Available at
www.ElsevierComputerScience.com
POWERED BY SCIENCE DIRECT®

Information Processing Letters 89 (2004) 123–130

Information Processing Letters

www.elsevier.com/locate/ipl

Probabilistic weak simulation is decidable in polynomial time ☆

Christel Baier [a], Holger Hermanns [b,c], Joost-Pieter Katoen [c,*]

[a] *Institut für Informatik I, University of Bonn, Römerstraße 164, D-53117 Bonn, Germany*
[b] *Department of Computer Science, Saarland University, D-66123 Saarbrücken, Germany*
[c] *Department of Computer Science, University of Twente, P.O. Box 217, 7500 AE Enschede, The Netherlands*

Received 17 June 2003; received in revised form 1 August 2003

Communicated by D. Basin

Abstract

This paper considers a weak simulation preorder for Markov chains that allows for stuttering. Despite the second-order quantification in its definition, we present a polynomial-time algorithm to compute the weak simulation preorder of a finite Markov chain.

Keywords: Algorithms; Computational complexity; Markov chains; Performance evaluation; Simulation preorder

1. Introduction

Markov chains [20,21] are one of the most important classes of stochastic processes. They are usually described by higher-level formalisms such as stochastic variants of Petri nets, process algebras, or automata networks. As the size of the Markov chain typically grows exponentially with the size of the high-level description, the infamous state-space explosion problem is frequently encountered in practice. To combat this problem, reduction techniques based on *lumpability* [20] are often employed. This allows for the computation of steady-state and transient-state probabilities on the quotient of the Markov chain under lumping equivalence.

This paper considers a Milner-style *weak simulation preorder* on Markov chains, first introduced in [5]. Whereas lumpability relates states that mutually mimic all individual steps, weak simulation requires that one state can mimic all stepwise behavior of the other, but not the converse, and—in contrast to strong simulation relations—only requires this for certain ("observable") transitions and not for other ("silent") transitions. This allows for a more radical reduction of the state space than using lumpability, while preserving (non-trivial) cumulative transient-state probabilities such as the probability to reach a set of goal states within a given time bound [5].

Efficient algorithms to construct the quotient space under (strong or weak) lumpability can be obtained with a variant of the partitioning-splitter technique for labeled transition systems [23,16,11]. The strong simulation preorder can be computed by solving a network flow problem [2]. The major contribution of this

☆ Supported by the VOSS (Validation of Stochastic Systems) project, funded by the NWO (Netherlands Organization for Scientific Research) and the DFG (German Research Council).

* Corresponding author.
E-mail address: katoen@cs.utwente.nl (J.-P. Katoen).

0020-0190/$ – see front matter
doi:10.1016/j.ipl.2003.10.001

paper is a polynomial-time decision algorithm for the weak simulation preorder on Markov chains. The crux of the algorithm is to consider the check whether a state simulates another one as a linear programming (LP) problem. By showing that also the decision problem for weak simulation belongs to the complexity class P, a complete picture on the complexity of deciding branching-time relations on Markov chains is obtained. This result is more surprising than the corresponding one in the non-probabilistic setting as the definition of weak simulation on Markov chains relies on a second-order quantification over partitionings of the direct successors of states (rather than the transitive closure of the transition relation).

The results are presented for weak simulation on continuous-time Markov chains (CTMCs) and carry over in a straightforward manner to weak simulation for discrete-time Markov chains (DTMCs).

2. Markov chains

Definition 1. A distribution on countable set S is a function $\mu : S \to [0, 1]$ with $\sum_{s\in S} \mu(s) \leqslant 1$. Distribution μ on S is called stochastic if $\sum_{s\in S} \mu(s) = 1$. Let $Dist(S)$ denote the collection of all distributions on S.

States of Markov chains are labeled by atomic propositions, i.e., elementary statements about states referring to, e.g., the number of customers in a queue, the state color, or the like. Let AP be a fixed, finite set of atomic propositions.

Definition 2. A (labeled) CTMC $\mathcal{M}$ is a tuple $(S, \boldsymbol{R}, L)$ where:

- S is a finite set of states,
- $\boldsymbol{R} : S \times S \to \mathbb{R}_{\geqslant 0}$ is the *rate matrix*,
- $L : S \to 2^{AP}$ is a labeling function that assigns to each state $s \in S$ the set $L(s)$ of atomic propositions that are valid in s.

