Available online at www.sciencedirect.com
SCIENCE DIRECT®
Theoretical Computer Science 356 (2006) 325–355
Theoretical Computer Science
www.elsevier.com/locate/tcs

Branching bisimulation for probabilistic systems: Characteristics and decidability

Suzana Andova[a], Tim A.C. Willemse[b,c,*]

[a] *Department of Computer Science, Twente University P.O. Box 217, 7500 AE Enschede, The Netherlands*
[b] *Institute for Computing and Information Sciences (ICIS), Radboud University Nijmegen P.O. Box 9010, 6500 GL Nijmegen, The Netherlands*
[c] *Department of Mathematics and Computer Science, Eindhoven University of Technology P.O. Box 513, 5600 MB Eindhoven, The Netherlands*

Abstract

We address the concept of abstraction in the setting of probabilistic reactive systems, and study its formal underpinnings for the *strictly alternating model* of Hansson. In particular, we define the notion of *branching bisimilarity* and study its properties by studying two other equivalence relations, viz. coloured trace equivalence and branching bisimilarity using maximal probabilities. We show that both alternatives coincide with branching bisimilarity. The alternative characterisations have their own merits and focus on different aspects of branching bisimilarity. Coloured trace equivalence can be understood without knowledge of probability theory and is independent of the notion of a *scheduler*. Branching bisimilarity, rephrased in terms of maximal probabilities gives rise to an algorithm of polynomial complexity for deciding the equivalence. Together they give a better understanding of branching bisimilarity. Furthermore, we show that the notions of branching bisimilarity in the alternating model of Hansson and in the non-alternating model of Segala differ: branching bisimilarity in the latter setting turns out to discriminate between systems that are intuitively branching bisimilar.

Keywords: Branching bisimulation; Probabilistic systems; Alternating model; Abstraction

1. Introduction

One of the hallmarks of process theory is the notion of *abstraction*. Abstractions allow one to reason about systems in which details, unimportant to the purposes at hand, have been hidden. It is an invaluable tool when dealing with complex systems. Research in process theory has made great strides in coping with abstraction in areas that focus on functional behaviours of systems. However, when it comes to theories focusing on functional behaviours *and* extra-functional behaviours such as probabilistic behaviour, we suddenly find that many issues are still unresolved.

This paper addresses abstraction in the setting of systems that have both non-deterministic and probabilistic traits, hereafter referred to as *probabilistic systems*. The model that we use throughout this paper to describe such systems is that of *graphs* that adhere to the strictly alternating regime as studied by Hansson [13], rather than the non-alternating model [19,20] as proposed by Segala et al. In particular, we study the notion of *branching bisimilarity* for this model.

* Corresponding author. Institute for Computing and Information Sciences (ICIS), Radboud University Nijmegen P.O. Box 9010, 6500 GL Nijmegen, The Netherlands. Tel.: +31 24 365 2634; fax: +31 24 365 2525.
E-mail address: timw@cs.ru.nl (T.A.C. Willemse).

0304-3975/$ - see front matter © 2006 Elsevier B.V. All rights reserved.
doi:10.1016/j.tcs.2006.02.010

The need for this particular equivalence relation is already convincingly argued by e.g. van Glabbeek and Weijland in [11], and by Groote and Vaandrager in [12]. Recall that branching bisimilarity for probabilistic systems has been defined earlier for the non-alternating model by Segala and Lynch [20] and a variation on that notion was defined by Stoelinga [21]. However, we stress that the differences between the alternating model and the non-alternating model lead to incompatibilities of the notions of branching bisimilarity in both settings. In fact, these differences are a key motivation for our investigation: while our notion of branching bisimulation satisfies the properties commonly attributed to it, the existing notions turn out to be too strict in their current phrasing (as we explain in detail in Section 7), and discriminate between systems that are intuitively branching bisimilar.

Van Glabbeek and Weijland [11] showed that a key property of branching bisimilarity is that it preserves the branching structure of processes, i.e. it preserves computations together with the *potentials* in all intermediate states of a system that are passed through, even when unobservable events are involved. Roughly speaking, the potentials are the options the system has to branch and behave. This property sets branching bisimilarity apart from weak bisimilarity, which does not have the property. They illustrated this property by defining two new equivalences, called *concrete coloured trace equivalence* (in a setting without abstraction) and *coloured trace equivalence* (in a setting with abstraction), which both use colours to code for the potentials. Subsequently, they showed that strong bisimilarity and concrete coloured trace equivalence coincide, proving that colours can indeed be used to code for the potentials of a system. Next, they showed that also branching bisimilarity and coloured trace equivalence coincide, and both are strictly finer than weak bisimilarity. This proved that branching bisimilarity indeed preserves the branching structure of the system.

Although our setting is considerably more complex than the non-probabilistic setting, the key concept of preservation of potentials should still hold. We show that this is indeed the case by defining probabilistic counterparts of concrete coloured trace equivalence and coloured trace equivalence, and show that these coincide with strong bisimilarity and branching bisimilarity, respectively. A major advantage of (concrete) coloured trace equivalence is that it can be understood without knowledge of probability theory and without appealing to schedulers.

Another property of branching bisimilarity (one that is due to the alternating model, and which can also be found for weak bisimilarity [18]), is the preservation of *maximal probabilities*. We show that branching bisimilarity can be rephrased in terms of such maximal probabilities, thus yielding another alternative definition of branching bisimulation. Apart from the more appetising phrasing that this yields, this result is also at the basis of the complexity results for deciding branching bisimilarity. We also provide the algorithm for deciding branching bisimilarity.

Both alternative characterisations of branching bisimulation have their own merits and focus on orthogonal aspects. We emphasise that together, these are instrumental in understanding branching bisimulation and its properties for probabilistic systems.

This paper is outlined as follows. In Section 2, we introduce the semantic model we use in the remainder of this paper, together with the notions of strong bisimulation and branching bisimulation. In Section 3, we prove that branching bisimulation can be rephrased in terms of maximal probabilities, and we discuss the decidability of branching bisimulation in detail. Section 4 formalises the notions of *colours* and *blends*. Then, in Sections 5 and 6 we define *concrete coloured trace equivalence* and *coloured trace equivalence* and we show that these two equivalence relations coincide with strong bisimilarity and branching bisimilarity, respectively. In Section 7 we give an overview of related work, which in turn provides the motivation for conducting this research in the first place. Section 8 summarises the results of this paper and addresses issues for further research.

2. Semantic model

We use *graphs* [1] to model probabilistic systems. The graphs we consider follow the strictly alternating regime of Hansson [13]. They can be used to describe systems that have both non-deterministic and probabilistic characteristics.

Graphs consist of two types of nodes: *probabilistic nodes* and *non-deterministic nodes*. These nodes are connected by two types of directed edges, called *probabilistic transitions* and *non-deterministic transitions*. The latter are labelled with *actions* from a set of action labels, representing atomic activities of a system or with the *unobservable event*, which is denoted τ and which is not part of the set of action labels of any graph. A graph, not containing τ-transitions is referred to as a *concrete graph*. The probabilistic transitions model the probabilistic behaviour of a system. We assume

[1] The model we use is also known as *Labelled Concurrent Markov Chains.*. We use the term *graph* to stay in line with [11].

the existence of a special node nil, which is not part of the set of nodes of any graph. This node is used as a terminal node for all graphs.

