06081 Abstracts Collection
Software Verification: Infinite-State Model Checking and Static Program Analysis
— Dagstuhl Seminar —

Parosh Aziz Abdulla[1], Ahmed Bouajjani[2] and Markus Müller-Olm[3]

[1] University of Uppsala, SE
parosh@docs.uu.se
[2] LIAFA - Université Paris VII, FR
ahmed.bouajjani@liafa.jussieu.fr
[3] Westfälische Wilhelms-Universität Münster, DE
mmo@math.uni-muenster.de

Abstract. From 19.02.06 to 24.02.06, the Dagstuhl Seminar 06081 "Software Verification: Infinite-State Model Checking and Static Program Analysis" was held in the International Conference and Research Center (IBFI), Schloss Dagstuhl. During the seminar, several participants presented their current research, and ongoing work and open problems were discussed. Abstracts of the presentations given during the seminar are put together in this paper. The first section describes the seminar topics and goals in general. Links to extended abstracts or full papers are provided, if available.

Keywords. Software verification, infinite-state systems, static program analysis, automatic analysis

06081 Executive Summary – Software Verification: Infinite-State Model Checking and Static Program Analysis

Software systems are present at the very heart of many daily-life applications, such as in communication (telephony and mobile Internet), transportation, energy, health, etc. Such systems are very often *critical* in the sense that their failure can have considerable human/economical consequences. In order to ensure reliability, development methods must include *algorithmic analysis and verification techniques* which allow (1) to detect automatically defective behaviors of the system and to analyze their source, and (2) to check that every component of a system conforms to its specification.

Two important and quite active research communities are particularly concerned with this challenge: the community of computer-aided verification, especially the community of (infinite-state) model checking, and the community of

static program analysis. From 19.02.06 to 24.02.06, 51 researchers from these two communities met at the Dagstuhl Seminar 06081 "Software Verification: Infinite-State Model Checking and Static Program Analysis" in order to improve and deepen the mutual understanding of the developed technologies and to trigger collaborations. During the seminar which was held at the International Conference and Research Center (IBFI), Schloss Dagstuhl, several participants presented their current research, and ongoing work and open problems were discussed. Abstracts of the presentations given during the seminar are put together in this paper. Links to extended abstracts or full papers are provided, if available.

Keywords: Infinite-state systems, model checking, program analysis, software verification

Joint work of: Abdulla, Parosh Aziz; Bouajjani, Ahmed; Müller-Olm, Markus

Extended Abstract: http://drops.dagstuhl.de/opus/volltexte/2006/797

Invisible Safety for Distributed Protocols

Ittai Balaban (Courant Institute - New York, USA)

The method of "Invisible Invariants" has been applied successfully to protocols that assume a "symmetric" underlying topology, be it cliques, stars, or rings. In this paper we show how the method can be applied to proving safety properties of distributed protocols running under arbitrary topologies. Many safety properties of such protocols have reachability predicates, which, on first glance, is beyond the scope of the Invisible Invariants method. To overcome this difficulty, we present a technique, called "coloring," that allows, in many instances, to replace the second order reachability predicates by first order predicates, resulting in properties that are amenable to Invisible Invariants, where "reachable" is replaced by "colored." We demonstrate our techniques on several distributed protocols, including a variant on Luby's Maximal Independent Set protocol, the Leader Election protocol used in the IEEE 1394 (Firewire) distributed bus protocol, and various distributed spanning tree algorithms.

All examples have been tested using the symbolic model checker TLV.

Keywords: Model-checking, parameterized systems, shape analysis, deductive verification, distributed protocols

Joint work of: Balaban, Ittai; Pnueli, Amir; Zuck, Lenore

Abstraction, Loops and Falsification

Thomas Ball (Microsoft Research, USA)

Finitary abstraction is useful for helping to prove properties of programs but less helpful for proving the presence of errors.

In particular, to prove the presence of an error, one must show that loops on the way to the error state terminate. Existing abstractions for underapproximation (such as *must* transitions in modal transition systems) do not address this problem as they help prove loops are non-terminating (rather than terminating) for some inputs. We demonstrate that by simple local analysis over a modal transition system and basic properties of the underlying concrete system, it is possible to infer transitive must relationships, without resorting to abstraction refinement.