For state $s \in S$, the exit rate $E(s)$ is defined by:

$$E(s) = \sum_{s'\in S} \boldsymbol{R}(s, s').$$

A state s for which $E(s) = 0$ is called *absorbing*, otherwise it is called non-absorbing. It should be noted that Definition 2 does not require $\boldsymbol{R}(s, s) = -\sum_{s'\neq s} \boldsymbol{R}(s, s')$, as is often found in textbooks on CTMCs. The usual *infinitesimal generator matrix* $\boldsymbol{Q}$ is obtained from the rate matrix by subtracting the row-sums from the diagonal:

$$\boldsymbol{Q}(s, s') = \begin{cases} \boldsymbol{R}(s, s) - E(s) & \text{if } s = s', \\ \boldsymbol{R}(s, s') & \text{otherwise.} \end{cases}$$

In the traditional interpretation, at the end of a stay in state s, the system will move to a different state. According to Definition 2, state s has a self-loop if $\boldsymbol{Q}(s, s) > 0$. We thus allow the system to occupy the same state before and after taking a transition. The inclusion of self-loops neither alters the transient nor the steady-state behavior of the CTMC and is also treated in this way in, among others, the textbook [26].

The rates intuitively specify the average delays of the transitions. More precisely, the exit rate $E(s)$ denotes that the probability of taking a transition from s within t time units equals $1 - \mathrm{e}^{-E(s)\cdot t}$. $\boldsymbol{R}(s, s') = \lambda > 0$ means that with probability $1 - \mathrm{e}^{-\lambda\cdot t}$ the transition from s to s' is enabled within the next t time units—provided the current state is s. If $\boldsymbol{R}(s, s') = 0$ then there is no transition possible from state s to s'. Let $\mathsf{Post}(s) = \{s' \in S \mid \boldsymbol{R}(s, s') > 0\}$ the set of successor states of state s. If $\boldsymbol{R}(s, s') > 0$ for more than one state s', a *race* between the outgoing transitions from s exists. The probability that the transition from the current state s to s' is taken within the next t time units is:

$$\frac{\boldsymbol{R}(s, s')}{E(s)} \cdot \bigl(1 - \mathrm{e}^{-E(s)\cdot t}\bigr).$$

$\boldsymbol{P}(s, s') = \boldsymbol{R}(s, s')/E(s)$ denotes the probability that the delay of going from s to s' "finishes before" the delay of any other outgoing transition from s. Hence, $\boldsymbol{P}(s, s')$ is the time-abstract probability of moving from s to s' in a single step.

Important equivalence notions on CTMCs are lumping equivalences, also known as bisimulations. Let $\mathcal{M} = (S, \boldsymbol{R}, L)$ be a CTMC and R an equivalence relation on S. R is a *strong bisimulation* [9,15] on $\mathcal{M}$ if for $(s_1, s_2) \in R$:

$$L(s_1) = L(s_2) \quad \text{and} \quad \boldsymbol{R}(s_1, C) = \boldsymbol{R}(s_2, C) \tag{1}$$

for all C in S/R, where $\boldsymbol{R}(s, C) = \sum_{s'\in C} \boldsymbol{R}(s, s')$. s_1 and s_2 in S are strongly bisimilar if there exists a

strong bisimulation R on $\mathcal{M}$ with $(s_1, s_2) \in R$. R is a *weak bisimulation* [7,4] on $\mathcal{M}$ if for all $(s_1, s_2) \in R$:

$$L(s_1) = L(s_2) \quad \text{and} \quad \boldsymbol{R}(s_1, C) = \boldsymbol{R}(s_2, C)$$

for all $C \neq [s_1]_R$.

3. Weak probabilistic simulation

For labeled transition systems, state s' simulates s if for each successor t of s there is a successor t' of s' that simulates t. Simulation of two states is thus defined in terms of simulation of their successor states [22,17]. In the probabilistic setting, the target of a transition is in fact a probability distribution, and thus, the simulation relation $\sqsubseteq$ needs to be lifted from states to distributions. This is done using *weight functions* [19,18].

Definition 3. Let $\mu \in \mathit{Dist}(X)$ and $\mu' \in \mathit{Dist}(Y)$ and $\sqsubseteq \subseteq X \times Y$. Then $\mu \preceq \mu'$ if and only if there exists a weight function $\Delta : X \times Y \to [0, 1]$ for $\sqsubseteq$ such that:

(1) $\Delta(x, y) > 0$ implies $x \sqsubseteq y$,
(2) $\mu(x) = K_1 \cdot \sum_{y \in Y} \Delta(x, y)$ for any $x \in X$,
(3) $\mu'(y) = K_2 \cdot \sum_{x \in X} \Delta(x, y)$ for any $y \in Y$,

where $K_1 = \sum_{x \in X} \mu(x)$ and $K_2 = \sum_{y \in Y} \mu'(y)$.

Intuitively, Δ distributes a probability distribution over a set X to a distribution over a set Y such that the total probability assigned by Δ to $y \in Y$ equals the original probability $\mu'(y)$ on Y (up to a factor $1/K_2$). In a similar way, the total probability mass of $x \in X$ that is assigned by Δ must coincide with the probability $\mu(x)$ on X (up to a factor $1/K_1$). (Note that $K_1 = K_2 = 1$ for stochastic μ and μ'.) Δ is a probability distribution on $X \times Y$ such that the probability to select (x, y) with $x \sqsubseteq y$ is one. In addition, the probability to select an element in $\sqsubseteq$ whose first component is x equals $\mu(x)$, and the probability to select an element in $\sqsubseteq$ whose second component is y equals $\mu'(y)$.