Definition 1. A *graph* is a 7-tuple $\langle N, P, s, \mathrm{Act}, \rightarrow, \rightsquigarrow, \mathsf{pr}\rangle$, where
- N is a non-empty finite set of *non-deterministic nodes*. We write N_{nil} for the set $N \cup \{\mathsf{nil}\}$.
- P is a non-empty finite set of *probabilistic nodes*. We write P_{nil} for the set $P \cup \{\mathsf{nil}\}$.
- $s \in P$ is the *initial node*, also called *root*.
- Act is a finite set of *action labels*. We abbreviate the set $\mathrm{Act} \cup \{\tau\}$ with Act_τ.
- $\rightarrow \subseteq N \times \mathrm{Act}_\tau \times P_{\mathsf{nil}}$ is the *non-deterministic transition relation*. We require that for all $n \in N$, there is at least one $(n, a, p) \in \rightarrow$ for some $a \in \mathrm{Act}_\tau$ and $p \in P_{\mathsf{nil}}$.
- $\rightsquigarrow \subseteq P \times N$ is a *probabilistic transition relation*.
- $\mathsf{pr} : \rightsquigarrow \rightarrow (0, 1]$ is a total function for which $\sum_{n \in N} \mathsf{pr}(p, n) = 1$ for all $p \in P$.

We write $n \xrightarrow{a} p$ rather than $(n, a, p) \in \rightarrow$ and $p \rightsquigarrow n$ rather than $(p, n) \in \rightsquigarrow$. The set of all graphs is denoted $\mathbf{G}$. In the remainder of this paper, $\mathsf{x}, \mathsf{y}, \ldots$ range over $\mathbf{G}$. We write N_x, P_x, s_x, etc. for the components of the graph x, and use S_x to denote the union $P_\mathsf{x} \cup N_\mathsf{x}$. We write $S_{\mathsf{nil},\mathsf{x}}$ for the set $S_\mathsf{x} \cup \{\mathsf{nil}\}$. When x is the only graph under consideration, or when no confusion can arise, we drop the subscripts altogether.

As a derived notion, we introduce the *cumulative probability* $\mu : S_{\mathsf{nil}} \times 2^{S_{\mathsf{nil}}} \rightarrow [0, 1]$, which yields the total probability of reaching a set of nodes via probabilistic transitions: $\mu(p, \mathcal{M}) \stackrel{\mathrm{def}}{=} \sum_{n \in \mathcal{M} \cap N} \mathsf{pr}(p, n)$ if $p \in P$ and 0 otherwise.

There are several variations on the graph model that we use throughout this paper. In [18], a more liberal version is considered, in which the alternation between probabilistic transitions and non-deterministic transitions is not as strict as in our model: in between two probabilistic transitions, one or more non-deterministic transitions may be specified. Other variations allow for non-deterministic nodes as starting nodes. From a theoretical point of view, these variations do not add to the expressive power of the model, and the theory outlined in this paper easily transfers to those models.

2.1. Strong bisimulation

Equivalence relations can be seen as a characterisation of the discriminating power of specific observers. Strong bisimilarity [16] is known to capture the capabilities of one of the most powerful observers that still has some appealing properties. It compares the stepwise behaviour of nodes in graphs and relates nodes when this behaviour is found to be identical.

Definition 2. Let x and y be graphs, let $N = N_\mathsf{x} \cup N_\mathsf{y}$ and let $P = P_\mathsf{x} \cup P_\mathsf{y}$. A symmetric relation $\mathcal{R} \subseteq N_{\mathsf{nil}}^2 \cup P^2$ is a *strong bisimulation relation* when for all nodes s and t for which $s \mathcal{R} t$ holds, we have
(1) if $s \in N$ and $t \in N$ and $s \xrightarrow{a} s'$ then there is some t', such that $t \xrightarrow{a} t'$ and $s' \mathcal{R} t'$ holds.
(2) if $s \in P$ and $t \in P$ then $\mu(s, \mathcal{M}) = \mu(t, \mathcal{M})$ holds for all $\mathcal{M} \in (N_{\mathsf{nil}} \cup P)_{/\mathcal{R}}$.
We say that x and y are *strongly bisimilar*, denoted $\mathsf{x} \underline{\leftrightarrow} \mathsf{y}$, iff there is a strong bisimulation relation $\mathcal{R}$ such that $s_\mathsf{x} \mathcal{R} s_\mathsf{y}$.

A corollary of requirement 2 in the definition of strong bisimilarity is that all probabilistic nodes that can be related by some strong bisimulation relation share the same cumulative probability of reaching another equivalence class. This justifies the overloading of the notation μ for cumulative probability to denote the probability of reaching a set of nodes from an entire equivalence class rather than from a single node. For a strong bisimulation relation $\mathcal{R}$, we define $\mu([s]_\mathcal{R}, \mathcal{M}) \stackrel{\mathrm{def}}{=} \mu(s, \mathcal{M})$ for arbitrary $s \in P$ and arbitrary $\mathcal{M} \in (N_{\mathsf{nil}} \cup P)_{/\mathcal{R}}$.

Proposition 3. $\underline{\leftrightarrow}$ *is an equivalence relation on* $\mathbf{G}$.

2.2. Paths, probabilities and schedulers

A decomposition of a graph into a set of so-named *computation trees* is necessary for further quantitative analysis of the graph: rather than conducting the analysis on the graph itself, the computation trees are analysed.

The decomposition requires all non-determinism in the graph to be resolved. This is typically achieved by employing a *scheduler* (also known as adversary or policy). A scheduler resolves the non-determinism by selecting at most one of possibly many non-deterministic transitions in each non-deterministic nodes. A computation tree is then obtained from the graph by resolving a non-deterministic choice according to the scheduler and keeping probabilistic information for the relevant nodes. Dependent on the type of scheduler, this choice is based on e.g. some history, randomisation or local information.

We subsequently formalise the notion of schedulers. Let x be a graph. A *path* starting in a node $s_0 \in S_{\mathsf{nil}}$ is an alternating finite sequence $c \equiv s_0\ l_1 \ldots l_n\ s_n$, or an alternating infinite sequence $c \equiv s_0\ l_1\ s_1\ \ldots$ of nodes and labels, where for all $i \geqslant 1$, $s_i \in S_{\mathsf{nil}}$ and $l_i \in \mathrm{Act}_\tau \cup (0, 1]$ and

(1) for all nodes $s_j \in N$ $(j \geqslant 0)$, we require $s_j \stackrel{l_{j+1}}{\rightarrow} s_{j+1}$.
(2) for all nodes $s_j \in P$ $(j \geqslant 0)$, we require $s_j \rightsquigarrow s_{j+1}$ and $l_{j+1} = \mathsf{pr}(s_j, s_{j+1})$.

Paths always consist of at least one node (its starting node). For a path c starting in s_0, we write $\mathsf{first}(c) = s_0$ for the initial node of c and, if c is a finite path, we write $\mathsf{last}(c)$ for the last node of c. The set of all nodes occurring in c is denoted $\mathsf{nodes}(c)$. We denote the *trace* of c by $\mathsf{trace}(c)$, which is the sequence of *action labels* from the set Act_τ that occur in c. The *concatenation* of two paths is again a path: given a finite path $c \equiv s_0\ l_1 \ldots l_n\ s_n$ (for $n \geqslant 0$) and a path c' with $\mathsf{last}(c) = \mathsf{first}(c')$, we denote their concatenation by $c \circ c'$ and it is defined as the path $s_0\ l_1 \ldots l_n c'$. If $c \equiv (s_0\ l_1\ s_1) \circ c'$ we write $\mathsf{rest}(c) = c'$.