We show how these transitive must relationships enable us to demonstrate the presence of rarely occurring errors.

Keywords: Finitary abstraction, loops, termination analysis, modal transition systems

Joint work of: Ball, Thomas; Sagiv, Mooly

Analysis of Dynamic Communicating Systems by Hierarchical Abstraction

Jörg Bauer (Universität des Saarlandes, D)

We propose a new abstraction technique for verifying topology properties of dynamic communicating systems (DCS), a special class of infinite-state systems. DCS are characterized by unbounded creation and destruction of objects along with an evolving communication connectivity or topology. We employ a lightweight graph transformation system to specify DCS. Hierarchical Abstraction (HA) computes a bounded over-approximation of all topologies that can occur in a DCS directly from its transformation rules. HA works in two steps. First, for each connected component, called cluster, of a topology, objects sharing a common property are summarized to one abstract object. Then isomorphic abstract connected components are summarized to one abstract component, called abstract cluster. This yields a conservative approximation of all graphs that may occur during any DCS run. The technique is implemented.

Keywords: Graph transformation, abstract interpretation, shape analysis

Joint work of: Bauer, Jörg; Wilhelm, Reinhard

Full Paper: http://drops.dagstuhl.de/opus/volltexte/2006/727

Lazy Shape Analysis

Dirk Beyer (EPFL - Lausanne, CH)

Many software model checkers are based on predicate abstraction. Values of variables in branching conditions are represented abstractly using predicates. The strength of this approach is its path-sensitive nature.

However, if the control flow depends heavily on the values of memory cells on the heap, the approach does not work well, because it is difficult to find 'good' predicate abstractions to represent the heap. In contrast, shape analysis can lead to a very compact representation of data structures stored on the heap. We combine shape analysis and predicate abstraction, and integrate shape analysis into the software model checker BLAST. Because shape analysis is expensive, we do not apply it globally. Instead, we ensure that shapes are computed and stored locally, only where necessary for proving the verification goal. To achieve this, we extend lazy abstraction refinement, which so far has been used only for predicate abstractions, to shapes. This approach does not only increase the precision of model checking and shape analysis taken individually, but also increases the efficiency of shape analysis (we do not compute shapes where not necessary). We implemented the technique by extending BLAST with calls to TVLA, and evaluated it on several C programs manipulating data structures, with the result that the combined tool can now automatically verify programs that are not verifiable using either shape analysis or predicate abstraction on its own.

Keywords: Software model checking, shape analysis, counterexample-guided abstraction refinement, interpolation, predicate abstraction

Joint work of: Beyer, Dirk; Henzinger, Thomas A.; Théoduloz, Grégory

Full Paper: http://drops.dagstuhl.de/opus/volltexte/2006/728

See also: Dirk Beyer, Thomas A. Henzinger, and Grégory Théoduloz. Lazy Shape Analysis. In Proc. of CAV 2006, to appear.

Automatic Termination Proofs for Programs with Shape-Shifting Heaps

Dino Distefano (Queen Mary College - London, GB)

We describe a new program termination analysis designed to handle imperative programs whose termination depends on the mutation of the program's heap. We first describe how an abstract interpretation can be used to construct a finite number of relations which, if each is well-founded, implies termination. We then give an abstract interpretation based on separation logic formulae which tracks the depths of pieces of heaps. Finally, we combine these two techniques to produce an automatic termination prover. We show that the analysis is able to prove the termination of loops extracted from Windows device drivers that could not be proved terminating before by other means; we also discuss a previously unknown bug found with the analysis.

Keywords: Termination analysis, heaps, separation logic

Joint work of: Distefano, Dino; Berdine, Josh; Cook, Byron; O'Hearn, Peter

Abstraction Refinement with BDDs and Craig Interpolation

Javier Esparza (Universität Stuttgart, D)

Counterexample-guided abstraction refinement (CEGAR) has proven to be a powerful method for software model-checking. In this paper, we investigate this concept in the context of sequential (possibly recursive) programs whose statements are given as BDDs. We examine how Craig interpolants can be computed efficiently in this case and propose a new, special type of interpolants. Moreover, we show how to treat multiple counterexamples in one refinement cycle. We have implemented this approach as an extension of the model-checker Moped and report on experiments.