Example 4. Let $X = \{s, t\}$ and $Y = \{u, v, w\}$ with $\mu(s) = \frac{2}{9}$, $\mu(t) = \frac{2}{3}$ and $\mu'(u) = \frac{1}{3}$, $\mu'(v) = \frac{4}{9}$ and $\mu'(w) = \frac{1}{9}$; $K_1 = K_2 = \frac{8}{9}$. Note that μ and μ' are both sub-stochastic. Let $\sqsubseteq = (X \times Y) \setminus \{(s, w)\}$. We have

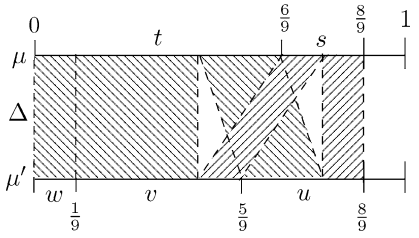

Fig. 1. Weight function for $\sqsubseteq$.

$\mu \preceq \mu'$, as, e.g., weight function Δ (cf. Fig. 1) defined by $\Delta(s, u) = \Delta(s, v) = \Delta(t, w) = \frac{1}{8}$, $\Delta(t, v) = \frac{3}{8}$ and $\Delta(t, u) = \frac{1}{4}$ satisfies the constraints of Definition 3.

The intention of a simulation preorder on CTMCs is to ensure that state s_2 simulates s_1 if and only if (i) s_2 is "faster than" s_1 and (ii) the time-abstract behavior of s_2 simulates that of s_1. This is achieved by the following definition that additionally incorporates a notion of *stuttering* [8,13].

Definition 5 [5,4]. Let $\mathcal{M} = (S, \boldsymbol{R}, L)$ be a CTMC. Relation $\sqsubseteq \subseteq S \times S$ is a *weak simulation* if and only if for all states $s_1, s_2 \in S$ with $s_1 \sqsubseteq s_2$ we have that $L(s_1) = L(s_2)$ and there exist functions $\Delta : S \times S \to [0, 1]$, $\delta_i : S \to [0, 1]$ and sets $U_i, V_i \subseteq S$ (for $i = 1, 2$) with:

$$U_i = \{u_i \in \mathsf{Post}(s_i) \mid \delta_i(u_i) > 0\} \quad \text{and}$$
$$V_i = \{v_i \in \mathsf{Post}(s_i) \mid \delta_i(v_i) < 1\}$$

such that:

(1) $v_1 \sqsubseteq s_2$ for any $v_1 \in V_1$ and $s_1 \sqsubseteq v_2$ for any $v_2 \in V_2$,
(2) $\Delta(u_1, u_2) > 0$ implies $u_i \in U_i$ and $u_1 \sqsubseteq u_2$,
(3) if $K_1 > 0$ and $K_2 > 0$ then for any $w \in S$:

$$K_1 \cdot \sum_{u_2 \in U_2} \Delta(w, u_2) = \delta_1(w) \cdot \boldsymbol{P}(s_1, w)$$

and

$$K_2 \cdot \sum_{u_1 \in U_1} \Delta(u_1, w) = \delta_2(w) \cdot \boldsymbol{P}(s_2, w),$$

(4) $K_1 \cdot E(s_1) \leqslant K_2 \cdot E(s_2)$,

where $K_i = \sum_{u_i \in U_i} \delta_i(u_i) \cdot \boldsymbol{P}(s_i, u_i)$ for $i = 1, 2$.

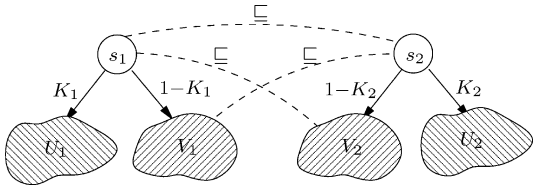

Fig. 2. Simulation scenario.

In the sequel, $\sqsubseteq$ denotes a weak simulation. Definition 5 can be justified as follows. The successor states of s_i are grouped into the subsets U_i and V_i. Although it is not required that U_i and V_i are disjoint, to understand the definition first consider $U_i \cap V_i = \emptyset$. (The fact that we allow a nonempty intersection has technical reasons that will be explained below.) K_i denotes the total probability to move from s_i within one transition to a state in U_i. Vice versa, with probability $1 - K_i$, in state s_i a transition to some state in V_i is made (cf. Fig. 2). The first condition states that the grouping of successor states into V_i and U_i is such that any state in V_2 simulates s_1 and that s_2 simulates any state in V_1. Intuitively, we interpret the moves from s_i to a V_i-state as silent transitions. The first condition of Definition 5 thus guarantees that any such transition is a "stutter" step. The second and third condition require the existence of a weight function Δ that relates the conditional probabilities to move from s_1 to a U_1-state and the conditional probabilities for s_2 to move to a U_2-state. Thus, Δ is a weight function for the probability distributions $\delta_i(\cdot) \cdot \boldsymbol{P}(s_i, \cdot)/K_i$. Intuitively, the transitions from s_i to a U_i-state are considered as observable moves and the second and third condition are the continuous versions of similar conditions for strong simulation in the discrete-time case [19]. Finally, the fourth condition states that s_2 is "faster than" s_1 in the sense that the total rate to move from s_2 to a U_2-state is at least the total rate to move from s_1 to a U_1-state.

