Theoretical Computer Science ••• (••••) •••–•••

Contents lists available at ScienceDirect

Theoretical Computer Science

www.elsevier.com/locate/tcs

Distributed probabilistic input/output automata: Expressiveness, (un)decidability and algorithms[☆]

Sergio Giro[a,*], Pedro R. D'Argenio[b], Luis María Ferrer Fioriti[c]

[a] *Fakultät für Informatik, Technische Universität München, Germany*
[b] *Universidad Nacional de Córdoba, FaMAF – CONICET, Córdoba, Argentina*
[c] *Saarland University, Computer Science, Saarbrücken, Germany*

A R T I C L E I N F O

Keywords:
Probabilistic systems
Distributed systems
Nondeterminism
Interleaving
Markov decision processes
Partial observation

A B S T R A C T

Probabilistic model checking computes the probability values of a given property quantifying over all possible schedulers. It turns out that maximum and minimum probabilities calculated in such a way are over-estimations on models of distributed systems in which components are loosely coupled and share little information with each other (and hence arbitrary schedulers may result too powerful). Therefore, we introduced definitions that characterise which are the schedulers that properly capture the idea of distributed behaviour in probabilistic and nondeterministic systems modelled as a set of interacting components.

In this paper, we provide an overview of the work we have done in the last years which includes: (1) the definitions of distributed and strongly distributed schedulers, providing motivation and intuition; (2) expressiveness results, comparing them to restricted versions such as deterministic variants or finite-memory variants; (3) undecidability results—in particular the model checking problem is not decidable in general when restricting to distributed schedulers; (4) a counterexample-guided refinement technique that, using standard probabilistic model checking, allows to increase precision in the actual bounds in the distributed setting; and (5) a revision of the partial order reduction technique for probabilistic model checking. We conclude the paper with an extensive review of related work dealing with similar approaches to ours.

1. Introduction

Markov decision processes (MDPs) are widely used in diverse fields ranging from ecology to computer science. They are useful to model and analyse systems in which both probabilistic and nondeterministic choices interact. MDPs can be automatically analysed using quantitative model checkers such as PRISM [24] or LiQuor [10].

Since MDPs contain nondeterministic choices (in addition to probabilistic steps), the model checking problem is to find out the largest or smallest probability of reaching a goal under any possible resolution of the nondeterministic choices, a concrete instance being "the probability of arrival of a package is at least 0.95 no matter how the package is routed". The resolution of such nondeterminism is given by the so-called *schedulers* (called also adversaries or policies—see e.g. [4,28]) which choose an enabled transition after each finite execution path of the system.

[☆] Supported by DFG Graduiertenkolleg 1480 (PUMA), ANPCYT Project PAE-PICT 02272, SeCyT-UNC Program 05/BP02 and Project 05/B497, EU 7FP grant agreement 295261 (MEALS), DFG as part of SFB/TR 14 AVACS, and DFG/NWO Bilateral Research Programme ROCKS. This article was written while the first author was on leave at the Department of Computer Science, FCEIA, Universidad Nacional de Rosario.

[*] Corresponding author.

E-mail addresses: sergio.giro.ar@gmail.com (S. Giro), dargenio@famaf.unc.edu.ar (P.R. D'Argenio), ferrer@depend.cs.uni-saarland.de (L.M. Ferrer Fioriti).

0304-3975/$ – see front matter © 2013 Elsevier B.V. All rights reserved.
http://dx.doi.org/10.1016/j.tcs.2013.07.017

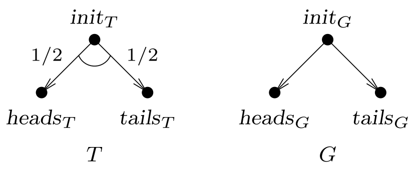

Fig. 1. T tosses a coin and G has to guess.