Keywords: Abstraction refinement, interpolation

See also: J. Esparza, S. Kiefer, and S. Schwoon. Abstraction refinement with Craig Interpolation and Symbolic Pushdown Systems. In Proc. of TACAS 2006, LNCS 3920, pp. 489-503, Springer-Verlag, 2006.

Analysis of Recursive Markov Chains, Recursive Markov Decision Processes, and Recursive Stochastic Games

Kousha Etessami (University of Edinburgh, GB)

Recursive Markov Chains (RMCs) are a natural abstract model of procedural probabilistic programs and other systems involving both recursion and probability. RMCs define a class of denumerable Markov chains with a rich theory generalizing that of multi-type Branching (stochastic) Processes and Stochastic Context-Free Grammars, and they are tightly related to probabilistic Pushdown Systems. Recursive Markov Decision Processes (RMDPs) and Recursive Stochastic Games (RSGs) extend RMCs with a controller and two adversarial players, respectively.

In a series of recent papers we have studied central algorithmic analysis and verification questions for RMCs, RMDPs, and RSGs, providing some strong upper and lower bounds on the complexity of key algorithmic problems.

I will provide a broad survey of this work, indicate some of the main techniques involved in our analyses, discuss potential application domains, and indicate some of the many directions for future research.

This talk describes joint work with Mihalis Yannakakis (Columbia University) contained in a series of recent papers that appear at: STACS'05, TACAS'05, ICALP'05, QEST'05, and STACS'06, and in a submitted paper.

Keywords: Recursive Markov Chains, Recursive Markov Processes, Recursive Stochastic Games

Joint work of: Etessami, Kousha; Yannakakis, Mihalis

Robust Interpretation of Metric-Time Properties: Continuous Reasoning Meets Formal Methods

Martin Fränzle (Universität Oldenburg, D)

We transfer the concept of robust interpretation from arithmetic first-order theories to metric-time temporal logics. The idea is that the interpretation of a formula is robust iff its truth value does not change under small variation of the constants in the formula. Exemplifying this on Duration Calculus (DC), our findings are that the robust interpretation of DC is equivalent to a multi-valued interpretation that uses the real numbers as semantic domain and assigns Lipschitz-continuous interpretations to all operators of DC. Obviously, such continuity permits various approximation schemes, thus allowing to generalize findings obtained on one trajectory to the whole set of sufficiently close trajectories. This includes approximations between discrete and dense time, thus permitting exploitation of discrete-time (semi-)decision procedures on dense-time properties.

Keywords: Robustness, metric-time temporal logic, duration calculus

Joint work of: Fränzle, Martin; Hansen, Michael R.

Quantitative Aspects of Dynamic Memory Program Analysis

Radu Iosif (VERIMAG - CNRS, F)

In this talk we survey on several recent works that attempt to combine program analysis techniques such as Hoare Logic and Regular Model Checking with quantitative reasoning, by taking into consideration the sizes of the data structures handled by the program. The advantages of combining shape with numeric information are threefold. First, more powerful shape properties can be expressed, for instance the fact that a tree is balanced, or that the sum of the lengths of two lists equals the length of a third one. Second, the semantics of the program can be expressed in a more precise way. Third, one can use existing results in the fields of number theory and extended automata to define decidable classes of programs, for which safety and termination properties can be verified automatically.

Keywords: Program analysis, dynamic memory, number theory, counter automata, tree automata

Joint work of: Iosif, Radu; Bozga, Marius; Bouajjani, Ahmed; Habermehl, Peter; Lakhnech, Yassine; Moro, Pierre; Vojnar, Tomas

Verifying Systems with Dynamically Evolving Structure using Graph Transformation

Barbara König (Universität Stuttgart, D)

Graph transformation systems can be used to model systems with dynamically evolving structures, which are hard to verify due to features such as their infinite state space and the creation and deletion of objects during runtime. We propose to approximate graph transformation systems by Petri nets using an unfolding technique. In a case study we verify insertion into red-black trees by modelling it using graph rewriting and by applying our approximation method.