In most cases $\delta_i(s) \in \{0, 1\}$ for any state s, i.e., δ_i is the characteristic function of U_i, and the sets U_i and V_i are disjoint. In general, though, U_i and V_i may contain *fragments* of states. That is, we deal with functions δ_i where $0 \leqslant \delta_i(s) \leqslant 1$. Intuitively, the $\delta_i(s)$-fragment of state s belongs to U_i, while the remaining part (the $(1-\delta_i(s))$-fragment) of s belongs to V_i. The use of fragments of states is discussed in the following example. It is needed to guarantee the transitivity of the weak simulation preorder.

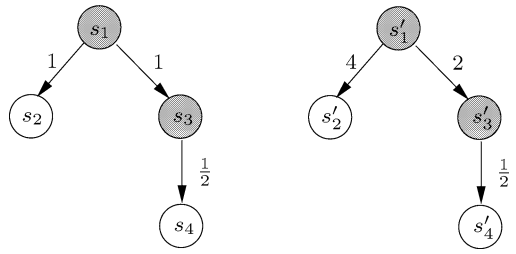

Fig. 3. An example of simulation using fragments of states.

Example 6. Consider the two CTMCs depicted in Fig. 3, where $L(s_1) = L(s_3) = L(s'_1) = L(s'_3) = \{a\}$; the other states are labeled by b. Intuitively, s_1 is "slower than" s'_1. However, when we require the sets U_i, V_i in Definition 5 to be disjoint, then $s_1 \not\sqsubseteq s'_1$. This can be seen as follows. We have $s_1 \not\sqsubseteq s'_3$ (and hence, $V_2 = \emptyset$) as s_1 moves with rate 1 to a b-state while the total rate for s'_3 to move to a b-state is smaller (i.e., $\frac{1}{2}$). Hence, the only chance to define the components in Definition 5 is $V_2 = \emptyset$ and $U_2 = \{s'_2, s'_3\}$. Because s'_3 and s_2 are not comparable using the simulation preorder (as they have different labels), we would have to define $U_1 = \{s_2, s_3\}$ and $V_1 = \emptyset$. But then, the weight-function condition (i.e., condition (3) of Definition 5) is violated because s_1 moves with probability $\frac{1}{2}$ to a b-state while the probability for s'_1 to reach a b-state within one step is $\frac{2}{3}$.

On the other hand, when we allow s_3 to be split: one half of s_3 belongs to U_1, one half to V_1, i.e., $\delta_1(s_3) = \frac{1}{2}$ and $U_1 = \{s_2, s_3\}$, $V_1 = \{s_3\}$ then we get that with $\delta_1(s_2) = \delta_2(s'_2) = \delta_2(s'_3) = 1$, $U_2 = \{s'_2, s'_3\}$, $V_2 = \emptyset$, and $\sqsubseteq = \{(s_1, s'_1), (s_2, s'_2), (s_3, s'_1), (s_4, s'_4), (s_3, s'_3), (s_2, s'_4), (s_4, s'_2)\}$ the conditional probabilities for the U_i-states are related via $\preceq$. Note that $K_1 = \frac{1}{2} + \frac{1}{4} = \frac{3}{4}$ and $K_2 = 1$. Hence, we may deal with $\Delta(s_2, s'_2) = \frac{2}{3}$ and $\Delta(s_3, s'_3) = \frac{1}{3}$.

Remark 7. Suppose $s_1 \sqsubseteq s_2$ and one of the states is absorbing. If s_2 is absorbing (i.e., $E(s_2) = 0$) then $K_1 \cdot E(s_1) = 0$. Hence, either s_1 is absorbing or $K_1 = 0$. In the latter case, $U_1 = \emptyset$, i.e., all successor states of s_1 belong to V_1 and are simulated by s_2 (by condition (1) in Definition 5). If s_1 is absorbing, then s_2 may be an arbitrary state with $L(s_1) = L(s_2)$. The observation that absorbing state s_1 is simulated by any state s_2 with the same labeling is natural for any type of

simulation that abstracts from silent moves. Note that in absorbing states of a CTMC just time advances.

4. Algorithm for weak simulation

Theorem 1. *Given a CTMC with finite state space S, the quotient space with respect to the weak simulation preorder $\sqsubseteq$ can be computed in time and space* $O(poly(|S|))$.

