen, Germany)

Joint work of Küchlin, Wolfgang; Sinz, Carsten; Zengler, Christoph; Walter, Rouven
Main reference C. Sinz, A. Kaiser, W. Küchlin, "Formal methods for the validation of automotive product configuration data," AI EDAM: Artificial Intelligence for Engineering Design, Analysis and Manufacturing, 17(1):75–97, 2003.
URL http://dx.doi.org/10.1017/S0890060403171065

Automotive production is based on product configuration with very high variance, especially for German premium car manufacturers. Vehicle configuration is structured in two levels. High-level configuration (HLC) is concerned with the configuration of customer car orders from sales options such as motors, seats, etc. Low-level configuration (LLC) is concerned with the selection of the necessary parts for a car order from the bill-of-materials (BOM), which is the list of all parts necessary for an entire line of cars. Documentation of both HLC and LLC is usually based on Boolean logic. For a number of years, we have successfully shipped verification systems based on SAT-solving to the automotive industry.

Some verification issues concerned with HLC are the computation of options which are necessary, possible, or impossible, for every car order. Verification of LLC is concerned with the computation of BOM materials (parts or software) which can never be used in any order, which would be missing for some orders, or which would be multiply selected for some orders.

More recent issues include e. g., model counting the number of car orders in the HLC, the explanation of proof results, or the reconfiguration of orders.

References

1 Wolfgang Küchlin and Carsten Sinz. Proving consistency assertions for automotive product data management. *J. Automated Reasoning*, 24(1–2):145–163, February 2000. (Special issue: Satisfiability in the Year 2000).

2 Andreas Kübler, Christoph Zengler, and Wolfgang Küchlin. Model counting in product configuration. In Inês Lynce and Ralf Treinen, editors, *Proc. First Int'l Workshop on Logics for Component Configuration (LoCoCo)*, volume 29 of *EPTCS*, pp. 44–53, 2010.

3 Carsten Sinz, Andreas Kaiser, and Wolfgang Küchlin. Formal methods for the validation of automotive product configuration data. *Artificial Intelligence for Engineering Design, Analysis and Manufacturing*, 17(1):75–97, January 2003. Special issue on configuration.

4 Christoph Zengler and Wolfgang Küchlin. Boolean quantifier elimination for automotive configuration – a case study. In Charles Pecheur and Michael Dierkes, editors, *Formal Methods for Industrial Critical Systems – 18th Int'l Workshop, FMICS 2013*, volume 8187 of *LNCS*, pp. 48–62. Springer, 2013.

3.9 Boolean reasoning requires smart propositional encodings

Daniel Le Berre (Artois University, Lens, France)

Boolean reasoning has been used both in product and software configuration, with both success and failure stories. Encoding a problem into a Boolean satisfaction or optimization problem requires a lot of expertize: there are numerous ways to translate high level constraints into clauses, and intermediate solutions based on custom constraint propagators do exist. A great encoding is typically not a Boolean model of the initial problem but a way to describe a problem specific propagator for a Boolean engine. The choice of the Boolean input language (SAT, MAXSAT, Pseudo-Boolean Optimization) as well as the Boolean engine used may also have a deep impact on performances. It is thus important to model those problems into a high level input language such as ASP (Answer Set Programming), Alloy, MiniZinc, Copris/Scarab and to reuse state-of-the-art translators to produce Boolean formulas.

3.10 Configuration Evolution

Leonardo Gresta Paulino Murta (Federal University Fluminense – Niteroi, Brazil)

Both Product and Software Configuration can and do evolve over time. However, low attention is being provided to this problem. The discipline of Configuration Management can help to shed some light on this subject. In this talk, we introduce some basic Configuration Management concepts, discuss why general purpose Version Control Systems provide poor support for controlling evolution of more elaborate artifacts, and discuss some challenges of versioning Product and Software Configurations.

3.11 Configuration in Variability-Rich Software Ecosystems

Klaus Schmid (Universität Hildesheim, Germany)

License Creative Commons BY 3.0 Unported license
© Klaus Schmid
Joint work of Schmid, Klaus; Eichelberger, Holger; El-Sharkawy, Sascha; Kroeher, Christian; Brummermann, Hendrik; Keunecke, Markus
Main reference K. Schmid, "Variability Support for Variability-Rich Software Ecosystems," in Proc. of the 4th Int'l Workshop on Product Line Approaches in Software Engineering (PLEASE'13), pp. 5–8, IEEE, 2013.
URL http://dx.doi.org/10.1109/PLEASE.2013.6608654

While software product lines are a way to support a single organization to create a range of products from common assets with variability, an ecosystem involves a number of organizations that produce software and services, which mutually enrich each other. In an ecosystem the final "system" is only created at a later point, when the decision is made which software from which organizations will be combined to form the final system. Of course, both situations may happen simultaneously: each – or at least some – organization may employ a product line approach. This is what we call a *variability-rich software ecosystem* [1].

In such an ecosystem it is important to describe the composition of the individual parts as well as the variability of the individual product lines, both of which can be seen as forms of configuration. Moreover, the situation of an ecosystem leads to additional requirements for configuring each individual product line. Examples for this are that some sort of default modeling should be supported [2] so that each composition results in a complete configuration, some sort of modularization should be supported (e. g., like in CVL [3], using interfaces), and so forth. Besides the demands such a situation creates for the configuration itself, it also creates demands on the way the instantiation is performed (e. g., support for partial instantiation).

As a reaction to these demands, we created the EASy-Producer tool [4]. This supports the configuration and instantiation of variability-rich ecosystems. As part of this effort a specific variability modeling language (IVML) and a variability instantiation languages (VIL) were developed and implemented.

References

1 K. Schmid. Variability Support for Variability-Rich Software Ecosystems, 4th International Workshop on Product Line Approaches in Software Engineering (PLEASE) at the International Conference on Software Engineering (ICSE), 5–8, 2013.
2 H. Brummermann, M. Keunecke, K. Schmid. Formalizing distributed evolution of variability in information system ecosystems. Proceedings of the 6th Workshop on Variability Modeling of Software-Intensive Systems (VaMoS '12), ACM, 11–19, 2012.
3 CVL. Common variability language, 2012. http://www.omgwiki.org/variability/doku.php?id=start&rev=1351084099, Online accessed: 25.06.2014.
4 K. Schmid and E. Almeida. Product Line Engineering. IEEE Software, Vol. 30, No. 4, 24–30, 2013.

3.12 Performance Prediction in the Presence of Feature Interactions

Norbert Siegmund (Universität Passau, Germany)

Joint work of Norbert Siegmund, Sergiy Kolesnikov, Christian Kästner, Sven Apel, Don Batory, Marko Rosenmüller, Gunter, Saake
Main reference N. Siegmund, S. S. Kolesnikov, C. Kästner, S. Apel, D. Batory, M. Rosenmüller, G. Saake, "Predicting Performance via Automated Feature-Interaction Detection," in Proc. of the 34th Int'l Conf. on Software Engineering (ICSE'12), pp. 167–177, IEEE, 2012.
URL http://dx.doi.org/10.1109/ICSE.2012.6227196

Customizable programs and program families provide user-selectable features allowing users to tailor the programs to the application scenario. Beside functional requirements, users are often interested in non-functional requirements, such as a binary-size limit, a minimized energy consumption, and a maximum response time.

In our work, we aim at predicting a configuration's non-functional properties for a specific workload based on the user-selected features [2, 3]. To this end, we quantify the influence of each selected feature on a non-functional property to compute the properties of a specific configuration. Here, we concentrate on performance only.

Unfortunately, the accuracy of performance predictions may be low when considering features only in isolation, because inaccurate predictions. many factors influence performance. Usually, a property is program-wide: it emerges from the presence and interplay of multiple features. For example, database performance depends on whether a search index or encryption is used and how both features interplay. If we knew how the combined presence of two features influences performance, we could predict a configuration's performance more accurately. Two features interact (i. e., cause a performance interaction) if their simultaneous presence in a configuration leads to an unexpected performance, whereas their individual presences do not.

We improve the accuracy of predictions in two steps: (i) We detect which features interact and (ii) we measure to what extent they interact. In our approach, we aim at finding the sweet spot between prediction accuracy, measurement effort, and generality in terms of being independent of the application domain and the implementation technique. The distinguishing property of our approach is that we neither require domain knowledge, source code, nor complex program-analysis methods, and our approach is not limited to special implementation techniques, programming languages, or domains.

Our key idea to determine which features interact is the following: We measure each feature twice. In the first run, we try to measure the performance influence of the feature in isolation by measuring the variant that has the smallest number of additionally selected features. The second run, aims at maximizing the number of features such that all possible interactions that may influence on performance materialize in the measurement. If the influence of the feature in isolation differs with the influence when combined with other features, we know that this feature interacts. In the second step, we perform several sampling heuristics, such as pair-wise sampling, to determine the actual combinations of interacting features that cause interactions.

Our evaluation is based on six real-world case studies from varying domains (e. g., databases, encoding libraries, and web servers) using different configuration techniques. Our experiments show an average prediction accuracy of 95 percent, which is a 15 percent improvement over an approach that takes no interactions into account [1].

References

1 N. Siegmund, S. Kolesnikov, C. Kästner, S. Apel, D. Batory, M. Rosenmüller, and G. Saake. Predicting Performance via Automated Feature-Interaction Detection. In *Proc. ICSE*, pp. 167–177. IEEE, 2012.