Keywords: Graph transformation systems, Petri nets, verification

Automata-based Techniques in Symbolic Model Checking

Axel Legay (University of Liège, B)

"(ω-)Regular model checking" is the name of a family of symbolic techniques for analyzing infinite-state systems in which states are represented by (in)finite words, and sets of states by finite automata on these objects, and transitions by finite automata operating on pairs of state encodings, i.e. finite-state transducers. In this context, the central problem is then to compute the iterative closure of a finite-state transducer. A simple technique for iterating transducers is the one based on widening principle. In this approach the idea is simply to compute successive approximations of the closure in order to guess it. In the first part of this talk, we investigate the possibilities of applying widening techniques in case where only specific techniques have been exploited so far. Finding that existing generic techniques are often not applicable in cases easily handled by specific techniques (systems with integers, systems with reals, heterogenous systems, ...), we have developed new approaches to iterating transducers. These approaches build on earlier work, but exploit a number of new conceptual and algorithmic ideas, often induced with the help of experiments, that give it a broad scope, as well as good performance. In the second part of the talk, we address the problem of verifying liveness properties using the regular model checking framework. We show how this problem can be reduced to the one of computing the transitive closure of a transducer.

Keywords: Regular model checking, infinite-state systems, automatic analysis

Flat counter automata almost everywhere!

Jérôme Leroux (LaBRI - Bordeaux, F)

This paper argues that flatness appears as a central notion in the verification of counter automata.

A counter automaton is called flat when its control graph can be "replaced", equivalently w.r.t. reachability, by another one with no nested loops.

From a practical view point, we show that flatness is a necessary and sufficient condition for termination of accelerated symbolic model checking, a generic semi-algorithmic technique implemented in successful tools like FAST, LASH or TReX.

From a theoretical view point, we prove that many known semilinear subclasses of counter automata are flat: reversal bounded counter machines, lossy vector addition systems with states, reversible Petri nets, persistent and conflict-free Petri nets, etc. Hence, for these subclasses, the semilinear reachability set can be computed using a *uniform* accelerated symbolic procedure (whereas previous algorithms were specifically designed for each subclass).

Keywords: Symbolic representation, infinite state system, acceleration, meta-transition

Joint work of: Leroux, Jérôme; Sutre, Grégoire

Full Paper: http://drops.dagstuhl.de/opus/volltexte/2006/729

Full Paper:
http://www.labri.fr/perso/leroux/papiers/LerouxSutre-ATVA05.ps

Inferring Network Invariants Automatically

Martin Leucker (TU München, D)

Verification by network invariants is a heuristic to solve uniform verification of parameterized systems. Given a system P, a network invariant for P is a system that abstracts the composition of every number of copies of P running in parallel. If there is such a network invariant, by reasoning about it, uniform verification with respect to the family $P[1] \parallel \cdots \parallel P[n]$ can be carried out.

In this talk, we propose a procedure searching systematically for a network invariant satisfying a given safety property. The search is optimized using a combination of Angluin's and Biermann's learning/inference algorithms for improving successively possible invariants. We also show how to reduce the learning problem to SAT, allowing efficient SAT solvers to be used, which turns out to yield a very competitive learning algorithm. The overall search procedure finds a minimal such invariant, if it exists.

Keywords: Network invariance, learning, SAT solving

Joint work of: Grinchtein, Olga; Jonsson, Bengt; Leucker, Martin; Piterman, Nir

Thread-Modular Verification as Abstract Interpretation

Alexander Malkis (MPI für Informatik - Saarbrücken, D)

Consider a multithreaded program with a fixed number of finite-state threads communicating via shared variables. The problem is whether there is a computation from an initial state to an error state. We examine the algorithm of Flanagan and Qadeer for this problem. This algorithm computes a superset of states that occur in computations starting in the initial state. Until now no description of this superset was known. We characterize this superset in the framework of abstract interpretation. We show that this algorithm implements Cartesian abstraction.

Keywords: Cartesian abstraction, static analysis, abstract interpretation, threads, shared variables, safety

Joint work of: Malkis, Alexander; Podelski, Andreas

The Zeno Problem for Dense-Timed Petri Nets

Richard Mayr (North Carolina State University, USA)

Dense-Timed Petri Nets are an extension of Petri nets in which each token is equipped with a real-valued clock. The Zeno problem is the question whether there exists a Zeno-computation from a given marking, i.e., an infinite computation which takes only a finite amount of time. This question is hard for dense time, because (unlike for discrete time) an infinite Zeno-computation can have infinitely many time-passing phases of decreasing length (e.g., 2^{-n}, for n=1,2,...). We show the decidability of the Zeno problem by a (partial) encoding of timed Petri nets into a subclass of untimed transfer nets. Furthermore, the related question if there exist arbitrarily fast computations from a given marking in a timed Petri net is also decidable. On the other hand, the existence of an infinite non-Zeno computation (i.e., an infinite computation taking infinite time) is undecidable.

