ISSN 2186-7437

NII Shonan Meeting Report

No. 2016-4

Higher-order model checking

Naoki Kobayashi
Luke Ong
Igor Walukiewicz

March 14–17, 2016

National Institute of Informatics
2-1-2 Hitotsubashi, Chiyoda-Ku, Tokyo, Japan

Higher-order model checking

Organizers:
Naoki Kobayashi (Tokyo University)
Luke Ong (University of Oxford)
Igor Walukiewicz (CNRS, Bordeaux University)

March 14–17, 2016

Finite state model checking has been widely studied and successfully applied to system verification. The main theme of this meeting, higher-order model checking, is a generalization of finite state model checking, obtained by replacing finite state models with more expressive models called recursion schemes. Higher-order model checking (HOMC) has found applications in analysis of object-oriented and concurrent programs with recursion and higher-order procedures.

Recursion schemes are a kind of simply-typed grammar for generating possibly infinite ranked trees. A recursion scheme is a finite system of equations, defining a finite set of higher-order functions by mutual recursion. The order of a recursion scheme is given by the highest type-theoretic order of the functions defined by it. From a programming language perspective, recursion schemes may be viewed as programs (i.e. closed, ground-type terms) of the simply-typed lambda calculus with recursion, constructed from a set of uninterpreted function symbols. Higher-order model checking is the model checking of trees generated by recursion schemes. The higher-order model checking problem asks, given a recursion scheme $\mathcal{G}$ and a correctness property ϕ, whether the tree generated by $\mathcal{G}$ satisfies ϕ.

Topics of the workshop are:

- Extensions of HOMC: beyond simple types (e.g. untyped and recursively typed recursion schemes, higher-type Böhm trees); beyond omega-regular properties: finitary parity, omega-B, stack unboundedness, etc.
- Algorithms for HOMC: HorSatZDD, Preface, C-SHORe, etc.
- Higher-order grammars and pushdown automata: Context sensitivity of unsafe Maslov languages and other open problems. Effective denotational semantics and strategy-aware models for HOMC: Compositional approaches to HOMC.

Overview of Talks

Concurrent Hyland-Ong Games

Pierre Clairambault, ENS Lyon

This talk will be an overview of recent work in collaboration with Simon Castellan and Glynn Winskel on concurrent game semantics for higher-order programming languages.

Game semantics are an important tool for the semantics and verification of higher-order programs. They are central in Ong's original decidability proof for MSO on HORS, and recent developments by Tsukada and Ong on compositional HOMC. The close line of work of algorithmic game semantics exploits effective representations of higher-order effectful programs provided by game semantics to develop verification algorithms. Those rely on an important body of prior work on games model for sequential higher-order languages. For modeling concurrent programs though, the easiest choice is to represent them as non-deterministic sequential strategies, wiring in the non-determinism of the scheduler. This causes a combinatorial explosion problem and blurs the causal structure of the program.

In recent work, we have developed a new framework based on event structures for the game semantics of higher-order concurrent languages that avoids interleavings, and focuses on the causal structure of programs. In this talk I will give an overview of our work. I will first give a high-level description of the model, relying on examples from higher-order concurrent effectful languages (e.g. Idealized Parallel Algol – IPA). Then I will describe two applications: firstly, a causal version of Ghica and Murawski's interleaving-based fully abstract games model of IPA. Secondly, a new interpretation of the sequential language PCF. Unlike the traditional one, the interpretation is parallel, and displays independent branches of computation. However, just as the traditional one, our model has a finite definability property and its extensional collapse is fully abstract.

