Complexity of Equivalence Checking Problems

Zdeněk Sawa

Dept. of Computer Science, FEI,
Technical University of Ostrava,
17. listopadu 15,
CZ-708 33 Ostrava, Czech Republic
Zdenek.Sawa@vsb.cz

Abstract. The article summarizes the results of the author in the area of automated verification of systems and concurrency theory. These results are concerning the computational complexity of equivalence checking problems. The main aim of the paper is to present intuitive non-formal overview of these results.

Keywords: verification, equivalence checking, computational complexity

1 Introduction

One of the main problems in a design of any complicated system is to ensure the correctness of the design. Simulation and testing are the standard techniques used for this purpose. While they are very useful in the early stages of development and allow to discover many bugs in the system, they have a serious drawback that one can never be sure that there are not some other more subtle bugs in the system. This problem becomes more apparent in the design of systems that are composed of many components that run concurrently and may interact with each other in complicated ways. Examples of such systems are operating systems, network communication protocols, microprocessors, distributed and parallel algorithms, and traffic control systems.

It becomes obvious that some *formal methods* are necessary in the design of such systems. While simulation and testing explore *some* of the possible behaviours of the system, the formal methods usually allow to verify *all* possible behaviours of the system.

The formal methods provide the designer with the necessary mathematical tools that can be used in the construction of rigorous mathematical proofs of the correctness of the system. The construction of such proof can be done either by hand or may be automated by use of some sort of verification software tools. The latter approach is very attractive since the construction of proofs by hand is very tedious and error prone.

The design and analysis of algorithms that can be used in such verification tools is one the main research topics in the area of verification and concurrency theory. These areas have their origins in more traditional areas of computer science like automata theory, theory of formal languages, semantics of programming languages, and Petri nets.

Formalisms based on relations between input and output are usually not appropriate for modeling of systems with ongoing behaviour, so such systems are usually modeled as labelled transition systems. This means that a system has some set of possible states and a set of possible transitions between these states. The set of states can be infinite. The transitions are often labelled with actions from some finite set of actions. A labelled transition system can be viewed as an abstraction of the behaviour of the systems, as the semantics of systems.

There are many possible ways how labelled transition systems can be produced and described. The most often used possibilities include:

- different types of automata, like finite state automata, pushdown automata, counter machines, and so on,
- process algebras where a system in described by a set of algebraic equations, examples of process algebras are CCS, CSP, and π-calculus,
- Petri nets.

There are two main types of problems studied in the concurrency theory — model checking and equivalence checking:

- *Model checking* is a problem where we have given some system (resp. description of a system) and some desired property of the system expressed as a formula in some temporal logic, and we are asked if the system satisfies the given property. There are many different types of temporal and modal logics such as LTL, CTL, CTL*, μ-calculus, and different fragments of these logics.
- *Equivalence checking* is a problem where we have given (descriptions of) two systems and we are asked whether these systems are equivalent with respect to some notion of equivalence. There are many possible ways how equivalence of systems can be defined, and many different types of equivalence was proposed in literature. These equivalences were organized into the linear-time / branching time spectrum (Fig. 1), see [8]. In fact, as equivalence checking problems we can consider also problems where we decide some general relations between systems, not necessarily equivalences.

  One important application of equivalence checking is a situation where one of the systems represents a specification and the other an implementation and we want to check whether their behaviours are the same.

There are also other types of problems, for example one promising approach is to try produce automatically a system that satisfies some given specification.