2 N. Siegmund, M. Rosenmüller, C. Kästner, P. Giarrusso, S. Apel, and S. Kolesnikov. Scalable Prediction of Non-functional Properties in Software Product Lines. In *Proc. SPLC*, pp. 160–169. IEEE, 2011.

3 N. Siegmund, M. Rosenmüller, C. Kästner, P. Giarrusso, S. Apel, and S. Kolesnikov. Scalable Prediction of Non-functional Properties in Software Product Lines: Footprint and Memory Consumption. *Information and Software Technology*, 55(3):491–507, 2013.

3.13 A minimal introduction to product configuration

Juha Tiihonen (Aalto University, Finland)

A minimal history of product configuration includes rule-based configurators, early model-based configurators, mainstream configuration environments and mass customization toolkits [4]. The main ideas of product configuration modeling [2, 5]) cover connection-based, resource-based, structure-based, and function-based approaches. The related concepts are treated uniformly in an object oriented manner with the availability of taxonomic hierarchies with refinement, abstraction, and applicability of attributes. An integral part of product configuration modeling is supporting the variable compositional structure of components and functions/features. Common feature modeling concepts of basic, cardinality based and extended feature models including "complex" constraints [1] can therefore be easily mapped to concepts of the product configuration community. A demonstration with product configuration system WeCoTin [6] showed how the example model of [1] can be modeled and configured. Research challenges include personalized configuration, unification of configuration and feature models, community-based configuration, standardized configuration knowledge representations, intelligent user interfaces for configuration knowledge acquisition, intelligent testing and debugging, unobtrusive preference elicitation, and processes for intelligent systems development [3]. It is concluded that product configuration has a long and successful history and product configurators are applied relatively widely. Product configuration modeling techniques can be directly applied for representing many if not most feature models. It seems that many aspects of variability modeling from the product configuration community could be carried from product configuration community to software configuration community. However, management of variability is just one aspect of software product family modeling.

References

1 Benavides, D., Segura, S., & Ruiz-Cortes, A. (2010). Automated analysis of feature models 20 years later: A literature review. Information Systems, 35(6), 615-636. DOI: 10.1016/j.is.2010.01.001

2 Felfernig, A. (2007). Standardized configuration knowledge representations as technological foundation for mass customization. Engineering Management, IEEE Transactions On, 54(1), 41–56. DOI: 10.1109/TEM.2006.889066

3 Felfernig, A., Hotz, L., Bagley, C., & Tiihonen, J. (2014). Chapter 15 – Configuration-Related Research Challenges. In A. Felfernig, L. Hotz, C. Bagley & J. Tiihonen

(Eds.), Knowledge-Based Configuration (pp. 191–195). Boston:Morgan Kaufmann. DOI: 10.1016/B978-0-12-415817-7.00015-3
4 Hotz, L., Felfernig, A., Günter, A., & Tiihonen, J. (2014). Chapter 2 – A Short History of Configuration Technologies. In A. Felfernig, L. Hotz, C. Bagley & J. Tiihonen (Eds.), Knowledge-Based Configuration (pp. 9–19). Boston: Morgan Kaufmann. DOI: 10.1016/B978-0-12-415817-7.00002-5
5 Soininen, T., Tiihonen, J., Männistö, T., & Sulonen, R. (1998). Towards a general ontology of configuration. AI EDAM, 12(4), 357–372.
6 Tiihonen, J., Heiskala, M., Anderson, A., & Soininen, T. (2013). WeCoTin – A practical logic-based sales configurator. AI Communications, 26(1), 99–131. DOI: 10.3233/AIC-2012-0547

3.14 Challenges of topological variability

Andrzej Wasowski (IT University of Copenhagen, Denmark)

Main reference T. Berger, S. Stanciulescu, O. Ogaard, O. Haugen, B. Larsen, A. Wasowski, "To Connect or Not to Connect: Experiences from Modeling Topological Variability," to appear in the Proc. of the 18th Int'l Software Product Line Conf. (SPLC'14).

Classic variability models are non-structural. Both feature models and decision models focus on capturing sets of parameters, their names and dependencies between them. These are then used to configure the piecefal of software in question. One well known publicly available example is the Linux kernel, having a configurator driven by a simple variability model.

In installation engineering, a bit differently than in software, there is need for modelling component types, and their connections (topologies). Configurators derived from such models are used by installation engineers to design specifications for particular deployments. I present the problem using the example of fire alarm systems of a Norwegian vendor, Autronica. The presentation explains the shortcomings of variability models known from software product lines area for specifications of such systems.

This work has been funded by the ARTEMIS project VARIES on variability in safety critical systems.

3.15 Strategically Optimizing Product Portfolios

Patrick Wischnewski (Logic4Business – Saarbrücken, Germany)

The steadily increasing number of product variants is leading to a steady increase in costs and time expenditures in development, production and sales. Even more, designing variants with respect to the actual market situation in order to exactly meet the customer's demand requires optimization methods that enable the manufactures to strategically optimize their product portfolio.

Because of the discrete structure of the products, the respective optimization procedures are expensive from a computational point of view. Therefore, in order to successfully develop procedures that efficiently perform strategic optimizations for industrial size problems, two directions of research are necessary. The first direction is towards efficient optimization procedures. The second direction is towards efficient encodings of the problem of exactly matching the properties of the optimization procedure.

In several industrial projects we have successfully used SAT-based optimization procedures. In these projects we observed that finding the right encoding for a product portfolio and adjusting the optimization procedure respectively was the key for efficiently performing strategic optimizations in terms of the product portfolio.

4 Summary of Closing Discussion

The closing discussion of the seminar focused on identifying future research directions in product and software configuration.

During the discussion, the idea of easy-to-use but expressive configuration languages emerged as the main vision for the future. These two language properties were identified as at odds with each other, making achieving this vision challenging. The participants pondered whether such languages would be generic or domain-specific and questioned how much representational adequacy generic languages could achieve.

In addition to support for ease of modeling, such languages should also provide efficient reasoning capabilities without burdening the user. The discussion explored the idea of targeting a range of solvers using intelligent solver-selection logic and translators determining the most efficient problem encodings. The languages should also support a smooth transition between different computational classes, without the need to reformulate existing configuration models.

A challenge posed by the expressive-language vision is the ability to quickly and reliably classify a given problem based on its characteristics and identify the most appropriate solvers and encodings. The participants agreed about the need to create a body of knowledge classifying configuration problems and the most appropriate solving techniques for each class.

The participants also recognized usable configurators as an additional important research direction. Existing works on this topic are very sparse; however, tool builders and problem modelers require clear guidance on how to design effective interactions with users. This direction requires multi-disciplinary efforts, including human-machine interface, cognitive science, and experimental research.

Finally, the participants postulated that achieving progress in the field would require creating widely accessible collections of benchmark problems. They also identified the diversity of configuration languages and tasks as main challenges in creating such benchmarks. Standard formats such as DIMACS in the SAT community greatly simplify creating benchmarks. However, other communities offer some positive experiences in addressing these challenges. For example, the CSP community has created the MiniZinc language, which is used as a frontend for wide range of solvers.

Participants

- Michel Aldanondo
University of Toulouse, FR
- Danilo Beuche
pure-systems GmbH – Magdeburg, DE
- Nikolaj Bjorner
Microsoft Res. – Redmond, US
- Krzysztof Czarnecki
University of Waterloo, CA
- Conrad Drescher
SAP AG – Walldorf, DE
- Andreas Falkner
Siemens AG – Wien, AT
- Jianmei Guo
University of Waterloo, CA
- Albert Haag
SAP AG – Walldorf, DE
- Holger H. Hoos
University of British Columbia – Vancouver, CA
- Lothar Hotz
HITeC e. V. / Universität Hamburg, DE
- Arnaud Hubaux
ASML – Veldhoven, NL
- Dietmar Jannach
TU Dortmund, DE
- Christian Kästner
Carnegie Mellon University, US
- Eunsuk Kang
MIT – Cambridge, US
- Charles Krueger
BigLever – Austin, US
- Wolfgang Küchlin
Universität Tübingen, DE
- Daniel Le Berre
Artois University – Lens, FR
- Tomi Männistö
University of Helsinki, FI
- Leonardo G. P. Murta
Federal University Fluminense – Niteroi, BR
- Klaus Schmid
Universität Hildesheim, DE
- Norbert Siegmund
Universität Passau, DE
- Markus Stumptner
University of South Australia Adelaide, AU
- Juha Tiihonen
Aalto University, FI
- Andrzej Wasowski
IT Univ. of Copenhagen, DK
- Patrick Wischnewski
Logic4Business – Saarbrücken, DE
- Ed Zulkoski
University of Waterloo, CA

Report from Dagstuhl Seminar 14181

Multi-agent Systems and their Role in Future Energy Grids

Edited by

Michael N. Huhns[1], Wolfgang Ketter[2], Ryszard Kowalczyk[3], Fabrice Saffre[4], and Rainer Unland[5]

1 **University of South Carolina, US,** huhns@sc.edu
2 **Erasmus University – Rotterdam, NL,** wketter@rsm.nl
3 **Swinburne University – Melbourne, AU,** rkowalczyk@swin.edu.au
4 **BT Research and Innovation – Ipswitch, UK,** fabrice.saffre@bt.com
5 **Universität Duisburg-Essen, DE,** rainer.unland@icb.uni-due.de

Abstract

This report documents the program and the outcomes of Dagstuhl Seminar 14181 "Multi-agent systems and their role in future energy grids". A number of recent events (e. g. Fukushima, Japan, and the largest blackout in history, India) have once again increased global attention on climate change and resource depletion. The evaluation of the feasibility of current approaches for future energy generation, distribution, transportation, and consumption has become an important requirement for most countries. There is a general consensus on the need for a fundamental transformation of future energy grids. The development of an information and communication technology (ICT) support infrastructure was identified as the key challenge in the design of an end-to-end smart grid. A multiagent system, with agents located at the edges and nodes of the grid and representing the interests of end-users, distributors, and providers, enables intelligent decisions to be made at each node in the electric power distribution network (grid). The seminar fostered discussions among experts from all relevant disciplines is to develop the foundation for the necessary interdisciplinary solution from engineering, computer science, and business management. The outcome was an understanding and identification of the requirements on the information systems for future smart grids.