This result is proven by presenting a polynomial-time algorithm that computes the weak simulation preorder of a given CTMC. Let $(S, \boldsymbol{R}, L)$ be a CTMC. The main procedure of our algorithm (cf. Algorithm 1) computes the weak simulation preorder in an iterative manner as for the non-probabilistic case. Starting from the trivial preorder

$$R = \{(s_1, s_2) \in S \times S \mid L(s_1) = L(s_2)\}$$

pairs (s_1, s_2) are successively removed from R if s_1 has a transition that cannot be "simulated" by a transition of s_2 where simulation is understood with respect to the current relation R. This process is continued until no such pair of states is left in R. The loop invariant of this procedure is that R is coarser than $\sqsubseteq$.

```
(* Input: CTMC with finite state space S    *)
(* Output: the weak simulation preorder ⊑ *)

R := {(s1, s2) ∈ S × S | L(s1) = L(s2)};
while ∃(s1, s2) ∈ R such that s1 ⋢ s2 do
   R := R \ {(s1, s2)}
od
return R                                  (* R = ⊑ *)
```

Algorithm 1. Schema for computing $\sqsubseteq$.

Several improvements of this naive schema are possible, e.g., in the style of [14]. However, in this paper we concentrate on the differences to the non-probabilistic setting. The computational procedure explicitly relies on a test whether s_2 simulates (under a fixed R) s_1. In order to do so, according to Definition 5 we need a method to check whether components δ_i, U_i, V_i, K_i, Δ can be constructed for (s_1, s_2). We show that this problem can be reduced to a *linear programming* (LP) *problem*. Thus, checking whether one state weakly simulates another one amounts to check whether a certain set of linear inequations and equations has a solution. This is done as follows. Let $(s_1, s_2) \in R$. Note that by construction of R, s_1 and s_2 are equally labeled. Let:

$$s{\downarrow_R} = \{s' \in S \mid (s', s) \in R\}$$

be the downward closure of s with respect to R, and similarly

$$s{\uparrow_R} = \{s' \in S \mid (s, s') \in R\}$$

be the upward closure of s with respect to R.

First, distinguish the following cases:

(1) $\mathsf{Post}(s_1) \subseteq s_2{\downarrow_R}$. Then, the conditions in Definition 5 are fulfilled for (s_1, s_2) by setting:

$$V_1 = \mathsf{Post}(s_1), \quad U_1 = \emptyset, \quad \text{and} \quad U_2 = \mathsf{Post}(s_2).$$

(2) $E(s_2) = 0$, i.e., s_2 is absorbing. Then the observations in Remark 7 can be applied to check whether the conditions of Definition 5 are fulfilled for (s_1, s_2).

Note that these checks can be done in polynomial time. We now consider the remaining case. Assume s_2 is non-absorbing and s_1 has at least one successor state $u_1 \in \mathsf{Post}(s_1)$ such that $u_1 \notin s_2{\downarrow_R}$. As $u_1 \not\sqsubseteq s_2$, and all states in V_1 have to be simulated by s_2 (cf. condition (1) of Definition 5), state $u_1 \in U_1$. Thus, $K_1 > 0$, and, by condition (4) of Definition 5, $K_2 > 0$.

Consider the following variables:

- x and y which stand for the values $x = 1/K_1$ and $y = 1/K_2$, respectively.
- x_u for $u \in S$ with $(u, s_2) \in R$ which stands for the value $x_u = \delta_1(u)/K_1$.
- y_u for $u \in S$ with $(s_1, u) \in R$ which stands for the value $y_u = \delta_2(u)/K_2$.
- z_{u_1,u_2} for each pair of states $(u_1, u_2) \in R$.

We write $\Delta(u_1, u_2)$ instead of z_{u_1,u_2}, and put:

$$x_u = x \quad \text{if } u \notin \setminus s_2{\downarrow_R},$$
$$y_u = y \quad \text{if } u \notin \setminus s_1{\uparrow_R}.$$

This is justified as follows. Each state u in $\mathsf{Post}(s_1) \setminus s_2{\downarrow_R}$ has to be put completely in U_1. Thus, $\delta_1(u) = 1$, and hence:

$$x_u = x = \frac{1}{K_1} = \frac{\delta_1(u)}{K_1}.$$

By a symmetric argument, we put $y_u = y$ if $u \notin s_1{\uparrow_R}$.

