A Local Algorithm for Checking Probabilistic Bisimilarity

Yuxin Deng
Shanghai Jiao Tong University, China
Email: deng-yx@cs.sjtu.edu.cn

Wenjie Du
Shanghai Normal University, China
Email: wenjiedu@shnu.edu.cn

***Abstract*—Bisimilarity is one of the most important relations for comparing the behaviour of formal systems in concurrency theory. Decision algorithms for bisimilarity in finite state systems are usually classified into two kinds: global algorithms are generally efficient but require to generate the whole state spaces in advance, and local algorithms combine the verification of a system's behaviour with the generation of the system's state space, which is often more effective to determine that one system fails to be related to another. Although local algorithms are well established in the classical concurrency theory, the study of local algorithms in probabilistic concurrency theory is not mature. In this paper we propose a polynomial time local algorithm for checking probabilistic bisimilarity. With mild modification, the algorithm can be easily adapted to decide probabilistic similarity with the same time complexity.**

***Keywords*-concurrency; probabilistic bisimilarity; local algorithm; probabilistic labelled transition systems;**

I. Introduction

In the last three decades a wealth of behavioural equivalences have been proposed in concurrency theory. Among them, *bisimilarity* [38], [41] is probably the most studied one as it admits a suitable semantics and an elegant co-inductive proof technique. It can also be given an efficient decision procedure [40]. Given a labelled transition system (LTS) with n states and m transitions, the partition refinement algorithm of Paige and Tarjan takes time $O(m \log n)$ to generate all bisimulation equivalence classes. This algorithm belongs to a class of *global* algorithms (e.g. [19], [22]), as classified in [10], which require an LTS to be fully generated a priori. However, in many cases, one may be able to determine that one process fails to be related to another by examining only a fraction of the state space. One would like to have a verification algorithm that exploits this fact. Another class of algorithms, called *local* or *"on the fly"* algorithms (e.g. [21], [6], [33], [35]), combine the verification of a system's behaviour with the generation of the system's state space.

Fernandez and Mouier [21] first proposed an "on the fly" algorithm for checking behavioural equivalences and preorders. Let $\mathcal{L}_1$ and $\mathcal{L}_2$ be two LTSs with initial states s_0 and t_0. To decide if s_0 is bisimilar to t_0, the algorithm of [21] performs depth-first searches (DFS for short) on the product LTS $\mathcal{L}_1||\mathcal{L}_2$. During a round of DFS, it is possible to reach a state, say $(s||t)$, which has already been visited because of a loop, but not yet analyzed. In this case, s and t are *assumed* to be bisimilar and the DFS continues. If the two states are found to be not bisimilar after finishing searching the loop, then we know that a wrong assumption was used. So another round of DFS has to be performed, with one piece of new information, namely the two states are not bisimilar. Moreover, the basic idea illustrated above is also applicable to checking similarity. In [33] Lin lifted Fernandez and Mouier's algorithm to handle value-passing processes. The idea was later on used in [13] to verify open bisimulation.

In recent years, probabilistic constructs have been proven useful for giving quantitative specifications of system behaviour. The first papers on probabilistic concurrency theory [23], [8], [32] proceed by *replacing* nondeterministic with probabilistic constructs. The reconciliation of nondeterministic and probabilistic constructs starts with [24] and has received a lot of attention in the literature [48], [44], [34], [43], [25], [36], [4], [29], [39], [9], [46], [37], [16], [17], [15]. We shall also work in a framework that features the co-existence of probability and nondeterminism.

Decision algorithms for probabilistic bisimilarity and similarity have been studied in [18], [2], [1], [3], [49] for fully probabilsitic processes and in [26], [5], [1], [49] for nondeterministic probabilistic processes. However, all these algorithms are global because they require the whole state space of a system to be fully generated in advance. In [47] a local algorithm in the style of [33] is proposed to decide probabilistic bisimilarity, but it is tailored to check equivalence relations and thus cannot handle probabilistic similairty which is only a preorder relation.

In this paper we propose a local algorithm of checking probabilistic bisimilarity for finitary probabilistic labelled transition systems (pLTSs) which admit both probability and nondeterminism. The basical idea is from [21] but we adapt the algorithm of [33] by adding a procedure for comparing two distributions. This is motivated from the definition of probabilistic bisimulation. To see if two states s, t are related by a probabilistic bisimulation relation $\mathcal{R}$, we need to check that each transition $s \xrightarrow{a} \Delta$ must be matched by some transition $t \xrightarrow{a} \Theta$ such that Δ and Θ are related by a relation $\mathcal{R}^\dagger$ which is lifted from $\mathcal{R}$ to distributions. Inspired by [1], we check the validity of $\Delta\ \mathcal{R}^\dagger\ \Theta$ via a network based technique by checking if the maximum flow of an appropriately constructed network $\mathcal{N}(\Delta, \Theta, \mathcal{R})$ is 1. In the worst case, our algorithm runs in time $O(n^7/\log n)$ and space $O(n^2)$.

In contrast to [47], our algorithm can be easily adapted to check probabilistic simialrity, while keeping the time and space complexities unchanged.

Outline of the paper: The paper proceeds by recalling the definition of probabilistic similarity and bisimilarity and some basic properties in Section II. In Section III we present a local algorithm for checking probabilistic bisimilarity and its adaption for checking probabilistic similarity. Section IV concludes the paper.