The set of *all paths* starting in s_0 is denoted $\mathsf{Path}(s_0)$ and the set of finite paths starting in s_0 is denoted $\mathsf{Path}_f(s_0)$. A path c is a *maximal path* iff c is a finite path with $\mathsf{last}(c) = \mathsf{nil}$ or c is an infinite path. The set of maximal paths starting in s_0 is denoted $\mathsf{Path}_m(s_0)$.

Definition 4. A *scheduler* of paths starting in a node s_0 is a partial function $\sigma : \mathsf{Path}_f(s_0) \mapsto (\rightarrow \cup \{\bot\})$ (where $\bot$ represents "halt"). If, for some $c \in \mathsf{Path}_f(s_0)$, $\sigma(c)$ is defined we require that the following two conditions are met:

(1) if $\mathsf{last}(c) \in N$, then $\sigma(c) = \bot$ or $\sigma(c) = \mathsf{last}(c) \stackrel{a}{\rightarrow} t$ for some a and t.
(2) if $\mathsf{last}(c) \in P_{\mathsf{nil}}$, then $\sigma(c) = \bot$.

Moreover, we impose the following two sanity restrictions on σ: for all paths $c \in \mathsf{Path}_m(s_0) \cap \mathsf{Path}_f(s_0)$, we have $\sigma(c) = \bot$ and for all paths $c \in \mathsf{Path}_f(s_0)$ with $\mathsf{last}(c) \in N$, we require that $\sigma(c)$ is defined. We denote the set of *all* schedulers of a node s_0 by $\mathsf{Sched}(s_0)$. When defining schedulers, we will often leave the extra definitions that are required to meet these sanity restrictions implicit and focus on the remaining rules.

Note that the second condition in the definition of a scheduler expresses that a finite path c ending in a probabilistic node can only be scheduled (if scheduled at all) to $\bot$. In case such a path is not scheduled, then σ is defined for all extensions of this path by a probabilistic transition. This is also illustrated in Example 7 at the end of this section.

For most practical purposes, we are not interested in *all* paths of a graph, but only in those paths that are scheduled by a given scheduler. Let $\sigma \in \mathsf{Sched}(s_0)$ be a scheduler of a node s_0 in a graph x. We write $\mathsf{SPath}(s_0, \sigma)$ for the set of all finite and infinite paths $c \equiv s_0\ l_1\ s_1 \ldots$, where for each $s_i \in N$ we have $\sigma(s_0\ l_1\ s_1\ \ldots\ s_i) = s_i \stackrel{l_{i+1}}{\rightarrow} s_{i+1}$. The set of *maximal scheduled paths* starting in s_0 that is induced by σ is denoted $\mathsf{SPath}_m(s_0, \sigma)$ and contains all infinite scheduled paths and all finite scheduled paths c for which $\sigma(c) = \bot$.

Note that our sanity restrictions on schedulers turn finite maximal paths into finite maximal scheduled paths (since the former are necessarily scheduled to $\bot$). This is required for a proper definition of the probability space and a probability measure.

Several types of schedulers are defined in the literature, such as randomised schedulers, determinate schedulers and history dependent schedulers. For the exhibition of the theory, we do not fix a specific type of schedulers, but in Section 3.3 we show that a particular type of scheduler, so-called *simple schedulers* are sufficiently powerful for our purposes.

Definition 5. Let $s_0 \in S_{\mathsf{nil}}$ be a node and let $\sigma \in \mathsf{Sched}(s_0)$ be a scheduler. We say that σ is a *simple* scheduler if for all $c, c' \in \mathsf{Path}_f(s_0)$ with $\mathsf{last}(c) = \mathsf{last}(c')$, $\sigma(c) = \sigma(c')$.

Obviously, for a graph x the set of all schedulers that can be defined for a given node s_0 may be infinite, while the set of all simple schedulers for that graph *is finite*. This fact will be used in Section 3.3 where an algorithm for deciding branching bisimulation on graphs is given.

Definition 6. A *probabilistic tree* is a 7-tuple $\langle N, P, s, \mathrm{Act}, \rightarrow, \rightsquigarrow, \mathsf{pr}\rangle$, where
- N is a non-empty countable set of non-deterministic nodes.
- P is a non-empty countable set of probabilistic nodes.
- $\rightarrow : N \times \mathrm{Act}_\tau \rightarrow P_{\mathsf{nil}}$ is the non-deterministic transition function.
- $s \in P$, Act, $\rightsquigarrow$ and pr are defined as in Definition 1.

Graphs and probabilistic trees differ with respect to the non-deterministic branching degree that is allowed. While graphs have finite non-deterministic branching degree, probabilistic trees have branching degree 1. In other words, all non-deterministic transitions are uniquely determined by a pair consisting of a non-deterministic node and an action label. Furthermore, the set of nodes of a graph are necessarily finite, while probabilistic trees can have infinitely many nodes. It is well-known that probabilistic trees can be used to represent fully probabilistic systems (see e.g. [1,5]).

Every scheduler $\sigma \in \mathsf{Sched}(s_0)$ for a graph x defines a probabilistic tree $\mathsf{CT}_{\mathsf{x}}(s_0, \sigma)$ whose nodes are finite scheduled paths in x. The probabilistic and non-deterministic transitions of $\mathsf{CT}_{\mathsf{x}}(s_0, \sigma)$ are uniquely defined by the transition relations of x and σ in the obvious way. The probabilistic tree $\mathsf{CT}_{\mathsf{x}}(s_0, \sigma)$ is called a *computation tree* starting in s_0 and induced by σ. When no confusion can arise we omit the index x. The probabilistic transition relation $\rightsquigarrow$ of x is used to define a probability on a finite path in $\mathsf{CT}_{\mathsf{x}}(s_0, \sigma)$. These probabilities are then employed to define a probability measure for the probability space associated to σ. We proceed with the formal definitions. Let $c \equiv s_0\, l_1 \ldots l_n\, s_n$ be a finite path. Then, the probability of c, denoted $\mathbf{P}(c)$, is defined as:
(1) $\mathbf{P}(c) = \prod_{l_i \in (0,1]} l_i$ if at least one $l_i \in (0, 1]$ for $1 \leqslant i \leqslant n$.
(2) $\mathbf{P}(c) = 1$ otherwise.

Let c be a finite scheduled path. Then, the *basic cylinder* of c, induced by σ is given by

$$c{\uparrow} = \{c' \in \mathsf{SPath}_m(s_0, \sigma) \mid \ c \text{ is a prefix of } c'\}. \tag{1}$$

The probability measure of $c{\uparrow}$, denoted by $\mathcal{P}(c{\uparrow})$ is defined as $\mathcal{P}(c{\uparrow}) = \mathbf{P}(c)$. The probability space $(\Omega_\sigma, \mathcal{F}_\sigma, \mathcal{P}_\sigma)$ induced by $\sigma \in \mathsf{Sched}(s_0)$ is defined as follows: [2]
(1) $\Omega_\sigma = \mathsf{SPath}_m(s_0, \sigma)$.
(2) $\mathcal{F}_\sigma$ is the smallest sigma-algebra on $\mathsf{SPath}_m(s_0, \sigma)$ that contains all basic cylinders $c{\uparrow}$ for c a finite scheduled σ-path.
(3) $\mathcal{P}_\sigma$ is a probability measure on $\mathcal{F}_\sigma$, and is completely defined by $\mathcal{P}(\cdot)$.

