Journal of Computer and System Sciences 91 (2018) 82–103

Contents lists available at ScienceDirect

Journal of Computer and System Sciences

www.elsevier.com/locate/jcss

A generic framework for checking semantic equivalences between pushdown automata and finite-state automata

Antonín Kučera [a,*], Richard Mayr [b]

[a] Faculty of Informatics, Masaryk University, Botanická 68a, CZ-60200 Brno, Czech Republic
[b] University of Edinburgh, School of Informatics, LFCS, 10 Crichton Street, Edinburgh EH8 9AB, UK

ARTICLE INFO

Article history:
Received 19 January 2017
Received in revised form 5 September 2017
Accepted 7 September 2017
Available online 19 September 2017

Keywords:
Pushdown automata
Semantic equivalences
Bisimulation

ABSTRACT

For a given process equivalence, we say that a process g is *fully equivalent* to a process f of a transition system $\mathcal{T}$ if g is equivalent to f and every reachable state of g is equivalent to some state of $\mathcal{T}$. We propose a generic method for deciding full equivalence between pushdown processes and finite-state processes applicable to every process equivalence satisfying certain abstract conditions. Then, we show that these conditions are satisfied by bisimulation-like equivalences (including weak and branching bisimilarity), weak simulation equivalence, and weak trace equivalence, which are the main conceptual representatives of the linear/branching time spectrum. The list of particular results obtained by applying our method includes items which are first of their kind, and the associated upper complexity bounds are essentially optimal.

1. Introduction

One of the main paradigms in formal verification is *equivalence-checking*, where the correctness of a given *implementation* is demonstrated by proving semantic equivalence with its intended behavior called the *specification*. Formally, the implementation and the specification are understood as *processes*, i.e., states in labeled transition systems, and the semantic equivalence is some equivalence over the class of all processes. Equivalence proofs are often long and tedious, especially when the implementation uses unbounded data structures such as counters, stacks, or queues making the state space infinite. A natural question is whether such proofs can be produced automatically, i.e., whether a given process equivalence is decidable in a given class of processes, and what is the associated complexity. The *equivalence-checking* problem has been considered for various process equivalences and various classes of infinite-state processes in the last decades; we refer to, e.g., [49,18,37,9,43,11,53] for surveys of some subfields.

A special variant of the equivalence-checking problem is *regular equivalence-checking*, where the specification is a finite-state process. Hence, an instance of the regular equivalence-checking problem is a process g of a (possibly infinite-state) transition system $\mathcal{U}$, and a process f of a finite-state transition system $\mathcal{T}$. The question is whether g and f are equivalent for some fixed process equivalence. In general, the process g may reach states that are not equivalent to any state of $\mathcal{T}$, i.e., the system $\mathcal{T}$ does not necessarily characterize the *state space* of g up to the chosen equivalence. This motivates the problem of *full regular equivalence-checking*, where we require that g and f are *fully equivalent*, i.e., they are equivalent and each state reachable from g is equivalent to some state of $\mathcal{T}$. The concept of full equivalence was introduced in [40] and studied in

* Corresponding author.
E-mail address: tony@fi.muni.cz (A. Kučera).

http://dx.doi.org/10.1016/j.jcss.2017.09.004
0022-0000/

[47], where it was shown that, for a large class of process equivalences, the problem of full regular equivalence-checking is reducible to the model-checking problem with a slightly extended version of the branching-time logic EF.

In this paper, we restrict our attention to implementations definable by *pushdown automata (PDA)*, a widely accepted model[1] for sequential programs with recursive procedure calls (see, e.g., [2,3,20,22,21]). The operational behavior of a given PDA Δ is formally defined by the associated transition system $\mathcal{T}_\Delta$, where the states are the configurations of Δ and the transitions are determined by the rules of Δ in the natural way (see Section 2). Hence, $\mathcal{T}_\Delta$ has infinitely many states. We use PDA^k to denote the subclass of PDA processes where the underlying pushdown automaton has at most k control states. Due to historical reasons, we also refer to PDA^1 processes as BPA processes.[2]

1.1. Our contribution

We give a generic algorithm for the full regular equivalence-checking problem where the implementation (i.e., the process g) is a PDA process. The algorithm is applicable to every process equivalence satisfying certain abstract criteria, and we show that these criteria are met by bisimulation-like equivalences (incl. weak, early, delay, and branching bisimilarity), weak simulation equivalence, and weak trace equivalence. These equivalences are the main conceptual representatives of the linear/branching time spectrum [57,58], and the applicability of the presented algorithm extends to many (if not all) equivalences in this spectrum by modifying the techniques used for the aforementioned representatives. For PDA^k processes, where $k \geq 1$ is a fixed constant, the obtained algorithms are essentially *optimal*.