Seminar April 27 to May 2, 2014 – http://www.dagstuhl.de/14181
1998 ACM Subject Classification C.3 Special-Purpose and Application-Based Systems, I.2.11 Distributed Artificial Intelligence
Keywords and phrases Multiagent Systems, Demand-Side Management, Smart Grid, Eneregy Distribution
Digital Object Identifier 10.4230/DagRep4.4.37

1 Executive Summary

Michael N. Huhns
Wolfgang Ketter
Ryszard Kowalczyk
Fabrice Saffre

Due to the depletion of scarce resources for energy production and the problems associated with climate change, there is widespread interest in new approaches formanaging energy generation, distribution, transportation, and consumption. Overall we must find a way to combine the economics, physics, physical components, and governmental policies and regulations of energy systems, while satisfying the personal preferences of consumers. The goal is to create a global Smart Grid.

The main differences between the current and the envisioned future grid are the production ecosystem on one hand and the information exchange on the other. Current grids traditionally rely on a comparatively stable number of large power plants that produce a constant and predictable amount of power, as well as on smaller power plants that can be activated quickly if demand requires it. Power and information flow from the supply side to the demand side. This is reflected in the underlying business models, which are mainly dictated by the prices the few big producers can achieve on the global market and by the costs for transmitting the power through a distribution system usually owned by private companies. This will change as more and more renewable and distributed generation technologies spread to a household level.

The distinction between producer and consumer will become increasingly blurred as the flow of power as well as information among the resultant *prosumers* becomes bi-directional. The current grid operates at a high-voltage level suited for long distance delivery, while a prosumer-based network will be a more localized and low-voltage grid. Further, the increasing use of renewable sources will result in a less predictable generation pattern, a matter which in itself is raising a number of interesting challenges. In short, the new power grids will differ in magnitude and direction as well as in generation consistency, which will require a complete revision of the underlying business model as the currently predominant global (or, at least, national) market will be replaced by a number of local markets that will have to maintain the balance between supply and (individually generated) demand, i. e., market places for power generation as well as power consumption.

The development of an information and communication technology (ICT) support infrastructure will be the key challenge in the design of an end-to-end smart grid framework. This will require the capability to balance supply and demand and to handle complex operations. The efficient, real-time exchange of information and the coordinated decisions among many stake holders (consumers, distributors, transporters, and generators) have to be supported. This is not possible within the structure and practice of the current grid. Different levels of the grid (layout, control, ICT infrastructure, maintenance, failure handling, and business models), as well as the communication and cooperation among these levels, needs to be fully coordinated with all the other levels. To predict the emergent properties of the system under a range of different conditions and worst-case scenarios, extensive and effective simulation tools will be required. A solution to this large and very complex problem requires intelligent decisions to be made at each node in the electric power distribution network (grid), especially at the edges. To be manageable, the decisions must take advantage of locality constraints and end-user preferences. A multiagent system, with agents located at the edges and nodes of the grid and representing the interests of end-users, distributors, and providers, satisfies these requirements. It is thus the default system solution thatwas considered first and adopted at the Dagstuhl.

2 Table of Contents

3 Overview of Seminar Talks

The seminar featured an introductory talk by Fabrice Saffre, who outlined the issues in Demand-Side Management and contrasted them with the Demand-Response approach.

3.1 Demand-Side Management (DSM)

Fabrice Saffre (BT Research and Innovation – Ipswitch, UK)

The need for DSM arises as soon as:
1. The ratio between the supply of and demand for a resource fluctuates over time
2. The resource cannot be (efficiently) stockpiled for future use

We can find inspiration from biology based on the Lotka-Volterra predator-prey model, demonstrating how fluctuations between supply and demand are inevitable, but still must be controlled. The current approach to such control is the centrally orchestrated Demand Response model, favored by the power generation and distribution industry.

In contrast, DSM implies distributed choreography. It tries to shape the demand profile by providing incentives to consumers to time-shift their flexible loads away from periods during which resources are scarce. There are two problems with this:
1. It relies on the fiction that the average human is a rational being capable of identifying an optimal strategy
2. It disregards the “inconvenience cost” of doing things at unusual times

A solution to these problems is to employ intelligent agents to represent consumers, resulting in a multiagent system for control. Unlike humans, software agents can be built to specification, to predictably and reliably follow a set of algorithmic rules, and to not suffer the inconvenience of having a circadian rhythm. Putting agents in control of scheduling flexible loads (within the limits fixed by the owner) bears the promise of realising many more opportunities for DSM. The challenge is to define a set of rules that, when applied autonomously by each agent (based on information gathered through direct or indirect interaction with other agents), leads to the collective behavior of the entire population improving the match between supply and aggregated demand.

We postulate that game theory might not be the most suitable paradigm or source of inspiration. Moreover, market-based mechanisms, arguably game theory’s most successful offsprings/relatives/application, might not offer the best solution either. Competition between selfish rational entities with unrestricted access to global, transparent information not only requires complex reasoning (with the associated risk of delays and computational overhead), it can lead to chronic inefficiency. This is because the social optimum often differs from the Nash equilibrium. We conclude that
1. Demand-Side Management could lead to massive waste reduction through better utilization of transient resources
2. A successful, widely applicable DSM solution could hold the key to “sustainability without austerity”
3. It is often assumed that DSM always means flattening the load profile, i. e., “shaving the peaks, filling the troughs”

4. Although it remains true in some cases (e. g., for bandwith), it is no longer necessarily so, especially in the energy sector where intermittent renewable sources (solar, wind, ...) introduce variation on the supply side
5. Robust methods for incorporating dynamic targets and constraints on flexibility into DSM would be extremely valuable

3.2 Multiagent Systems Enabling the Smart Grid

Michael N. Huhns (University of South Carolina – Columbia, SC, US)

Energy allocation and distribution is a societal problem, which involves the management of scarce resources. Solving it requires a socio-technical system. The design of the system depends on verifying the following two hypotheses: (1) Participation: A sufficient number of people in a society can be motivated to participate either directly or indirectly via their intelligent software agents in the management of an essential and limited resource (electric power); (2) Stability: A system of interacting agents cooperating and competing for resources on behalf of a community of users will produce a controllable, stable, and prosocial allocation of resources.

Our approach relies on the following premises:

- Premise 1. Current pricing incentives are insufficient, because they are based on a history of past aggregate behavior and have little predictive value.
- Premise 2. The community of consumers exhibits rich social relationships and energy usage dependencies that can be handled better through peer-to-peer interactions rather than through centralized control.

3.3 Smart Grid Projects at University of Passau

Hermann de Meer (Universität Passau, DE)

Demand-Side Management (DSM) has been emerging as an important approach to mitigate volatile renewable power sources or other causes of demand and supply mismatches. We have developed the concepts of GreenSLAs (service level agreement) and GreenSDAs (supply demand agreement) as the basis of DSM schemes within the European projects ALL4Green and DC4Cities. While All4Green has been investigating peak shaving techniques within a demand response setting, DC4Cities has been more focused into following closely renewable energy sources as the pure basis of power supply within the confines of a smart city context. Both projects apply DSM in the context of existing automation frameworks of data centers (DC). DCs are large consumers of power while being relatively flexible in their power consumption profiles. Extensions of current grid operational conditions within the transition into the smart grid paradigm, introduce new dynamics and substantial risk potentials. To deal with such risks in a hybrid and multidisciplinary smart grid setting, has been the focus of the European HYRIM project.

3.4 Multiagent Systems Enabling the Smart Grid

Minjie Zhang (University of Wollongong, AU)

We briefly introduce two completed projects in agent-based modelling and simulation of power grid systems. In the first project, a Multi-Agent System (MAS) was proposed to automatically diagnose faults in a power grid network and to automatically execute emergency controls to prevent catastrophic failures of the network. A three-layer agent-based emergency control model was proposed to adaptively control the power grid system during its daily operations and emergencies, and an agent-based Q-learning approach was also proposed to restore the system automatically from faults and outages.

In the second project, MAS solutions were developed for managing a power distribution system by considering the renewable distributed generators. The proposed approaches employed the decentralized management to control distributed electric components in the distribution system adaptively for dynamically balancing power consumption and supply through using agent- based communication, decision-making, and cooperation. The MAS approaches required no dependency between agents and can be easily extended to any scale through using individual agents as "plug and operate" units.