Let $\mathsf{CT}(s_0, \sigma)$ be a computation tree for graph x. Recall that every node in $\mathsf{CT}(s_0, \sigma)$ is a finite path in x starting in s_0. We say that a node t of x *appears* (or, it has an *appearance*) in $\mathsf{CT}(s_0, \sigma)$ if there is a node c in $\mathsf{CT}(s_0, \sigma)$ such that $\mathsf{last}(c) = t$. In case we are also interested in the node c of $\mathsf{CT}(s_0, \sigma)$ that gives rise to an appearance of a node t of x in $\mathsf{CT}(s_0, \sigma)$, we say that t is *due to* node c. In general, there may be more nodes c, c' in $\mathsf{CT}(s_0, \sigma)$ to which t is due. To distinguish between these, we sometimes reason about the occurrence t_c when we mean that t is due to the node c in $\mathsf{CT}(s_0, \sigma)$. Note that from the context, it is always clear whether we mean the node c in the computation tree or the node t in the graph when we reason about a particular occurrence t_c. We say that t_c and $t_{c'}$ are *different* occurrences of t in $\mathsf{CT}(s, \sigma)$ iff $c \neq c'$.

Let t_c be an occurrence of t due to node c in $\mathsf{CT}(s, \sigma)$. Note that by definition, we have $c \in \mathsf{Path}_f(s)$ with $\mathsf{last}(c) = t$. The scheduler σ that is used to obtain the computation tree $\mathsf{CT}(s, \sigma)$ is said to *induce* a scheduler $(\sigma{-}c) \in \mathsf{Sched}(t)$. This scheduler is defined as follows:

$$\text{for all paths } c' \in \mathsf{Path}_f(t) \quad (\sigma{-}c)(c') = \sigma(c{\circ}c'). \tag{2}$$

Clearly, when we consider the path consisting of a single node s, we obtain $(\sigma{-}s)(c') = \sigma(c')$ for all c'. This induced scheduler $(\sigma{-}c)$ agrees with the original scheduler $\sigma \in \mathsf{Sched}(s)$, but its "starting" node is shifted towards some other node, and therefore, it only defines a computation tree that starts in $\mathsf{last}(c)$. This means that the computation tree $\mathsf{CT}(\mathsf{last}(c), (\sigma{-}c))$ yields a subtree of the computation tree $\mathsf{CT}(s, \sigma)$. Finally, we define the *depth* of an occurrence t_c, which is given by the depth of the node c in the computation tree.

The notions that we have introduced thus far are illustrated in Example 7.

[2] Note that we here overload the notation $\mathcal{P}$.

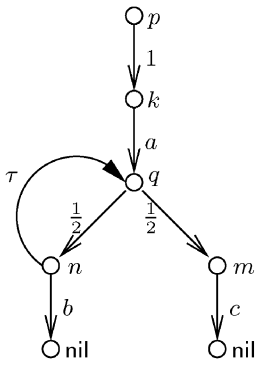

Fig. 1. A graph with an unobservable self-loop.

Example 7. Consider the graph of Fig. 1 with initial node p. The graph is not fully probabilistic because of the non-deterministic node n that has two outgoing non-deterministic transitions. It is also not a concrete graph because one of the transitions is labelled with τ.

The set of paths, starting in p is as follows:

$$\begin{aligned}
\mathsf{Path}(p) \;=\; & \{p,\ p\ 1\ k\} \cup \{p\ 1\ k\ a\ q\ (\tfrac{1}{2}\ n\ \tau\ q)^i \mid i \geqslant 0\ \} \\
\cup\ & \{p\ 1\ k\ a\ q\ (\tfrac{1}{2}\ n\ \tau\ q)^i\ \tfrac{1}{2}\ n \mid i \geqslant 0\} \\
\cup\ & \{p\ 1\ k\ a\ q\ (\tfrac{1}{2}\ n\ \tau q)^i\ \tfrac{1}{2}\ n\ b\ \mathsf{nil} \mid i \geqslant 0\} \\
\cup\ & \{p\ 1\ k\ a\ q\ (\tfrac{1}{2}\ n\ \tau\ q)^i\ \tfrac{1}{2}\ m \mid i \geqslant 0\} \\
\cup\ & \{p\ 1\ k\ a\ q\ (\tfrac{1}{2}\ n\ \tau\ q)^i\ \tfrac{1}{2}\ m\ c\ \mathsf{nil} \mid i \geqslant 0\} \\
\cup\ & \{p\ 1\ k\ a\ q\ (\tfrac{1}{2}\ n\ \tau\ q)^\omega\}
\end{aligned}$$

Among this set, the only infinite path is $p\ 1\ k\ a\ q\ (\frac{1}{2}\ n\ \tau\ q)^\omega$; all remaining paths are finite. The maximal paths are the infinite path and all paths that end in nil:

$$\begin{aligned}
\mathsf{Path}_m(p) \;=\; & \{p\ 1\ k\ a\ q\ (\tfrac{1}{2}\ n\ \tau\ q)^i\ \tfrac{1}{2}\ n\ b\ \mathsf{nil} \mid i \geqslant 0\} \\
\cup\ & \{p\ 1\ k\ a\ q\ (\tfrac{1}{2}\ n\ \tau\ q)^i\ \tfrac{1}{2}\ m\ c\ \mathsf{nil} \mid i \geqslant 0\} \\
\cup\ & \{p\ 1\ k\ a\ q\ (\tfrac{1}{2}\ n\ \tau\ q)^\omega\}
\end{aligned}$$

To illustrate the effect of a particular scheduler on the set of paths of a graph we consider the following scheduler:

$$\begin{cases} \sigma_1(p\ 1\ k) = k \xrightarrow{a} q \\ \sigma_1(c) & \text{undefined for other finite paths } c. \end{cases}$$

Note that we have left some parts of the definition of σ_1 implicit: finite maximal paths and paths ending in non-deterministic nodes are not (correctly) covered by σ_1. By our convention (see Definition 4), σ_1 assigns $\perp$ to those paths when they are not explicitly defined. The set of scheduled paths is:

$$\mathsf{SPath}(p, \sigma_1) = \left\{p,\ p\ 1\ k,\ p\ 1\ k\ a\ q,\ p\ 1\ k\ a\ q\ \tfrac{1}{2}\ n,\ p\ 1\ k\ a\ q\ \tfrac{1}{2}\ m\right\}.$$

The subset of maximal scheduled paths is

$$\mathsf{SPath}_m(p, \sigma_1) = \left\{p\ 1\ k\ a\ q\ \tfrac{1}{2}\ n,\ p\ 1\ k\ a\ q\ \tfrac{1}{2}\ m\right\}.$$

Remark that the scheduler is undefined for $p\ 1\ k\ a\ q$. This does not mean, however, that the scheduling stops at this point. On the contrary, it is defined for all extensions of the path $p\ 1\ k\ a\ q$, which are obtained using one of the specified probabilistic transitions, in this case for the paths $p\ 1\ k\ a\ q\ \frac{1}{2}\ n$ and $p\ 1\ k\ a\ q\ \frac{1}{2}\ m$.