On the Proof Theory of Infinitary Proofs

Amina Doumane,Université Paris Diderot

Infinitary and circular proofs are commonly used in fixed point logics. Being natural intermediate devices between semantics and traditional finitary proof systems, they are commonly found in completeness arguments, automated deduction, verification, etc. However, their proof theory is surprisingly underdeveloped. In particular, very little is known about the computational behavior of such proofs through cut elimination. Taking such aspects into account has unlocked rich developments at the intersection of proof theory and programming language theory. One would hope that extending this to infinitary calculi would lead, e.g., to a better understanding of recursion and corecursion in programming languages. Structural proof theory is mostly based on two fundamental properties of a proof system: cut elimination and focalization. The first one is only known to hold for restricted (purely additive) infinitary calculi, thanks to the work of Santocanale and Fortier; the second one has never been studied in

infinitary systems. In this work, we consider the infinitary proof system MALL for multiplicative and additive linear logic extended with least and greatest fixed points, and prove these two key results. We thus establish MALL as a satisfying computational proof system in itself, rather than just an intermediate device in the study of finitary proof systems.

Linear Dependent Types for Higher-order Model Checking

Marcon Gaboardi, University of Buffalo

Linear dependent types are a data-dependent generalization of bounded linear logic that have proved useful to reason about programs resource consumption. They give a convenient abstraction of PCF programs behavior including intensional and extensional information. We previously used them to describe program complexity and to ensure differential privacy. In this talk we will describe how linear dependent types can be extended to reason about reachability properties. Concretely we will provide an embedding of Kobayashi's intersection type system for reachability in linear dependent types. This embedding provides a different perspective on how to structure the typing information for higher order model checking. The embedding gives also evidence of the relations between linear dependent types and indexed linear logic. Ultimately, our goal is to provide a general way to describe different intensional and extensional information as higher order model checking problems.

This is ongoing work, joint with Charles Grellois.

Semantics of Linear Logic and Higher-Order Model Checking

Charles Grellois, University of Bologna

MSO properties can be checked over infinite trees using alternating parity automata. Alternation enables the automaton to duplicate or erase subtrees during its execution, a behavior strikingly similar to the one of the exponential of linear logic. Following this observation, we proved with Mellis that models of linear logic can be adapted in order to capture the higher-order model-checking problem for properties expressed by an alternating automaton A without parity condition: the interpretation of a higher-order recursion scheme G is the set of states from which the infinite tree $[\![G]\!]$ it computes is accepted by A.

To extend this model-theoretic approach, we refined the coloring policy of the Kobayashi-Ong type system and showed that it can be defined in a modal way. This enables us to extend models of linear logic with a coloring comonad, and a parity fixpoint operator, which interprets the recursion of higher-order recursion schemes inductively or coinductively, depending on the coloring information stored in the denotations of the model. We apply this extension to two models of linear logic, and notably to the Scott model of linear logic, in which the interpretation of a type is finite. This finiteness properties allows us to obtain a semantic proof of the decidability of higher-order model-checking, but also a decidability proof of the selection problem originally formulated by Carayol and Serre.

Unboundedness and the Analysis of Higher-Order Programs

Matthew Hague, Royal Holloway University of London

Recently, it has been shown that the diagonal problem for higher-order recursion schemes (HORS), and hence the simultaneous unboundedness problem, is decidable. From recent work by Zetzsche this means that we can construct the downward closure of the set of traces constructed by a given HORS. This also means we can construct the downward closure of the Parikh image of a HORS. In addition, the result also implies decidability of separability by piecewise testable languages, as well as the decidability of reachability of parameterised asynchronous networks of HORS.

Hence, we have a several new tools in the analysis of HORS. There are at least two future challenges: how can we use these tools to further the analysis of (concurrent) higher-order systems, and how can we put these tools into practice?

Partial Evaluation and Normalisation by Traversals

Neil D. Jones, DIKU, University of Copenhagen

Game semantics re-examined: The game semantics for PCF can be thought of as a PCF interpreter. In game semantics papers the denotation of an expression is a game strategy. When played, the game results in a traversal. A recent paper by Ong normalises simply typed lambda-expression by generating traversals.

A surprising consequence: it is possible to build a lambda calculus interpreter with none of the traditional implementation machinery: beta-reduction; environments binding variables to values; and "closures" and "thunks" for function calls and parameters. (Interestingly, this was implicitly visible in early work on full abstraction.)