4 Working Groups

The seminar participants self-organized into four working groups for discussions on physics-based models of energy grids so that they can be controlled, the control of such grids via agents, and the interactions among the agents and the humans and organizations they represent. The four groups were the following:

1. Multiagent systems for future energy grids, including dynamics and stability, robust / adaptive quality of service, self-organization, and emergent behavior
2. The smart grid as a network of networks
3. Market modelling, design, and simulation, including interaction with users, user behavior, and user privacy
4. Efficient scheduling, optimization, and control of energy and resources

4.1 Working Group 1: Multiagent Systems for Future Energy Grids

Wolfgang Renz (HAW – Hamburg, DE)

Joint work of Matthias Bürger, Michael N. Huhns, Christoph Mayer, and Peter Palensky

This working group focused on system dynamics: dynamics and stability, robustness, QoS, adaptivity, self organization and self*, and emergence. The basic question addressed was how to bridge the gap between a micro-model and macro-behavior using an empirical method based on data from sampling a parameter space to produce a roadmap for white-, gray-, and black-box models.

The recommendation is to create a stability testbed for complex energy systems. It would incorporate instability mechanisms in simplistic scenarios, with ideas on how to find them and offering collections of instability candidates. Its software architecture would be scalable and distributed. It would support plug-ins and interfaces among information-based and physics-based agents.

For an example of system dynamics, a households app switches on loads synchronously, thus causing higher prices. Consequently an industrial load switches off, which causes a reduced price. Households would profit from the lower prices at the expense of the industrial entity. This would recur when the industrial load is switched on again.

The identified features of a proposal for applied research are

- A starting situation where aggregation and brokerage are done asynchronously by humans and reaction times are measured in minutes
- A future situation where aggregation and brokerage are automated synchronously and reaction times are measured in seconds
- Example risks involve unstable power, price manipulation, distribution grid bottlenecks, and instabilities
- Objectives are to analyze risks, construct intelligent agents, and develop market rules, products, and demonstrations

4.2 Working Group 2: Network of Networks

Fabrice Saffre (BT Research and Innovation – Ipswitch, UK)

We started from the realization that many of the problems encountered when trying to match energy supply and demand, particularly in the face of intermittent, unpredictable, and/or small renewable generating facilities (solar panels, wind turbines, etc.), emerged from the difficulty of untangling the intricate web of interdependencies among increasingly flexible and/or "intelligent" loads (such as the Internet of Things). A centralized method taking into account all constraints and uncertainties to identify an optimal schedule rapidly becomes intractable as the number of prosumers increases. In fact, depending on the level of flexibility in individual loads, the approach can start to break down beyond a mere handful of loads due to combinatorial effects.

From these considerations, we set off to create and test an experimental distributed management framework to try and address these questions. The first step in the chosen approach consists in forming expanding clusters of prosumers whose individual energy consumption/generation patterns are identified as compatible. At this stage, by "compatible" we simply mean that there exists a combined schedule (taking into account any available flexibility) for which the aggregated supply/demand profile is "flatter" for the cluster as a whole than it would be for its individual constituents.

Note that we prefer using the expression "cluster" rather than the more loaded term "coalition," because we deliberately avoid using market-based dynamics or methods inspired from game theory in favor of a simpler "mechanistic" approach in which prospective groups are formed through random aggregation. The newly formed cluster is then created or discarded based on compatibility, and the process is repeated until no more successful pairings are found.

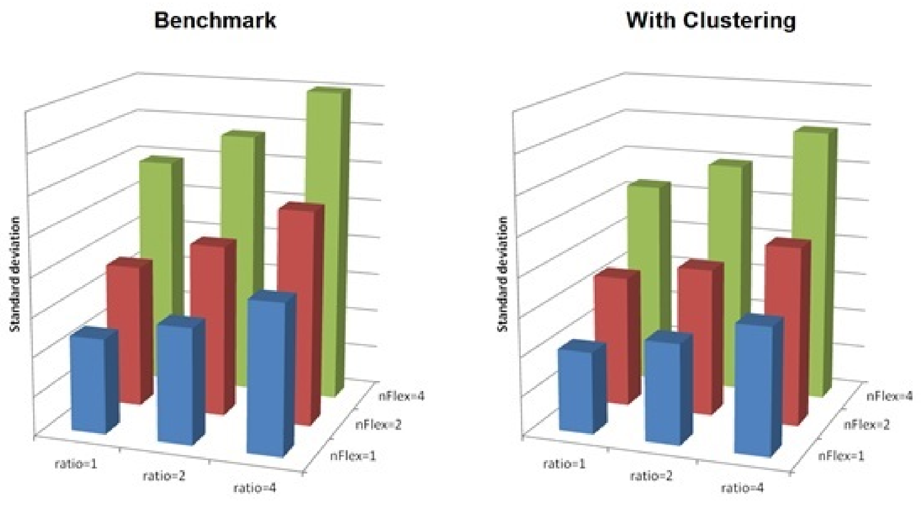


Figure 1 Early results.

Every so-formed cluster is considered a "meta-prosumer" and, in order to prevent a combinatorial explosion, its overall flexibility is restricted to a chosen target by "freezing" the schedule (execution time window) of some of the loads. Early results suggest that this heuristic approach can successfully flatten the aggregated profile (see Figure 1), approximating the optimal schedule at a minute fraction of the computational cost of an exhaustive search.

An additional challenge is found in the necessity to accommodate the limitations of the famously antiquated distribution grid, the topology of which severely constrains the amount of power that can "flow" from one node to another. Indeed, its "hierarchical" tree-like design is intended to provide adequate capacity for electricity to "cascade" from central generation facilities down to the end user, not to allow power generated at one leaf to be channelled to another, as a micro-grid scenario effectively requires.

Future work will involve taking such constraints into account when assessing the compatibility between members of a prospective cluster, as well as using realistically "peaky" 24-hour demand patterns. Both refinements are necessary in order to evaluate the potential of our prosumer clustering method in a practical deployment scenario.

4.3 Working Group 3: Market Modeling, Design, Simulation, and Interaction with Users, Including User Behavior and User Privacy

Gilbert Fridgen (University of Augsburg – Augsburg, DE)

Joint work of Sebastian Lehnhoff, Tobias Linnenberg, Tina Balke, Sonja Klingert, Costin Badica, Micha Kahlen, Michael Kaisers, and Wolfgang Ketter

In investigating markets, the group made the assumptions that there would be a local cooperative behind each feeder, it would work to maximize welfare but with individual constraints, it would have static and exclusive membership, it would operae as the sole trader on behalf of its members, and the agents of its members would behave cooperatively. The group outlined its objectives as

- Use PowerTAC and integrate it with a domain-specific simulator
- Provide high usability (for "normal" human users)
- Create a mixed-initiative system

It will be evaluated on use cases for training, analysis, and design. Its architecture will comprise a platform development / simulator made up of PowerTAC and other frameworks, a learning user interface agent, a preference elicitation tool (for single user preferences, such as "one is on holiday and won't need the energy and may thus provide it to its neighbor, who might need it for an EV"), and a scoping process. It would be useful in industry as training for traders, i. e., suppliers and aggregators, and as a micro-grid for energy cooperatives and local suppliers.

The open research questions are

- How to setup such an energy-cooperative?
  - how to align economic incentives?
  - how to setup a coordination scheme?
  - which communication mechanisms are needed?
- Is there a necessity for an infrastructure change, when we introduce energy cooperatives with agent-based control? (in terms of a "parallel network")
- How to build up agent-based cooperative business models for service-oriented providers? (in relation to getting them started)
- How to design, develop, and apply a decision support tool?

4.4 Working Group 4: Resource Efficient Scheduling, Optimization, and Control

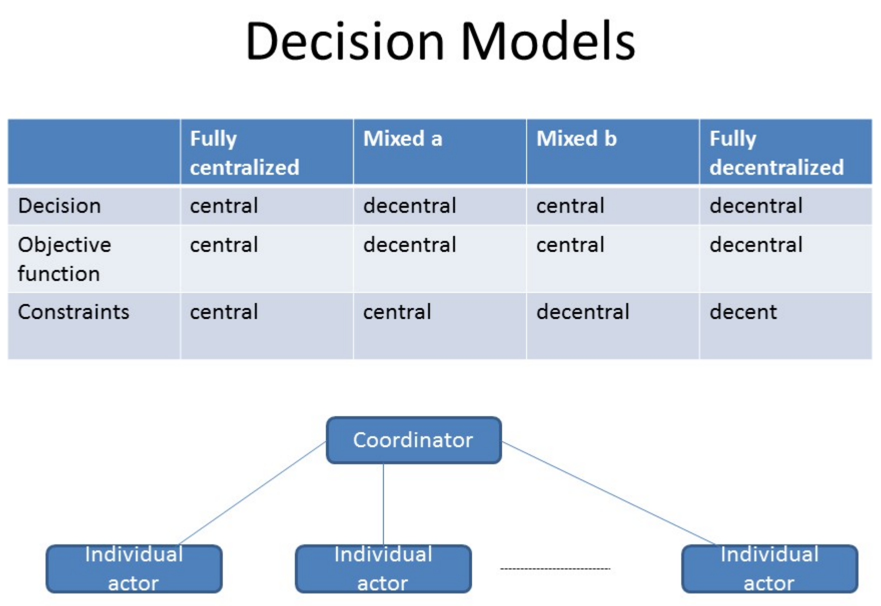
Marjan van den Akker (Utrecht University – Utrecht, NL)

Joint work of Christian Derksen, Ryszard Kowalczyk, Lars Mönch, Konstantina Valogianni, Eric van Heck, and Minjie Zhang

Utilizing resources efficiently requires different kinds of decisions to be made. These are shown in the Figures 2 and 3.