The second scheduler we consider is slightly more involved, and we use it to illustrate the probability of sets of paths. Let σ_2 be defined as follows:

$$\begin{cases} \sigma_2(p\ 1\ k) = k \xrightarrow{a} q \\ \sigma_2(p\ 1\ k\ a\ q\ \frac{1}{2}\ n) = n \xrightarrow{b} \mathsf{nil} \\ \sigma_2(p\ 1\ k\ a\ q\ (\frac{1}{2}\ n\ \tau\ q)^{10}\ \frac{1}{2}\ n) = n \xrightarrow{b} \mathsf{nil} \\ \sigma_2(c) & \text{undefined for other finite paths } c. \end{cases}$$

We find that the set of scheduled paths is as follows:

$$\mathsf{SPath}(p, \sigma_1) = \left\{ p,\ p\ 1\ k,\ p\ 1\ k\ a\ q,\ p\ 1\ k\ a\ q\ \tfrac{1}{2}\ n,\ p\ 1\ k\ a\ q\ \tfrac{1}{2}\ m,\ p\ 1\ k\ a\ q\ \tfrac{1}{2}\ n\ b\ \mathsf{nil} \right\}.$$

Its subset of maximal scheduled paths is

$$\mathsf{SPath}_m(p, \sigma_1) = \left\{ p\ 1\ k\ a\ q\ \tfrac{1}{2}\ n\ b\ \mathsf{nil},\ p\ 1\ k\ a\ q\ \tfrac{1}{2}\ m \right\}$$

The probability $\mathcal{P}$, for various paths is as follows: $\mathcal{P}((p)\uparrow) = \mathcal{P}((p\ 1\ k)\uparrow) = \mathcal{P}((p\ 1\ k\ a\ q)\uparrow) = 1$, while $\mathcal{P}((p\ 1\ k\ a\ q\ \frac{1}{2}\ n)\uparrow) = \frac{1}{2}$, $\mathcal{P}((p\ 1\ k\ a\ q\ \frac{1}{2}\ n\ b\ \mathsf{nil})\uparrow) = \frac{1}{2}$ and $\mathcal{P}((p\ 1\ k\ a\ q\ \frac{1}{2}\ m)\uparrow) = \frac{1}{2}$. Note that even though σ_2 is defined for a path such as $p\ 1\ k\ a\ q\ (\frac{1}{2}\ n\ \tau\ q)^{10}\ \frac{1}{2}\ n$, this finite path is not a node in $\mathsf{CT}(p, \sigma_2)$, since it is not a scheduled path by σ_2.

The last scheduler that we consider is defined as follows:

$$\begin{cases} \sigma_3(p\ 1\ k) = k \xrightarrow{a} q & \\ \sigma_3(c) = n \xrightarrow{\tau} q & \text{if } \mathsf{last}(c) = n \\ \sigma_3(c) = p \xrightarrow{c} \mathsf{nil} & \text{if } c \text{ is such that } \mathsf{last}(c) = m \\ \sigma_3(c) & \text{undefined for other finite paths } c. \end{cases}$$

Note that σ_3 is a *simple* scheduler, as it always schedules a non-deterministic transition on the basis of the last non-deterministic node of the path. The set of scheduled paths is as follows:

$$\begin{aligned} \mathsf{SPath}(p, \sigma_3) = {} & \{p,\ p\ 1\ k\} \cup \{p\ 1\ k\ a\ q\ (\tfrac{1}{2}\ n\ \tau\ q)^i \mid i \geqslant 0\} \\ & \cup \{p\ 1\ k\ a\ q\ (\tfrac{1}{2}\ n\ \tau\ q)^i\ \tfrac{1}{2}\ n \mid i \geqslant 0\} \\ & \cup \{p\ 1\ k\ a\ q\ (\tfrac{1}{2}\ n\ \tau\ q)^i\ \tfrac{1}{2}\ m \mid i \geqslant 0\} \\ & \cup \{p\ 1\ k\ a\ q\ (\tfrac{1}{2}\ n\ \tau\ q)^i\ \tfrac{1}{2}\ m\ c\ \mathsf{nil} \mid i \geqslant 0\} \cup \{p\ 1\ k\ a\ q\ (\tfrac{1}{2}\ n\ \tau\ q)^\omega\} \end{aligned}$$

and

$$\mathsf{SPath}_m(p, \sigma_3) = \left\{ p\ 1\ k\ a\ q\ (\tfrac{1}{2}\ n\ \tau\ q)^i\ \tfrac{1}{2}\ m\ c\ \mathsf{nil} \mid i \geqslant 0 \right\} \cup \left\{ p\ 1\ k\ a\ q\ (\tfrac{1}{2}\ n\ \tau\ q)^\omega \right\}.$$

For every $c \in \mathsf{SPath}_f(p, \sigma_3)$ we have $c\uparrow = \mathsf{SPath}_m(p, \sigma_3)$ and $\mathcal{P}(\mathsf{SPath}_m(p, \sigma_3)) = 1$. Furthermore, the node n appears in the computation tree $\mathsf{CT}(p, \sigma_3)$. It has several different occurrences. For instance, consider the nodes $c_1 \equiv p\ 1\ k\ a\ q\ \frac{1}{2}\ n\ \tau\ q\ \frac{1}{2}\ n$ and $c_2 \equiv p\ 1\ k\ a\ q\ \frac{1}{2}\ n\ \tau\ q\ \frac{1}{2}\ n\ \tau\ q\ \frac{1}{2}\ n$ in $\mathsf{CT}(p, \sigma_3)$. Then we say that the occurrences n_{c_1} and n_{c_2} are different occurrences of n in $\mathsf{CT}(p, \sigma_3)$.

Note that $\mathsf{CT}(p, \sigma_1)$ and $\mathsf{CT}(p, \sigma_2)$ are finite computation trees while $\mathsf{CT}(p, \sigma_3)$ is infinite.

2.3. Branching bisimulation

Strong bisimilarity is most appropriate when considering *concrete graphs*, but the equivalence is too fine in a setting with abstraction. This is because it treats the unobservable event τ as if it were any other observable event. While abstraction is of utmost importance in the analysis of probabilistic systems, it is also one of the harder notions to grasp. This is because the unobservable event (represented by the action τ) plays an almost diabolical role: while the τ itself

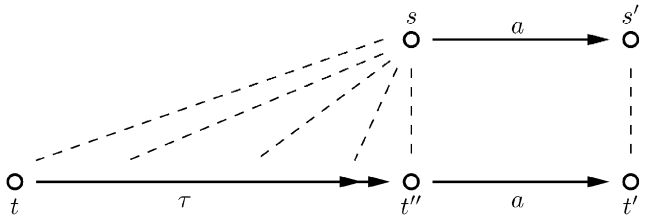

Fig. 2. Branching bisimulation in a non-probabilistic setting.

may not be visible, its effect might be noted by the disabling or enabling of some observable events. For instance, while the inspection of a coin that has been put in a coffee-machine may be unobservable, it manifests itself through a (consistent) rejection of that particular coin. This illustrates that we cannot bluntly remove all τ actions from a graph: only the ones that do not manifest themselves may be removed. We call such τ actions *inert*.

The equivalence relation we define in this section is called *branching bisimilarity*. It is strictly in between strong bisimilarity and weak bisimilarity (for the latter see e.g. [18]). Branching bisimilarity enjoys several pleasing properties. Unlike strong bisimilarity, it treats the inert τ actions as unobservable. Further, in contrast to weak bisimilarity, it preserves the non-deterministic branching structure of graphs. This is due to the fact that it differentiates between τ actions that are truly inert and τ actions that are not really inert.

We briefly repeat one of the central ideas behind branching bisimulation from the non-probabilistic setting (see e.g. [2,11]). The crucial point in that setting is that a node t can be related to a node s by a branching bisimulation relation *only* whenever all (observable) transitions $s \xrightarrow{a} s'$ from node s can be matched by transitions $t \xrightarrow{\tau} \cdots \xrightarrow{\tau} t'' \xrightarrow{a} t'$ from node t such that t' can again be related to s' by the branching bisimulation relation. Unlike in e.g. weak bisimulation or delay bisimulation, it is required that this sequence of τ transitions traverses through nodes that *all* can be related to s (see Fig. 2). In our setting, the sequences of transitions readily translate to paths.

