Information Societies Technology (IST) Programme

Future and Emerging Technologies - Open Domain

Project acronym: Advance

Project full title:

Advanced Validation Techniques
for Telecommunication Protocols

Contract no: IST-1999-29082

Deliverable 1

Methods and Tools for Extended Automata

Year 1 : October 2000 - September 2001

Date of preparation: October 2001

Contributors: Paris, Liège, Edinburgh, Uppsala, Grenoble

1 Introduction

This document summarizes the research results obtained by the Advance partners during the first 12 months of the project, related to developing methods and tools for analyzing systems modeled by extended automata. The results are grouped into different categories.

2 Analyzing new types of extended automata

2.1 Systems with procedures

The Edinburgh site has worked on the verification of systems with (possible recursive and mutually recursive) procedures [ES01c, EKS01]. They can be seen as automata extended with one stack. In the presence of recursion the size of the stack may grow arbitrarily, and so the system may exhibit an infinite state space. In the absence of recursion, procedure calls can in principle be replaced by the body of the procedure (inlining), yielding a flat systems; however, this may make the size of the system blow-up.

In the protocols area this work finds two applications:

- Large protocols, like some of the studied by the Grenoble site, make extensive use of procedures (usually non recursive).
- Functional languages like Erlang, used by Ericsson to implement protocols, are inherently recursive. Any model-checking approach requires modeling the stack.

In [ES01c], one presents a model-checker for Boolean programs with (possibly recursive) procedures and the temporal logic LTL. The checker is guaranteed to terminate even for (usually faulty) programs in which the depth of the recursion is not bounded. The algorithm uses automata to finitely represent possibly infinite sets of stack contents and BDDs to compactly represent finite sets of values of Boolean variables. The checker is illustrated with some examples, and compared with the Bebop tool of Ball and Rajamani.

In [EKS01], one examines an extension of the model-checking problem for LTL, namely model-checking with regular valuations. The problem is solved via two different techniques, with an eye on efficiency — both techniques can be shown to be essentially optimal. These methods are applicable to problems in different areas, e.g., data flow analysis, analysis of systems with checkpoints, etc., and provide a general, unifying and efficient framework for solving these problems.

The problem of analyzing programs based on recursive procedures is also addressed in [BHM01]. This paper introduces a more refined model called BPA(Z) that can model not only recursive dependencies, but also the passing of an integer parameter to subroutines. Moreover, this parameter can be tested against conditions expressible in Presburger-arithmetic. This new and more expressive model can still be analyzed automatically. The authors define Z-input 1-CM, a new class of one-counter machines that take integer numbers as input, to describe sets of configurations of BPA(Z). It is shown that the $Post^*$ (the set of successors) of a set of BPA(Z)-configurations described by a Z-input 1-CM can be effectively constructed. The Pre^* (set of predecessors) of a regular set can be effectively constructed as well. However, the Pre^* of a set described by a Z-input 1-CM cannot be represented by a Z-input 1-CM in general and has an undecidable membership problem. The paper then develops a new temporal logic based on reversal-bounded counter machines that can be used to describe properties of BPA(Z), and shows that the model-checking problem is decidable for that logic.

2.2 Systems modeled by timed Petri nets

In [AN01], one addresses the problem of analyzing systems modeled by (unbounded) *Timed Petri Nets (TPNs)* where each token is equipped with a real-valued clock representing the "age" of the token. Each arc in the net is provided with a subinterval of the natural numbers, restricting the ages of the tokens traveling the arc. The main idea consists of applying methodology which has been developed earlier based on the theory of *better quasi orderings (BQOs)*, to derive an efficient constraint system for automatic verification of safety properties for TPNs. A prototype based on this method has been implemented and applied for verifying a parametrized version of Fischer's protocol.

2.3 Systems with real variables

Previous work has shown that the combined linear arithmetic of real and integer variables can be dealt with by using finite automata on infinite words. Unfortunately, this involves some difficult and delicate to implement algorithms. In [BJW01], it is shown, using topological arguments, that only a restricted class of automata on infinite words are necessary for handling real and integer linear arithmetic. This allows the use of substantially simpler algorithms and opens the path to the implementation of a usable system for handling this combined theory. The results of this paper have indeed been implemented and made available in the tool LASH [LAS]. The application of these results to the symbolic state-space exploration of timed systems is the subject of ongoing work.