The available tools for model checking such as PRISM [24] or LiQuor [10] calculate the worst-case probability considering all possible schedulers. However, in distributed systems, some schedulers correspond to unrealistic resolutions of the nondeterminism (as we illustrate below) thus resulting in overly pessimistic worst-case probabilities. A restricted class of schedulers was proposed to cope with this problem in previous literature—see e.g. [13,9,8,12,16]. We call these schedulers *distributed schedulers*, since in these settings there is a *local* scheduler for each component and so the resolution of the nondeterminism is *distributed* among the different components.

In this paper, we investigate different subclasses of distributed schedulers in order to answer to which extent these subclasses are able to attain the worst-case probability. The subclasses we consider are strongly related to the development of techniques for MDP analysis. As an example, if the class of all schedulers is considered, worst-case probabilities of reachability properties are attained by schedulers that are both Markovian—i.e. the decision is based on the current state of the execution, disregarding the previous history—and deterministic—i.e. the schedulers themselves have no probabilistic choices, see [4]. The existence of this subclass ensures that the worst-case probability can be found by exhaustive search (notice that more efficient methods exist [4]). Hence, one may like to know to which extent these results hold in case the schedulers are restricted to be distributed.

1.1. *Unrealistic worst cases and distributed schedulers*

A scheduler is a function mapping paths to transitions (or, in the more general case, paths to distributions on transitions). Given that the execution up to some state s is known (namely, the history path), the scheduler "chooses" to perform one transition out of all transitions enabled in state s.

The following example illustrates the problem that motivates the introduction of distributed schedulers: a man tosses a coin and another one has to guess heads or tails. Fig. 1 depicts the models of these two men in terms of MDPs. Man T, who tosses the coin, has only one transition which represents the toss of the coin: with probability $\frac{1}{2}$ he moves to state $heads_T$ and with probability $\frac{1}{2}$ he moves to state $tails_T$. Instead, man G has two possible transitions, each one representing his choice: $heads_G$ or $tails_G$. An *all-knowing* scheduler for this system may let G guess the correct answer with probability 1 according to the following sequence: first, it lets T toss the coin, and then it chooses for G the transition leading to heads if T tossed a head or the transition leading to tails if T tossed a tail. Therefore, the supremum probability of guessing obtained by quantifying over these all-knowing schedulers is 1, even if T is a smart player that always hides the outcome until G reveals his choice. As a consequence, quantitative model checkers based on [4], though safe, yield an overestimation of the correct value. In this example, in which T and G do not share all information, we would like that the supremum probability of guessing (i.e., of reaching any of the states $(heads_T, heads_G)$ or $(tails_T, tails_G)$) is $\frac{1}{2}$.

This observation is fundamental in distributed systems in which components share little information with each other, as well as in security protocols, where the possibility of information hiding is a fundamental assumption [6]. Similar phenomena to the one we illustrated have been observed in [28] from the point of view of compositionality and studied in [12,13,9] in different settings. Distributed schedulers are also related to the partial-information policies of [12].

In order to avoid considering these unrealistic behaviours, *distributed* schedulers were proposed in previous literature. *Local* schedulers for each component of the system are defined in the usual way (that is, the choices are based on the complete history of the component) and distributed schedulers are defined to be the schedulers that can be obtained by composing these local schedulers. We remark that the "all-knowing" scheduler of the example would not be a valid scheduler in this new setting since the choice for G depends on information which is external to (and not observable by) G. In contrast, a local scheduler for G takes the decision having no information about the actual state of T, and so the choice cannot conveniently vary according to the outcome of T.

Previous work in the area either deals with nondeterminism in a unique manner (regardless whether it originates from local choices or from the interleaving) or simply focuses on local choices avoiding the resolution of interleaving nondeterminism (either by assuming full synchronisation [13] or by model construction [9]; see Section 7 for a detailed comparison). If we allow interleaving nondeterminism, the schedulers can also be restricted to handle this nondeterminism in a realistic way. So, we motivate a restriction to distributed schedulers in this direction, and define the *strongly distributed* schedulers as the schedulers complying with such restriction.

1.2. *Overview of the paper*

This article surveys the state of the art of model checking for distributed probabilistic systems modelled as a network of interconnecting probabilistic I/O automata. It collects and summarises the work that we have done in the last years since