Before we turn to the definition of branching bisimulation, we fix some shorthand notation to ease notational burden and to capture the ideas depicted by Fig. 2 in a formal framework. Let c be an arbitrary finite path. Then the path c *satisfies* a path-predicate ϕ, denoted by $c \;\mathsf{sat}\; \phi$, is defined as follows for the following path-predicates:

(1) $c \;\mathsf{sat}\; s \Longrightarrow_{\mathcal{M}} s'$ iff $\mathsf{first}(c) = s$, $\mathsf{last}(c) = s'$, $\mathsf{trace}(c) = \tau^*$ and $\mathsf{nodes}(c) \subseteq \mathcal{M}$.
(2) $c \;\mathsf{sat}\; s \Longrightarrow_{\mathcal{M}} \cdot \xrightarrow{a} s''$ iff $\exists c' : c \equiv c'as''$ and $c' \;\mathsf{sat}\; s \Longrightarrow_{\mathcal{M}} \mathsf{last}(c')$.
(3) $c \;\mathsf{sat}\; s \Longrightarrow_{\mathcal{M}} \cdot \rightsquigarrow s''$ iff $\exists c'\, \exists l \in (0, 1] : c \equiv c'ls''$ and $c' \;\mathsf{sat}\; s \Longrightarrow_{\mathcal{M}} \mathsf{last}(c')$.

Note that by requiring that c is a finite path, we have $\mathsf{last}(c) = s''$ in the last two path-predicates. Moreover, we also find that $\mathsf{last}(c') \xrightarrow{a} s''$ (resp. $\mathsf{pr}(\mathsf{last}(c'), s'') = l$). The intuition behind the path-predicates is that a finite number of nodes from the set $\mathcal{M}$ may be visited, provided that this does not require the execution of an observable action (unless, as is stated for the second path-predicate, it is the last action and $s'' \in \mathcal{M}$).

Proposition 8. *Let $s, s' \in S_{\mathsf{nil}}$, $a \in \mathrm{Act}_\tau$ and $l \in (0, 1]$. Let $\mathcal{M} \subseteq S_{\mathsf{nil}}$ be such that $s \in \mathcal{M}$. Then*

(1) $s \;\mathsf{sat}\; s \Longrightarrow_{\mathcal{M}} s$.
(2) $(s\ a\ s') \;\mathsf{sat}\; s \Longrightarrow_{\mathcal{M}} \cdot \xrightarrow{a} s'$.
(3) $(s\ l\ s') \;\mathsf{sat}\; s \Longrightarrow_{\mathcal{M}} \cdot \rightsquigarrow s'$.

Let σ be a scheduler, and let $\mathcal{M}, \mathcal{M}'$ be sets of nodes. Let $\mathcal{B}_\sigma(s \overset{a}{\Longrightarrow}_{\mathcal{M}} \mathcal{M}')$ be the set of all maximal σ-scheduled paths that start in s and silently (i.e. using τ actions) traverse through a set of nodes $\mathcal{M}$ and reach a node in $\mathcal{M}'$ by executing a given a action ($a \in \mathrm{Act}_\tau$). More concretely, let $\mathcal{B}_\sigma(s \overset{a}{\Longrightarrow}_{\mathcal{M}} \mathcal{M}')$ be defined as follows:

$$\begin{aligned}\mathcal{B}_\sigma(s \overset{a}{\Longrightarrow}_{\mathcal{M}} \mathcal{M}') = \{c \in \mathsf{SPath}(s, \sigma) \mid \sigma(c) = \bot \text{ and either}\\ c \;\mathsf{sat}\; s \Longrightarrow_{\mathcal{M}} \cdot \xrightarrow{a} s', s' \in \mathcal{M}', \text{ or}\\ c \;\mathsf{sat}\; s \Longrightarrow_{\mathcal{M}} \cdot \rightsquigarrow s', s' \in \mathcal{M}', a = \tau, \text{ or}\\ c \equiv s, a = \tau, \mathcal{M} = \mathcal{M}'\}.\end{aligned} \tag{3}$$

When $a = \tau$, we generally write $\mathcal{B}_\sigma(s \Longrightarrow_{\mathcal{M}} \mathcal{M}')$ instead of $\mathcal{B}_\sigma(s \overset{\tau}{\Longrightarrow}_{\mathcal{M}} \mathcal{M}')$. Next, we overload the function μ to denote the *normalised* cumulative probability. Given two disjoint, non-empty sets of nodes $\mathcal{M}$ and $\mathcal{M}'$ and a node

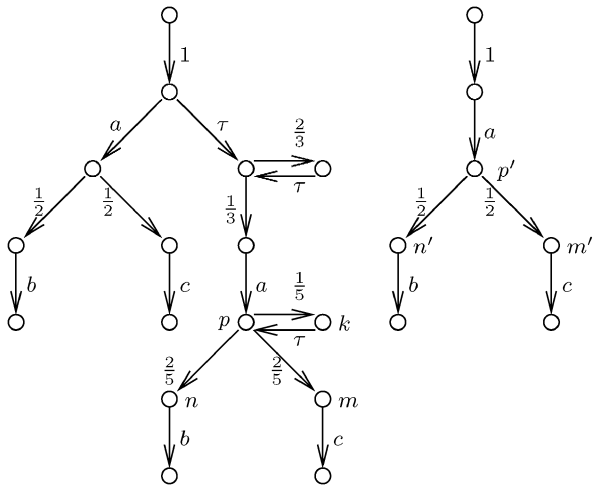

Fig. 3. Two branching bisimilar graphs.

$p \in \mathcal{M}$, the function $\mu_{\mathcal{M}}(p, \mathcal{M}')$ is used to denote the probability of entering $\mathcal{M}'$ from p (in one step) weighted by the probability of remaining in $\mathcal{M}$. Formally, we have

$$\mu_{\mathcal{M}}(p, \mathcal{M}') = \begin{cases} \dfrac{\mu(p, \mathcal{M}')}{1 - \mu(p, \mathcal{M})} & \text{if } p \in P \text{ and } \mu(p, \mathcal{M}) \neq 1. \\ 0 & \text{otherwise.} \end{cases} \tag{4}$$

Definition 9. Let x and y be graphs. Let $N = N_{\mathsf{x}} \cup N_{\mathsf{y}}$, $P = P_{\mathsf{x}} \cup P_{\mathsf{y}}$, $S = S_{\mathsf{x}} \cup S_{\mathsf{y}}$ and $S_{\mathsf{nil}} = S \cup \{\mathsf{nil}\}$. Let $\mathcal{R}$ be an equivalence relation on S_{nil}. $\mathcal{R}$ is a *branching bisimulation relation* when for all nodes s and t for which $s\mathcal{R}t$ holds, we have
(1) if $s \in N$ and $s \xrightarrow{a} s'$, then there is a scheduler σ such that $\mathcal{P}(\mathcal{B}_\sigma(t \overset{a}{\Longrightarrow}_{[t]_\mathcal{R}} [s']_\mathcal{R})) = 1$.
(2) if $s \in P$, then for some scheduler σ, $\mu_{[s]_\mathcal{R}}(s, \mathcal{M}) = \mathcal{P}(\mathcal{B}_\sigma(t \Longrightarrow_{[t]_\mathcal{R}} \mathcal{M}))$ for all $\mathcal{M} \in S_{\mathsf{nil}/\mathcal{R}} \setminus \{[s]_\mathcal{R}\}$.
We say that x and y are *branching bisimilar*, denoted $\mathsf{x} \leftrightarrow_b \mathsf{y}$, iff there is a branching bisimulation relation $\mathcal{R}$ on S_{nil}, such that $s_{\mathsf{x}} \mathcal{R} s_{\mathsf{y}}$.