For agent-based distributed hierarchical decision making in future energy micro-grids, the group identified the following research questions:

1. How to balance supply-and-demand at different scales (e. g., planning vs. control)?
2. What are the decision making entities in different schemes? (e. g., local vs. global and makers vs. accountability)
3. What are the decision criteria? Global: cost/efficiency, QoS, CO2 ... Local: min cost (energy, depreciation), satisfy needs (supply guarantee), ...
4. How to align, via feedback and feed forward loops, capacity creation and utilization?
5. How to support/enable/facilitate the alignment (feedback and dynamic trade-offs/priorities) between the concerns of physical + economical + environmental + convenience/flexibility + regulatory ... E.g., different objective functions and different constraints; minimum cost vs. maximum QoS; minimum cost + user flexibility as constraints

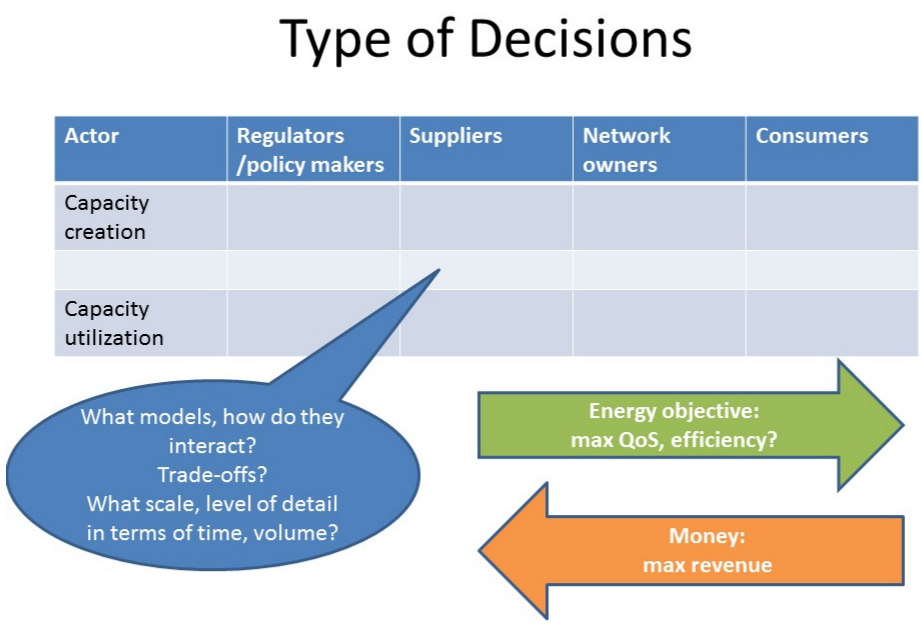


Decision Models

	Fully centralized	Mixed a	Mixed b	Fully decentralized
Decision	central	decentral	central	decentral
Objective function	central	decentral	central	decentral
Constraints	central	central	decentral	decent

Coordinator
Individual actor
Individual actor
Individual actor

Figure 2 Decision models.



Type of Decisions

Actor	Regulators /policy makers	Suppliers	Network owners	Consumers
Capacity creation				
Capacity utilization				

What models, how do they interact? Trade-offs? What scale, level of detail in terms of time, volume?
Energy objective: max QoS, efficiency?
Money: max revenue

Figure 3 Type of decisions.

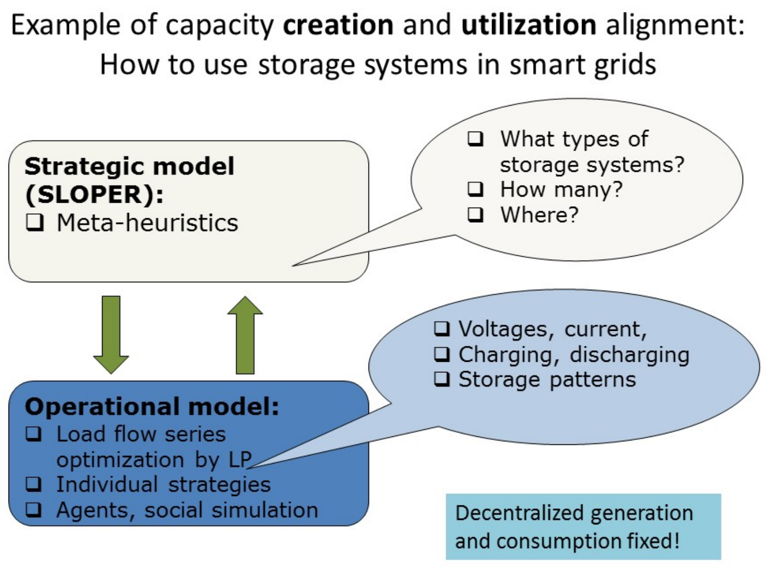

Example of capacity **creation** and **utilization** alignment:
How to use storage systems in smart grids
Strategic model (SLOPER):
❑ Meta-heuristics
❑ What types of storage systems?
❑ How many?
❑ Where?
❑ Voltages, current,
❑ Charging, discharging
❑ Storage patterns
Operational model:
❑ Load flow series optimization by LP
❑ Individual strategies
❑ Agents, social simulation
Decentralized generation and consumption fixed!

Figure 4 Example of capacity creation and utilization alignment.

The group recommended that the following project-staged approach would be effective. Start simple with a capacity utilization case, then extend with capacity creation, alignment cycle, etc. Specifically,

- DM Modeling (1, 2) (decision roles, decision types) considering different DM schemes (centralized, decentralized, mixed 1, mixed 2)
- Framework design (e. g., modularize/decompose into hierarchy (abstract level/layers/-modules?), decide on scheme for each layer, agent roles, interaction, communication, relationships/organization)
- Mechanism/techniques design (e. g., LP for centralized, auction for broker, meta-heuristics, machine learning, etc.)
- Evaluation (theoretical, experimental, etc.)
- Validation (case study, expert reviews, etc.)
- And then iterate . . .

5 Open Problems

Open problems were described throughout the previous sections, particularly in the working group summaries.

Participants

- Marco Aiello
University of Groningen, NL
- Costin Badica
University of Craiova, RO
- Tina Balke
Univ. of Surrey – Guildford, GB
- Matthias Bürger
Universität Stuttgart, DE
- Liana Cipcigan
Cardiff University, GB
- Hermann de Meer
Universität Passau, DE
- Christian Derksen
Universität Duisburg-Essen, DE
- Gilbert Fridgen
Universität Bayreuth, DE
- Hanno Hildmann
NEC Laboratories Europe – Heidelberg, DE
- Michael N. Huhns
University of South Carolina – Columbia, US
- Micha Kahlen
Erasmus Univ. – Rotterdam, NL
- Michael Kaisers
CWI – Amsterdam, NL
- Stamatis Karnouskos
SAP Research – Karlsruhe, DE
- Wolfgang Ketter
Erasmus Univ. – Rotterdam, NL
- Sonja Klingert
Universität Mannheim, DE
- Matthias Klusch
DFKI – Saarbrücken, DE
- Ryszard Kowalczyk
Swinburne University – Melbourne, AU
- Winfried Lamersdorf
Universität Hamburg, DE
- Sebastian Lehnhoff
OFFIS – Oldenburg, DE
- Tobias Linnenberg
Universität der Bundeswehr – Hamburg, DE
- Christoph Mayer
OFFIS – Oldenburg, DE
- Lars Mönch
FernUniversität in Hagen, DE
- Sascha Ossowski
University Rey Juan Carlos – Madrid, ES
- Peter Palensky
Austrian Institute of Technology – Wien, AT
- Wolfgang Renz
HAW – Hamburg, DE
- Fabrice Saffre
BT Research – Ipswich, GB
- Rainer Unland
Universität Duisburg-Essen, DE
- Konstantina Valogianni
Erasmus Univ. – Rotterdam, NL
- Marjan van den Akker
Utrecht University, NL
- Eric van Heck
Erasmus Univ. – Rotterdam, NL
- Minjie Zhang
University of Wollongong, AU

Report from Dagstuhl Perspectives Workshop 14182

Categorical Methods at the Crossroads

Edited by

Samson Abramsky[1], John Baez[2], Fabio Gadducci[3], and Viktor Winschel[4]

1 **University of Oxford, GB,** samson.abramsky@cs.ox.ac.uk
2 **University of California – Riverside, US,** baez@math.ucr.edu
3 **University of Pisa, IT,** gadducci@di.unipi.it
4 **Universität Mannheim, DE,** winschel@rumms.uni-mannheim.de

Abstract

This report documents the program and the outcomes of Dagstuhl Seminar 14182 "Perspectives Workshop: Categorical Methods at the Crossroads". The aim of the meeting was to investigate the potential of category theory as a paradigm for mathematical modeling and applied science. The envisaged application areas included computation, physics, biology, complex systems, social and cognitive science and linguistics. Many of these areas were indeed tackled in the variety of topics dealt with during the workshop.

Each working day followed the same structure: two survey lectures during the morning, followed by three shorter talks in the afternoon, and closed by a working group session. During these sessions the attendants split into several groups according to the main thematic areas that had been identified on the first day. Both surveys and talks are reported in the "Overview" section of the report, while a wrap-up of the discussions that occurred inside the working groups is reported in the "Working Groups" section.