The linear program now consists of the following equations and inequalities:

$$\sum_{u_1 \in u_2\downarrow_R} \Delta(u_1,u_2) = \boldsymbol{P}(s_2,u_2)\cdot y_{u_2} \quad \text{for } u_2 \in S,$$

$$\sum_{u_2 \in u_1\uparrow_R} \Delta(u_1,u_2) = \boldsymbol{P}(s_1,u_1)\cdot x_{u_1} \quad \text{for } u_1 \in S,$$

$$\sum_{u_1 \in S} x_{u_1}\cdot \boldsymbol{R}(s_1,u_1) = E(s_1),$$

$$\sum_{u_2 \in S} y_{u_2}\cdot \boldsymbol{R}(s_2,u_2) = E(s_2),$$

$$x \geqslant 1,$$

$$y \geqslant 1,$$

$$x \geqslant x_u \geqslant 0 \quad \text{if } u \in s_2\downarrow_R,$$

$$y \geqslant y_u \geqslant 0 \quad \text{if } u \in s_1\uparrow_R,$$

$$y\cdot E(s_1) \leqslant x\cdot E(s_2).$$

This LP problem has $O(|S|^2)$ variables and $4\cdot|S|+5$, i.e., $O(|S|)$ equations. It is justified as follows. The first two equations correspond to condition (3) in Definition 5, rewritten as:

$$\sum_{u_1 \in U_1} \Delta(u_1,u_2) = \frac{\delta_2(u_2)\cdot \boldsymbol{P}(s_2,u_2)}{K_2} = \boldsymbol{P}(s_2,u_2)\cdot \underbrace{\frac{\delta_2(u_2)}{K_2}}_{=y_{u_2}}$$

and similar for the symmetric condition for u_2. The range of u_1 can be restricted to $u_2\downarrow_R$ as—due to condition (2) of Definition 5—for the other cases $\Delta(u_1,u_2)=0$. The third and fourth equations formalize the requirements for K_i:

$$\sum_{u_i \in S} \delta_i(u_i)\cdot \boldsymbol{R}(s_i,u_i) = K_i\cdot E(s_i), \quad i=1,2.$$

The requirements $x \geqslant 1$ and $y \geqslant 1$ guarantee that

$$0 < K_1 \leqslant 1 \quad \text{and} \quad 0 < K_2 \leqslant 1$$

while the conditions $x \geqslant x_u \geqslant 0$ and $y \geqslant y_u \geqslant 0$ ensure that $0 \leqslant \delta_i(u) \leqslant 1$, for $i=1,2$. Finally, the last inequality is obtained by rewriting the rate condition:

$$K_1\cdot E(s_1) \leqslant K_2\cdot E(s_2) \quad \text{by}$$

$$\frac{1}{K_2}\cdot E(s_1) \leqslant \frac{1}{K_1}\cdot E(s_2).$$

It is not difficult to see that any solution to the above LP problem induces components δ_i, U_i, V_i, K_i and Δ such that the conditions in Definition 5 are fulfilled. Vice versa, components δ_i, U_i, V_i, K_i and Δ as in Definition 5 induce a solution of the above linear program.

Example 8. The linear equations obtained for checking whether s_1 is simulated by s_1' for the CTMCs in Fig. 3 given that R equals $\sqsubseteq$ are as follows. For illustration purposes we use a particular solution (witnessing that $s_1 \sqsubseteq s_1'$) in our explanations. The state space is comprised of the disjoint union of the two CTMCs. The variables solving the system are: $x=\frac{4}{3}$, $y=1$, $x_{s_1}=0$, $x_{s_2}=x_{s_4}=x=\frac{4}{3}$, $x_{s_3}=\frac{2}{3}$, and $y_{s_1'}=y_{s_2'}=y_{s_3'}=y_{s_4'}=y=1$. For any other state t, x_t and y_t equal 0. The side conditions on x, y and x_s and $y_{s'}$ and the rate condition (last equation, i.e., $1\cdot 2 \leqslant \frac{4}{3}\cdot 6$) are straightforwardly fulfilled. The condition on the exit rate of state s_1' (fourth equation) amounts to

$$y_{s_2'}\cdot \boldsymbol{R}(s_1',s_2') + y_{s_3'}\cdot \boldsymbol{R}(s_1',s_3') = E(s_1')$$

which is satisfied as $y_{s_2'} = y_{s_3'}$. In a similar way we obtain for the exit rate condition of s_1 (third equation):

$$x_{s_2}\cdot \boldsymbol{R}(s_1,s_2) + x_{s_3}\cdot \boldsymbol{R}(s_1,s_3) = E(s_1).$$

To illustrate the weight function condition, consider the second equation. The interesting cases occur for s_1 through s_4. For $u_1=s_1$ we obtain:

$$\underbrace{\sum_{u_2 \in \{s_1'\}} \Delta(s_1,u_2)}_{=0} = \underbrace{\boldsymbol{P}(s_1,s_1)\cdot x_{s_1}}_{=0}.$$

A similar equation is obtained for state s_4. For $u_1=s_2$ we yield, using $s_2\uparrow_R = \{s_2', s_4'\}$:

$$\underbrace{\Delta(s_2,s_2')}_{=\frac{2}{3}} + \underbrace{\Delta(s_2,s_4')}_{=0} = \underbrace{\boldsymbol{P}(s_1,s_2)\cdot x_{s_2}}_{=\frac{1}{2}\cdot\frac{4}{3}}.$$

Finally, it can be checked that for s_3 with $s_3\uparrow_R = \{s_3'\}$ the obtained equation is also satisfied. To summarize, the LP problem has a solution, and therefore s_1' simulates s_1 under R.

