cost
EUROPEAN COOPERATION IN SCIENCE AND TECHNOLOGY

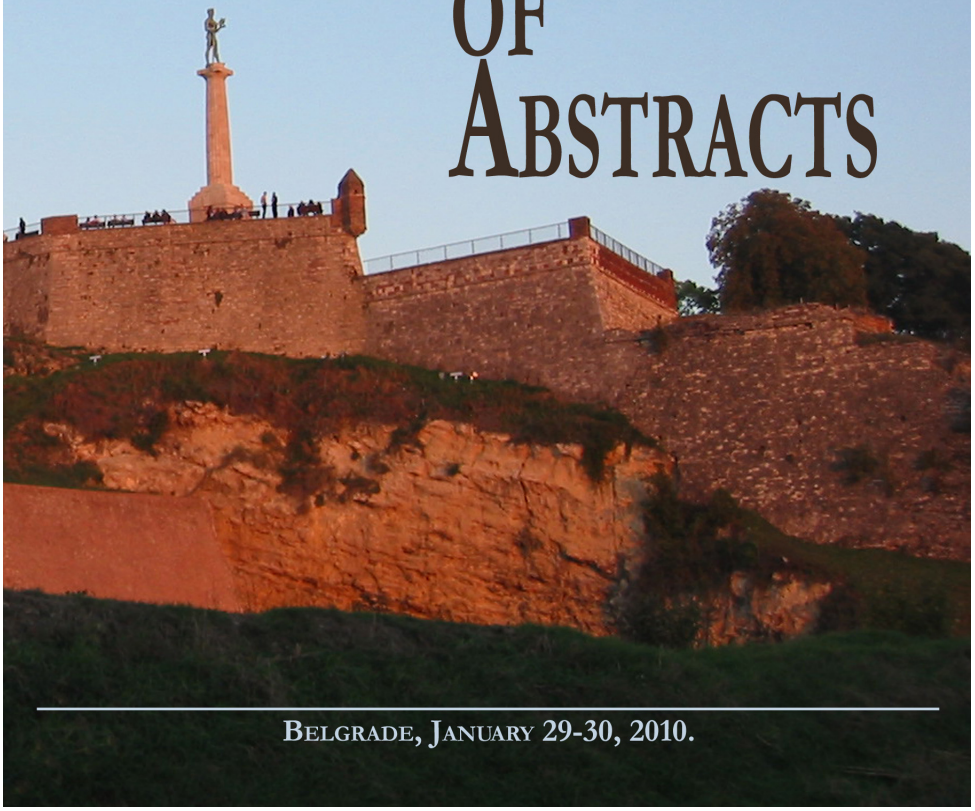
University of Belgrade

Faculty of Mathematics

Automated Reasoning GrOup

COST IC0901 - Rich Model Toolkit
WG1 and WG2 Meeting and
Third Workshop on Formal and Automated
Theorem Proving and Applications

Book of Abstracts



Belgrade, January 29-30, 2010.

COST Action IC0901 WG1 and WG2 Meeting
and
Third Workshop on Formal and Automated
Theorem Proving and Applications

http://argo.matf.bg.ac.rs/events/2010/fatpa2010.html

Book of Abstracts

January 29-30, 2010, Belgrade, Serbia

COST Action IC0901 Chair:
Viktor Kunčak (EPFL, Lausanne, Switzerland)

COST Action IC0901 Working Group 1 (Rich Model Language) Chair:
Tobias Nipkow (TU Munich, Germany)

COST Action IC0901 Working Group 1 (Rich Model Language) Deputy Chair:
Paul Jackson (University of Edinburgh, United Kingdom)

COST Action IC0901 Working Group 2 (Decision Procedures) Chair:
Maria Paola Bonacina (University of Verona, Italy)

COST Action IC0901 Working Group 2 (Decision Procedures) Deputy Chair:
Armin Biere (Johannes Kepler University, Linz, Austria)

FATPA Workshop Chair:
Predrag Janičić (University of Belgrade, Serbia)

Preface

This booklet contains abstracts of talks given at the joint meeting of:

COST Action IC0901 WG1 and WG2 Meeting
and
Third Workshop on Formal and Automated Theorem Proving and Applications

held at the University of Belgrade on January 29-30, 2010. The meeting was attended by 35 participants coming from 16 institutions from 11 countries (Austria (1), Czech Republic (1), Denmark (1), Finland (2), France (3), Germany (2), Italy (1), Norway (1), Serbia (17), Switzerland (3), United Kingdom (3)).

Given the close connection between the scope of two events, we scheduled a single, joint programme. The programme consisted of 19 presentations, illustrating state-of-the-art research performed in a number of European academic and industrial centres. The talks were (sometimes rather loosely) divided into the five categories: SAT solving, SMT solving, formal and automated theorem proving, applications of theorem proving, and logical foundations.

More details about the meeting can be found online: `http://argo.matf.bg.ac.rs/events/2010/fatpa2010.html`

For the success of the meeting, we are grateful to the invited speaker Ralph-Johan Back, all other speakers and all participants. We are also grateful to the COST organization for support within the COST Action IC0901 (`http://richmodels.org`) and to the Faculty of Mathematics, University of Belgrade which was the host institution of the meeting. We thank Ružica Piskač, Philippe Suter, and Milan Banković for making notes on discussions after the talks, and Filip Marić and Philipp Rümmer for providing photographs used in this booklet.

Viktor Kunčak,
Assistant Professor at EPFL, Switzerland,
Chair of Management Committee of COST Action IC0901
and
Predrag Janičić,
Associate professor at the Faculty of Mathematics,
University of Belgrade, Serbia
FATPA Workshop Chair and
Management Committee Member for COST Action IC0901

Participants

- Ralph-Johan Back (Åbo Akademi University, Turku, Finland)
  `http://users.abo.fi/backrj`
- Milan Banković (University of Belgrade, Serbia)
  `http://www.matf.bg.ac.rs/~milan`
- Marc Bezem (University of Bergen, Norway) [COST Action MC Member]
  `http://www.ii.uib.no/~bezem`
- Sascha Böhme (TU Munich, Germany) [COST Action WG Member]
  `http://www4.in.tum.de/~boehmes`
- Johannes Eriksson (Åbo Akademi University, Turku, Finland)
  `http://users.abo.fi/johannes.eriksson`
- Silvia Ghilezan (University of Novi Sad, Serbia) [COST Action MC Member]
  `http://imft.ftn.ns.ac.yu/~silvia`