Keywords: Verification, timed Petri nets, Zeno

On Probabilistic Program Equivalence and Refinement

Joël Ouaknine (Oxford University, GB)

We study notions of equivalence and refinement for probabilistic programs formalized in the second-order fragment of Probabilistic Idealized Algol.

Probabilistic programs implement randomized algorithms: a given input yields a probability distribution on the set of possible outputs. Intuitively, two programs are equivalent if they give rise to identical distributions for all inputs. We show that equivalence is decidable by studying the fully abstract game semantics of probabilistic programs and relating it to probabilistic finite automata. For terms in beta-normal form our decision procedure runs in time exponential in the syntactic size of programs; it is moreover fully compositional in that it can handle open programs (probabilistic modules with unspecified components).

In contrast, we show that the natural notion of program refinement, in which the input-output distributions of one program uniformly dominate those of the other program, is undecidable.

Keywords: Probabilistic Idealized Algol, game semantics, program equivalence

Joint work of: Ouaknine, Joël; Murawski, Andrzej

See also: Andrzej S. Murawski and Joël Ouaknine: On Probabilistic Program Equivalence and Refinement. In Proc. of CONCUR 2005, LNCS 3653, pp. 156-170, Springer-Verlag, 2005.

Verifying Properties of Well-founded Linked Lists

Shaz Qadeer (Microsoft Research, USA)

We present a novel method for verifying programs that manipulate linked lists, based on two new predicates that characterize reachability of heap cells. These predicates allow reasoning about both acyclic and cyclic lists uniformly with equal ease. The crucial insight behind our approach is that a circular list invariably contains a *distinguished head cell* that provides a handle on the list. This observation suggests a programming methodology that requires the heap of the program at each step to be *well-founded*, i.e., for any field f in the program, every sequence $u.f, u.f.f, \ldots$ contains at least one head cell. We believe that our methodology captures the most common idiom of programming with linked data structures. We enforce our methodology by automatically instrumenting the program with updates to two auxiliary variables representing these predicates and adding assertions in terms of these auxiliary variables.

To prove program properties and the instrumented assertions, we provide a first-order axiomatization of our two predicates. We also introduce a novel induction principle made possible by the well-foundedness of the heap. We use our induction principle to derive from two basic axioms a small set of additional first-order axioms that are useful for proving the correctness of several programs.

We have implemented our method in a tool and used it to verify the correctness of a variety of nontrivial programs manipulating both acyclic and cyclic singly-linked lists and doubly-linked lists. We also demonstrate the use of indexed predicate abstraction to automatically synthesize loop invariants for these examples.

Keywords: Software verification, heap-allocated data structures, theorem proving

Full Paper:
http://research.microsoft.com/~qadeer/docs/popl06.lists.pdf

A Complete Abstract Interpretation Framework for Coverability Properties of Well Structured Transition Systems

Jean-Francois Raskin (Université Libre de Bruxelles, B)

We present an abstract interpretation based approach to solve the coverability problem of well-structured transition systems. Our approach distinguishes from other attempts in that (1) we solve this problem for the whole class of well-structured transition systems using a forward algorithm. So, our algorithm has to deal with possibly infinite downward closed sets. (2) Whereas other approaches have a non generic representation for downward closed sets of states, which turns out to be hard to devise in practice, we introduce a generic representation requiring no additional effort of implementation.

Keywords: Reachability analysis, infinite state systems, well structured transition systems, Petri nets

Full Paper:
http://www.ulb.ac.be/di/ssd/cfv/publications.html

See also: Pierre Ganty, Jean-Francois Raskin, and Laurent Van Begin. A Complete Abstract Interpretation Framework for Coverability Properties of WSTS. In Proc. of VMCAI'06, LNCS 3855, pp. 49–64, Springer-Verlag, 2006.