2.4 Fair parametric extended automata

In [ABLS01], one addresses the problem of verifying safety and liveness properties of infinite-state systems, using symbolic reachability analysis. The models that are considered are *fair parametric extended automata*, i.e., counter automata with parametric guards, supplied with fairness conditions on their transitions. In previous work, it has been shown that symbolic reachability analysis using acceleration techniques can be used to generate finite abstractions (symbolic graphs) of the original infinite-state model. In this work, it is shown that this analysis can be also used to introduce fairness conditions on the generated abstract model allowing to model-check liveness properties. It is shown first how to translate faithfully the fairness conditions of the infinite-state original model to conditions on the generated finite symbolic graph. Then, it is shown that one can also synthesize automatically new fairness conditions allowing to eliminate infinite paths in the symbolic graph which do not correspond to valid behaviors in the original model. These infinite paths correspond to abstractions of boundedly iterable (nested) loops. Techniques are given for deciding this bounded iterability for a class of components in the symbolic graph. The application of these techniques has been illustrated with nontrivial examples.

2.5 Systems with FIFO channels

In automated verification of protocols, one source of complications is that channels may have unbounded capacity, in which case a naive model of the protocol is no longer finite state. Symbolic techniques have therefore been developed for representing the contents of unbounded channels. In [AJ01a], some of these techniques are surveyed and applied to a simple leader election protocol. The protocols that are considered have entities modeled as finite state machines which communicate by sending messages from a finite alphabet over unbounded channels; this is a framework for which many techniques have been developed. A more general model is also considered, in which messages may belong to an unbounded domain of values which may be compared according to a total ordering relation: The motivation is to study protocols with timestamps or priorities. The authors show how techniques from the previous setting can be extended to this more general model, but also show that reachability quickly becomes undecidable if channels preserve the ordering between messages.

2.6 Effective lossy queue languages

Although the set of reachable states of a lossy channel system (LCS) is regular, it is well-known that this set cannot be constructed effectively. In [ABB01], one characterizes significant classes of LCS for which the set of reachable states can be computed. Furthermore, it is shown that, for slight generalizations of these classes, computability can no longer be achieved. To carry out this study, one defines *rewriting systems* which capture the behavior of LCS, in the sense that (i) they have a FIFO-like semantics and (ii) their languages are downward closed with respect to the substring relation. The main result of the paper shows that, for context-free rewriting systems, the corresponding language can be computed. An interesting consequence of this result is that it yields a characterization of classes of meta-transitions whose post-images can be effectively constructed. These meta-transitions consist of sets of nested loops in the control graph of the system, in contrast to previous works on meta-transitions in which only single loops are considered. Essentially the same proof technique we use to show the result mentioned above allows also to establish a result in the theory of $0L$-systems, i.e., context-free parallel rewriting systems. It is proven that the downward closure of the language generated by any $0L$-system is effectively regular.

3 General results on methodology of verification

In [AJ01b], the authors present a general model of infinite-state systems, and describe a standard algorithm for reachability analysis of such systems. The contribution of this paper consists in finding conditions under which the algorithm can be fully automated. The idea consists of performing backward reachability analysis. Using an iterative procedure, one generates successively larger approximations of the set of all states from which a given final state is reachable. The classes of systems that are considered are such that these approximations are *well quasi-ordered*, implying that the iterative procedure always terminates. Starting from these general termination conditions, one derives several computations models for which reachability is decidable. Many of these models are extensions of those existing in the literature. Using a well-known reduction from safety properties to reachability properties, the algorithm introduced in the paper can be used to decide large classes of safety properties for infinite-state systems. A motivation for this approach is the long-term desire to build general tools for verification of infinite-state systems, which implies that one should employ principles applicable across a rather wide range of such systems.

In [Bou01], a general approach is introduced for verifying algorithmically infinite-state systems. This approach is based on the use of languages (not necessarily regular) as symbolic representations of infinite sets of configurations, and the use of rewriting systems as models for these systems. The verification problem is then reduced to the problem of computing closures of languages under rewriting systems. The paper presents in a uniform manner several applications of this approach to different kinds of infinite-state systems: Pushdown systems, FIFO-channel systems, and parametrized networks of identical processes.

In [AB01], the computational power of several models of dynamical systems is investigated under infinitesimal perturbations of their dynamics. This study considers models for discrete and continuous time dynamical systems: Turing machines, Piecewise affine maps, Linear hybrid automata and Piecewise constant derivative systems (a simple model of hybrid systems). One associates with each of these models a notion of perturbed dynamics by a small ε (w.r.t. to a suitable metrics), and define the *perturbed reachability relation* as the intersection of all reachability relations obtained by ε-perturbations, for all possible values of ε. It is shown that for the four kinds of models we consider, the perturbed reachability relation is co-recursively enumerable, and that any co-r.e. relation can be defined as the perturbed reachability relation of such models. A corollary of this result is that systems that are robust, i.e., their reachability relation is stable under infinitesimal perturbation, are decidable.