In words, branching bisimilarity requires all non-deterministic transitions (i.e. also the inert τ transitions) emanating from a node in an equivalence class to be schedulable from *all* nodes related to that node, with probability 1. We say that all nodes in the same equivalence class have the same *potentials*. The second condition requires that a single scheduler for a node can be used to simulate the normalised cumulative probability of a related probabilistic node. This particular scheduler can be employed to find a "silent" path (i.e. a path with unobservable actions only) through a set of nodes that are related to the originating node before it leaves this class of nodes and reaches another equivalence class. This last step is done either via the execution of another τ action or by a probabilistic transition.

Example 10. Consider the two graphs of Fig. 3. We find that the two graphs are branching bisimilar. For instance, the non-deterministic node k and the probabilistic node p' are in the same equivalence class. This can be seen as follows. Say $\mathcal{R}$ is the branching bisimulation relation. We have $\mu_{[p']_\mathcal{R}}(p', [n']_\mathcal{R}) = \mu_{[p']_\mathcal{R}}(p', [m']_\mathcal{R}) = \frac{1}{2}$. To mimic these probabilities, we can define a scheduler $\sigma \in \mathsf{Sched}(k)$ as follows:

$$\begin{cases} \sigma(k\ (\tau\ p\ \frac{1}{5}\ k)^i) = k \xrightarrow{\tau} p & \text{for all } i \geqslant 0, \\ \sigma(k\ \tau\ p\ (\frac{1}{5}\ k\ \tau\ p)^i\ \frac{2}{5}\ n) = \bot & \text{for all } i \geqslant 0, \\ \sigma(k\ \tau\ p\ (\frac{1}{5}\ k\ \tau\ p)^i\ \frac{2}{5}\ m) = \bot & \text{for all } i \geqslant 0, \\ \sigma(c') & \text{is undefined for any other finite path } c'. \end{cases}$$

Using this scheduler, we find that $\mathcal{P}(\mathcal{B}_\sigma(k \Longrightarrow_{[k]_\mathcal{R}} [n]_\mathcal{R})) = \mathcal{P}(\mathcal{B}_\sigma(k \Longrightarrow_{[k]_\mathcal{R}} [m]_\mathcal{R})) = \frac{1}{2}$. To see that node p' is in the same equivalence class as node k, we must show the existence of a scheduler that mimics the non-deterministic τ-transition of node k with probability 1. This boils down to preventing p' from leaving its own class, which is achieved by the scheduler $\eta \in \mathsf{Sched}(p')$, defined as $\eta(p) = \bot$. *Nota Bene*: in Section 7, we use the same example to illustrate that using branching bisimulation in the *non-alternating* setting, the two graphs are not branching bisimilar. The crux turns out to be the non-deterministic node k.

3. Branching bisimulation: Maximal probabilities and decidability

Finding a branching bisimulation relation between two graphs can be quite hard. The culprit is the fact that in both conditions of the branching bisimulation relation definition, a quantification over an infinite set of schedulers appears. From this set, a scheduler must be picked that meets the conditions of the bisimulation relation. Moreover, this feat must be repeated for all nodes of the two graphs, making the entire process of checking for branching bisimulation rather cumbersome and even problematic to automate.

As we will show in this section, the above problems are not insurmountable. For instance, Philippou et al. [18] showed that weak bisimilarity can be rephrased in terms of maximal probabilities. Since branching bisimulation and weak bisimulation are closely related, this raises the question whether also branching bisimulation might be rephrased in terms of maximal probabilities. In Section 3.2 we give an affirmative answer to this question. This result allows us to narrow down the choice of schedulers to those schedulers that induce maximal probabilities.

This result is at the basis of a decision procedure for branching bisimulation. Instead of the infinite set of schedulers that must be checked in Definition 9, we can now narrow down the search criterion to those schedulers that induce a maximal probability.

We first introduce some auxiliary notation in Section 3.1. Some of this notation will only be used in the main proofs in Section 3.2, in which we show that Definition 9 can be rephrased in terms of maximal probabilities. Then, in Section 3.3 we provide results for deciding branching bisimulation, together with the algorithm for doing so.

3.1. Preliminaries

For the remainder of this section, we fix a graph x. Let $a \in \mathrm{Act}_\tau$ and $\mathcal{M}, \mathcal{M}' \subseteq S_{\mathsf{nil}}$, and let $s \in S_{\mathsf{nil}}$. In Section 2.3 (Eq. 3), we introduced the notation $\mathcal{B}_\sigma(s \overset{a}{\Longrightarrow}_{\mathcal{M}'} \mathcal{M})$ for a set of scheduled paths that silently traverse through $\mathcal{M}'$ before executing action a and reaching $\mathcal{M}$. The probability of this set of paths is highly dependable on the scheduler σ. Given that the set of probabilities is ordered, we can search for the *maximal* probability among this set by selecting an appropriate scheduler. We introduce the following notation:

$$\mathcal{P}_{\max}(s \overset{a}{\Longrightarrow}_{\mathcal{M}'} \mathcal{M}) \overset{\text{def}}{=} \max_{\sigma \in \mathsf{Sched}(s)} \mathcal{P}(\mathcal{B}_\sigma(s \overset{a}{\Longrightarrow}_{\mathcal{M}'} \mathcal{M})). \tag{5}$$

If $a = \tau$ we omit a and we simply write $\mathcal{P}_{\max}(s \Longrightarrow_{\mathcal{M}'} \mathcal{M})$. Note that even though the maximal probability is a unique number, this does not mean that there is necessarily a single scheduler that induces this maximal probability.

The following series of propositions are useful in understanding the interplay between maximal probabilities, branching bisimulation and (operations on) schedulers.

Proposition 11. *Let $\mathcal{R}$ be a branching bisimulation relation on S_{nil}. Let $a \in \mathrm{Act}_\tau$ be an action and let $\mathcal{M} \in S_{\mathsf{nil}/\mathcal{R}}$ be an equivalence class. For all nodes $s \in S_{\mathsf{nil}}$ with $\mathcal{P}_{\max}(s \overset{a}{\Longrightarrow}_{[s]_\mathcal{R}} \mathcal{M}) = 0$ we find that for every $t \in [s]_\mathcal{R}$, $\mathcal{P}_{\max}(t \overset{a}{\Longrightarrow}_{[t]_\mathcal{R}} \mathcal{M}) = 0$.*

Proof. The result follows directly from the definition of branching bisimulation. Namely, the existence of a node $t \in [s]_\mathcal{R}$ for which $\mathcal{P}_{\max}(t \overset{a}{\Longrightarrow}_{[t]_\mathcal{R}} \mathcal{M}) > 0$ is in immediate conflict with the assumptions $\mathcal{P}_{\max}(s \overset{a}{\Longrightarrow}_{[s]_\mathcal{R}} \mathcal{M}) = 0$ and $s \mathcal{R} t$. □

Now let us assume that $\mathcal{R}$ is an equivalence relation on S_{nil}. Let $a \in \mathrm{Act}_\tau$ and $\mathcal{M} \in S_{\mathsf{nil}/\mathcal{R}}$. Let $\sigma \in \mathsf{Sched}(s)$ be a scheduler such that $\mathcal{P}_{\max}(s \stackrel{a}{\Longrightarrow}_{[s]_\mathcal{R}} \mathcal{M}) = \mathcal{P}(\mathcal{B}_\sigma(s \stackrel{a}{\Longrightarrow}_{[s]_\mathcal{R}} \mathcal{M}))$. Let t be a node in x such that $s\mathcal{R}t$ and let t_c be an occurrence in the computation tree $\mathsf{CT}(s, \sigma)$.