WYSINWYX: What You See Is Not What You eXecute

Thomas Reps (University of Wisconsin - Madison, USA)

What You See Is Not What You eXecute: computers do not execute source-code programs; they execute machine-code programs that are generated from source code. Not only can the WYSINWYX phenomenon create a mismatch between what a programmer intends and what is actually executed by the processor, it can cause analyses that are performed on source code (or intermediate representations constructed from source code) to fail to detect bugs and security vulnerabilities. This issue arises regardless of whether one's favorite approach to assuring that programs behave as desired is based on theorem proving, model checking, or abstract interpretation.

To address the WYSINWYX problem, we have developed algorithms – based on model checking and abstract interpretation – to recover information from

stripped executables about the memory-access operations that the program performs. These algorithms are used in the CodeSurfer/x86 tool to construct intermediate representations that are used for browsing, inspecting, and analyzing stripped x86 executables. In addition to providing information about memory-access operations, CodeSurfer/x86 can be used to extract models of executables in the form of weighted pushdown systems.

Keywords: Static analysis, low-level code, memory-access analysis

Joint work of: Reps, Thomas; Balakrishnan, Gogul; Lim, Junghee; Teitelbaum, Tim

Applications of the First-Order Theory of Boolean Algebras of Sets with Linear Cardinality Constraints

Peter Revesz (University of Nebraska, USA)

The first-order theory of Boolean algebras of sets with linear cardinality constraints was recently shown to be decidable and to admit existential and universal quantifier elimination. This logic allows the expression of interesting conditions on the size of set or pointer linked data structures. We consider those cases under which these expressions can be shown to imply invariants at fixed locations of programs that use these types of data structures.

Keywords: Linear cardinality constraints, quantifier elimination, decidability, program invariants

Termination Proofs for Systems Code

Andrei Rybalchenko (EPFL - Lausanne, CH)

Termination of (operating) system components is crucial for the successful system behavior. We present the main building blocks of our tool for the automated termination proofs of C code fragments (containing nested loops, gotos, mutually recursive function calls, pointer aliasing etc). These blocks are transition invariants, transition predicate abstraction, and abstraction refinement for termination. Transition invariants are auxiliary assertions for proving termination that can be computed by least fixed point iteration. We apply transition predicate abstraction to make such iteration effective. We discover the relevant transition predicates from counterexamples. Finally, we describe how we extend the SLAM software model checker to automatically compute transition invariants, which gives us a tool for proving termination of C programs.

Keywords: Termination, refinement, binary reachability

Joint work of: Cook, Byron; Podelski, Andreas; Rybalchenko, Andrey

Verification of Nondeterministic Channel Systems with Probabilistic Message Losses

Philippe Schnoebelen (ENS - Cachan, F)

We introduce NPLCS's, a model for asynchronous communication protocols where messages can be lost according to probabilistic laws, and investigate the decidability of qualitative linear-time verification problems. Beside its application domain, the specificity of this research is that the operational semantics for our model are infinite-state Markovian decision processes.

Keywords: Probabilistic and nondeterministic infinite-state systems, lossy channel systems

Joint work of: Baier, Christel; Bertrand, Nathalie; Schnoebelen, Philippe

Reachability Analysis of Multithreaded Software with Asynchronous Communication

Stefan Schwoon (Universität Stuttgart, D)

We introduce asynchronous dynamic pushdown networks (ADPN), a new model for multithreaded programs in which pushdown systems communicate via shared memory. ADPN generalizes both CPS (concurrent pushdown systems) and DPN (dynamic pushdown networks). We show that ADPN exhibit several advantages as a program model. Since the reachability problem for ADPN is undecidable even in the case without dynamic creation of processes, we address the bounded reachability problem, which considers only those computation sequences where the (index of the) thread accessing the shared memory is changed at most a fixed given number of times. We provide efficient algorithms for both forward and backward reachability analysis. The algorithms are based on automata techniques for symbolic representation of sets of configurations.

This talk is based on joint work with Ahmed Bouajjani, Javier Esparza, and Jan Strejček that appeared in FSTTCS 2005.

Keywords: Model checking, pushdown systems, concurrency

Joint work of: Bouajjani, Ahmed; Esparza, Javier; Schwoon, Stefan; Strejček, Jan

Full Paper:
http://www.fmi.uni-stuttgart.de/szs/publications/info/schwoosn.BESS05b.shtml

See also: Proceedings of FSTTCS 2005 and Technical Report 2005/06 (Universität Stuttgart, Fakultät Informatik, Elektrotechnik und Informationstechnik).

