Information and Computation 244 (2015) 134–171

Contents lists available at ScienceDirect

Information and Computation

www.elsevier.com/locate/yinco

Polynomial time decision algorithms for probabilistic automata

Andrea Turrini [a,*], Holger Hermanns [b]

[a] State Key Laboratory of Computer Science, Institute of Software, Chinese Academy of Sciences, Beijing, China
[b] Saarland University – Computer Science, Saarbrücken, Germany

ARTICLE INFO

Article history:
Received 12 July 2014
Received in revised form 7 April 2015
Available online 26 July 2015

Keywords:
Probabilistic automata
Weak probabilistic bisimulation
Branching probabilistic bisimulation
Weak probabilistic simulation
Branching probabilistic simulation
Linear programming problem
Polynomial decision algorithm

ABSTRACT

Deciding in an efficient way weak probabilistic bisimulation in the context of probabilistic automata is an open problem for about a decade. In this work we close this problem by proposing a procedure that checks in polynomial time the existence of a weak combined transition satisfying the step condition of the bisimulation. This enables us to arrive at a polynomial time algorithm for deciding weak probabilistic bisimulation, and also branching probabilistic bisimulation. We furthermore present several extensions to interesting related problems, in particular weak and branching probabilistic simulation, setting the ground for the development of more effective and compositional analysis algorithms for probabilistic systems.

1. Introduction

Probabilistic automata (*PA*s) constitute a mathematical framework for the specification of probabilistic concurrent systems [1,2]. Probabilistic automata extend classical concurrency models in a simple yet conservative fashion. In probabilistic automata, there is no global notion of time, and probabilistic experiments can be performed inside a transition. This embodies a clear separation between probability and nondeterminism, and is represented by transitions of the form $s \xrightarrow{a} \mu$, where s is a state, a is an action label, and μ is a probability measure on states. Labeled transition systems are instances of this model family, obtained by restricting to Dirac measures (assigning full probability to single states). Thus, foundational concepts and results of standard concurrency theory are retained in full and extend smoothly to the model of probabilistic automata. The *PA* model is akin to Markov decision processes (*MDP*s) [3], and its foundational beauty can be paired with powerful model checking techniques, as implemented for instance in the PRISM tool [4]. Variations of this model are Labeled Concurrent Markov Chains (*LCMC*s) and alternating Models [5–7]. We refer the interested reader to [8] for a survey on *PA*s and other models.

If facing a concrete probabilistic system, we can conceive several different *PA* models to reflect its behavior. For instance, we can use different state names, encode diverse information in the states, represent internal computations with different action labels and/or different granularity, and so on. *Bisimulation relations* constitute a powerful tool allowing us to check whether two models describe in some sense the same system. They are then called bisimilar. The bisimilarity of two systems can be viewed in terms of a game played between a challenger and a defender. In each step of the infinite bisimulation game, the challenger chooses one automaton, makes a step, and the defender matches it with a step of the other automaton.

* Corresponding author.
E-mail addresses: turrini@ios.ac.cn (A. Turrini), hermanns@cs.uni-saarland.de (H. Hermanns).

http://dx.doi.org/10.1016/j.ic.2015.07.004
0890-5401/© 2015 Elsevier Inc. All rights reserved.

Depending on how we want to treat internal computations, this leads to *strong* and *weak* bisimulations [9]: the former requires that each single step of the challenger automaton is matched by an identically labeled single step of the defender automaton, the latter allows the matching up to internal computation steps. These probabilistic bisimulations can be twisted by allowing (or disallowing) the defender to match the challenger's step by a convex combination of enabled probabilistic transitions. That corresponds to the use of randomized schedulers, as we will explain below. It results in a spectrum of four bisimulations: strong [1,6,5], strong probabilistic [1], weak [7,1], and weak probabilistic [1] bisimulation.

In the classical setting, *branching* bisimulation [10] is often advocated as a refined alternative to *weak* bisimulation. It shares all the core properties, but is finer in that it requires that internal computation steps to be considered irrelevant may not pass through intermediate states with distinguishable behavior. Among others, this has benefits with respect to logical characterizations, and these benefits carry over to the setting of probabilistic automata, yielding branching probabilistic bisimulation [11].

Besides comparing automata, bisimulation relations allow us to reduce the size of an automaton without changing its properties (i.e., with respect to sets of logic formulae satisfied by it). This is particularly useful to alleviate the state explosion problem notoriously encountered in model checking, provided efficient decision algorithms exist that can be used as minimization algorithms.