4 New tools and techniques

4.1 Efficient state-space exploration of Petri nets

The Edinburgh site is working on the development of the unfolding technique for state space exploration. In this technique a concurrent system, represented as a Petri net, is unfolded into a usually larger acyclic net, equivalent to the first one with respect to a suitable equivalence notion. Different notions are used, depending on the type of property considered. In [ES01b], the equivalence notion is essentially "to have the same reachable markings". Four solutions to the reachability problem are studied and compared experimentally. In [ES01a], a technique is presented for automatically simplifying the system before constructing the unfolding. This paper profits from reduction techniques for Petri nets well established in the literature, and shows that in the present setting they can be sharpened, leading to larger reductions.

4.2 Incremental Construction of Abstractions

The paper [LBOB01] presents a methodology for constructing abstractions and refining them by analyzing counter-examples. The paper also presents a uniform verification method that combines abstraction, model-checking and deductive verification in a novel way. In particular, it allows and shows how to use the set of reachable states of the abstract system in a deductive proof even when the abstract model does not satisfy the specification and when it simulates the concrete system with respect to a weaker simulation notion than Milner's.

4.3 Slicing techniques

Conformance testing is still the main industrial validation technique for telecommunication protocols. The automatic construction of test cases based on the model approach is hindered by the state explosion problem. In [BFG01], a method is described that minimizes the impact of state explosion by reconsidering the test case generation at a higher level and by taking advantage of some static analysis techniques, in particular the slicing techniques. The specification is simplified by pipelining a set of three modules, each one implementing a different slicing technique.

4.4 The TReX tool

TReX [ABS01] is a tool for the automatic analysis of infinite-state systems modeled by means of extended automata. The models that can be analyzed are automata supplied with (unbounded) clocks, counters, and lossy FIFO-channels. Moreover, transitions may have guards where clocks and counters are compared with parametric bounds. The tool uses symbolic representations for infinite-state domains (regular expressions, parametric DBMs, arithmetical constraints), and forward/backward reachability analysis procedures, enhanced with acceleration techniques in order to help termination. TReX allows on-the-fly verification of safety properties, the generation of the reachability set which can be used, e.g., for parameter synthesis and for invariant generation, and the construction of a finite abstractions. The tool is interfaced with the IF environment allowing to use SDL as a specification language, and to interact with abstraction tools and finite model-checkers.

4.5 Counting the elements of finite-state set representations

The *Number Decision Diagram (NDD)* is a finite-state representation system for sets of integer vectors. In particular, NDDs can be used for representing the sets of solutions of arbitrary Presburger formulas, or the set of reachable states of some systems using unbounded integer variables. In [BL01], one addresses the problem of counting the number of distinct elements in a set of vectors represented as an NDD. An algorithm is described for performing an exact count without enumerating explicitly the vectors, which makes it capable of handling very large sets. As an auxiliary result, an efficient projection method is also developed, which allows to construct efficiently NDDs from quantified formulas, and thus makes it possible to apply the counting technique to sets specified by formulas. The algorithms introduced in this work have been implemented in the verification tool LASH [LAS], and applied successfully to various counting problems.

5 Applying extended automata analysis methods

5.1 The IF environment

In [BGM01], the Grenoble group summarizes its experience with the language IF, and the IF tool-set which is an open validation environment for distributed software systems. Indeed, face to the increasing complexity of such systems, none of the existing tools can cover by itself the whole validation process. The IF environment was built upon an expressive intermediate language and allows to connect several validation tools, providing most of the advanced techniques currently available. The results obtained on several large case-studies, including telecommunication protocols and embedded software systems, confirm the practical interest of this approach.

In [VER01] a new version of the language IF is defined. A compiler and a simulator have been implemented for it. The new version of the IF language has been used for specifying and simulating the PGM protocol [VR01].

5.2 The MASCARA case study

In [GJ01], one describes an experiment in verifying a real industrial protocol for wireless ATM, called MASCARA. Several tools have been used. SDL has been chosen as the specification language and the commercial tool ObjectGEODE has been used for creating, maintaining and modifying SDL descriptions. The IF tool-set has been used for generation, minimization and comparison of system models and verification of expected properties. All specification and verification tools are connected via the IF language, which has been defined as an intermediate representation for timed asynchronous systems as well as an open validation environment. Due to the complexity of the protocol, static analysis techniques, such as live variable analysis and program slicing, were the key to the success of this verification experiment. The results obtained give some hints concerning a methodology for the formal verification of real systems.

References

[AB01] E. Asarin and A. Bouajjani. Perturbed Turing Machines and Hybrid Systems. In *Proc. 6th IEEE Symp. on Logic in Computer Science (LICS'01)*, Boston (MA), USA, June 2001. IEEE.