II. Probabilistic bisimulation

A (discrete) probability distribution over a set S is a function $\Delta\colon S \to [0,1]$ with $\sum_{s\in S}\Delta(s) = 1$; the *support* of Δ is given by $\lceil\Delta\rceil = \{\, s\in S \mid \Delta(s) > 0\,\}$. We write $\mathcal{D}(S)$, ranged over by Δ, Θ, Φ, for the set of all distributions over S with finite support; these finite distributions are sufficient for the results of this paper. We also write $\overline{s}$ to denote the point distribution assigning probability 1 to s and 0 to all others, so that $\lceil\overline{s}\rceil = \{s\}$. If $p_i \geq 0$ and Δ_i is a distribution for each i in some finite index set I, and $\sum_{i\in I} p_i = 1$, then the probability distribution $\sum_{i\in I} p_i\cdot\Delta_i \in \mathcal{D}(S)$ is given by

$$(\sum_{i\in I} p_i\cdot\Delta_i)(s) \quad = \quad \sum_{i\in I} p_i\cdot\Delta_i(s)\ ; \tag{1}$$

we will sometimes write it as $p_1\cdot\Delta_1 + \ldots + p_n\cdot\Delta_n$ when the index set I is $\{1,\ldots,n\}$.

We now give the probabilistic generalisation of labelled transition systems:

Definition 2.1: A *probabilistic labelled transition system* (pLTS)[1] is a triple $\langle S, L, \to\rangle$, where

1) S is a set of states,
2) L is a set of transition labels,
3) $\to$ is a subset of $S\times L\times\mathcal{D}(S)$.

As with LTSs, we usually write $s \xrightarrow{\alpha} \Delta$ for $(s,\alpha,\Delta)\in\to$, $s\xrightarrow{\alpha}$ for $\exists\Delta : s\xrightarrow{\alpha}\Delta$ and $s\to$ for $\exists\alpha\colon s\xrightarrow{\alpha}$. An LTS may be viewed as a degenerate pLTS, one in which only point distributions are used. A pLTS is *finitely branching* if the set $\{\Delta \mid s\xrightarrow{\alpha}\Delta, \alpha\in L\}$ is finite; if moreover S is finite, then the pLTS is *finitary*.

In the probabilistic setting, the definitions of bisimulation-like equivalences are somewhat complicated by the fact that transitions go from states to distributions (see e.g. [31]). So we need to lift relations between states to relations between distributions (see e.g. [12]).

Definition 2.2: Given two sets S and T and a relation $\mathcal{R}\subseteq S\times T$. We lift $\mathcal{R}$ to a relation $\mathcal{R}^\dagger\subseteq\mathcal{D}(S)\times\mathcal{D}(T)$ by letting $\Delta\ \mathcal{R}^\dagger\ \Theta$ whenever

1) $\Delta = \sum_{i\in I} p_i\cdot\overline{s_i}$, where I is a countable index set and $\sum_{i\in I} p_i = 1$
2) For each $i\in I$ there is a state t_i such that $s_i\ \mathcal{R}\ t_i$
3) $\Theta = \sum_{i\in I} p_i\cdot\overline{t_i}$.

Note that in the decomposition of Δ, the states s_i are not necessarily distinct: that is, the decomposition is not in general unique, and similarly for the decomposition of Θ. For example, if $\mathcal{R} = \{(s_1,t_1),(s_1,t_2),(s_2,t_3),(s_3,t_3)\}$, $\Delta = \frac{1}{2}\overline{s_1} + \frac{1}{4}\overline{s_2} + \frac{1}{4}\overline{s_3}$, and $\Theta = \frac{1}{3}\overline{t_1} + \frac{1}{6}\overline{t_2} + \frac{1}{2}\overline{t_3}$, then $\Delta\ \mathcal{R}^\dagger\ \Theta$ holds because of the decompositions $\Delta = \frac{1}{3}\overline{s_1} + \frac{1}{6}\overline{s_1} + \frac{1}{4}\overline{s_2} + \frac{1}{4}\overline{s_3}$ and $\Theta = \frac{1}{3}\overline{t_1} + \frac{1}{6}\overline{t_2} + \frac{1}{4}\overline{t_3} + \frac{1}{4}\overline{t_3}$.

From the above definition, the next two properties follow. In fact, they are sometimes used in the literature as alternative methods of lifting relations (see e.g. [44], [31]).

Proposition 2.3:
1) Let Δ and Θ be distributions over S and T, respectively. Then $\Delta\ \mathcal{R}^\dagger\ \Theta$ iff there exists a weight function $w : S\times T\to[0,1]$ such that
   a) $\forall s\in S : \sum_{t\in T} w(s,t) = \Delta(s)$
   b) $\forall t\in T : \sum_{s\in S} w(s,t) = \Theta(t)$
   c) $\forall (s,t)\in S\times T : w(s,t) > 0 \Rightarrow s\ \mathcal{R}\ t$.
2) Let Δ, Θ be distributions over S and $\mathcal{R}$ be an equivalence relation. Then $\Delta\ \mathcal{R}^\dagger\ \Theta$ iff $\Delta(C) = \Theta(C)$ for all equivalence classes $C\in S/\mathcal{R}$, where $\Delta(C)$ stands for the accumulated probability $\sum_{s\in C}\Delta(s)$.