It looks promising to study game semantics from a new angle: their operational aspects. We apply partial evaluation to translate lambda-expressions into LLL, a low-level language. Further, this may give a new approach to an old topic: semantics-directed compiler generation.

This is joint work with Danii Berezun, St. Petersburg State University.

Overview of Higher-Order Model Checking Project at Tokyo

Naoki Kobayashi, Tokyo University

This talk gives an overview of our project on higher-order model checking and its applications to program verification and data compression. After recalling the higher-order model checking problem and how it can be applied to program verification, I will summarize the goal and current status of the project, and demonstrate some of the tools we have developed so far. If time permits, I will also provide a tutorial on type-based approach to higher-order model checking.

Higher-order Fixpoint Logic

Martin Lange, University of Kassel

Higher-Order Fixpoint Logic (HFL) is an extension of the modal mu-calculus by higher-order features, syntactically represented using a simply typed lambda calculus. Its formulas of order 0 form the modal mu-calculus and express properties of states in a transition system, i.e. predicates. Its formulas of order 1 express predicate transformers, i.e. mappings from predicates to predicates. Monotone predicate transformers form a complete lattice over any transition systems, thus a denotational semantics for fixpoint formulas over predicate transformers can be given. This principle easily extends to functions of higher order. We will introduce the syntax and semantics of HFL and give some examples of properties expressible in HFL (but not in the mu-calculus) thus trying to give some intuition on how such formulas can be read and understood. We will survey some results known about the expressive power and complexity of HFL, most importantly that the model checking problem for the order-k fragment is complete for k-EXPTIME.

Analyzing Time Complexity of Regular Expression Matching Based on Backtracking

Yasuhiko Minamide, Tokyo Institute of Technology

We develop an analysis for a regular expression that precisely determines the time complexity of its backtrack-based matching. More precisely, we decide i of $O(n^i)$ for a regular expression with polynomial time complexity. A regular expression is translated to a string-to-tree transducer with regular lookahead whose output represents the computation tree of backtrack-based matching. Then, we decide the order of the size increase of the transducer by extending the result of Aho and Ullman[1] for transducers without lookahead to those with regular lookahead. We will also talk about our experimental results.

Contextual approximation and higher-order procedures

Andrzej Murawski, University of Warwick

We investigate the complexity of deciding contextual approximation (refinement) in finitary Idealized Algol, a prototypical language combining higher-order types with state. Earlier work in the area established the borderline between decidable and undecidable cases, and focussed on the complexity of deciding approximation between terms in normal form.

In contrast, in this paper we set out to quantify the impact of locally declared higher-order procedures on the complexity of establishing contextual approximation in the decidable scenarios. We show that the obvious decision procedure based on exhaustive beta-reduction can be beaten. Further, by classifying regexes by levels, we give tight bounds on the complexity of contextual

[1] A. V. Aho and J. D. Ullman: Translations on a context free grammar, *Information and Control*, 19(5), 1971.

approximation for terms that may contain redexes up to level k, namely, (k-1)-EXPSPACE-completeness.

Interestingly, the bound is obtained by selective beta-reduction: redexes from level 3 onwards can be fired without losing optimality, whereas redexes of level up to 2 are handled by a dedicated decision procedure based on game semantics and a variant of pushdown automata.

This is joint work with Ranko Lazic.

Alternating Dependency Tree Automata, Higher-type Mu-calculus and Type-checking Games

Luke Ong, University of Oxford

The equivalence of alternating parity tree automata, modal mu-calculus, and parity games as definitional devices for tree languages is a keystone result underpinning the model checking of reactive systems. We lift this 3-way equi-expressivity result to languages of higher-type Bhm trees, which may be viewed as higher-order functions over trees.

Joint work with Matthew Hague, Steven Ramsay and Takeshi Tsukada.