- Florian Haftmann (TU Munich, Germany) [COST Action WG Member]
  http://www4.informatik.tu-muenchen.de/~haftmann
- Hugo Herbelin (INRIA — PPS, Paris, France)
  http://pauillac.inria.fr/~herbelin/index-eng.html
- Jelena Ivetić (University of Novi Sad, Serbia)
  http://imft.ftn.uns.ac.rs/~jelena/
- Paul Jackson (University of Edinburgh, United Kingdom) [COST Action MC Member, WG1 Deputy Chair]
  http://homepages.inf.ed.ac.uk/pbj
- Svetlana Jakšić (University of Novi Sad, Serbia)
  http://imft.ftn.uns.ac.rs/~svetlana/
- Predrag Janičić (University of Belgrade, Serbia) [COST Action MC Member]
  http://www.matf.bg.ac.rs/~janicic
- Barbara Jobstmann (CNRS/Verimag, Gieres, France) [COST Action MC Member]
  http://www-verimag.imag.fr/~jobstman
- Moa Johansson (University of Verona, Italy) [COST Action WG Member]
  http://homepages.inf.ed.ac.uk/s0199173
- Oliver Kullmann (Swansea University, United Kingdom)
  http://www.cs.swan.ac.uk/~csoliver/
- Viktor Kunčak (EPFL, Lausanne, Switzerland) [COST Action Chair]
  http://lara.epfl.ch/~kuncak
- Petar Maksimović (Mathematical Institute, Belgrade, Serbia)
- Filip Marić (University of Belgrade, Serbia)
  http://www.matf.bg.ac.rs/~filip
- Bojan Marinković (Mathematical Institute, Belgrade, Serbia)
  http://www.mi.sanu.ac.rs/~bojanm/
- Walther Neuper (Graz University of Technology, Austria)
  http://www.ist.tugraz.at/neuper
- Mladen Nikolić (University of Belgrade, Serbia)
  http://www.matf.bg.ac.rs/~nikolic
- Jovanka Pantović (University of Novi Sad, Serbia)
  http://imft.ftn.uns.ac.rs/~vanja/
- Vesna Pavlović (University of Belgrade, Serbia)
  http://www.matf.bg.ac.rs/~vesnap

- Danijela Petrović (University of Belgrade, Serbia)
  http://www.matf.bg.ac.yu/~danijela/
- Ružica Piskač (EPFL, Lausanne, Switzerland)
  http://icwww.epfl.ch/~piskac
- Miroslav Popović (University of Novi Sad, Serbia)
- Stefan Ratschan (Academy of Sciences, Prague, Czech Republic) [COST Action MC Member]
  http://www2.cs.cas.cz/~ratschan
- Alexis Saurin (INRIA — PPS, Paris, France)
  http://www.pps.jussieu.fr/~saurin
- Peter Schneider-Kamp (University of Southern Denmark, Odense, Denmark) [COST Action MC Member Substitute]
  http://www.imada.sdu.dk/~petersk
- Mirko Stojadinović(University of Belgrade, Serbia)
  http://www.matf.bg.ac.rs/~mirkos
- Sana Stojanović (University of Belgrade, Serbia)
  http://www.matf.bg.ac.rs/~sana
- Philippe Suter (EPFL, Lausanne, Switzerland)
  http://lara.epfl.ch/~psuter
- Philipp Rümmer (University of Oxford, United Kingdom) [COST Action MC Member Substitute]
  http://web.comlab.ox.ac.uk/people/Philipp.Ruemmer
- Milena Vujošević-Janičić (University of Belgrade, Serbia)
  http://www.matf.bg.ac.rs/~milena
- Dragiša Žunić (University of Novi Sad, Serbia)

Contents

Opening Remarks

Predrag Janičić
University of Belgrade, Serbia

Invited Lecture

1 Invariant Based Programming

Ralph-Johan Back
Åbo Akademi University, Turku, Finland

Abstract

Invariant based programming is an approach to constructing programs where the basic program situations (pre- and postconditions and intermediate (loop) invariants) are identified and formalized before we connect these situations with executable transitions. The program is built in a stepwise manner, each step adding (or deleting or changing) either a situation or a transition between situations. We continuously check that each step preserves the correctness of the

program built this far. Invariant based programming is now taught in introductory programming courses in our university, and seems to provide a simple and intuitive method for introducing formal methods and correctness concerns early on in the programming education. Invariant based programming is carried out within a visual formalism that we call invariant diagrams. All the information needed for checking program correctness is directly displayed in the diagram. This means that correctness can be checked directly by analysing the diagram, without the need for learning complicated programming logic proof rules.

The talk will describe the basic idea of invariant based programming and exemplify the approach with a simple example. We will identify the proof obligations that together establish the correctness of the constructed program, and explain the central role that automatic verification of these proof conditions (using SMT solvers) plays. We describe the basic workflow of invariant based programming, and our experiences from teaching this approach to both experienced programmers and to first year CS students.

Slides: http://argo.matf.bg.ac.rs/events/2010/slides/Back_InvariantBasedProgramming.pdf

Discussion

Predrag Janičić: What are your first experiences with using invariant based programming in developing real-world applications?
Re: For instance, we use it for implementing complex geometrical algorithms, like finding paths between islands in an archipelago. What proves to be the most difficult part is building an appropriate domain-specific theory, not other things.

There were additional discussions and questions by Paul Jackson and Oliver Kullman.

Towards Goals of the COST Action IC0901

2 Towards a Rich Model Toolkit

Viktor Kunčak
EPFL, Lausanne, Switzerland

Abstract

I will outline a rationale for Rich Model Language (RML) as a unifying notation for specifying computer systems. I argue that classical higher order logic (such as an Isabelle/HOL fragment) appears a good candidate for RML

and describe our experience with using it within our Jahob verification system. I will describe our current progress in developing a new reasoning system (written in the Scala) that uses a similar input language. I illustrate how the higher order logic view makes it easy to explain a combination result for expressive classes of formulas. Finally, I point out to design questions in using RML to describe verification and synthesis problems in addition to the validity of typical fragments of logic.

Slides: http://argo.matf.bg.ac.rs/events/2010/slides/Kuncak_TowardsARichModelToolkit.pdf

Discussion

Paul Jackson: What aspects of HOL did you need in Jahob? Did you need parameterized types?
Re: Parameterized types were useful in constructs such as if-then-else and function updates. By the time we generate e.g. SMT-LIB proof obligations, all types are ground.

There was an additional discussion and a question about synthesis by Ralph-Johan Back.

SAT Solving

Session Chair: Peter Schneider-Kamp
University of Southern Denmark, Odense, Denmark

3 The OKlibrary, an Open-source Research Platform for SAT Solving (and Beyond)

Oliver Kullmann
Swansea University, United Kingdom

Abstract

At around the time of this workshop, the pre-alpha release of the OKlibrary (`http://www.ok-sat-library.org/`) will (hopefully) be released. I want to give some overview on the basic ideas of this "research platform", and what functionality is already available.

Slides: http://argo.matf.bg.ac.rs/events/2010/slides/Kullmann_TheOKlibraryAnOpenSourceResearchPlatformForSATSolving.pdf

Discussion

Filip Marić: Are you using a package management system?
Re: Packages are not available in a Linux distribution. Due to the complexity of the packages, they wrote their own small management system. It's currently hard-coded, and they recognize the necessity of such a solution, but the problem is that they would need something very specialized, so they cannot use existing solutions.