[ABB01] P. Abdulla, L. Boasson, and A. Bouajjani. Effective Lossy Queue Languages. In *Proc. 28th Intern. Coll. on Automata, Languages and Programming (ICALP'01)*, Crete, Greece, July 2001. Lecture Notes in Computer Science 2076, Springer-Verlag.

[ABLS01] A. Annichini, A. Bouajjani, Y. Lakhnech, and M. Sighireanu. Analysis of Fair Extended Automata. In *Proc. 8th Intern. Static Analysis Symposium (SAS'01)*, Paris, France, July 2001. Lecture Notes in Computer Science 2126, Springer-Verlag.

[ABS01] A. Annichini, A. Bouajjani, and M. Sighireanu. TReX: A Tool for Reachability Analysis of Complex Systems. In *Proc. 13th Intern. Conf. on Computer Aided Verification (CAV'01)*, Paris, France, July 2001. Lecture Notes in Computer Science 2102, Springer-Verlag.

[AJ01a] Parosh Aziz Abdulla and Bengt Jonsson. Channel representations in protocol verification. In *Proc. CONCUR 2001, 12th Int. Conf. on Concurrency Theory*, volume 2154 of *Lecture Notes in Computer Science*, pages 1–15, 2001.

[AJ01b] Parosh Aziz Abdulla and Bengt Jonsson. Ensuring completeness of symbolic verification methods for infinite-state systems. *Theoretical Computer Science*, 256:145–167, 2001.

[AN01] Parosh Aziz Abdulla and Aletta Nylén. Timed Petri nets and BQOs. In *Proc. ICATPN'2001: 22nd Int. Conf. on application and theory of Petri nets*, volume 2075 of *Lecture Notes in Computer Science*, pages 53 –70, 2001.

[BFG01] M. Bozga, J-Cl. Fernandez, and L. Ghirvu. Using static analysis to improve automated test generation. *Software Tools for Tehnology Transfer*, 2001. submitted.

[BGM01] M. Bozga, S. Graf, and L. Mounier. Automated validation of distributed software using the IF environment. In *Proceedings of Software Model Checking Workshop*, Electronic Notes in Theoretical Computer Science, Paris, France, July 2001.

[BHM01] A. Bouajjani, P. Habermehl, and R. Mayr. Automatic Verification of Recursive Procedures with one Integer Parameter. In *Proc. 26th Intern. Symp. on Mathematical Foundations of Computer Science (MFCS 2001)*, Marianske Lazne, Czech Republic, August 2001. Lecture Notes in Computer Science, Springer-Verlag.

[BJW01] B. Boigelot, S. Jodogne, and P. Wolper. On the use of weak automata for deciding linear arithmetic with integer and real variables. In *Proceedings of IJCAR '01*, number 2083 in LNAI, pages 611–625, 2001.

[BL01] B. Boigelot and L. Latour. Counting the solutions of Presburger equations without enumerating them. In *Proceedings of CIAA '01*, LNCS, 2001. To appear.

[Bou01] A. Bouajjani. Languages, Rewriting systems, and Verification of Infinte-State Systems. In *Proc. 28th Intern. Coll. on Automata, Languages and Programming (ICALP'01)*, Crete, Greece, July 2001. Lecture Notes in Computer Science 2076, Springer-Verlag. invited paper.

[EKS01] J. Esparza, A. Kucera, and Stefan Schwoon. Model-checking LTL with regular valuations for pushdown systems. In *Proceedings of TACS '01*, LNCS, 2001. To appear.

[ES01a] J. Esparza and C. Schröter. Net reductions for LTL model-checking. In *Proceedings of CHARME '01*, 2001. To appear.

[ES01b] J. Esparza and C. Schröter. Unfolding based algorithms for the reachability problem. *Fundamenta Informaticae*, (34), 2001. To appear.

[ES01c] J. Esparza and S. Schwoon. A BDD-based model checker for recursive programs. In *Proceedings of CAV '01*, number 2102 in LNCS, pages 324–336, 2001.

[GJ01] S. Graf and G. Jia. Verification experiments on the Mascara protocol. In *Proceedings of the SPIN'01 Workshop*, 2001. to appear.

[LAS] The Liège Automata-based Symbolic Handler (LASH). Available at `http://www.montefiore.ulg.ac.be/~boigelot/research/lash/`.

[LBOB01] Y. Lakhnech, S. Bensalem, S. Owre, and S. Berezin. Incremental verification by abstraction. In *TACAS 2001*, volume 2031 of *LNCS*, 2001.

[VER01] VERIMAG. The Language IF-2.0. Technical report, University of Grenoble, September 2001. Internal tech. report. Verimag. Deliverable 3 of project Advance.

[VR01] VERIMAG and France-Telecom R&D. Modeling PGM with IF-2.0. Technical report, University of Grenoble, September 2001. Internal tech. report. Verimag.