Models of Lambda-Calculus and Caucal Hierarchy for Weak Logics

Paweł Parys, University of Warsaw

We study weak logics in relationship to infinitary safe lambda-calculus, safe higher order recursive schemes, and the Caucal hierarchy. In particular, we study weak MSO logic extended by the unbounding quantifier (WMSO+U), expressing the fact that there exist arbitrarily large finite sets satisfying a given property. We prove two results for this logic. Firstly, we show that for every formula of WMSO+U there exists a finitary model of inifinitary safe lambda-calculus recognizing the set of infinitary lambda-terms which generate a tree satisfying the given formula. Secondly, we show how this implies that the corresponding Caucal hierarchy, obtained by unravellings and interpretations via WMSO+U, coincides with the standard Caucal hierarchy, obtained by unravellings and interpretations via MSO. In fact, the Caucal hierarchy remains the same also when only the first-order logic is allowed in interpretations.

This is joint work with Szymon Toruczyk, ongoing work that we are finishing and preparing for publication.

Higher-Order Horn Clauses and Higher-Order Model Checking

Steven Ramsay, University of Oxford

A standard approach to showing that the tree generated by a given recursion scheme satisfies a given property is to find an inductive invariant for the scheme that implies the property. For example, in the intersection type approach to higher-order model checking (HOMC), the goal is typically to construct a type

environment of a certain form, which then constitutes a symbolic representation of the invariant. In this work, we consider the problem of finding higher-order inductive invariants in a purely logical setting, namely: the satisfiability problem for (constrained) higher-order horn clauses. Viewed as a general formulation of higher-order constraint solving, the problem has a much broader appeal than recursion scheme model checking, yet we argue that much of the technology already developed by the HOMC community can be made highly effective at solving it. In particular, we describe an adaptation of Kobayashi's Hybrid Algorithm to the problem and highlight its similarities to McMillan's Lazy Annotation algorithm (as used in first-order horn clause solving).

Towards Finite-Dimensional Feature Types for Synthesis

Jakob Rehof, Technical University Dortmund

In the context of Combinatory Logic Synthesis we investigate the type inhabitation problem as a logical foundation for component-oriented synthesis. From a type-theoretic standpoint, we are led to investigating the complexity and expressive power of fragments of type systems containing combinations of schematic types and intersection types. In this talk we will describe an ongoing effort to design finite-dimensional restrictions of the intersection type system which can express logical feature vectors of interest in synthesis and schematic program construction. Our search for such systems has led to a fine-grained analysis of borderlines dividing undecidable and decidable fragments of the intersection typed lambda calculus, and where finite-dimensional restrictions may suggest new lines of division orthogonal to known principles of restriction such as rank or order.

Denotations for Parity Automata

Sylvain Salvati, INRIA Bordeaux

In this talk, we will present a model of lambda Y-calculus that recognizes the same languages as parity automata. In particular, we propose an explicit representation of the interpretation of fixpoints.

Verifying Relational Properties of Functional Programs by First-Order Refinement

Ryosuke Sato, University of Tokyo

Much progress has been made recently on fully automated verification of higher-order functional programs, based on refinement types and higher-order model checking. Most of those verification techniques are, however, based on *first-order* refinement types, hence unable to verify certain properties of functions (such as the equality of two recursive functions and the monotonicity of a function, which we call *relational properties*). To relax this limitation, we introduce a restricted form of higher-order refinement types where refinement predicates can refer to functions, and formalize a systematic program transformation to reduce type checking/inference for higher-order refinement types

to that for first-order refinement types, so that the latter can be automatically solved by using an existing software model checker. We also prove the soundness of the transformation, and report on implementation and experiments.

Temporal Verification of Higher-Order Functional Programs

Tachio Terauchi, School of Information Science, Japan Advanced Institute of Science and Technology

We present an automated approach to verifying arbitrary omega-regular properties of higher-order functional programs. Previous automated methods proposed for this class of programs could only handle safety properties or termination, and our approach is the rst to be able to verify arbitrary omega-regular liveness properties.