Proof:

1) ($\Rightarrow$) Suppose $\Delta\ \mathcal{R}^\dagger\ \Theta$. By Definition 2.2, we can decompose Δ and Θ such that $\Delta = \sum_{i\in I} p_i\cdot\overline{s_i}$, $\Theta = \sum_{i\in I} p_i\cdot\overline{t_i}$, and $s_i\ \mathcal{R}\ t_i$ for all $i\in I$. We define the weight function w by letting $w(s,t) = \sum\{p_i \mid s_i = s, t_i = t, i\in I\}$ for any $s\in S, t\in T$. This weight function can be checked to meet our requirements.
   a) For any $s\in S$, it holds that
   $$\begin{array}{ll} & \sum_{t\in T} w(s,t) \\ = & \sum_{t\in T}\sum\{p_i \mid s_i = s, t_i = t, i\in I\} \\ = & \sum\{p_i \mid s_i = s, i\in I\} \\ = & \Delta(s) \end{array}$$
   b) Similarly, we have $\sum_{s\in S} w(s,t) = \Theta(t)$.
   c) For any $s\in S, t\in T$, if $w(s,t) > 0$ then there is some $i\in I$ such that $p_i > 0$, $s_i = s$, and $t_i = t$. It follows from $s_i\ \mathcal{R}\ t_i$ that $s\ \mathcal{R}\ t$.

   ($\Leftarrow$) Suppose there is a weight function w satisfying the three conditions in the hypothesis. We construct the index set $I = \{(s,t) \mid w(s,t) > 0, s\in S, t\in T\}$ and probabilities $p_{(s,t)} = w(s,t)$ for each $(s,t)\in I$.
   a) It holds that $\Delta = \sum_{(s,t)\in I} p_{(s,t)}\cdot\overline{s}$ because, for any $s\in S$,
   $$\begin{array}{ll} & (\sum_{(s,t)\in I} p_{(s,t)}\cdot\overline{s})(s) \\ = & \sum_{(s,t)\in I} w(s,t) \\ = & \sum\{w(s,t) \mid w(s,t) > 0, t\in T\} \\ = & \sum\{w(s,t) \mid t\in T\} \\ = & \Delta(s) \end{array}$$

[1]Essentially the same model has appeared in the literature under different names such as *NP-systems* [27], *probabilistic processes* [28], *simple probabilistic automata* [43], *probabilistic transition systems* [29] etc. Furthermore, there are strong structural similarities with *Markov Decision Processes* [42], [17].

b) Similarly, we have $\Theta = \sum_{(s,t)\in I} w(s,t) \cdot \overline{t}$.
c) For each $(s,t) \in I$, we have $w(s,t) > 0$, which implies $s \mathcal{R} t$.

Hence, the above decompositions of Δ and Θ meet the requirement of the lifting $\Delta \mathcal{R}^\dagger \Theta$.

2) ($\Rightarrow$) Suppose $\Delta \mathcal{R}^\dagger \Theta$. By Definition 2.2, we can decompose Δ and Θ such that $\Delta = \sum_{i\in I} p_i \cdot \overline{s_i}$, $\Theta = \sum_{i\in I} p_i \cdot \overline{t_i}$, and $s_i \mathcal{R} t_i$ for all $i \in I$. For any equivalence class $C \in S/\mathcal{R}$, we have that

$$\begin{array}{rcl}\Delta(C) & = & \sum_{s\in C} \Delta(s) \\ & = & \sum_{s\in C} \sum\{p_i \mid i \in I, s_i = s\} \\ & = & \sum\{p_i \mid i \in I, s_i \in C\} \\ & = & \sum\{p_i \mid i \in I, t_i \in C\} \\ & = & \Theta(C)\end{array}$$

where the equality in the third line is justified by the fact that $s_i \in C$ iff $t_i \in C$ since $s_i \mathcal{R} t_i$ and $C \in S/\mathcal{R}$.

($\Leftarrow$) Suppose, for each equivalence class $C \in S/\mathcal{R}$, it holds that $\Delta(C) = \Theta(C)$. We construct the index set $I = \{(s,t) \mid s \mathcal{R} t \text{ and } s,t \in S\}$ and probabilities $p_{(s,t)} = \frac{\Delta(s)\Theta(t)}{\Delta([s]_\mathcal{R})}$ for each $(s,t) \in I$, where $[s]_\mathcal{R}$ stands for the equivalence that contains s.

a) It holds that $\Delta = \sum_{(s,t)\in I} p_{(s,t)} \cdot \overline{s}$ because, for any $s \in S$,

$$\begin{array}{rl} & (\sum_{(s,t)\in I} p_{(s,t)} \cdot \overline{s})(s) \\ = & \sum_{(s,t)\in I} p_{(s,t)} \\ = & \sum\{\frac{\Delta(s)\Theta(t)}{\Delta([s]_\mathcal{R})} \mid s \mathcal{R} t,\ t \in S\} \\ = & \sum\{\frac{\Delta(s)\Theta(t)}{\Delta([s]_\mathcal{R})} \mid t \in [s]_\mathcal{R}\} \\ = & \frac{\Delta(s)}{\Delta([s]_\mathcal{R})} \sum\{\Theta(t) \mid t \in [s]_\mathcal{R}\} \\ = & \frac{\Delta(s)}{\Delta([s]_\mathcal{R})} \Theta([s]_\mathcal{R}) \\ = & \frac{\Delta(s)}{\Delta([s]_\mathcal{R})} \Delta([s]_\mathcal{R}) \\ = & \Delta(s)\end{array}$$

b) Similarly, we have $\Theta = \sum_{(s,t)\in I} p_{(s,t)} \cdot \overline{t}$.
c) For each $(s,t) \in I$, we have $s \mathcal{R} t$.