While most of these problems may be solved automatically for finite-state systems, the problems become undecidable when we consider them in full generality. The interesting question is where exactly lies the borderline between decidable and undecidable verification problems and what is the computational

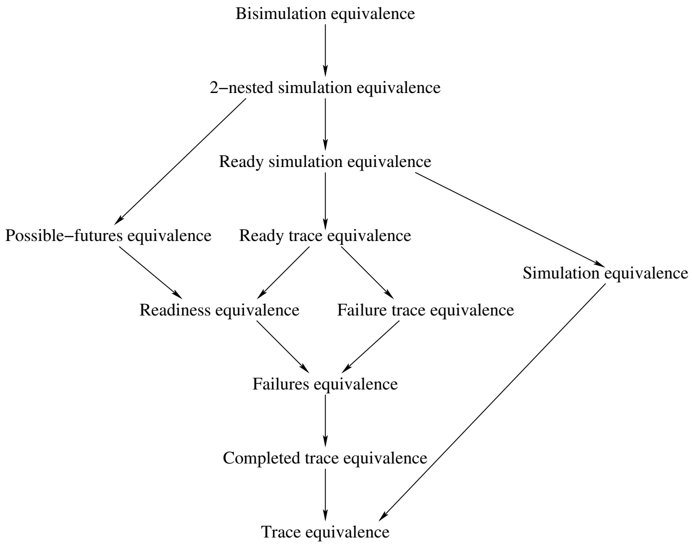

Fig. 1. Linear time / branching time spectrum

complexity of decidable problems, i.e., the question where are the limits of automated verification.

One of the most serious obstacles in the design of efficient verification algorithms is the phenomenon known as a 'state explosion'. This problem appears when we have a system composed of many components. These components may have small state spaces, but the state space of the whole system can be exponentially larger than descriptions of its components. Unfortunately it shows up that the state explosion is unavoidable in many cases and that algorithms solving such problems require exponential time.

For more information about model checking and equivalence checking, see for example [1] or [9].

The rest of the paper contains short informal descriptions of results obtained by the author in the area.

2 Own results

2.1 *EXPTIME*-hardness of Equivalence Checking of Non-Flat Systems

Let us consider systems composed by parallel composition from explicitly given finite-state systems where some actions may be hidden in the resulting system, i.e., they can be replaced by 'invisible' τ actions. Such systems are an example of so called 'non-flat' systems. 'Flat' systems are systems with explicitly given sets of states and transitions, on the other hand, 'non-flat' systems are build from flat systems and have some internal structure. The equivalence checking problem for the above mentioned non-flat systems was considered by A. Rabinovich in [5]. He proved there that the problem is *PSPACE*-hard for any relation that lies between bisimulation equivalence and trace preorder. He mentioned that the problem is *EXPTIME*-complete for bisimulation equivalence, and he formulated a conjecture that the problem is in fact *EXPTIME*-hard for any relation between bisimulation equivalence and trace preorder.

The Rabinovich's conjecture was approved by the author in [7]. This *EXPTIME*-hardness lower bound result can be easily extended to other types of non-flat systems, for example to 1-safe Petri nets. A new auxiliary model called reactive linear bounded automata (RLBA) was also introduced in this paper. The use of RLBA considerably simplified the proof and allows very simple generalization of the result to other types of non-flat systems.

2.2 *PTIME*-hardness of Equivalence Checking of Flat Systems

It was shown by the author and his supervisor in [6] that equivalence checking is *PTIME*-hard in case of flat systems for any relation between bisimulation equivalence and trace preorder. The result implies that equivalence checking can not be efficiently parallelized unless $NC = PTIME$ which is generally conjectured to be very unlikely.

The paper [7] contains alternative and simpler proof of the same result.

2.3 A Method for Proving *DP*-hardness of Verification Problems Concerning One-Counter Automata

One-counter automata are finite-state automata equipped with a counter. The the counter may be incremented by 1, decremented by 1, and tested if it is equal to zero. Such automata can be viewed as a special case of pushdown automata where the stack alphabet contains only one symbol (and a special marker of the bottom). One-counter automata are one of the simplest examples of infinite state systems.

One-counter nets form a subclass of one-counter automata where we can not test the zero value of the counter, only non-zero. Such systems are equivalent to Petri nets with at most one unbounded place.

A general method that can be used to show *DP*-hardness of many problems concerning one-counter automata and one counter nets was presented in [2]. Note that *DP*-hardness of a problem implies that the problem is *NP*-hard and also *coNP*-hard.