Our approach is automata-theoretic, and extends our recent work on binary-reachability-based approach to automated termination verication of higher-order functional programs to fair termination published in ESOP 2014. In that work, we have shown that checking disjunctive well-foundedness of (the transitive closure of) the calling relation is sound and complete for termination. The extension to fair termination is tricky, however, because the straightforward extension that checks disjunctive well-foundedness of the fair calling relation turns out to be unsound, as we shall show in the paper. Roughly, our solution is to check fairness on the transition relation instead of the calling relation, and propagate the information to determine when it is necessary and sufcient to check for disjunctive well-foundedness on the calling relation. We prove that our approach is sound and complete. We have implemented a prototype of our approach, and confirmed that it is able to automatically verify liveness properties of some non-trivial higher-order programs.

Negations in Refinement Intersection Type Systems

Takeshi Tsukada, University of Tokyo

Refinement intersection type systems are a basic tool for higher-order model checking that is important from theoretical and practical points of views. In general, a derivation gives a witness of derivability whereas there is no simple witness of underivability of a given judgement. In this talk, we study a type system that proves underivability of a judgement.

Given a type τ that refines a simple-type A, its negation $\neg\tau$ is defined as the type for terms that do not have type τ, that means, $\not\vdash t : \tau$ if and only if $\vdash t : \neg\tau$ (for every t of simple type A). So the derivability of $\vdash t : \neg\tau$ is equivalent to the underivability of $\vdash t : \tau$. We show that the negation is a definable connective for certain intersection type systems. Hence, for such a system, the set of types is closed under all Boolean operations as the class of regular languages.

Model-Checking Program Equivalence in Interface Middleweight Java

Nikos Tzevelekos, Queen Mary University of London

Using game semantics, we investigate the problem of verifying contextual

equivalences in Interface Middleweight Java (IMJ), an imperative object calculus in which program phrases are typed using interfaces. In particular, we show how to determine the decidability status of problem instances (over a fixed type signature) by examining the position of methods inside the term type and the types of its free identifiers. Our results build upon the recent fully abstract game semantics of IMJ. Decidability is proved by translation into visibly pushdown register automata over infinite alphabets with fresh-input recognition, and the procedure has been implemented into a new tool called Coneqct.

Relational Verification of Functional Programs via Induction-based Horn Constraint Solving

Hiroshi Unno, University of Tsukuba

This talk presents an automated method for verification of higher-order functional programs. The main advantage of the proposed method is that it can verify relational specifications (e.g., the equivalence, associativity, commutativity, distributivity, monotonicity, idempotency, and non-interference) where multiple function calls need to be analyzed simultaneously. As in previous work, our method reduces verification problems into constraint solving problems of Horn clauses with unknown predicate variables. To enable relational verification, we propose a novel Horn constraint solving method based on inductive theorem proving: the method reduces Horn constraint solving to validity checking of first-order formulas with inductively-defined predicates, which are then checked by induction on the derivation of the predicates. We here use an SMT solver to automate inductive proofs. The use of Horn clauses, which have recently been considered as a common intermediate language for verification, enables our method to be applied to relational verification across programs in various paradigms such as imperative, logic, concurrent, as well as functional ones. Furthermore, our novel combination of Horn constraint solving and inductive theorem proving extends the reach of induction-based verification from pure total functions to impure partial procedures in various paradigms. We have implemented a relational verification tool for the OCaml functional language based on the proposed method and obtained promising results in preliminary experiments.

The Diagonal Problem for Higher-Order Recursion Schemes is Decidable

Igor Walukiewicz, CNRS, Bordeaux University

A non-deterministic recursion scheme recognizes a language of finite trees. This very expressive model can simulate, among oth- ers, higher-order pushdown automata with collapse. We show the decidability of the diagonal problem for schemes. This result has several interesting consequences. In particular, it gives an algorithm to compute the downward closure of languages of words recognized by schemes. In turn, this has immediate application to separability problems and reachability analysis of concurrent systems.

This is joint work with Lorenzo Clemente, Warsaw University; Pawel Parys, Warsaw University; and Sylvain Salvati, INRIA Bordeaux.