4 A Methodology for Comparison and Ranking of SAT Solvers

Mladen Nikolić
University of Belgrade, Serbia

Abstract

Weighing improvements to modern SAT solvers and comparison of two or more arbitrary solvers is a challenging, but important task. Relative performance of solvers is usually assessed by running them on some set of SAT instances and comparing the number of solved instances and their running time in some simple manner. In this paper we point to shortcomings of this approach and advocate a more rigorous, statistically founded methodology. We present such methodology and results of its application.

Slides: http://argo.matf.bg.ac.rs/events/2010/slides/Nikolic_AMethodologyForComparisonAndRankingOfSATSolvers.pdf

Discussion

Oliver Kullmann: About the work of Brglez: it was "rejected" by the community on the basis that solvers sometimes want to exploit the ordering, so shuffling was really creating different problems.

Re: The purpose of shuffling is to force the solver to explore different parts of the search-space. Another way to do it is to use different seeds. Shuffling is one way to average out.

?: About the benchmarks: the industrial set was more affected by the shuffling. Any idea why?
Re: Shuffling does not necessarily degenerate performance. In the original form, the number of solved instances was around the middle (between min and max). Crafted benchmarks of one family, are usually characterized by some parameter (e.g., the number of pigeons in pigeonhole problem). Complexity of solving can be exponential with respect to that parameter, so for parameter values large enough, formulae are too hard regardless of shuffling. On industrial examples, shuffling can bring instances below or above the threshold if there is a larger number of instances that is somewhere around the threshold.

Barbara Jobstmann: ... to pick up on that: so the number of solved instances did not really change? I.e., on the whole number of instances, does the "best solver" always solve the same amount?
Re: No (cf. the numbers).

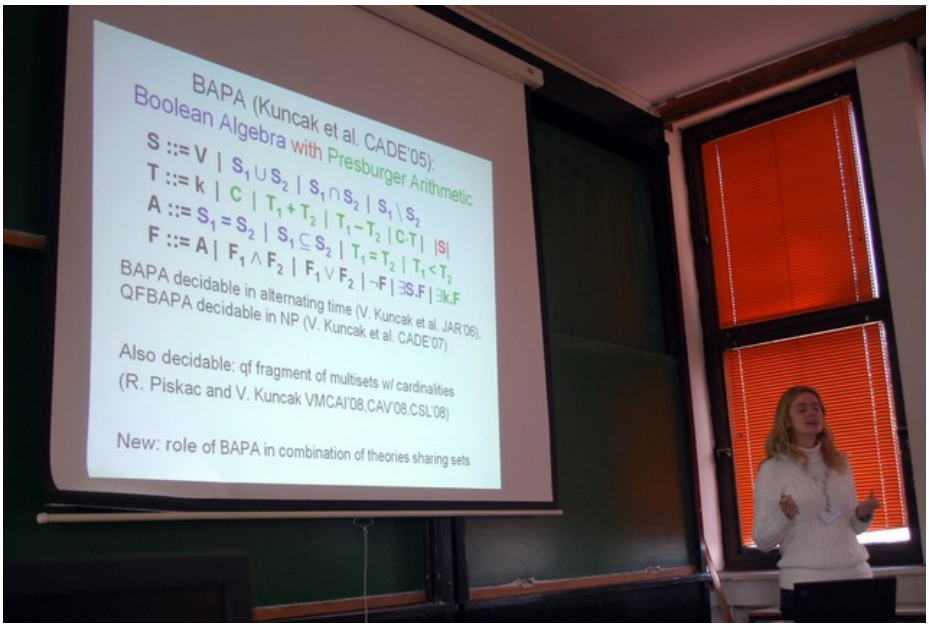
Philippe Suter: Was preprocessing used?
Re: Yes, the industrial benchmarks were already preprocessed when included in the corpus for the competition.

?: Your tests were made using variants of only one solver (MiniSAT). Have you made tests using some other solvers?
Re: Not yet, but the methodology is applicable to all solvers.

SMT Solving

Session Chair: Paul Jackson
University of Edinburgh, United Kingdom

5 Combining Theories with Shared Set Operations



Ružica Piskač
A joint work with Thomas Wies and Viktor Kuncak
EPFL, Lausanne, Switzerland

Abstract

We explore automated reasoning techniques for the non-disjoint combination of theories that share set variables and operations. We are motivated by applications to software analysis, where verification conditions are often expressed in a combination of such theories. The standard Nelson-Oppen result on combining theories does not apply in this setting, because the theories share more than just equalities. We state and prove a new combination theorem and use it to show the decidability of the satisfiability problem for a class of formulas obtained by applying propositional connectives to formulas belonging to:

1. Boolean algebra with Presburger arithmetic (with quantifiers over sets and integers);
2. weak monadic second-order logic over trees (with first and second order quantifiers);
3. two-variable logic with counting quantifiers;

4. the Bernays-Schönfinkel-Ramsey class of first-order logic with equality (with $\forall^*\exists^*$ quantifier prefix);

5. the quantifier-free logic of multisets with cardinality constraint.

We illustrate our result through verification conditions expressing properties of data structures.

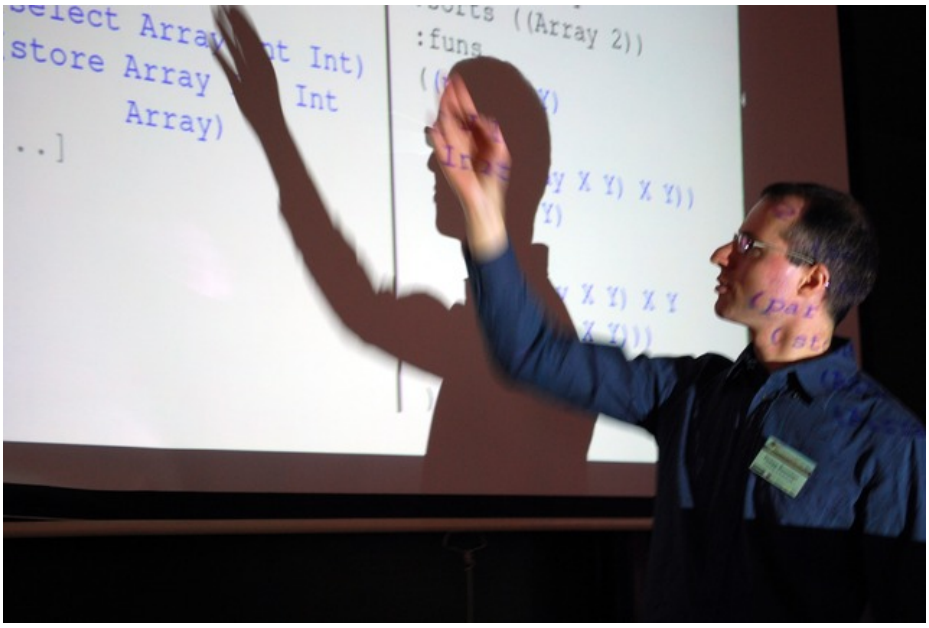
Slides: http://argo.matf.bg.ac.rs/events/2010/slides/Piskac_CombiningTheoriesWithSharedSetOperations.pdf

Discussion

Paul Jackson: Was this implemented, or what's the status?
Re: It's in the process. We've had some experiences with projections of WS1S. Projections between multisets and BAPA can be easily implemented, we're working on it. Two variable logic with counting, we won't implement it, however, we have something close that we presented at VMCAI this year (adding functions to sets), and this we could implement.