Accelerating Abstraction Refinement by Summarizing Loops

Nassim Seghir (MPI für Informatik - Saarbrücken, D)

Predicate based abstraction refinement is a technique that builds an abstract system in the domain of formulas constructed over a set of predicates using logical operators. The refinement process consists in finding new predicates and adding them to the predicate set when the abstraction is too coarse.

Most of the approaches proposed so far use the weakest precondition to infer new predicates. Such approaches may diverge in the presence of loops. Moreover, predicates that can be inferred are limited to formulas obtainable via syntactic substitutions over condition expressions present in the program.

This work proposes a technique based on summarizing loops to avoid the divergence of the refinement process. A loop summary is an over-approximation of the relation between program variables before and after the execution of the loop. Using our method, we were able to verify safety properties that neither methods based on weakest precondition nor interpolation methods were able to verify.

Keywords: Predicate abstraction, abstraction refinement, loop summaries

Exact XML Type Checking in Polynomial Time

Helmut Seidl (TU München, D)

Macro tree transducers (mtts) have been shown to be an expressive formalism for reasoning about XSLT-like document transformations. Here, we are concerned with techniques for precise type-checking. *Inverse type inference*, though, is known to have exponential worst-case complexity already for top-down transformations without parameters. Instead, we propose *forward* inference based on precise characterizations of output languages of transducers. Using this approach, we exhibit that type-checking for call-by-value mtts with few parameters is polynomial — given that the output type is specified by a deterministic automaton and that every input node is visited by the mtt only constantly often. Based on *context-free* tree grammars, we also propose a fast approximative type-checking algorithm which works for general mtts. Finally, we extend our approach to *macro forest transducers* which additionally support concatenation as a builtin output operation.

Keywords: Macro tree transducers, document transformation, XSLT, inverse type inference

Joint work of: Seidl, Helmut; Perst, Thomas; Maneth, Sebastian

Constraint-based Static Analysis of Programs

Henny Sipma (Stanford University, USA)

We present a constraint-based approach to static analysis of programs. The approach involves the following steps: (1) Fix a template property, that is, a property of a certain shape with unknown coefficients; (2) Provide the conditions for the property to hold; (3) Encode the conditions as a system of constraints using rules that allow computing consequences; (4) Solve the constraints; (5) Substitute the solutions in the template property; all concrete properties thus obtained are properties of the system of the desired kind. We demonstrate how this approach can be used to generate linear inequality invariants, linear ranking functions, and polynomial equality invariants.

The advantage of this approach is that it does not require widening.

Controlling the complexity of the constraint system can be achieved by strengthening the conditions on the property or constraining the template property itself. Other advantages are that it can be used to generate any property that can be encoded as a system of constraints, and that it can generate properties in any domain that allows computation of consequences. A disadvantage of the approach is that the resulting system of constraints can have high complexity. Constraint solving, however, is a very active field of research, and any advances in this area directly improve scalability and precision of constraint-based program analysis.

Keywords: Static program analysis, constraint-based analysis, invariant generation, termination analysis

Joint work of: Sankaranarayanan, Sriram; Colon, Michael; Sipma, Henny; Bradley, Aaron; Manna, Zohar

Verifying Concurrent Message-Passing C Programs with Recursive Calls

Tayssir Touili (LIAFA - Université Paris VII, F)

We consider the model-checking problem for C programs with (1) data ranging over very large domains, (2) (recursive) procedure calls, and (3) concurrent parallel components that communicate via synchronizing actions. We model such programs using communicating pushdown systems, and reduce the reachability problem for this model to deciding the emptiness of the intersection of two context-free languages L1 and L2. We tackle this undecidable problem using a CounterExample Guided Abstraction Refinement (CEGAR) scheme. We implemented our technique in the model checker MAGIC and found a previously unknown bug in a version of a Windows NT Bluetooth driver.

Joint work of: Chaki, S.; Clarke, E.; Kidd, N.; Reps, T.; Touili, T.