Perspectives Workshop April 27 to May 2, 2014 – http://www.dagstuhl.de/14182
1998 ACM Subject Classification D.3.1 Formal Definitions and Theory, F.3.1 Specifying and Verifying and Reasoning about Programs, F.3.2 Semantics of Programming Languages, F.4.1 Mathematical Logic, I.2.1 Applications and Expert Systems
Keywords and phrases category theory, concurrency, economics, game theory, logics, quantum computing, semantics
Digital Object Identifier 10.4230/DagRep.4.4.49

1 Executive Summary

Fabio Gadducci

Since the 1960s, category theory has been recognised as a powerful conceptual framework and a flexible specification language. The range of research areas where categorical methods found application is quite wide: from physics, economics, and linguistics to many branches of mathematics, especially algebraic geometry, algebraic topology, and logic. And, of course, computer science: possibly the discipline, apart from mathematics, where these methods have been most wholeheartedly adopted. Indeed, they have become part of the standard "tool-box" in many areas of theoretical informatics, from programming languages to automata, from process calculi to type theory.

Despite their flexibility and expressiveness, a more general acceptance of categorical methods has been hindered by the perceived difficulties of the formalism. As a consequence, many researchers in different communities share the feeling of under-exploitation of the potentialities of category theory to their areas of interest. This Dagstuhl Perspectives Workshop seminar brought together people from various disciplines that are interested in the application of categorical tools in their research area, from computer science towards venues such as economics, mathematics, and physics.

Besides the benefits to the understanding of each topic that came from a plurality of voices in a discussion, the meeting tried to address more general concerns. As far as some disciplines are concerned, the workshop helped the reconciliation of different research strands that uses categorical tools. Most importantly though, the workshop aimed at building bridges between disciplines, by reviewing the variety of uses of categorical methods in different fields and trying to find common abstractions that allow the same structures and concepts to be recognized as they arise in different settings, and to be transferred from one area to another.

In order to put on firm grounds the foundations of a common language, each working day included two survey lectures during the morning, which presented a variety of topics where categorical methods play a major role. These were followed by three shorter talks in the early afternoon, which presented active areas and innovative application for these methods. The day was closed by a working group session: during these sessions the attendants split into several groups according to the main thematic areas that had been identified on the first day. The variety of topics dealt with in the workshop was large, and the suggested application areas included (quantum) computation, physics, biology, complex systems, economic, social and cognitive science, and linguistics. Indeed, some of the items span more than one discipline, e. g. game theory, and the list is definitively not exhaustive.

Although the scope of the workshop was broad, the over-arching research theme was to develop categorical methods as a unified approach to the modeling of complex systems, and category theory as a paradigm for mathematical modeling and applied science. To this end, the overall purpose of the workshop was to start developing a coherent research community applying categorical methods to a wide range of disciplines. Under these terms, the workshop has been indeed successful. Laying out a common mathematical language and finding analogies among apparently distant concepts from unrelated disciplines provided a basis for fruitful cross-disciplinary interactions also by facilitating a "technology transfer". Concretely, this led to the fostering of new collaborations among the participants, and the preliminary exploration of new directions and research themes.

2 Table of Contents

3 Overview of Talks

3.1 Categories at the crossroads

Samson Abramsky (University of Oxford, GB)

This talk laid out some of the themes for the workshop. The main aim of the meeting was formulated as considering category theory as a new paradigm for mathematical modelling and applied science. The envisaged application areas include computation, physics, biology, complex systems, social and cognitive science and linguistics. Much of the work using category theory within Computer Science to date has been within the field of semantics of programming languages. It was suggested that the potential scope for the application of categorical methods within CS is much wider than this, and it may be fruitful to loosen the traditional role of a preconceived syntax. Examples of current work in this spirit includes coalgebra and the use of diagrammatic and graphical formalisms. New territories to conquer include algorithms and complexity, intensional computation, and logic in AI. As an example of the last of these, valuation algebras were discussed as a foundational formalism for local computation and generic inference, and it was shown how they could be formulated very naturally in categorical terms. As another case study, the application of categorical methods in physics, in particular to the study of contextuality and non-locality, was described. This use of categorical methods leads to the recognition of general structures which arise in many situations outside the quantum realm, and to a general notion of contextual semantics, with a wide range of applications.

Some general methodological principles and advantages of category theory as a methodology for mathematical modelling were articulated. Finally, some practical issues involved in developing the community working in this area, and achieving fruitful interactions with and impact upon the wider community, were discussed.

3.2 Network theory

John C. Baez (University of California – Riverside, US)

Nature and the world of human technology are full of networks. Researchers in many fields draw diagrams of networks: flow charts, electrical circuit diagrams, signal-flow graphs, Bayesian networks, Feynman diagrams and the like. Mathematicians and computer scientists know that in principle these diagrams fit into a common framework: category theory. But we are still far from a unified theory of networks. We give an overview of the theory as it stands now, with an emphasis on topics for future research.

3.3 Introduction to differential and tangent categories

Robin Cockett (University of Calgary, CA)

Differential categories come in two complementary colours: as *tensor* differential categories and as *Cartesian* differential categories. Tensor differential categories have their roots in linear logic and the work of Thomas Ehrhard who provided several key examples of these settings. The coKleisli category of a tensor differential category is a Cartesian differential category – although it is not the case that a Cartesian differential category need to arise. Cartesian differential categories provide the categorical semantics for the "resource λ-calculus" (Curian, Boudel) which is equivalently the "differential λ-calculus" (Ehrhard, Regnier) – both these systems arose from computer science considerations. Similarly, the differentiation of combinatorial species (Joyal) can be organized to provide a Cartesian differential category – and the differentiation there is the same as the differentiation of datatypes (Huet, Abbott, McBride). These applications are significant as they illustrate an key feature of differential categories: they are "additive" settings which do not assume the presence of negatives. Of course, the standard differential from elementary calculus – which absolutely assumes negatives – also gives an example of a Cartesian differential category.

While differential categories collect a wide range of examples they fail to explain examples arising from differential geometry. Furthermore, there are now a number of important variations on differential geometry. For example: synthetic differential geometry, known as SDG (Kock, Lawvere), convenient differential geometry (Kriegl, Michor), and tropical differential geometry (Mikhalkin). This last example is of special significant to this development as negatives are not assumed. In order to capture these settings – and the standard settings from differential geometry and algebraic geometry – Geoff Cruttwell and I introduced the notion of a "tangent category". It turned out that we were not the first to have developed these ideas. Rosicky some thirty years earlier had introduced the notion of a tangent functor. His excellent but brief paper on the subject had received one citation in that time and, indeed, it would have escaped our notice but for Anders Kock who brought it to our attention.

A tangent category is essentially a category with a tangent functor in the sense of Rosicky – although reformulated so that negatives are not required. This associates naturally to each object a tangent bundle, $p : T(A) \to A$, satisfying certain axioms. Significantly the notion not only captures all the settings mentioned above, but also the notion of a Cartesian differential category. Even more significantly, once one reformulates the notion of "vector bundles" for the setting, the category of such bundles over any object naturally forms a Cartesian differential category – this prompted us to name this abstract reformulation a "differential bundle" to emphasize this connection.

At the time that Rosicky's paper was written many of the above connections could not have been made – the surrounding geography of ideas had not yet been developed. In retrospect, however, one can see that his notion – suitably generalized to the additive setting – unifies the broad subject matter of differential geometry. This makes it reasonable to speak of the subject embodied by tangent categories as the subject of "abstract differential geometry".

3.4 Category theory for the masses: A tale of food, spiders and Google

Bob Coecke (University of Oxford, GB)

We will demonstrate the following. Category theory, usually conceived as some very abstract form of meta-mathematics, is present everywhere around us. Explicitly, we show how it provides a kindergarten version of quantum theory as well as a new process-based foundation of it, how it helps to automate quantum reasoning, and how it will help Google to understand sentences given the meaning of their words.

3.5 Categories in cognition: An integrative model for multi-systems

Andrée Ehresmann (University of Amiens, FR)

Biological, cognitive or socio-economic systems are evolutionary, self-organized, multi^3 systems meaning: multi-level hierarchy, multi-agent cooperation/concurrence, multi-temporality. While most models retain only some of these properties, the *Memory Evolutive Systems* (developed with J.-P. Vanbremeersch since several years) propose an integrative approach to such systems, accounting for their local/global internal dynamics "in the making" and their emergence properties. MES rely on various categorical tools: "partial" (pre)sheaves, (co)limits and hierarchical categories, (pro-)sketches, leading to the Emergence and Double Complexification Theorems. The talk gives (http://ehres.pagesperso-orange.fr/Dagstuhl.pdf) give a brief summary of the main characteristics of MES, emphasizing some difficult computational problems. For more details, cf. our book *Memory Evolutive Systems: hierarchy, emergence, cognition* (Elsevier, 2007). For recent applications of MES to cognition, to creativity, to innovation in design and to anticipation, cf. the site http://ehres.pagesperso-orange.fr.

3.6 Intermodelling, queries and Kleisli categories

Tom Maibaum (McMaster University – Hamilton, CA)

After a brief review of some CT based approaches used in software engineering research, we explore the specification and maintenance of relationships between models, which are vital for Model Driven Engineering. We show that a wide class of such relationships can be specified in a compact and precise manner if inter-model mappings involve derived model elements computed by corresponding queries. Composition of such mappings is not straightforward and requires specialized algebraic machinery. We present a formal framework, in which such machinery can be generically defined for a wide class of meta-model definitions, and thus important inter-modeling scenarios can be algebraically specified and formalized.