---

6 The SMT-LIB 2 Standard: Overview and Proposed New Theories



Philipp Rümmer
Oxford University Computing Laboratory, United Kingdom

Abstract

SMT-LIB is a standardised format for interfacing SMT solvers and collecting verification conditions, as well as a library of verification benchmarks. Recently, version 2 of the SMT-LIB format has been proposed, which simplifies the format in various ways, besides providing a framework for parametric theory specifications (theories with parametric polymorphism) that makes SMT-LIB more flexible when working with many theories at the same time. In my talk, I will give an overview of SMT-LIB 2 and the changes compared to version 1.

In the second part of the talk, I will introduce several theories relevant for program verification that our group has proposed for inclusion in SMT-LIB 2: theories of (finite) sets, relations, lists, and maps, as well as a theory of floating point arithmetic following the IEEE 754-2008 standard.

Slides: http://argo.matf.bg.ac.rs/events/2010/slides/Ruemmer_TheSMT-LIB2StandardOverviewAndProposedNewTheories.pdf

Discussion

Philippe Suter: Is there any plan for a standard to represent models, when a formula is found to be satisfiable?
Re: In interactive mode, you should be able to query the solver with, e.g., ground terms, and get a valuation back.

Viktor Kunčak: What is the difference between arrays (which are not finite) and function symbols?
Re: One difference is that arrays are first-class objects, so you could quantify over them. In the quantifier-free logics, there's no real difference. *Paul Jackson:* You cannot really use an array for functions which return infinitely many different values.

Additional questions discussed:
Filip Marić: Is there any progress in standardizing an API for SMT solvers?

Paul Jackson: About IEEE floating point standard, what do you mean that it is ambiguous: so it seems that even people who are trained in writing standards can't come up with precise ones.

---

7 Uniform Reduction to SAT and SMT

Predrag Janičić
A joint work with Filip Marić
University of Belgrade, Serbia

Abstract

We will present a novel approach for specification and solving a wide class of problems, including CSP and verification problems. The approach uses an imperative-declarative specification language and problem specifications are transformed into propositional or first-order logic formulae and then solved by SAT or SMT solvers. We will also present our prototype implementation of the approach — a tool *URSA Major*, that employs a SAT solver ArgoSAT and several SMT solvers for bit-vector arithmetic, linear arithmetic, etc.

Slides: http://argo.matf.bg.ac.rs/events/2010/slides/JanicicMaric_UniformReductionToSATAndSMT.pdf

Discussion

Viktor Kunčak: You mention restrictions on the bounds of loops and if-then-else expressions. What does it mean for an expression to be symbolic?
Re: It means that it depends on the unknowns.

Viktor Kunčak: Is there a way to lift the bound on if-then-else expressions?
Re: Yes, it is possible, although would not be very elegant to implement. Instead we have conditional expressions with very elegant implementation.

Peter Schneider-Kamp: If you have an NP-complete problem, can you use this approach if conditions depend on unknowns?
Re: Yes. You can use conditional operators to solve NP complete problems, and it is simple to express. However you have at some point to pay for the cost, so the formulas will be complex.

Oliver Kullmann: Regarding the translation, consider a problem with many different possible encodings/translations, can the system accommodate this?
Re: The system lets you choose your own encoding, the user has full control over the encoding that will be used. Some encodings may take more time for making a specification, but the user essentially has the control.

Viktor Kunčak: For the 8-queens, what would happen if you added a break when the variable becomes false, would this help or not?
Re: You can't do it, because such a condition would depend on unknowns (symbolic values), so you couldn't know when to break.

---

8 Proving SPARK VCs with SMT solvers. Implications for the Rich Model Language

Paul Jackson
University of Edinburgh, United Kingdom

Abstract

SPARK is a subset of Ada used primarily in high-integrity systems in the aerospace, defence, rail and security industries. Formal verification of SPARK programs is supported by tools produced by the UK company Praxis. These tools include a verification condition generator and an automatic prover. The proof of verification conditions (VCs) ensures the correctness of program assertions and the absence of exceptions such as array indices out-of-bounds and divide by zeroes.

We report here on our experiences in using popular SMT solvers such as CVC3, Yices and Z3 to prove SPARK VCs. Our experiments cover the use of both solver-specific input languages and the SMT-LIB standard language.

We observe that significant work is required to handle the variety of types (e.g. array types, record types, integer subrange types, ordered enumeration types) found in SPARK VCs when translating these VCs into the SMT-LIB language. In this light, we discuss what types ought to be supported in the Rich Model Language, and also bring in reflections on our past experiences

with the rich type systems of interactive theorem provers such as Nuprl as PVS.

Slides: http://argo.matf.bg.ac.rs/events/2010/slides/Jackson_ProvingSPARKVCsWithSMTSolvers.pdf

Discussion

Viktor Kunčak: Question about the described translation. It seems to be important to pay attention to these reductions. I'm not aware of much published work documenting reductions from expressive languages to less expressive ones. I would be interested to see a precise description of what you're doing.
Re: There are examples in the submitted journal paper.

Viktor Kunčak: Within the Rich Models Toolkit initiative, such translations would clearly be interesting. In what programming language is this implemented?
Re: It's in C++.

Ralph-Johan Back: What is Praxis promising regarding their software/methods? Are they actually saying they're constructing error-proof programs.
Re: For legal reasons, they make absolutely no promises, but they describe their use of formal methods, justify how rigorous there are, etc.

Ralph-Johan Back: So it's important for them to audit the methods they're using. In particular they need to generate understandable proofs.

Viktor Kunčak: A comment, and merging into the general discussion: you seem to like type systems but users don't like to be forced to use them. In PL, there's the notion of soft typing. Do you think we can have something similar: ie. formulas have their meanings, you don't have to check their types but you get more guarantees/meaning if you do.
Re: Checking only the skeletons could be seen as some sort of soft typing.

Viktor Kunčak: Let's say I just do simple checking, not of the dependent types, for instance. Should I still expect the laws of FOL etc. to hold?
Re: I don't know, I'd say probably not.

Viktor Kunčak: Somehow, programs with soft types still execute even if you don't prove they won't be run-time errors. In Isabelle, for instance, they are ways to work around the lack of subranges, etc., so I wonder if there's a way to write such "incomplete" specifications, and have tools that can analyze these in more details, but we wouldn't have to.
Re: Indeed, we could envisage a different translation which would be less precise, but proving the result is not an absolute guarantee.