Environment Abstraction for Parameterized Systems

Helmut Veith (TU München, D)

Many aspects of computer systems are naturally modeled as parameterized systems which renders their automatic verification difficult. In well-known examples such as cache coherence protocols and mutual exclusion protocols, the unbounded parameter is the number of concurrent processes which run the same distributed algorithm. We introduce environment abstraction as a tool for the verification of such concurrent parameterized systems. Environment abstraction enriches predicate abstraction by ideas from counter abstraction; it enables us to reduce concurrent parameterized systems with unbounded variables to precise abstract finite state transition systems which can be verified by a finite state model checker. We demonstrate the feasibility of our approach by verifying the safety and liveness properties of Lamport's bakery algorithm and Szymanski's mutual exclusion algorithm.

Keywords: Model checking, software verification, predicate abstraction, parameterized systems

Joint work of: Veith, Helmut; Clarke, Edmund; Talupur, Muralidhar

Abstract Regular Model Checking and Its Application to Verification of Programs with Dynamic Data Structures

Tomas Vojnar (Techn. University - Brno, CZ)

The talk presents the abstract regular model checking framework combining regular model checking, which is a generic technique for verifying a wide range of infinite-state and parameterised systems, with the approach of automated abstraction with counterexample-guided refinement. In regular model checking, finite-state automata are used for finitely representing infinite, but regular sets of reachable configurations. We present several different abstraction schemas applicable over the finite-state automata in order to make regular model checking terminate as often as possible (in general, termination cannot be guaranteed as we are dealing with undecidable problems) and also to cope with the state explosion problem related to the size of the automata encountered. We further present a way how abstract regular model checking can be used for verifying programs with dynamic singly-linked dynamic structures, which involves a way of encoding memory configurations of such programs as words and program statements as word transducers. We also comment on possible extensions of abstract regular model checking from words to trees and on ways of using this new framework for verifying programs with more general classes of dynamic linked data structures.

Keywords: Regular model checking, abstraction, program verification, dynamic-linked data structures, pointers

Joint work of: Bouajjani, Ahmed; Habermehl, Peter; Moro, Pierre; Rogalewicz, Adam; Vojnar, Tomas

Full Paper:
http://www.fit.vutbr.cz/~vojnar/Publications/bhmv-lists-05.ps.gz

See also: A. Bouajjani, P. Habermehl, P. Moro, and T. Vojnar. Verifying Programs with Dynamic 1-Selector-Linked Structures in Regular Model Checking. In Proc. of TACAS'05, LNCS 3440, pp. 13–29, Springer-Verlag, 2005.

On the Decidability of Metric Temporal Logic

James Worrell (Oxford University, GB)

Metric Temporal Logic (MTL) is a widely-studied real-time extension of Linear Temporal Logic. In this talk we consider a fragment of MTL, called Safety MTL, capable of expressing properties such as invariance and time-bounded response. Our main result is that the model-checking and satisfiability problems for Safety MTL are decidable. These are the first positive decidability result for MTL over dense-time omega-words that do not involve restricting the precision of the timing constraints, or the granularity of the semantics; the proof heavily uses the techniques of infinite-state verification.

Keywords: Metric temporal logic, model-checking, satisfiability

A Logic of Reachable Patterns in Linked Data-Structures

Greta Yorsh (Tel Aviv University, IL)

We define a new decidable logic for expressing and checking invariants of programs that manipulate dynamically-allocated objects via pointers and destructive pointer updates. The main feature of this logic is the ability to limit the neighborhood of a node that is reachable via a regular expression from a designated node. The logic is closed under boolean operations (entailment, negation) and has a finite model property. The key technical result is the proof of decidability. We show how to express precondition, postconditions, and loop invariants for some interesting programs. It is also possible to express properties such as disjointness of data-structures, and low-level heap mutations. Moreover, our logic can express properties of arbitrary data-structures and of an arbitrary number of pointer fields. The latter provides a way to naturally specify postconditions that relate the fields on entry to a procedure to the fields on exit. Therefore, it is possible to use the logic to automatically prove partial correctness of programs performing low-level heap mutations.

Keywords: Heaps, logic, decidability

Joint work of: Yorsh, Greta; Rabinovich, Alexander; Sagiv, Mooly; Meyer, Antoine; Bouajjani, Ahmed

Full Paper:
http://www.cs.tau.ac.il/~gretay/papers/LRP.abs.html