The basic idea is that we can define a certain fragment of Presburger arithmetic, called OCL (One-Counter Logic), such that deciding the truth of closed formulas in this fragment can be easily reduced to different verification problems for one-counter automata. Deciding the truth in OCL is *DP*-hard and so it implies *DP*-hardness of all these problems. Every formula of OCL can be in some sense 'implemented' by the corresponding pair of automata and this construction proceeds inductively on a structure of a formula.

The particular results that were obtained using this method are:

– Equivalence checking is *DP*-hard for one-counter nets for every relation between bisimulation equivalence and simulation preorder.
– Deciding simulation preorder and simulation equivalence is *DP*-hard for a one-counter automaton and finite-state system (in both directions).
– Model checking problem with a one-counter net and a formula from the branching time temporal logic EF (the logic EF is a fragment of CTL) is *DP*-hard.

The article [3] where this results are described in more detail was accepted to publication in the magazine Information and Computation.

2.4 Undecidability of Deciding Simulation Equivalence for One-Counter Automata

It was proved in [4] that the problem of deciding simulation equivalence (resp. simulation preorder) is undecidable for one-counter automata.

3 Overview of Publications:

This section contains an overview of publications where the results were published in chronological order:

– P. Jančar, F. Moller, and Z. Sawa – Simulation problems for one-counter machines, in Proceedings of SOFSEM'99, [4].
– Z. Sawa and P. Jančar – *P*-hardness of equivalence testing on finite-state processes, in Proceedings SOFSEM 2001, [6].
– P. Jančar, A. Kučera, F. Moller, and Z. Sawa – Equivalence-checking with one-counter automata: A generic method for proving lower bounds, in Proceedings of FoSSaCS 2002, [2].
– Z. Sawa – Equivalence Checking of Non-flat Systems Is EXPTIME-hard, in Proceedings of CONCUR 2003, [7].
– P. Jančar, A. Kučera, F. Moller, and Z. Sawa. DP Lower Bounds for Equivalencei-Checking and Model-Checking of One-Counter Automata, to appear in Information and Computation, [3].

References

1. E.M. Clarke, Jr., O. Grumberg, and D.A. Peled. *Model Checking.* MIT Press, 1999.
2. P. Jančar, A. Kučera, F. Moller, and Z. Sawa. Equivalence-checking with one-counter automata: A generic method for proving lower bounds. In *Proceedings of FoSSaCS 2002*, volume 2303 of *LNCS*, pages 172–186. Springer, 2002.
3. P. Jančar, A. Kučera, F. Moller, and Z. Sawa. DP Lower Bounds for Equivalencei-Checking and Model-Checking of One-Counter Automata. To appear.
4. P. Jančar, F. Moller, and Z. Sawa. Simulation problems for one-counter machines. In *Proceedings of SOFSEM'99*, volume 1725 of *LNCS*, pages 404–413. Springer, 1999.

5. A. Rabinovich. Complexity of equivalence problems for concurrent systems of finite agents. *Information and Computation*, 139(2):111–129, 15 December 1997.
6. Z. Sawa and P. Jančar. *P*-hardness of equivalence testing on finite-state processes. In *Proc. SOFSEM 2001 (Piestany, Slovak Rep., November 2001)*, volume 2234 of *Lecture Notes in Computer Science*, page 326. Springer, 2001.
7. Z. Sawa. Equivalence Checking of Non-flat Systems Is EXPTIME-hard. In *Proc. CONCUR 2003*, volume 2761 of *Lecture Notes in Computer Science*, pages 237–250. Springer, 2003.
8. R.J. van Glabbeek. The Linear Time - Branching Time Spectrum. In J.C.M. Baeten and J.W. Klop, editors, *Proceedings of CONCUR '90, Theories of Concurrency: Unification and Extension*, volume 458 of *Lecture Notes in Computer Science*, pages 278–297. Springer-Verlag, Berlin, 1990.
9. *Handbook of Process Algebra*, Elsevier, 2001.