Hence, the above decompositions of Δ and Θ meet the requirement of the lifting $\Delta \mathcal{R}^\dagger \Theta$.

■

Definition 2.4: A binary relation $\mathcal{R} \subseteq S \times S$ is a *simulation* if whenever $s \mathcal{R} t$:

- if $s \xrightarrow{a} \Delta$, there exists some Θ such that $t \xrightarrow{a} \Theta$ and $\Delta \mathcal{R}^\dagger \Theta$.

The relation $\mathcal{R}$ is a *bisimulation* if both $\mathcal{R}$ and $\mathcal{R}^{-1}$ are simulations. *Similarity* (resp. *Bisimilarity*), written $\preceq$ (resp. $\sim$), is the union of all simulations (resp. bisimulations).

Bisimilarity can be approximated by a family of inductively defined relations. Similarity can be approximated in a similar way.

Definition 2.5: Let S be the state set of an LTS. We define:

- $\sim_0 := S \times S$
- $s \sim_{n+1} t$, for $n \geq 0$, if
  1) if $s \xrightarrow{a} \Delta$, there exists some Θ such that $t \xrightarrow{a} \Theta$ and $\Delta \sim_n{}^\dagger \Theta$;
  2) if $t \xrightarrow{a} \Theta$, there exists some Δ such that $s \xrightarrow{a} \Delta$ and $\Delta \sim_n{}^\dagger \Theta$.
- $\sim_\omega := \bigcap_{n\geq 0} \sim_n$

In general, $\sim$ is a strictly finer relation than $\sim_\omega$. However, the two relations coincide when limited to finitely branching pLTSs.

Proposition 2.6: On finitely branching pLTSs, $\sim_\omega$ coincides with $\sim$.

Proof: It is trivial to show by induction that $s \sim t$ implies $s \sim_n t$ for all $n \geq 0$, thus $s \sim_\omega t$.

Now we show that $\sim_\omega$ is a bisimulation. Suppose $s \sim_\omega t$ and $s \xrightarrow{a} \Delta$. We have to show that there is some Θ with $t \xrightarrow{a} \Theta$ and $\Delta \sim_\omega{}^\dagger \Theta$. Consider the set

$$T := \{\Theta \mid t \xrightarrow{a} \Theta \wedge \Delta \not\sim_\omega{}^\dagger \Theta\}.$$

For each $\Theta \in T$, we have $\Delta \not\sim_\omega{}^\dagger \Theta$, which means that there is some $n_\Theta > 0$ with $\Delta \not\sim_{n_\Theta}{}^\dagger \Theta$. Since t is finitely branching, T is a finite set. Let $N = max\{n_\Theta \mid \Theta \in T\}$. It holds that $\Delta \not\sim_N{}^\dagger \Theta$ for all $\Theta \in T$, since by a straightforward induction on m we can show that $s \sim_n t$ implies $s \sim_m t$ for all $m, n \geq 0$ with $n > m$. By the assumption $s \sim_\omega t$ we know that $s \sim_{N+1} t$. It follows that there is some Θ with $t \xrightarrow{a} \Theta$ and $\Delta \sim_N{}^\dagger \Theta$, so $\Theta \notin T$ and hence $\Delta \sim_\omega{}^\dagger \Theta$. By symmetry we also have that if $t \xrightarrow{a} \Theta$ then there is some Δ with $s \xrightarrow{a} \Delta$ and $\Delta \sim_\omega{}^\dagger \Theta$. ■

In the sequel, we consider finitary LTSs, which are finitely branching, thus allow us to use the above proposition.

III. Checking probabilistic bisimulation

We see from Definition 2.4 that to check if a relation $\mathcal{R}$ is a bisimulation we need to check if two distributions are related by a lifted relation $\mathcal{R}^\dagger$. The latter can be solved by using network-based techniques, as already observed in [1].

Networks: We briefly recall the basic definitions of networks. More details can be found in e.g. [20]. A *network* is a tuple $\mathcal{N} = (N, E, \bot, \top, c)$ where (N, E) is a finite directed graph (i.e. N is a set of nodes and $E \subseteq N \times N$ is a set of edges) with two special nodes $\bot$ (the *source*) and $\top$ (the *sink*) and a *capability* c, i.e. a function that assigns to each edge $(v, w) \in E$ a non-negative number $c(v, w)$. A *flow function* f for $\mathcal{N}$ is a function that assigns to edge e a real number $f(e)$ such that

- $0 \leq f(e) \leq c(e)$ for all edges e.
- Let $in(v)$ be the set of incoming edges to node v and $out(v)$ the set of outgoing edges from node v. Then, for each node $v \in N \backslash \{\bot, \top\}$,

$$\sum_{e\in in(v)} f(e) = \sum_{e\in out(v)} f(e).$$

The *flow* $F(f)$ of f is given by

$$F(f) \ = \sum_{e\in out(\bot)} f(e) - \sum_{e\in in(\bot)} f(e).$$

The *maximum flow* in $\mathcal{N}$ is the supremum (maximum) over the flows $F(f)$, where f is a flow function in $\mathcal{N}$.