Using efficient well-known methods for solving LP problems, the test whether a state weakly simulates another one can be performed in polynomial time. Note that it suffices to check whether the LP problem

has a solution; the solution itself is not needed. In the main algorithm where pairs (s_1, s_2) are successively removed from R, the number of iterations is bounded $|S|^2$. Thus, one has to solve $|S|^2$ LP problems, each of size quadratic in S and solvable in polynomial time. To summarize, the weak simulation preorder on CTMCs can be computed in polynomial time. The weak simulation preorder of a DTMC (see [4]) can be computed with a slightly adapted version of this algorithm.

Note that *any* solution of the above linear inequality system is sufficient for our purposes (i.e., yields the components δ_1, δ_2 and Δ and the derived sets U_i, V_i and values K_i). That is, we do no have to solve an optimization problem. However, it is well known that the problem of solving linear programming problems and the problem of solving linear inequalities are polynomial equivalent, i.e., can be solved with the same algorithms plus a polynomial-time transformation [24,27].

For computing the *strong* simulation preorder one can use the same schema (Algorithm 1) where a network flow algorithm can be applied to check whether the current relation R is a strong simulation [2]. The question arises whether for the weak simulation preorder, the presented linear inequalities can be rewritten as a network flow problem (and thus, can be solved with simpler algorithms than those for solving general linear programming problems). Of course, under certain circumstances, this is possible, e.g., for CTMCs without stutter steps as then the strong and weak simulation preorder coincide. However, in general, the above linear inequalities do not have the form of a network flow problem. The question whether there is general technique that transforms the above linear inequality system into a network flow problem (or another problem-type for which simpler solutions exists) is open.

5. Related work

Decision algorithms for equivalences and preorders have been reported in the literature for various variants of Markov chains. Checking lumpability (or: strong bisimulation) on Markov chains can be done in time $O(m \cdot \log n)$, where n is the number of states and m is the number of transitions [11]. This algorithm can also be employed for weak bisimulation. In the discrete-time case, checking strong bisimulation takes $O(m \cdot \log n)$ time [16], whereas weak bisimulation takes $O(n^3)$ time [1,3]. The incorporation of nondeterminism may yield an exponential time complexity [10], however, for special models with probability and nondeterminism or variants of weak bisimulation, polynomial-time algorithms can be established [6,10, 25]. The computation of the strong simulation preorder, i.e., a stuttering-free simulation, for DTMCs (and Markov decision processes) can be reduced to a maximum flow problem [2] and has a worst case time complexity of $O((m \cdot n^6 + m^2 \cdot n^3)/\log n)$.

6. Conclusions

This paper presented a polynomial-time algorithm for computing the weak simulation preorder ($\sqsubseteq$) of a finite-state Markov chain. The crux of our algorithm is to consider the check whether a state simulates another one as a linear programming problem. Improvements to our basic algorithm are not considered here, but we expect that techniques from, e.g., [12,14,28], can be employed to speed up the algorithm.

Weak simulation has some interesting properties. The kernel of $\sqsubseteq$, i.e., $\sqsubseteq \cap \sqsubseteq^{-1}$, is coarser than weak bisimulation. Moreover, $\sqsubseteq$ preserves bounds on probabilistic reachability properties in the following sense [5]. Let $T \subseteq S$ be a nonempty set of states, $s_1, s_2 \in S$ be states of the CTMC and d a positive real number. Then:

$$s_1 \sqsubseteq s_2 \quad \Rightarrow \quad \mathrm{Prob}\big(s_1 \overset{\leqslant d}{\rightsquigarrow} T\big) \leqslant \mathrm{Prob}\big(s_2 \overset{\leqslant d}{\rightsquigarrow} T\big),$$

where $\mathrm{Prob}(s \overset{\leqslant d}{\rightsquigarrow} T)$ denotes the probability to reach some state in T within d time units when starting in state s.

References

[1] S. Andova, J. Baeten, Abstraction in probabilistic process algebra, in: T. Margaria, W. Yi (Eds.), Tools and Algorithms for the Construction and Analysis of Systems, in: Lecture Notes in Comput. Sci., Vol. 2031, Springer, Berlin, 2001, pp. 204–219.

[2] C. Baier, B. Engelen, M. Majster-Cederbaum, Deciding bisimilarity and similarity for probabilistic processes, J. Comput. System Sci. 60 (1) (2000) 187–231.

[3] C. Baier, H. Hermanns, Weak bisimulation for fully probabilistic systems, in: O. Grumberg (Ed.), Computer-Aided Verification, in: Lecture Notes in Comput. Sci., Vol. 1256, Springer, Berlin, 1997, pp. 119–130.