Rich Model Language: Panel Discussion

Panel Discussion Moderator: Viktor Kunčak
EPFL, Lausanne, Switzerland

Paul Jackson: Shouldn't we settle for a language that can easily be processed by tools, as opposed to by humans?
Re: I agree that we should separate core (abstract) from concrete syntax. We can then add a pretty human-readable syntax within tools. We could do a similar things for the type system: have simple types with parametric polymorphism, and add more sophisticated soft typing approaches on top.

Paul Jackson: I would argue for a language without parametric types (or at least that these types should not have to be inferred).

Re: One way to do this is to always force the introduction of a type with a variable (even if it is a type variable). I'd like to point you to the page: `http://richmodels.org/rmt` which we can use to write down ideas and try to make them converge. There are already some ideas up there.

Predrag Janičić: An organizational issue: do you have some ideas about the timeline of the language definition? How do you see the evolution of this language?
Re: I would suggest that for now the main goal would be to have, by SVARM in Edinburgh, a candidate RML format and, as illustration, tools that use RML-like input or output.

Formal and Automated Theorem Proving

Session Chair: Filip Marić
University of Belgrade, Serbia

9 Efficient Proof Reconstruction for the SMT Solver Z3

Sascha Böhme
TU Munich, Germany

Abstract

The Satisfiability Modulo Theories (SMT) solver Z3 can generate proofs of unsatisfiability. We present independent checking of these proofs in the theorem prover Isabelle/HOL, and in particular focus on the question of efficiency. Detailed performance data shows that LCF-style proof reconstruction can be as fast as proof search in Z3. Moreover, our implementations outperform previous work in this field, often by orders of magnitude.

Slides: http://argo.matf.bg.ac.rs/events/2010/slides/Boehme_EfficientProofReconstructionForTheSMTSolverZ3.pdf

Discussion

Filip Marić: Did you have some cooperation with Microsoft on this?
Re: We had quite a few email exchanged, mostly to understand what they were doing. I tried to have them improve parts of the proof reconstruction procedure, but it looks like it's not their priority.

Filip Marić: Are all the theories supported by Z3 working with proof reconstruction?
Re: Not as now, for instance non-linear arithmetic is not.

Filip Marić: There was an initiative in SMT community to standardize proofs, so does this work apply to other solvers than Z3?
Re: There were plans but they did not go very far, so currently all the formats are different.

Paul Jackson: Did you find any bug in Z3 solver?
Re: Yes, but all of them are bugs in building proof representations, not in the main part of the solver.

Viktor Kunčak: Is there an option in Z3 to switch off the proof generation and how much the efficiency is improved that way?
Re: Yes, you can switch off the proof generation and it improves the efficiency, but I don't have exact numbers at the moment.

---

10 Integrating Isabelle/HOL And Functional Programming – Current Trends

Florian Haftmann
TU Munich, Germany

Abstract

We demonstrate two recent developments concerning Isabelle/HOL and functional programming:

1. Haskabelle is a pragmatic translator from Haskell to Isabelle/HOL It allows reasoning about (a substantial subset of) Haskell programs by turning them into corresponding Isabelle/HOL specifications.
2. Inductively defined predicates are frequently used in formal specifications. Using the theorem prover Isabelle, we describe an approach to turn a class of systems of inductively defined predicates by proof into a system of equations using data flow analysis . Thus we extend the scope of code generation in Isabelle/HOL from functional to functional-logic programs while leaving the trusted foundations of code generation itself intact.

Slides: http://argo.matf.bg.ac.rs/events/2010/slides/Haftmann_IntegratingIsabelleHOLAndFunctionalProgramming-CurrentTrends.pdf

Discussion

Paul Jackson: Can you compare this with previous work, for eg. Coq approach?
Re: The main difference is that we constructively repeat the pencil proof, thus we get proper equational theorems in the sense of the LCF kernel. Besides that Coq cannot deal with multiple solutions.

Viktor Kunčak: If we compare this with approaches for logic programming languages, what are the main differences?
Re: The big difference to Prolog is that we have no unification, and I think it's quite close to Mercury. There are probably more efficient ways to implement some aspects of logic programming than what we're doing.

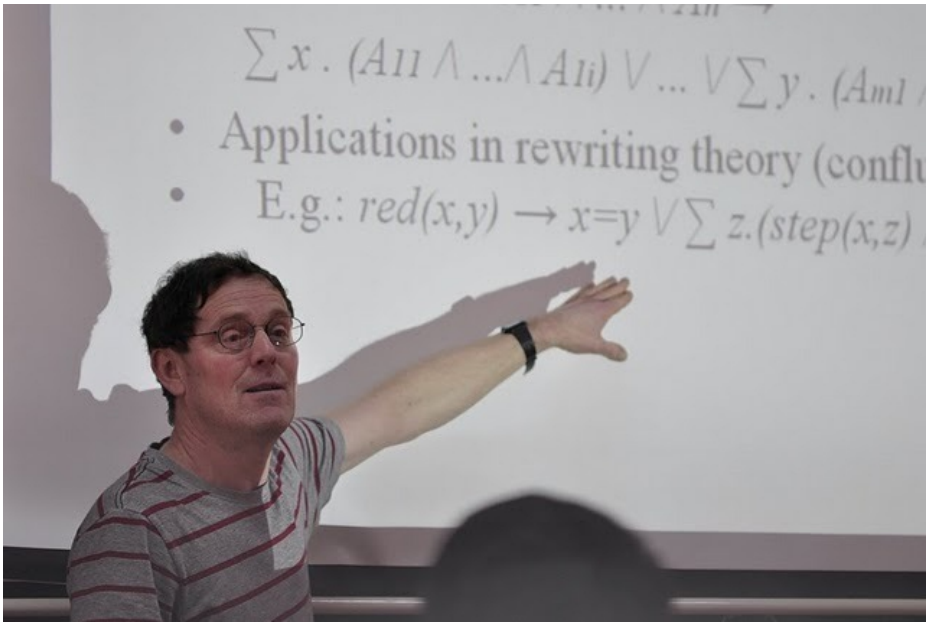
Viktor Kunčak: For example, when we were discussing executable fragments of RML, I think what you're doing is one way to expand these fragments. Do you have a subset of a functional language that you can translate back and forth from Haskell to Isabelle?
Re: Haskabelle does nothing else than a translation from text to text, which is not really semantic translation. It probably wouldn't be too hard to characterize the programs we support. We have a concise representation of what we can import from Haskell (as data types). This is a quite precise description of what we're doing.

Paul Jackson: Have you tried this out on large example? On some large inductive definitions?
Re: We have quite large examples, for instance semantics of a fragment of C++, so it really does scale.

Paul Jackson: So how fast does the interpreter run them?
Re: We don't have benchmarks for that.. it's functional but I can't really tell how fast or slow it is.