3.7 Coalgebraic methods in epistemic game theory

Martin Meier (Institute for Advanced Studies – Vienna, AT)

Economic game theory has be concerned with incomplete and imperfect information since the seminal papers of Harsany in the Sixties. Harsanyi already then has been thinking about the problem of modelling expectations of players and concretely about the expectations about expectations, i. e. hierarchies of mutual expectations of the players involved. He considered the mathematical constructions too difficult and the problem was not tackled until much later in the work of Mertens, Zamit or Heifetz. In 2004 finally Moss and Viglizzo showed how to construct the type space of the hierarchical expectations as final coalgebras. This close connection of categorical methods with traditional modelling problems in economic game theory is an example where recent research begins to consider how category theory can help to construct the complicated informational or epistemic structures about who knows and beliefs what in epistemic game theory.

3.8 Some categorical approaches to concurrency theory

Ugo Montanari (University of Pisa, IT)

Concurrency can be studied from two different viewpoints: operational and abstract. In the former sense, concurrency defines equivalence classes of computations such that concurrent events can be executed in any order. Abstract concurrency observes causality dependencies between events and consequently defines certain notions of bisimilarity. Category theory is useful in both cases, offering natural notions of: (i) deterministic and non-deterministic concurrent computations; and (ii) coalgebraic transition systems labelled with event dependencies. About (i) we summarize the theory of P/T nets and pre-nets as developed by Meseguer, Sassone, Baldan, Bruni and the author. About (ii) we consider the notion of causal tree by Darondeau and Degano as the prototypical definition of causality, and derive from it a coalgebra in a presheaf category whose index category contains partial orders on events and monotone mappings. Applying a result by Ciancia, Kurz and the author, it is then possible to collapse such a coalgebra on a much smaller, often finite one, corresponding to the notion of causal automaton by Pistore, developed long time ago, with some ingenuity.

3.9 The next 700 logics

Till Mossakowski (University of Magdeburg, DE)

In this talk we analyse and clarify the method to establish and clarify the scope of logic theorems offered within the theory of institutions The method presented pervades a lot of abstract model theoretic developments carried out within institution theory. The power of the proposed general method is illustrated with the examples of (Craig) interpolation and (Beth) definability, as they appear in the literature of institutional model theory. Both case studies illustrate a considerable extension of the original scopes of the two classical theorems. Our presentation is rather narrative with the relevant logic and institution theory concepts introduced and explained gradually to the non-expert audience.

3.10 Abstract nonsense about gaming

Dusko Pavlovic (University of Hawaii – Manoa, US)

Game theory has largely been developed as the theory of equilibria that emerge from interactions between rational players. Since equilibria invariably arise as fixed points of diverse dynamics, which are conveniently presented as the morphisms in various categories, it is not surprising that the categorical presentations of a great variety of gaming processes turn out to be natural, and even fairly succinct. E. g., Moss and Viglizzo showed in CMCS 2004 how to present Harsanyi's type spaces as final coalgebras, and I proposed in CALCO 2009 (also http://arxiv.org/abs/0905.3548) a framework for deriving and analyzing categorically a broad range of gaming equilibria, old and new. These treatments, however did not lead to any new results in either game theory or category theory, but perhaps pointed to a convenient language. In this workshop talk, I discuss a dark alley of game theory where the categorical presentation seems to shine some light. The dark alley is the notion of bounded rationality, as gaming where the players are logically and computationally limited, so that they may fail to reach the optimizations that are too hard to construct. This is not only a very natural concept, going back to Herbert Simon's work from the late 1950s, but also a tremendously important practical problem, fully in the focus of experimental and behavioral game theory. Yet the underlying theory seems surprisingly thin. The main technical tool of the theory of bounded rationality is the view of strategies as state machines, and the technical results are in the tune of "this state machine strategy is the best response in the populations that consist of these 6 (or so) state machine strategies". The more desirable proofs that a state machine strategy is the best response against *all possible* state machine strategies generally remain beyond reach, because quantifying over all state machines is hard. And if we cannot calculate the best response strategies, then it is of course even harder to calculate any equilibria. A natural way beyond this obstacle is to look at strategies as Turing machines, or computable functions of some other form. This seems much better, because there are universal Turing machines, and they make quantifying over all Turing machines into an effective operation. Alas, it is fairly easy to construct games such that the best response to computable strategies is not computable. The point of my talk was to present a categorical model that seems to show the way out of this impasse. It captures the games in which the players do not just optimize following their respective utilities, but can also use deceit to outsmart the opponent, thus leading to *learning equilibria*. Some of the presented ideas originated in conversations with Viktor Winschel during his visits to Egham and Honolulu.

3.11 Equational Logic for PROPs

Gordon Plotkin (University of Edinburgh, GB)

We would like to program with various forms of graphs. For example, they occur in rule-based modelling in biology, representing individual proteins or populations of them. The various forms of dags used include: Milner's place graphs, link graphs and bigraphs, and Danos *et alii*'s kappa graphs. We would particularly like a general algebraic framework including textual notation. We could then consider equational technology: matching, rewriting, etc, whether qualitative or quantitative. In this talk we begin such a project by seeking an equational logic for dags. From a categorical point of view these form certain symmetric monoidal theories (aka PROPS), and so we approach the problem by analogy with standard equational logic and Lawvere theories. As a warm-up we consider linear equational logic and operads, which are intermediate between Lawvere theories and PROPs.

3.12 Categories in quantum theory

Peter Selinger (Dalhousie University – Halifax, CA)

I started by reviewing the basic postulates of quantum mechanics, and what they mean in the context of quantum computing. Then I reviewed the graphical languages of symmetric monoidal categories and of compact closed categories. The central concept of categorical quantum mechanics is that of a dagger compact closed category, first defined by Abramsky and Coecke. Many properties of finite-dimensional Hilbert spaces (i. e., complex linear algebra, or finite-dimensional quantum mechanics) can be axiomatised within arbitrary dagger compact closed categories. Finally, I reviewed completeness theorems and completely positive maps, which unify the concepts of unitary transformations and measurements.

3.13 Brzozowski's Algorithm (Co)Algebraically

Alexandra Silva (Radboud University Nijmegen, NL)

We show a proof of correctness of Brzozowski's minimization algorithm using a combined algebraic and coalgebraic view on automata. We discuss how this is an example of a potentially wider application of categorical methods to algorithms.

3.14 A topos of probabilistic concepts

Alex Simpson (University of Edinburgh, GB)

In *A review of probability theory*, Terence Tao stresses that 'probabilistic concepts' are distinguished by being preserved under 'extensions' to the sample space. Such extensions allow one, for example, to freely introduce new independent random variables. The general principle can be reformulated from a category-theoretic perspective: 'probabilistic concepts' organise themselves into *presheaves* over the category of sample spaces and extensions.

In this talk, we take the category of measure-preserving maps (identified mod 0) between standard probability spaces as the category of sample spaces. We observe that presheaves of probabilistic interest (e. g., presheaves corresponding to individual probability spaces, presheaves of random elements) form *sheaves* for the atomic Grothendieck topology. Thus we are led to the stronger idea of 'probabilistic concepts' as sheaves. Various constructions on (pre)sheaves are discussed, including a notion of independent product. The main result is a characterisation of the sheaf property in terms of conditional independence.

As future work, we hope to develop the idea that the resulting boolean sheaf topos offers a mathematical universe combining set-theoretic and probabilistic primitives, somewhat in the spirit of the universe envisaged in David Mumford's *The dawning of the age of stochasticity*.

4 Working Groups

4.1 Network theory

John C. Baez (University of California – Riverside, US)

Besides the author of the abstract, the participants included Fabio Gadducci, Barbara König, Ugo Montanari, Pawel Sobociński, and Jamie Vicary.

In the working group sessions on Network Theory, we discussed a general framework for networks with inputs and outputs, such as electrical circuit diagrams, signal flow diagrams, and the like. The basic idea is to consider a category where objects are labelled finite sets and morphisms are labelled graphs having designated sets of "input" and "output" vertexes. In fact, this kind of category should extend to a bi-category where the 2-morphisms are graph rewrites. Furthermore, this bi-category should be symmetric monoidal, allowing us to compose networks "in parallel" as well as in series.

We investigated how to prove these claims. Jamie noted that Michael Shulman's idea for constructing symmetric monoidal bi-categories should be useful, at least in the simpler case where the 2-morphisms are all isomorphisms. We explored that and found no obstacle. For bi-categories where the 2-morphisms are graph rewrites, Andrea and Fabio's work may prove useful. Pawel pointed out that the technique of double pushout rewriting, used to study graph rewriting, makes use of a concept that has a nice 2-categorical formulation: namely, adhesive categories.

Wrap up. In short, researchers from rather different areas met and discovered a body of common techniques, which could be used to solve some open problems.

4.2 Game theory

Viktor Winschel and Philipp Zahn (University of Mannheim, DE)

Besides the authors of the abstract, the participants included Robin Cockett, Martin Meier, Jeffrey C. Morton, Dusko Pavlovic, and Peter Selinger.

Viktor notes that reflexivity in economics and computer science is at the core of both sciences. In computer science this is due to the need to discuss code as data or the universality of Turing machines. In economics it arises because the modeler of the society is in the society hence the model and the modeled interact. There are many existing modeling issues that are ultimately in need of a reflexive treatment. An example is the learning equilibrium he defined with Dusko where they use the existence of the universal Turing machine to define a learning equilibrium in a game theoretical setting where strategies, the learning mechanism, process strategies, the actual ones for choosing the moves of a player in a game.

Dusko asserts that the minority game is an interesting example for applying the concept of a learning equilibrium because traditional analysis is not really helpful.

After a brief discussion of mixed strategy equilibria and their interpretation, Martin remarks that there are alternatives to viewing players as really randomising. One prominent example being “purification” by Harsanyi.