More specifically, we show that, given a PDA Δ and a finite-state system $\mathcal{T}$, the full equivalence between the processes of Δ and $\mathcal{T}$ is representable by a finite relation $\mathcal{B}$ called *base*. All pairs of fully equivalent processes can be generated from $\mathcal{B}$ by applying simple substitution rules assuming that the chosen process equivalence is a *right PDA congruence* (see Definition 5). Then, we show how to compute the base $\mathcal{B}$ as the greatest fixed-point of a certain monotonic function. This monotonic function depends on another function called *expansion* which must be tailored specifically for each process equivalence so that the criteria of Definition 12 are satisfied. Finally, we show how to design an appropriate expansion for the concrete process equivalences mentioned above. The list of particular results obtained in this way includes the following:

(a) Branching bisimilarity [59] between PDA^k and finite-state processes is decidable in polynomial time. To the best of our knowledge, this is the first result about computational tractability of branching bisimilarity for systems with infinitely many states (the same actually applies to early and delay bisimilarity). Branching bisimilarity plays a distinguished role in the semantics of systems with silent moves [56], similarly as strong bisimilarity [50] for processes without silent moves.

(b) For weak simulation equivalence, we prove that full equivalence between PDA^k and finite-state processes is decidable in polynomial time. Since checking (non-full) weak simulation equivalence between PDA^k and finite-state processes is **EXPTIME**-complete even for BPA [46], this result shows that full regular equivalence-checking can be more tractable than "ordinary" regular equivalence-checking.

(c) For weak trace equivalence, we show that full equivalence between PDA^k and finite-state processes is decidable in polynomial space, and the problem is **PSPACE**-hard even for BPA. Since checking (weak) trace equivalence between BPA and finite-state processes is undecidable, we see that full regular equivalence-checking can be even "more decidable" than regular equivalence-checking.

Another generic outcome of our method is an algorithm deciding whether a given finite-state process is the $\sim$-quotient of a given PDA process for a given semantic equivalence $\sim$. Here we need to assume that $\sim$ is *preserved under quotients* (see Definition 18) which is not really restrictive because most of the existing process equivalences satisfy this property [40,42].

1.2. Related work

Language equivalence is undecidable for general nondeterministic PDA and BPA [30]. However, for the deterministic subclass (dPDA), language equivalence is decidable [51] (see also [55,34]). The computational complexity of this problem is open and no nontrivial lower bound is known. For the subclass of deterministic one-counter automata, language equivalence-checking is **NL**-complete [8].

Checking bisimulation equivalence is decidable for PDA processes [52]. A nonelementary lower bound has been shown in [6] (see also [35]), improving the previous **EXPTIME** lower bound of [46]; the exact complexity is still open. However, bisimilarity is known to be **PSPACE**-complete for the subclass of one-counter automata [7]. In the context of bisimilarity-checking, a special attention has been devoted BPA which are strictly less expressive than PDA w.r.t. bisimulation-like equivalences. The first positive result is due to Baeten, Bergstra, and Klop [4] who proved the decidability of strong bisimilarity for *normed* BPA (a PDA is normed if the stack can be emptied from every reachable configuration). Simpler proofs were given later in [14,26,32], and there is even a polynomial-time algorithm [28]. The decidability result was extended to all (not necessarily

[1] From the language-theoretic point of view, the definition of PDA adopted in this paper corresponds to the subclass of real-time PDA. The concept of ε-transitions is replaced by "silent" transitions with a distinguished label τ which may (but do not have to) be treated in a special way by a given semantic equivalence.

[2] The "BPA" acronym stands for Basic Process Algebra, a natural fragment of ACP [5]. BPA algebra is expressively equivalent (up to strong bisimilarity) to PDA processes with one control state.