The test whether $\Delta\ \mathcal{R}^\dagger\ \Theta$*:* Suppose $\mathcal{R}\subseteq S\times S$ and $\Delta,\Theta\in\mathcal{D}(S)$. We will see that the question whether $\Delta\ \mathcal{R}^\dagger\ \Theta$ can be reduced to a maximum flow problem in a suitably chosen network. Let $S'=\{s'\mid s\in S\}$ where s' are pairwise distinct new states, i.e. $s'\in S'$ for all $s\in S$. We create two states $\bot$ and $\top$ not contained in $S\cup S'$ with $\bot\neq\top$. We associate with the pair (Δ,Θ) the following network $\mathcal{N}(\Delta,\Theta,\mathcal{R})$.

- The nodes are $N=S\cup S'\cup\{\bot,\top\}$.
- The edges are $E = \{(s,t')\mid (s,t)\in\mathcal{R}\} \cup \{(\bot,s)\mid s\in S\}\cup\{(s',\top)\mid s\in S\}$.
- The capability c is defined by $c(\bot,s)=\Delta(s)$, $c(t',\top)=\Theta(t)$ and $c(s,t')=1$ for all $s,t\in S$.

The following lemma is taken from Lemma 5.1 of [1].

Lemma 3.1: The following statements are equivalent.

1) There exists a weight function w for (Δ,Θ) with respect to $\mathcal{R}$.
2) The maximum flow in $\mathcal{N}(\Delta,\Theta,\mathcal{R})$ is 1.

Corollary 3.2: $\Delta\ \mathcal{R}^\dagger\ \Theta$ iff the maximum flow in $\mathcal{N}(\Delta,\Theta,\mathcal{R})$ is 1.

Proof: Combining Proposition 2.3(1) and Lemma 3.1. ■

Corollary 3.2 provides a method for deciding whether $\Delta\ \mathcal{R}^\dagger\ \Theta$. We construct the network $\mathcal{N}(\Delta,\Theta,\mathcal{R})$ and compute the maximum flow with well-known methods, as sketched in Algorithm 1.

Algorithm 1 Check$(\Delta,\Theta,\mathcal{R})$

```
Input: A nonempty finite set $S$, distributions
    $\Delta,\Theta\in\mathcal{D}(S)$ and $\mathcal{R}\subseteq S\times S$
Output: If $\Delta\ \mathcal{R}^\dagger\ \Theta$ then "yes" else "no"
Method:
    Construct the network $\mathcal{N}(\Delta,\Theta,\mathcal{R})$
    Compute the maximum flow $F$ in $\mathcal{N}(\Delta,\Theta,\mathcal{R})$
    If $F<1$ then return "no" else "yes".
```

As shown in [7], computing the maximum flow in a network can be done in time $O(n^3/\log n)$ and space $O(n^2)$, where n is the number of nodes in the network. So we immediately have the following result.

Lemma 3.3: The test whether $\Delta\ \mathcal{R}^\dagger\ \Theta$ can be done in time $O(n^3/\log n)$ and space $O(n^2)$.

Checking probabilistic bisimilarity: We now present a bisimilarity-checking algorithm by adapting the algorithm originally proposed in [33] for value-passing processes.

The main procedure in the algorithm is **Bisim**(s,t). It starts with the initial state pair (s,t), trying to find the

Algorithm 2 Bisim(s,t)

```
Bisim$(s,t)=\{$
$NotBisim:=\{\}$
fun Bis$(s,t)$=$\{$
    $Visited:=\{\}$
    $Assumed:=\{\}$
    Match$(s,t)\}$
$\}$ handle $WrongAssumption\Rightarrow$ Bis$(s,t)$
return Bis$(s,t)$

Match$(s,t)=$
$Visited:=Visisted\cup\{(s,t)\}$
$b=\bigwedge_{a\in A}$ MatchAction$(s,t,a)$
if $b=false$ then
  $NotBisim:=NotBisim\cup\{(s,t)\}$
  if $(s,t)\in Assumed$ then
    raise $WrongAssumption$
  end if
end if
return $b$

MatchAction$(s,t,a)=$
for all $s\xrightarrow{a}\Delta_i$ do
  for all $t\xrightarrow{a}\Theta_j$ do
    $b_{ij}=$ MatchDistribution$(\Delta_i,\Theta_j)$
  end for
end for
return $(\bigwedge_i(\bigvee_j b_{ij}))\underline{\wedge(\bigwedge_j(\bigvee_i b_{ij}))}$

MatchDistribution$(\Delta,\Theta)=$
Assume $\lceil\Delta\rceil=\{s_1,...,s_n\}$ and $\lceil\Theta\rceil=\{t_1,...,t_m\}$
$\mathcal{R}:=\{(s_i,t_j)\mid$ Close$(s_i,t_j)=true\}$
return Check$(\Delta,\Theta,\mathcal{R})$

Close$(s,t)=$
if $(s,t)\in NotBisim$ then
  return $false$
else if $(s,t)\in Visited$ then
  $Assumed:=Assumed\cup\{(s,t)\}$
  return $true$
else
  return Match$(s,t)$
end if
```

smallest bisimulation relation containing the pair by matching transitions from each pair of states it reaches. It uses three auxiliary data structures:

- $NotBisim$ collects all state pairs that have already been detected as not bisimilar.
- $Visited$ collects all state pairs that have already been visited.
- $Assumed$ collects all state pairs that have already been visited and assumed to be bisimilar.