The Complexity of Downward Closure Comparisons

Georg Zetzsche, Universite Paris Diderot

The downward closure of a language is the set of all (not necessarily contiguous) subwords of its members. It is well-known that the downward closure of every language is regular. One advantage of abstracting a language by its downward closure is that then, equivalence and inclusion become decidable.

It has recently been shown by Hague, Kochems, and Ong that downward closures are computable for higher-order pushdown automata. However, the current method yields no upper bound on the complexity of such a computation. This talk will present recent results on complexity issues surrounding downward closures. Aside from general algorithms and possible approaches to obtain upper bounds in the case of HOPA, we will discuss a lower bound result for the abovementioned equivalence and inclusion problem for HOPA.

List of Participants

- Pierre Clairambault, CNRS & ENS Lyon, France
- Amina Doumane PPS, Universit Paris Diderot, France
- Charles Grellois, Universit Paris Diderot & University of Dundee France
- Marco Gaboardi, University of Dundee, UK
- Matthew Hague Royal Holloway, University of London, UK
- Atsushi Igarashi, Kyoto University, Japan
- Jennifer Jochems, University of Oxford, UK
- Neil Jones DIKU, University of Copenhagen, Denmark
- Naoki Kobayashi, The University of Tokyo, Japan
- Martin Lange, University of Kassel, Germany
- Martin Lester, University of Oxford, UK
- Rupak Majumdar, MPI-SWS, Germany
- Yasuhiko Minamide, Tokyo Institute of Technology, Japan
- Andrzej Murawski, University of Warwick, UK
- Luke Ong, University of Oxford, UK
- Pawel Parys, University of Warsaw, Poland
- Steven Ramsay, University of Oxford, UK
- Jakob Rehof, Dortmund Technical University, Germany
- Sylvain Salvati, INRIA, France
- Ryosuke Sato, University of Tokyo, Japan
- Techie Terauchi, JAIST, Japan
- Peter Thiemann, Universität Freiburg, Germany
- Takeshi Tsukada, The University of Tokyo, Japan
- Nikos Tzevelekos, Queen Mary University of London, UK
- Hiroshi Unno, University of Tsukuba, Japan
- Igor Walukiewicz CNRS, Bordeaux, France
- Georg Zetzsche, LSV Cachan, Germany

Meeting Schedule

Check-in Day: March 13 (Sun)

- Welcome Banquet

Day1: March 14 (Mon)

9:00 - 9:45 Luke Ong

9:45 - 10:30 Takeshi Tsukada

10:30 - 11:00 Tea break

11:00 - 11:45 Charles Grellois

11:45 - 14:00 Lunch break

14:00 - 14:45 Neil Jones

14:45 - 15:30 Pierre Clairambault

15:30 - 16:00 Tea break

16:00 - 16:45 Amina Doumane

16:45 - 17:10 Ryosuke Sato

17:10 - 17:35 Yasuhiko Minamide

Day2: March 15 (Tue)

9:00 - 9:45 Naoki Kobayashi

9:45 - 10:30 Steven Ramsay

10:30 - 11:00 Tea break

11:00 - 11:45 Tachio Terauchi

11:45 - 14:00 Lunch break

14:00 - 14:45 Jakob Rehof

14:45 - 15:30 Sylvain Salvati

15:30 - 16:00 Tea break

16:00 - 16:45 Hiroshi Unno

16:45 - 17:30 Martin Lange

Day3: March 16 (Wed)

9:00 - 9:45 Georg Zetzsche

9:45 - 10:30 Matthew Hague

10:30 - 11:00 Tea break

11:00 - 11:45 Igor Walukiewicz

11:45 - 13:30 Lunch break

14:00 - Excursion

Day4: March 17 (Thu)

9:00 - 9:45 Pawel Parys

9:45 - 10:30 Nikos Tzevelekos

10:30 - 11:00 Tea break

11:00 - 11:45 Marco Gaboardi

11:45 - 12:15 Andrzej Murawski