---

11 Automating Coherent Logic



Marc Bezem
University of Bergen, Norway

Abstract

Coherent Logic (CL) is a fragment of FOL which extends the clausal fragment used in resolution logic. We discuss the pros and cons of automated reasoning in CL and give an overview of the project ACL. The goal of ACL is to support automated reasoning in proof assistants (logical frameworks based on typed lamda calculus) such as Coq and Isabelle.

Slides: http://argo.matf.bg.ac.rs/events/2010/slides/Bezem_AutomatingCoherentLogic.pdf

Discussion

Predrag Janičić: If the conjecture is already coherent, you would just go and apply your rules, but could you try instead to try the prove the goal by negating it and refuting it, and couldn't it then be more efficient?
Re: I never tried that, but my natural reaction would be "don't touch it", and use what I presented, but I should say I have a bias towards working in this fragment.

Viktor Kunčak: Do you have function symbols in coherent logic, in general?
Re: Not yet, but we know how to eliminate them. We could certainly try to extend it with function symbols, the theory would remain complete, but the effect on the prover has to be assessed. It's a good point, because we could then compare the approaches of handling them "natively" and eliminating them.

Viktor Kunčak: Just a remark that is seems that this would work well for proof-based synthesis.

---

12 Conjecture Synthesis for Inductive Theory Formation

Moa Johansson
University of Verona, Italy

Abstract

For my PhD, I developed a system called IsaCoSy, which automatically performs synthesis of non-trivial conjectures in inductive theories. IsaCoSy takes an Isabelle theory as input, and produce conjectures of increasing size about the available datatypes and functions. The synthesis process is made tractable by only synthesising irreducible term, enforced by applying constraints generated from the currently known theorems. The conjectures are passed through Isabelle's counter- example checker, and then passed on to an automated inductive prover. In the future, we hope that conjecture synthesis techniques may assist users finding routine lemmas in novel theory developments, and perhaps have applications in loop-invariant generation.

I will also briefly mention some of my new research interests in Verona, within the domain of combining SMT solvers and F.O provers.

Slides: http://argo.matf.bg.ac.rs/events/2010/slides/Johansson_ConjectureSynthesisForInductiveTheoryFormation.pdf

Discussion

Viktor Kunčak: About the generation process; am I correct to assume that by construction this only generates type-correct terms?
Re: Yes.

Viktor Kunčak: If you compare the number of conjectures for which you can't find counter-examples, and the number that IsaPlanner manages to prove, what is the relationship between these numbers?
Re: In the examples I examined, everything that came out of the counter-example generator was true (no false positives), for the tree example, one of the theorems could not be proved automatically. For the lists, you sometimes need to do generalization of the accumulator argument, so some manual work.

Sascha Bḧme: Have you considered adding these new theorems to the Isabelle distribution?
Re: No, but it's a good idea.

Barbara Jobstmann: Have you considered guiding the theorem generation procedure so that it doesn't generate theorems looking like the ones that were wrong? You could use a fitting function (la genetic programming) to drive the search towards correct lemmas.
Re: I haven't really looked into techniques for avoiding non-theorems.

Applications of Theorem Proving

Session Chair: Barbara Jobstmann
CNRS/Verimag, Gieres, France

13 Automatic Checking of Invariant Diagrams

Johannes Eriksson
Åbo Akademi University, Turku, Finland

Abstract

We have implemented the Socos environment to support Invariant Based Programming. Socos consists of a diagrammatic interface connected to a theorem prover. The diagrammatic interface allows construction of invariant based programs in drawing program like settings. The basic idea is to use an efficient automatic proof strategy to minimize the number of conditions that have to be considered by the programmer, and let the remaining conditions be proved interactively in a proof assistant. We have developed a prototype tool which uses Simplify as a back-end prover and evaluated it in case studies. The tool that we are at the present developing is an extension to Eclipse and is based on the verification system PVS. It takes advantage of the integration of the SMT solver Yices into PVS, using Yices as the default decision procedure for discharging conditions. Conditions that were not proved automatically can be proved interactively in PVS.

We give a demonstration of the prototype tool, and present the architecture of the new tool we are currently working on.

Slides: http://argo.matf.bg.ac.rs/events/2010/slides/Eriksson_AutomaticCheckingOfInvariantDiagrams.pdf

Discussion

Barbara Jobstmann: I was wondering, how students use it: do they start from a program or from the specifications?
Re: It seems to work quite well in practice to start with the code and then try to verify it and refine the specifications until it works.

Viktor Kunčak: If the argument is that teaching a programming language is too hard, how about PVS?
Re: We teach it at various levels. For freshman we don't use PVS and then we can teach various subsets.

Ralpha-Johan Back: The question is maybe how difficult is it to teach logic in high-school? In our experience it works well with propositional logic, but they have more problems with quantifiers. So it seems to make sense to teach them either as a last course in high-school or first in university. I think once they master that they won't have problems grasping PVS. So I think it's a crucial notion that they need to understand.

Walther Neuper: What if somebody here wants to use this for his/her students, for exercises, can we get the program somewhere?
Re: We are in the process of releasing a new version in Spring, based on Eclipse and PVS and no other dependencies, but the current version is a prototype and hard to release.

---

14 Automated Timetabling Using a SAT Encoding

Filip Marić
University of Belgrade, Serbia

Abstract

In this talk, our experience in automated course timetabling for several high school and university departments in Belgrade will be presented. Timetabling is done using a propositional satisfiability (SAT) encoding. Timetable requirements are represented by propositional formulae and SAT solvers are used to search for their models. Each model represents a valid timetable. We describe an appropriate SAT encoding that makes possible to formulate a very wide set of different timetable requirements. We also give some techniques used to reduce the problem size. For instance, room allocation was done in a non standard and efficient way. The results obtained are encouraging and they show that this approach is sound and promising for other applications as well.

Slides: http://argo.matf.bg.ac.rs/events/2010/slides/Maric_AutomatedTimetablingUsingASATEncoding.pdf

Discussion

Viktor Kunčak: How many soft constraints do you usually manage to satisfy?
Re: Sometimes all of them, sometimes not. We usually consider the soft constraints for senior professor first :) Sometimes we completely ignore the ones for younger.

Viktor Kunčak: Do the given CPU times cover the whole process or just one run of the SAT solver?
Re: Just one run.

Oliver Kullmann: In the UK they have conferences on scheduling, etc. Everything seems to be proprietary software though, so this project could be very valuable if it was provided to others.

---

15 Program Languages with CTP Features?

Walther Neuper
Graz University of Technology, Austria

Abstract

Program languages based on Computer Algebra Systems (CAS) have been a success story for the last two decades. What features could we request from languages based on Computer Theorem Provers (CTP)? This talk gives some requirements raised by experiments with an educational math system based on CTP technology (`http://www.ist.tugraz.at/projects/isac`).

A software tutor basically faces two issues: (1) provide feedback on user input, and (2) guide the learners steps towards a solution of the problem to be mastered. CTP accomplishes (1) perfectly: a variety of provers provides checks of "correctness modulo a theory" for an input formula. User guidance might be accomplished by a program, interpreted in a single-stepping mode similar to a debugger: the system stops at certain "break points" and waits for the learners decision of how to proceed: with another step produced by the program or by input of her own formula, checked according to (1) above.