The core procedure, **Match**, is called from function **Bis** inside the main procedure **Bisim**. Whenever a new pair of states is encountered it is inserted into $Visited$. If two states fail to match each other's transitions then they are not bisimilar and the pair is added to $NotBisim$. If the current state pair has been visited before, we check whether it is in $NotBisim$. If this is the case, we return $false$. Otherwise, a loop has been detected and we make assumption that the two states are bisimilar, by inserting the pair into $Assumed$, and return $true$. Later on, if we find that the two states are not bisimilar after finishing searching the loop, then the assumption is wrong, so we first add the pair into $NotBisim$ and then raise the exception $WrongAssumption$, which forces the function **Bis** to run again, with the new information that the two states in this pair are not bisimilar. In this case, the size of $NotBisim$ has been increased by at least one. Hence, **Bis** can only be called for finitely many times. Therefore, the procedure **Bisim**(s,t) will terminate. If it returns $true$, then the set $(Visited - NotBisim)$ constitutes a bisimulation relation containing the pair (s,t).

The main difference from the local algorithm of checking non-probabilistic bisimilarity in [33] is the introduction of the procedure **MatchDistribution**(Δ,Θ), where we approximate $\sim$ by a binary relation $\mathcal{R}$ which is coarser than $\sim$ in general, and we check the validity of $\Delta\ \mathcal{R}^\dagger\ \Theta$. If $\Delta\ \mathcal{R}^\dagger\ \Theta$ does not hold, then $\Delta\ \sim^\dagger\ \Theta$ is invalid either and **MatchDistribution**(Δ,Θ) returns *false* correctly. Otherwise, the two distributions Δ and Θ are considered equivalent with respect to $\mathcal{R}$ and we move on to match other pairs of distributions. The correctness of the algorithm is stated in the following theorem.

Theorem 3.4: Given two finitary pLTSs with initial states s_0 and t_0, the function **Bisim**(s_0,t_0) terminates, and it returns *true* if and only if $s_0 \sim t_0$.

Proof: Let **Bis**$_i$ stand for the i-th execution of the function **Bis**. Let $Assumed_i$ and $NotBisim_i$ be the set $Assumed$ and $NotBisim$ at the end of **Bis**$_i$. When **Bis**$_i$ is finished, either a $WrongAssumption$ is raised or no $WrongAssumption$ is raised. In the former case, $Assumed_i \cap NotBisim_i \neq \emptyset$; in the latter case, the execution of the function **Bisim** is completed. From function **Close** we know that $Assumed_i \cap NotBisim_{i-1} = \emptyset$. Now it follows from the simple fact $NotBisim_{i-1} \subseteq NotBisim_i$ that $NotBisim_{i-1} \subset NotBisim_i$. Since we are considering finitary pLTSs, there is some j such that $NotBisim_{j-1} = NotBisim_j$, when all the non-bisimilar state pairs reachable from s_0 and t_0 have been found and **Bisim** must terminate.

For the correctness of the algorithm, we consider the relation $\mathcal{R}_i = Visited_i - NotBisim_i$, where $Visited_i$ is the set $Visited$ at the end of **Bis**$_i$. Let **Bis**$_k$ be the last execution of **Bis**. For each $i \leq k$, the relation $\mathcal{R}_i$ can be regarded as an approximation of $\sim$, as far as the states appeared in $\mathcal{R}_i$ are concerned. Moreover, $\mathcal{R}_i$ is a coarser approximation because if two states s,t are re-visited but their relation is unknown, they are assumed to be bisimilar. Therefore, if **Bis**$_k(s_0,t_0)$ returns $false$, then $s_0 \not\sim t_0$. On the other hand, if **Bis**$_k(s_0,t_0)$ returns $true$, then $\mathcal{R}_k$ constitutes a bisimulation relation containing the pair (s_0,t_0). This follows because **Match**$(s_0,t_0) = true$ which basically means that whenever $s\ \mathcal{R}_k\ t$ and $s \xrightarrow{a} \Delta_i$ there exists some transition $t \xrightarrow{a} \Theta_i$ such that **Check**$(\Delta,\Theta,\mathcal{R}_k) = true$, i.e. $\Delta\ {\mathcal{R}_k}^\dagger\ \Theta$. Indeed, this rules out the possibility that $s_0 \not\sim t_0$ as otherwise we would have $s_0 \not\sim_\omega t_0$ by Proposition 2.6, that is $s_0 \not\sim_n t_0$ for some $n > 0$. The latter means that some transition $s \xrightarrow{a} \Delta$ exists such that for all $t \xrightarrow{a} \Theta$ we have $\Delta \not\sim_{n-1}{}^\dagger\ \Theta$, i.e. Δ and Θ can be distinguished at level n, so a contradiction arises. ■

Below we consider the time and space complexities of the algorithm.

Theorem 3.5: Let s and t be two states in a pLTS with n states in total. The function **Bisim**(s,t) terminates in time $O(n^7/\log n)$ and space $O(n^2)$.

Proof: The number of state pairs is bounded by n^2. In the worst case, each execution of the function **Bis**(s,t) only yields one new pair of states that are not bisimilar. The number of state pairs examined in the first execution of **Bis**(s,t) is at most $O(n^2)$, in the second execution is at most $O(n^2-1)$, $\cdots$. Therefore, the total number of state pairs examined is at most $O(n^2+(n^2-1)+\cdots+1) = O(n^4)$. When a state pair (s,t) is examined, each transition of s is compared with all transitions of t labelled with the same action. Since the pLTS is finitely branching, we could assume that each state has at most c outgoing transitions. Therefore, for each state pair, the number of comparisons of transitions is bound by c^2. As a comparison of two transitions calls the function **Check** once, which requires time $O(n^3/\log n)$ by Lemma 3.3. As a result, examining each state pair takes time $O(c^2n^3/\log n)$. Finally, the worst case time complexity of executing **Bisim**(s,t) is $O(n^7/\log n)$.