[4] C. Baier, H. Hermanns, J.-P. Katoen, V. Wolf, Comparative branching-time semantics for Markov chains, in: R. de Simone, D. Lugiez (Eds.), Concurrency Theory, in: Lecture Notes in Comput. Sci., Vol. 2761, Springer, Berlin, 2003, pp. 492–508.
[5] C. Baier, J.-P. Katoen, H. Hermanns, B. Haverkort, Simulation for continuous-time Markov chains, in: L. Brim, et al. (Eds.), Concurrency Theory, in: Lecture Notes in Comput. Sci., Vol. 2421, Springer, Berlin, 2002, pp. 338–354.
[6] C. Baier, M. Stoelinga, Norm functions for probabilistic bisimulations with delays, in: J. Tiuryn (Ed.), Found. of Software Science and Computation Structures, in: Lecture Notes in Comput. Sci., Vol. 1784, Springer, Berlin, 2000, pp. 1–16.
[7] M. Bravetti, Revisiting interactive Markov chains, in: W. Vogler, K.G. Larsen (Eds.), Models for Time-Critical Systems, in: BRICS Notes Series NS-02-3, 2002, pp. 60–80.
[8] M. Brown, E. Clarke, O. Grumberg, Characterizing finite Kripke structures in propositional temporal logic, Theoret. Comput. Sci. 59 (1988) 115–131.
[9] P. Buchholz, Exact and ordinary lumpability in finite Markov chains, J. Appl. Probab. 31 (1994) 59–75.
[10] S. Cattani, R. Segala, Decision algorithms for probabilistic bisimulation, in: L. Brim, et al. (Eds.), Concurrency Theory, in: Lecture Notes in Comput. Sci., Vol. 2421, Springer, Berlin, 2002, pp. 371–385.
[11] S. Derisavi, H. Hermanns, W.H. Sanders, Optimal state-space lumping in Markov chains, Inform. Process. Lett. 87 (6) (2003) 309–315.
[12] R. Gentilini, C. Piazza, A. Policriti, Simulation as coarsest partition problem, in: J.-P. Katoen, P. Stevens (Eds.), Tools and Algorithms for the Construction and Analysis of Systems, in: Lecture Notes in Comput. Sci., Vol. 2280, Springer, Berlin, 2002, pp. 415–430.
[13] R.J. van Glabbeek, W.P. Weijland, Branching time and abstraction in bisimulation semantics, J. ACM 43 (3) (1996) 555–600.
[14] M.R. Henzinger, T. Henzinger, P.W. Kopke, Computing simulations on finite and infinite graphs, in: IEEE Symp. on Foundation of Comp. Sci., 1995, pp. 453–462.
[15] J. Hillston, A Compositional Approach to Performance Modelling, Cambridge Univ. Press, Cambridge, 1996.
[16] T. Hyunh, L. Tian, On some equivalence relations for probabilistic processes, Fund. Inform. 17 (1992) 211–234.
[17] B. Jonsson, Simulations between specifications of distributed systems, in: J.C.M. Baeten, J.F. Groote (Eds.), Concurrency Theory, in: Lecture Notes in Comput. Sci., Vol. 527, Springer, Berlin, 1991, pp. 346–360.
[18] C. Jones, G. Plotkin, A probabilistic powerdomain of evaluations, in: IEEE Symp. on Logic in Computer Science, 1989, pp. 186–195.
[19] B. Jonsson, K.G. Larsen, Specification and refinement of probabilistic processes, in: IEEE Symp. on Logic in Computer Science, 1991, pp. 266–277.
[20] J.G. Kemeny, J.L. Snell, Finite Markov Chains, Van Nostrand, New York, 1960.
[21] V.G. Kulkarni, Modeling and Analysis of Stochastic Systems, Chapman & Hall, London, 1995.
[22] R. Milner, Communication and Concurrency, Prentice-Hall, Englewood Cliffs, NJ, 1989.
[23] R. Paige, R. Tarjan, Three partition refinement algorithms, SIAM J. Comput. 16 (6) (1987) 973–989.
[24] C.H. Papadimitriou, K. Steiglitz, Combinatorial Optimization: Algorithms and Complexity, Dover, Mineola, NY, 1998.
[25] A. Philippou, I. Lee, O. Sokolsky, Weak bisimulation for probabilistic systems, in: C. Palamidessi (Ed.), Concurrency Theory, in: Lecture Notes in Comput. Sci., Vol. 1877, Springer, Berlin, 2000, pp. 334–349.
[26] M.L. Puterman, Markov Decision Processes: Discrete Stochastic Dynamic Programming, John Wiley & Sons, New York, 1994.
[27] A. Schrijver, Theory of Linear and Integer Programming, John Wiley & Sons, New York, 1986.
[28] L. Tan, R. Cleaveland, Simulation revisited, in: T. Margaria, W. Yi (Eds.), Tools and Algorithms for the Construction and Analysis of Systems, in: Lecture Notes in Comput. Sci., Vol. 2031, Springer, Berlin, 2002, pp. 480–495.