The talk tries to abstract experiences from the experimental prototype towards general features, which might be useful beyond eLearning and interesting for a "Rich-Model Toolkit", in particular with respect to the concept of "con-

text" in PolyML, which recently has been enhanced in cooperation with Isabelle (`http://isabelle.in.tum.de/index.html`).

Slides: http://argo.matf.bg.ac.rs/events/2010/slides/Neuper_ProgramLanguagesWithCTPFeatures.pdf

Discussion

Oliver Kullmann: It seems that the system can help good students. Does it help the bad students too?
Re: My experience is that as a teacher I never had enough time for the good students. The system is clearly designed for the good students, as the bad ones are the ones that school system forces us to take more care of. In my experience, you can give very interesting papers to good students, they will then tend to read them but skip the formalization parts. I believe an indirection through a computer can then help in that case.

Ralph-Johan Back: So, is your system aimed primarily to good or bad students?
Re: Primarily to good students, because it motivates them to explore on their own.

Logical Foundations

Session Chair: Silvia Ghilezan
University of Novi Sad, Serbia

16 Deciding Non-linear Numerical Constraints: an Overview

Stefan Ratschan
Institute of Computer Science, Academy of Sciences of the Czech Republic,
Prague, Czech Republic

Abstract

It is becoming increasingly clear that in order to reason about computer systems one also has to be able to reason about numerical constraints. However, there is not only the problem that in recent decades numerical constraints have been more in the focus of mathematicians than of computer scientists. But even within mathematics, corresponding research has been spread among several communities (e.g., computer algebra, numerical analysis, interval computation, global optimization). The talk will give an overview of the various available techniques with special emphasis on issues relevant for reasoning about computer systems.

Slides: http://argo.matf.bg.ac.rs/events/2010/slides/Ratschan_DecidingNonLinearNumericalConstraintsAnOverview.pdf

Discussion

Viktor Kunčak: Can you tell us a bit more about your tool?
Re: It can be downloaded at: `http://rsolver.sourceforge.net/` It is based on interval techniques, but there are also some other advanced techniques implemented inside. In general, it works more efficiently than classical symbolic techniques. However, if a problem is close to the border between robust/non-robust problems, symbolic techniques might work better.

Siliva Ghilezan: Are you planning to add some extensions towards the fuzzy decidability?
Re: No.

---

17 Decision Procedures for Algebraic Data Types with Abstractions

Philippe Suter
EPFL, Lausanne, Switzerland

Abstract

We describe a family of decision procedures that extend the decision procedure for quantifier-free constraints on recursive algebraic data types (term algebras) to support recursive abstraction functions. Our abstraction functions are catamorphisms (term algebra homomorphisms) mapping algebraic data type values into values in other decidable theories (eg. sets, multisets, lists, integers, booleans). Each instance of our decision procedure family is sound; we identify a widely applicable many-to-one condition on abstraction functions that implies the completeness. Complete instances of our decision procedure include the following correctness statements:

1. a functional data structure implementation satisfies a recursively specified invariant;

2. such data structure conforms to a contract given in terms of sets, multisets, lists, sizes or heights;

3. a transformation of a formula (or lambda term) abstract syntax tree changes the set of free variables in the specified way.

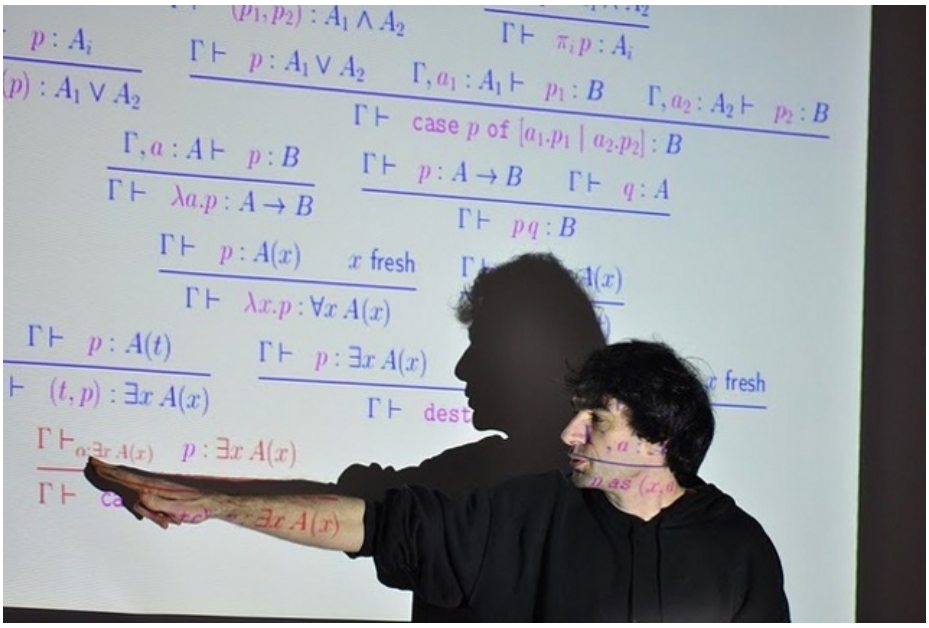
Slides: http://argo.matf.bg.ac.rs/events/2010/slides/Suter_DecisionProceduresForAlgebraicDataTypesWithAbstraction.pdf

Discussion

Barbara Jobstmann: Are you planning to implement your decision procedure?
Re: Yes.
Sivia Ghilezan: You mentioned the cardinality of the inverse images for sets and lists. What about multisets?
Re: You can use the formula for lists and use it as a lower bound.

---

18 Intuitionistically Proving Markov's Principle Thanks to Delimited Control



Hugo Herbelin
INRIA — PPS, Paris, France

Abstract

We add a bit of classical logic to intuitionistic logic and show that Markov's principle (i.e. not not exists x A(x) implies exists x A(x)) is derivable while still preserving the disjunction and existential properties of intuitionistic logic. Extraction of the constructive content of a classical proof of exists x A(x) is done using a control delimiter what corresponds to an internalisation of Friedman's A-translation.