Viktor emphasizes that there have been already several projects in between economics and computer science where reflexivity has been addressed. He has been involved in one with Samson Abramksy that uses coalgebras to define infinite games and coinduction to proof equilibria. In another with Achim Blumensath coalgebras are used to compose games from simpler units and which allows a reflexive feature of game being played in networks about the very structure of the network. he mentions an ongoing project with Paulo Oliva and Julian Hedges from computer science where higher order functions are used to define very general games. It allows to define a reflexive structure of players in a game solving the game in order to coordinate or differentiate in a very general sense. Currently the project encompasses a game simulation, equilibrium checking and solving engine for this general kind of games.

Dusko asks what is the result of Escardo/Oliva that is used.

Philipp replies that for the first paper there is no result being used but a mere clarification of concepts. The crucial elements are quantifiers and selection functions, both higher order functions. The first result is that one can represent standard game utility functions as a special case. As an example one can consider an agent choosing between different petrol stations looking for the cheapest fuel. Here the environment is the cost functions linking petrol station and cost of fuel. The quantifier is then a function mapping from this cost function into the minimal price whereas the selection function is mapping from the cost function into the optimal choice, i. e. the cheapest petrol station. Besides being able to instantiate the standard approach one can go much beyond. In particular, one can have outcomes with less structure but even more importantly one can consider goals of agents that comprising the whole environment and not only the outcomes, respectively the image of the outcome function. One example is a beauty contest where one agent is only interested in voting for the winner and not for one particular candidate. An example is a game with three judges and two candidates, judge three being an agent only motivated by voting for the winner of the voting contest. This agent can be represented by a fixed point quantifier, respectively selection function.

Viktor remarks that in some sense this is capturing context-dependency.

Jeffrey wonders how dynamic games fit into that framework, and how is the standard framework able to capture reflexivity?

Dusko replies that equilibrium modelling is one thing but the dynamics beyond are important and not addressed so far – von Neumann himself was opposing the Nash program.

To this effect, Jeffrey mentions a game, Nomic, introduced by Peter Suber, a lawyer, aiming at creating policy rules that are implementing good policies where the issue of constitutional rules versus other rules become important.

Dusko further affirms that in some sense the learning equilibrium is a step in this direction because there learning is a change of strategy. But moreover, there can be dynamics beyond the point where everything is learnt.

Philipp notes that in "rock, scissors, paper" there are empirical frequencies different from the standard result of equally randomising between the three different moves.

Jeffrey remarks that an interesting personal experience of playing Nomic was that loopholes turned out to become important and most often policies ended up being under attack by people successfully finding loopholes.

Robin asks where category theory occurs, i. e., how a categorical game theory looks like.

Philipp states that to some degree there is composition already in the selection function approach because a game theoretic setting is reduced to a decision problem at all stages.

Along this line, Martin states that you can do the same with standard game theory by looking at pathological games of one person.

Jeffrey wonders how can there be reflexivity in normal games.

And Peter further asks if there is a use of categories in economics besides game theory.

Viktor replies that in general category theory should be very suited also to macro economics as a framework for synthesis and system analysis. For example there might be a use of sheaf theory as a general approach for the transition from the local to the global level, hence macroeconomics. Here reflexivity arises explicitly as the Lucas critique by the need to consider that agents take the theory and the model of the politician into account, which the politician need to take into account as well. Or a so to say vertical reflection where the macro aggregates like interest rate depend on the discussion of the agents who in turn decide based on the interest rate. Similarly the general issue of institutional dynamics is about the context being depended on but also ruling the content.

Dusko notes that category theory in quantum physics was not "demanded" by the experimental side, it clearly came from the theoretical side.

As before, Martin asks if in physics peoples use categories outside of quantum physics.

Peter states that it is most prominent is quantum physics but there is also use outside of it. And people such as John Baez promoting the view that, for instance, higher order categories should play a more important role.

Martin notes that in physics in a situation with many particles it possible to focus on the nearest particles; those farther apart can be neglected. This is hardly the case in economics.

Peter remarks that in markets everything seems to be related to everything.

Dusko recalls that there is a branch of research that wants to use concepts from quantum physics to apply them to markets, in quantum economics and econophysics.

4.3 Semantics

Alexandra Silva (Radboud University Nijmegen, NL)

Besides the author of the abstract, the participants included Samson Abramsky, Andrea Corradini, Bartek Klin, Francois Metayer, Andy Pitts, Alex Simpson, Peter Selinger, and Sam Staton.

Francois asks if there are many parts of CT that are actually applied in CS, and wonders if one can enumerate the main areas that are applied. Andy thinks it is too wide a subject and that there is no such thing as the CT that is applied in CS. However, Samson points that the general approach is the same even if different sub-areas of CT are applied to different areas of CS/Math. Samson mentions the work by Baez on Higher Categories in Math and mentions that the fact that these are so hard to work with reduces the impact ... It is noted that homotopy type theory is attracting a lot of interest. A new trend seems to be to make models intensional, both in type theory and concurrency.

Peter asks if CT in semantics is on the rise. Alexandra replies that in programming languages there is more and more of a trend to use CT as a base for designing. Other examples mentioned by people are homotopy type theory, an area where people apply CT unapologetically, and theorem proving systems, which are becoming more and more usable and thus more widespread. and there CT will survive and thrive. It is noted that automated theorem proving is a good way to bring CT to the mainstream.

It is noted that coalgebra came from the CT community but has a life of its own ... they found interesting applications in automata theory. There seems however to be a lot of inbreeding, while on the contrary the communication to other communities is an important point. Learning to talk with different communities is an import an point in the quest of finding open problems in other areas where CT can make a contribution. Coinduction in theorem provers is still an open problem and though the communication issue with the theorem proving community is not such a big issue one still does not see enough cross-fertilization. Peter mentions that communication problems are also present in quantum computation though there one already sees contributions. He also argues that a weakness of semantics/CT is that the definitions play a key role. Having the right definitions makes the theorems trivial, which is the opposite of hard subjects where they have combinatorial proofs of theorems (and simple definitions). CT is very organized, good for compiler design with complicated inter-dependencies (and not so complicated algorithmic problems). In general, the audience agrees that people sees category theorists only as reconstructing the things they knew already, and that is a disadvantage, because we do not give them a good reason to care enough.

Andy wonders about linguistics. Samson tells that there are successful stories there. Word semantics based on a vector representation of words. This is used in information retrieval. The nice observation is that using tensor product you can compose information. Goes back to Lambek grammars. This is the subject of Bob Coecke's talk.

The audience consider crucial the two following questions: How will we promote CT? And should we be training pure CT? Also relevant is to understand the need of a venue, i. e., of a focused forum. There is of course a CT conference. but is this enough? Bartek proposes a forum like Highlights: 8 min talks for around 100 talks, with no claim of originality. As a general remark, it is true that there are several areas where CT is a commonly used tool: quantum, type theory, coalgebra, ... Each of these areas has its own workshop, but this may not be enough for further spreading the use of CT.

A few more areas are identified where it would be good to have an impact: security, algorithms & complexity, machine learning. Andy asks about the NSA take on category theory and Peter mentions they are actually interested in formal methods. Concerning security: proving the safety of a property means imagining all possible ways that an attacker can use. Could the attacker's behavior be abstracted by means of CT tools? How about security compositionally? Related to this, and a main conference such as ICALP, Samson mentions that going into track A might give fruits. In his talk he had an example related to oracles and this seems to be a good way to bring track A subjects to track B.

Wrap up. Category theory started as re-writing arguments of other areas. Now we have a very powerful toolkit. Compositionality is one of the strengths. Applied CT can use the toolkit in other areas, and in its spread computational tools will play a key role.

The book of Spivac is an interesting result of someone who came into CT from the outside. There is still need to have books/tutorials on how to use basic category theory.

Having a conference or a training network could create the synergy to produce the tutorials needed for a widespread adoption of CT.

Participants

- Samson Abramsky
University of Oxford, GB
- John C. Baez
University of California – Riverside, US
- Robin Cockett
University of Calgary, CA
- Bob Coecke
University of Oxford, GB
- Andrea Corradini
University of Pisa, IT
- Andrée Ehresmann
University of Amiens, FR
- Fabio Gadducci
University of Pisa, IT
- Ulrike Golas
Zuse Institut Berlin, DE
- Michael Johnson
Macquarie Univ. – Sydney, AU
- Bartek Klin
University of Warsaw, PL
- Barbara König
Universität Duisburg-Essen, DE
- Tom S. Maibaum
McMaster Univ. – Hamilton, CA
- Martin Meier
IHS – Wien, AT
- Francois Metayer
University Paris-Diderot, FR
- Ugo Montanari
University of Pisa, IT
- Jeffrey C. Morton
Universität Hamburg, DE
- Till Mossakowski
Universität Magdeburg, DE
- Fernando Orejas
UPC – Barcelona, ES
- Dusko Pavlovic
Univ. of Hawaii – Manoa, US
- Andrew M. Pitts
University of Cambridge, GB
- Gordon Plotkin
University of Edinburgh, GB
- Peter Selinger
Dalhousie Univ. – Halifax, CA
- Alexandra Silva
Radboud Univ. Nijmegen, NL
- Alex Simpson
University of Edinburgh, GB
- Pawel Sobocinski
University of Southampton, GB
- Sam Staton
Radboud Univ. Nijmegen, NL
- Jamie Vicary
University of Oxford, GB
- Viktor Winschel
Universität Mannheim, DE
- Philipp Zahn
Universität Mannheim, DE