The space requirement of the algorithm is easily seen to be $O(n^2)$, in view of Lemma 3.3. ■

Checking probabilistic similarity: With mild modification, the above algorithm can be adapted to check probabilistic similarity. We simply remove the underlined part in the function **MatchAction**; the rest of the algorithm remains unchanged. Similar to the analysis in Theorems 3.4 and 3.5, the new algorithm can be shown to correctly check probabilistic similarity over finitary pLTSs; its worst case time and space complexities are still $O(n^7/\log n)$ and $O(n^2)$, respectively.

IV. Concluding remarks

We have presented a polynomial time local algorithm for checking probabilistic bisimilarity over finitary pLTSs. With mild modification, it can be used to check probabilistic similarity. As far as we know, this is the first time that local algorithms are investigated in the area of probabilistic concurrency theory.

In the nonprobabilistic setting there is an approach to checking bisimilarity between two states by first constructing a *characteristic formula* [45] for one state in the modal μ-calculus [30] and check if the other state satisfies the formula. This approach yields efficient algorithms for checking behavioural relations [11]. In the probabilistic setting, characteristic formulae also exist in the probabilistic modal μ-calculus [14]. We believe that it is promising to check probabilistic bisimilarity in a logical way, along the line of consideration proposed in [11].

There are other local algorithms for checking nonprobabilistic behavioural relations. For example, Celikkan [6] proposed a preorder-checking algorithm by recursively constructing a graph whose vertices are pairs of related states. Mateescu and Oudot [35] proposed an algorithm by first encoding a bisimulation relation as a boolean equation system (BES) and then employing a local BES resolution algorithm. It is unclear if those algorithms can be adapted to the probabilistic setting but remain effective for checking behavioural relations.

Acknowledgment

We thank Dr. Lijun Zhang for the interesting initial discussion that we had. Deng would like to acknowledge the support of the National Natural Science Foundation of China (Grant No. 60703033).

References

[1] C. Baier, B. Engelen, and M. Majster-Cederbaum. Deciding bisimilarity and similarity for probabilistic processes. *Journal of Computer and System Sciences*, 60(1):187–231, 2000.

[2] C. Baier and H. Hermanns. Weak bisimulation for fully probabilistic processes. In *Proc. CAV'97*, volume 1254 of *LNCS*, pages 119–130. Springer, 1997.

[3] C. Baier, H. Hermanns, and J.-P. Katoen. Probabilistic weak simulation is decidable in polynomial time. *Information Processing Letters*, 89(3):123–130, 2004.

[4] E. Bandini and R. Segala. Axiomatizations for probabilistic bisimulation. In *Proc. ICALP'01*, volume 2076 of *LNCS*, pages 370–381. Springer, 2001.

[5] S. Cattani and R. Segala. Decision algorithms for probabilistic bisimulation. In *Proc. CONCUR'02*, volume 2421 of *LNCS*, pages 371–285. Springer, 2002.

[6] U. Celikkan. *Semantic Preorders in the Automated Verification of Concurrent Systems*. PhD thesis, North Carolina State University, 1995.

[7] J. Cheriyan, T. Hagerup, and K. Mehlhorn. Can a maximum flow be computed on O(nm) time? In *Proc. ICALP'90*, volume 443 of *LNCS*, pages 235–248. Springer, 1990.

[8] I. Christoff. Testing equivalences and fully abstract models for probabilistic processes. In *Proc. CONCUR'90*, volume 458 of *LNCS*, pages 126–140. Springer, 1990.

[9] R. Cleaveland, S. P. Iyer, and M. Narasimha. Probabilistic temporal logics via the modal mu-calculus. *Theoretical Computer Science*, 342(2-3):316–350, 2005.

[10] R. Cleaveland and O. Sokolsky. *Equivalence and Preorder Checking for Finite-State Systems*, chapter 12, pages 391–424. North-Holland, 2001.

[11] R. Cleaveland and B. Steffen. Computing behavioural relations, logically. In *Proc. ICALP'91*, volume 510 of *LNCS*, pages 127–138. Springer, 1991.

[12] Y. Deng and W. Du. Probabilistic barbed congruence. *ENTCS*, 190(3):185–203, 2007.

[13] Y. Deng and Y. Fu. Algorithm for verifying strong open bisimulation in full pi-calculus. *Journal of Shanghai Jiaotong University*, E-5(2):147–152, 2001.

[14] Y. Deng and R. van Glabbeek. Characterising probabilistic processes logically, 2009. Submitted.

[15] Y. Deng, R. van Glabbeek, M. Hennessy, and C. Morgan. Characterising testing preorders for finite probabilistic processes. *Logical Methods in Computer Science*, 4(4:4):1–33, 2008.

[16] Y. Deng, R. J. vanGlabbeek, M. Hennessy, C. C. Morgan, and C. Zhang. Remarks on testing probabilistic processes. *ENTCS*, 172:359–397, 2007.

[17] Y. Deng, R. J. vanGlabbeek, C. C. Morgan, and C. Zhang. Scalar outcomes suffice for finitary probabilistic testing. In *Proc. ESOP'07*, volume 4421 of *LNCS*, pages 363–378. Springer, 2007.