Slides: http://argo.matf.bg.ac.rs/events/2010/slides/Herbelin_IntuinisticallyProvingMarkovsPrincipleThanksToDelimitedControl.pdf

Discussion

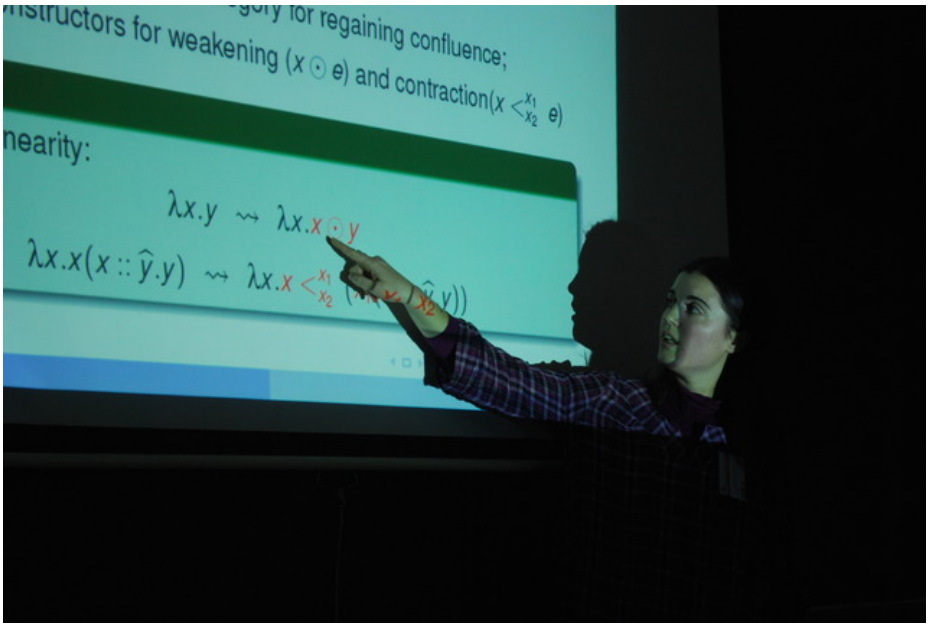
Viktor Kunčak: Could you give some other examples, beside "the" example in your slides?

Re: For example, as shown by Gödel and Kreisel, the proof of completeness of the classical first-order logic needs Markov's principle to be formalized intuitionistically.

Silvia Ghilezan: You have in your calculus the "call by value". Is there a possibility in your calculus to explore the "call by name"?
Re: There is a possibility by relaxing the call-by-value constraint on certain rules of the calculus. Then, some extra care has to be taken if one does not want to introduce non-determinism.

---

19 Intuitionistic Sequent-style Calculus with Explicit Structural Rules



Jelena Ivetić
A joint work with Silvia Ghilezan, Pierre Lescanne and Dragiša Žunić
University of Novi Sad, Serbia

Abstract

In this talk we extend the Curry-Howard correspondence to intuitionistic sequent calculus with explicit weakening and contraction. We study a system derived from λ^{Gtz} of Espirito Santo by adding explicit operators for weakening and contraction, which we call $\ell\lambda^{\mathsf{Gtz}}$. This system contains only linear terms. For the proposed calculus we introduce the type assignment system with simple types. The presented system has a natural diagrammatic representation, which is used for proving the subject reduction property. We prove the strong normalisation property by embedding $\ell\lambda^{\mathsf{Gtz}}$ into the simply typed $\lambda\mathsf{lxr}$ calculus of Kesner and Lengrand.

Slides: http://argo.matf.bg.ac.rs/events/2010/slides/Ivetic_IntuitionisticSequent-styleCalculusWithExplicitStructuralRules.pdf

Discussion

Marc Bezem: What can you do with terms in the CurryHoward isomorphism?
Re: We can compute the same things as with ordinary lambda calculus, but we can see the process of computation in more details, revealing the details that are usually hidden with implicit structural rules.

Programme

January 29, 2010

January 29, 2010.	
09:30—10:00	Registration
10:00—10:15	Opening Remarks
Invited Lecture	
10:15—11:00	Ralph-Johan Back (Åbo Akademi University, Turku, Finland): *Invariant Based Programming*
Goals of the COST Action IC0901	
11:00—11:30	Viktor Kunčak (EPFL, Lausanne, Switzerland): *Towards a Rich Model Toolkit*
11:30—12:00	Coffee break
Session *SAT Solving*; Session Chair: Peter Schneider-Kamp	
12:00—12:30	Oliver Kullmann (Swansea University, United Kingdom): The OKlibrary, an Open-source Research Platform for SAT Solving (and Beyond)
12:30—13:00	Mladen Nikolić (University of Belgrade, Serbia): A Methodology for Comparison and Ranking of SAT Solvers
13:00—14:30	Lunch break
Session *SMT Solving*; Session Chair: Paul Jackson	
14:30—15:00	Ružica Piskač (EPFL, Lausanne, Switzerland): Combining Theories with Shared Set Operations
15:00—15:30	Philipp Rümmer (Oxford University Computing Laboratory, United Kingdom): The SMT-LIB 2 Standard: Overview and Proposed New Theories
15:30—16:00	Predrag Janičić and Filip Marić (University of Belgrade, Serbia): Uniform Reduction to SAT and SMT
16:00—16:30	Coffee break
16:30—17:00	Paul Jackson (University of Edinburgh, United Kingdom): *Proving SPARK VCs with SMT solvers. Implications for the Rich Model Language*
Session *COST IC0901* ; Session Chair: Viktor Kunčak	
17:00—18:00	Rich Model Language: Panel Discussion
19:15—20:00 20:00—22:30	Visit to the Retrospective Exhibition of Paja Jovanović Dinner

January 30, 2010

January 30, 2010.	
Session *Formal and Automated Theorem Proving*; Session Chair: Filip Marić	
09:30—10:00	Sascha Böhme (TU Munich , Germany): *Efficient Proof Reconstruction for the SMT Solver Z3*
10:00—10:30	Florian Haftmann (TU Munich, Germany): *Integrating Isabelle/HOL And Functional Programming – Current Trends*
10:30—11:00	Marc Bezem (University of Bergen, Norway): *Automating Coherent Logic*
11:00—11:30	Moa Johansson (University of Verona, Italy): *Conjecture Synthesis for Inductive Theory Formation*
11:30—12:00	Coffee break
Session *Applications of Theorem Proving*; Session Chair: Barbara Jobstmann	
12:00—12:30	Johannes Eriksson (Åbo Akademi University, Turku, Finland): *Automatic Checking of Invariant Diagrams*
12:30—13:00	Filip Marić (University of Belgrade, Serbia): Automated Timetabling using a SAT Encoding
13:00—13:30	Walther Neuper (Graz University of Technology, Austria): *Program Languages with CTP Features?*
13:30—15:00 15:00—16:00	Lunch break Mini excursion: Knez Mihajlova Street and Kalemegdan
Session *Logical Foundations*; Session Chair: Silvia Ghilezan	
16:00—16:30	Stefan Ratschan (Academy of Sciences, Prague, Czech Republic): *Deciding Non-linear Numerical Constraints: an Overview*
16:30—17:00	Philippe Suter (EPFL, Lausanne, Switzerland): Decision Procedures for Algebraic Data Types with Abstractions
17:00—17:30	Coffee break
17:30—18:00	Hugo Herbelin (INRIA - PPS, Paris, France): *Intuitionistically Proving Markov's Principle Thanks to Delimited Control*
18:00—18:30	Jelena Ivetic (University of Novi Sad, Serbia): *Intuitionistic Sequent-style Calculus with Explicit Structural Rules*
19:45—22:30	Dinner