The efficiency of *PA* bisimulation decision algorithms is therefore of great interest, and several results have appeared in the literature. With respect to the strong relations, both strong and strong probabilistic bisimulations can be decided in polynomial time [12,13]. The situation relative to the weak bisimulation relations is more intricate: For *PA* weak bisimulation, there is no known decision algorithm; however, as shown in [14], *PA* weak bisimulation is not transitive and this severely limits its usefulness and the need of a decision algorithm. On the other hand, weak probabilistic bisimulation is indeed transitive. The only known decision algorithm for it is exponential [13] in the size of the probabilistic automaton. On the other hand, weak bisimulation on *LCMC*s can be computed in polynomial time [7]. In *LCMC*s, there is a strict separation of states with nondeterministic choices (over the transition to perform) and states with probabilistic effects, and this separation is exploited to arrive at polynomiality of the decision algorithm. Similarly to the weak bisimulation, also the branching bisimulation on alternating *LCMC*s can be decided in polynomial time [15] while no decision algorithm is known for *PA* branching probabilistic bisimulation.

In this context, it is worth to note that *LCMC* weak bisimulation [7] and *PA* weak probabilistic bisimulation [1] coincide [16] provided the *LCMC* is seen as an alternating *PA* in the sense that each *PA* state enables either a single transition leading to a probability measure over states, or possibly multiple transitions each with a single target state (or none at all). This alternation constraint on the structure of the *PA* is understood to be enough to reduce the complexity of the published decision procedure [13]. It enables to characterize states by their maximal jump probabilities per equivalence class, while this is not the case in general. As a consequence. the approach proposed in [7] cannot be used for ordinary *PA*s, see [13] for more details. Restricted versions of *PA* weak probabilistic bisimulations, such as normed [17] and delay [18] bisimulation, can be decided in polynomial time.

Lately, the model of *PA* has been enhanced with memoryless continuous time, integrated into the model of Markov automata [19–21]. This extension is also rooted in interactive Markov chains (*IMC*s) [22], another model with a well-understood compositional theory. *IMC*s are applied in a large spectrum of practical applications, ranging from networked hardware on chips [23] to water treatment facilities [24] and ultra-modern satellite designs [25]. The standard analysis trajectory for *IMC*s revolves around compositional applications of weak or branching bisimulation minimization, a strategy that employs polynomial time decision algorithms [22,26] and has been proven very effective [27,28,23]. Owed to the unavailability of effective algorithms for *PA* weak and branching probabilistic bisimulations, this compositional minimization strategy has thus far not been explored in the *PA* (or *MDP*) setting. We aim at making this possible, and furthermore, we intend to repeat and extend the successful applications of *IMC*s in the extended Markov automata setting. For this, polynomial time decision procedures for weak or branching probabilistic bisimulation on *PA* are the essential building blocks.

In this paper we show that *PA* weak probabilistic and branching probabilistic bisimulations can be decided in polynomial time, thus just as all other interesting bisimulations on *PA*. To arrive there, we provide a decision procedure that follows the standard partition refinement approach [13,29,30] and that is based on a Linear Programming (LP) problem. The crucial step is that we manage to generate and decide an LP problem that proves or disproves the existence of a weak step in time polynomial in the size of an automaton which in turn encodes a weak transition linear in its size. This enables us to decide in polynomial time whether the defender has a matching weak transition step - opposed to the exponential time required thus far [13] for this. Apart from this result, which closes successfully the open problem of [13], we show how the LP approach can be extended to branching steps and hence to branching probabilistic bisimulation. We further harvest the approach to decide in polynomial time the single-sided versions of the relations in question, weak probabilistic simulation and branching probabilistic simulation. As a result, the techniques developed in this paper provide a blueprint for deciding in polynomial time further weak relations from the simulation and bisimulation spectrum [31] in the context of probabilistic automata.

Organization of the paper. After the mathematical preliminaries in Section 2, we present in Section 3 the probabilistic automata model and the behavioral relations defined on it. Then, in Section 4, we recast the algorithm proposed in [13] that decides whether two probabilistic automata are weak probabilistic bisimilar and we present the algorithms for both weak probabilistic bisimulation and simulation, together with their complexity analysis. In Section 5 we introduce the polynomial