[18] S. Derisavi, H. Hermanns, and W. Sanders. Optimal state-space lumping in markov chains. *Information Processing Letters*, 87(6):309–315, 2003.

[19] A. Dovier, C. Piazza, and A. Policriti. An efficient algorithm for computing bisimulation equivalence. *Theoretical Computer Science*, 311(1-3):221–256, 2004.

[20] S. Even. *Graph Algorithms*. Computer Science Press, 1979.

[21] J.-C. Fernandez and L. Mounier. Verifying bisimulations "on the fly". In *Proc. FORTE'90*, pages 95–110. North-Holland, 1990.

[22] K. Fisler and M. Y. Vardi. Bisimulation minimization and symbolic model checking. *Formal Methods in System Design*, 21(1):39–78, 2002.

[23] A. Giacalone, C.-C. Jou, and S. A. Smolka. Algebraic reasoning for probabilistic concurrent systems. In *Proc. IFIP TC2 Working Conference on Programming Concepts and Methods*, pages 443–458. North-Holland, 1990.

[24] H. Hansson and B. Jonsson. A calculus for communicating systems with time and probabilities. In *Proc. IEEE Real-Time Systems Symposium*, pages 278–287. IEEE Computer Society Press, 1990.

[25] M. Huth and M. Kwiatkowska. Quantitative analysis and model checking. In *Proc. LICS'97*, pages 111–122. IEEE Computer Society, 1997.

[26] D. Huynh and L. Tian. On some equivalence relations for probabilistic processes. *Fundamenta Informaticae*, 17(3):211–234, 1992.

[27] B. Jonsson, C. Ho-Stuart, and W. Yi. Testing and refinement for nondeterministic and probabilistic processes. In *Proc. FTRTFT'94*, volume 863 of *LNCS*, pages 418–430. Springer, 1994.

[28] B. Jonsson and W. Yi. Compositional testing preorders for probabilistic processes. In *Proc. LICS'95*, pages 431–441. IEEE Computer Society, 1995.

[29] B. Jonsson and W. Yi. Testing preorders for probabilistic processes can be characterized by simulations. *Theoretical Computer Science*, 282(1):33–51, 2002.

[30] D. Kozen. Results on the propositional mu-calculus. *Theoretical Computer Science*, 27:333–354, 1983.

[31] K. G. Larsen and A. Skou. Bisimulation through probabilistic testing. *Information and Computation*, 94(1):1–28, 1991.

[32] K. G. Larsen and A. Skou. Compositional verification of probabilistic processes. In *Proc. CONCUR'92*, volume 630 of *LNCS*, pages 456–471. Springer, 1992.

[33] H. Lin. "On-the-fly instantiation" of value-passing processes. In *Proc. FORTE'98*, volume 135 of *IFIP Conference Proceedings*, pages 215–230. Kluwer, 1998.

[34] G. Lowe. Probabilistic and prioritized models of timed CSP. *Theoretical Computer Science*, 138:315–352, 1995.

[35] R. Mateescu and E. Oudot. Improved on-the-fly equivalence checking using boolean equation systems. In *Proc. SPIN'08*, volume 5156 of *LNCS*, pages 196–213. Springer, 2008.

[36] A. McIver and C. Morgan. An expectation-based model for probabilistic temporal logic. Technical Report PRG-TR-13-97, Oxford University Computing Laboratory, 1997.

[37] A. McIver and C. Morgan. Results on the quantitative mu-calculus. *ACM Transactions on Computational Logic*, 8(1), 2007.

[38] R. Milner. *Communication and Concurrency*. Prentice Hall, 1989.

[39] M. M. Mislove, J. Ouaknine, and J. Worrell. Axioms for probability and nondeterminism. *ENTCS*, 96:7–28, 2004.

[40] R. Paige and R. E. Tarjan. Three partition refinement algorithms. *SIAM Journal on Computing*, 16(6):973–989, 1987.

[41] D. Park. Concurrency and automata on infinite sequences. In *Proc. the 5th GI Conference*, volume 104 of *LNCS*, pages 167–183. Springer, 1981.

[42] M. L. Puterman. *Markov Decision Processes*. Wiley, 1994.

[43] R. Segala. Modeling and verification of randomized distributed real-time systems. Technical Report MIT/LCS/TR-676, PhD thesis, MIT, 1995.

[44] R. Segala and N. Lynch. Probabilistic simulations for probabilistic processes. In *Proc. CONCUR'94*, volume 836 of *LNCS*, pages 481–496. Springer, 1994.

[45] B. Steffen and A. Ingólfsdóttir. Characteristic formulae for processes with divergence. *Information and Computation*, 110:149–163, 1994.

[46] R. Tix, K. Keimel, and G. Plotkin. Semantic domains for combining probability and non-determinism. *ENTCS*, 129:1–104, 2005.

[47] P. Wu, C. Palamidessi, and H. Lin. Symbolic bisimulations for probabilistic systems. In *Proc. QEST'07*, pages 179–188. IEEE Computer Society, 2007.

[48] W. Yi and K. G. Larsen. Testing probabilistic and nondeterministic processes. In *Proc. PSTV'92*, volume C-8 of *IFIP Transactions*, pages 47–61. North-Holland, 1992.

[49] L. Zhang, H. Hermanns, F. Eisenbrand, and D. Jansen. Flow faster: efficient decision algorithms for probabilistic simulations. *Logical Methods in Computer Science*, 4(4:6):1–42, 2008.