Proposition 12. *Let $(\sigma{-}c) \in \mathsf{Sched}(t)$ be the scheduler induced by σ, as defined in Section* 2.2. *Then* $\mathcal{P}_{\max}(t \stackrel{a}{\Longrightarrow}_{[t]_\mathcal{R}} \mathcal{M}) = \mathcal{P}(\mathcal{B}_{(\sigma-c)}(t \stackrel{a}{\Longrightarrow}_{[t]_\mathcal{R}} \mathcal{M}))$.

Proof. We prove this by contradiction. Assume that $\mathcal{P}_{\max}(t \stackrel{a}{\Longrightarrow}_{[t]_\mathcal{R}} \mathcal{M}) > \mathcal{P}(\mathcal{B}_{(\sigma-c)}(t \stackrel{a}{\Longrightarrow}_{[t]_\mathcal{R}} \mathcal{M}))$. This implies $\mathcal{P}_{\max}(s \stackrel{a}{\Longrightarrow}_{[s]_\mathcal{R}} \mathcal{M}) > \mathcal{P}(\mathcal{B}_\sigma(s \stackrel{a}{\Longrightarrow}_{[s]_\mathcal{R}} \mathcal{M}))$ which contradicts our assumption that σ induces maximal probabilities. Therefore, we find that $\mathcal{P}_{\max}(t \stackrel{a}{\Longrightarrow}_{[t]_\mathcal{R}} \mathcal{M}) = \mathcal{P}(\mathcal{B}_{(\sigma-c)}(t \stackrel{a}{\Longrightarrow}_{[t]_\mathcal{R}} \mathcal{M}))$, which finishes the proof. □

Vice versa, assume that $\eta \in \mathsf{Sched}(t)$ is such that $\mathcal{P}_{\max}(t \stackrel{a}{\Longrightarrow}_{[t]_\mathcal{R}} \mathcal{M}) = \mathcal{P}(\mathcal{B}_\eta(t \stackrel{a}{\Longrightarrow}_{[t]_\mathcal{R}} \mathcal{M}))$, i.e. η induces the maximal probability of reaching $\mathcal{M}$ from t. Let σ be as defined above, and let $\sigma^+ \in \mathsf{Sched}(s)$ be the scheduler defined as follows:

$$\begin{cases} \sigma^+(c) = \sigma(c) & \text{if } c \text{ is such that } t \notin \mathsf{nodes}(c), \\ \sigma^+(c) = \eta(c_2) & \text{if } \exists c_1 : c \equiv c_1 \circ c_2 \text{ with } t \notin \mathsf{nodes}(c_1) \text{ and } \mathsf{first}(c_2) = t. \end{cases}$$

Proposition 13. *We find* $\mathcal{P}_{\max}(s \stackrel{a}{\Longrightarrow}_{[s]_\mathcal{R}} \mathcal{M}) = \mathcal{P}(\mathcal{B}_{\sigma^+}(s \stackrel{a}{\Longrightarrow}_{[s]_\mathcal{R}} \mathcal{M})) = \mathcal{P}(\mathcal{B}_\sigma(s \stackrel{a}{\Longrightarrow}_{[s]_\mathcal{R}} \mathcal{M}))$.

The remaining shorthand notations and propositions are used mainly in the proofs that appear in the next two sections. As such, they can be skipped on first reading this paper.

Definition 14. Let $s, t \in S_{\mathsf{nil}}$ be arbitrary nodes, and let $\sigma \in \mathsf{Sched}(s)$ be a scheduler starting in node s. Let $\mathcal{M}, \mathcal{M}' \subseteq S_{\mathsf{nil}}$ be subsets of S_{nil}. We introduce the following two shorthands:

$$\mathcal{B}_\sigma(s \stackrel{a}{\Longrightarrow}_{\mathcal{M}-t} \mathcal{M}') \stackrel{\text{def}}{=} \{c \in \mathcal{B}_\sigma(s \stackrel{a}{\Longrightarrow}_{\mathcal{M}} \mathcal{M}') \mid c \equiv s \text{ or } c \equiv (s\ l\ s') \circ c' \text{ for some } l \in \mathrm{Act}_\tau \cup (0, 1], \\ s' \in S_{\mathsf{nil}} \text{ and path } c' \text{ satisfying } t \notin \mathsf{nodes}(c')\}.$$

and

$$\mathcal{B}_\sigma(s \stackrel{a}{\Longrightarrow}_{\mathcal{M}+t} \mathcal{M}') \stackrel{\text{def}}{=} \mathcal{B}_\sigma(s \stackrel{a}{\Longrightarrow}_{\mathcal{M}} \mathcal{M}') \setminus \mathcal{B}_\sigma(s \stackrel{a}{\Longrightarrow}_{\mathcal{M}-t} \mathcal{M}').$$

In words, $\mathcal{B}_\sigma(s \stackrel{a}{\Longrightarrow}_{\mathcal{M}-t} \mathcal{M}')$ denotes the subset of $\mathcal{B}_\sigma(s \stackrel{a}{\Longrightarrow}_{\mathcal{M}} \mathcal{M}')$ containing all paths that do not pass through t; if $s = t$ then it starts in t but it never returns to node t again. The complement of this set is given by the subset $\mathcal{B}_\sigma(s \stackrel{a}{\Longrightarrow}_{\mathcal{M}+t} \mathcal{M}')$, which contains all paths that *do* pass through t at least once after leaving the root node. Now, when $s = t$ it denotes the set of paths that start in t *and* that returns to t at least once more.

Proposition 15. *For all $s, t, a, \mathcal{M}, \mathcal{M}'$ and σ:*

$$\mathcal{P}(\mathcal{B}_\sigma(s \stackrel{a}{\Longrightarrow}_{\mathcal{M}} \mathcal{M}')) = \mathcal{P}(\mathcal{B}_\sigma(s \stackrel{a}{\Longrightarrow}_{\mathcal{M}-t} \mathcal{M}')) + \mathcal{P}(\mathcal{B}_\sigma(s \stackrel{a}{\Longrightarrow}_{\mathcal{M}+t} \mathcal{M}')).$$

Proof. Standard result from probability theory. □

Proposition 16. *If for some $\sigma \in \mathsf{Sched}(s)$, we find $\mathcal{P}_{\max}(s \stackrel{a}{\Longrightarrow}_{[s]_\mathcal{R}} \mathcal{M}) = \mathcal{P}(\mathcal{B}_\sigma(s \stackrel{a}{\Longrightarrow}_{[s]_\mathcal{R}} \mathcal{M})) > 0$ then there is an occurrence s_c in $\mathsf{CT}(s, \sigma)$, satisfying $\mathcal{P}(\mathcal{B}_{(\sigma-c)}(s \stackrel{a}{\Longrightarrow}_{[s]_\mathcal{R}-s} \mathcal{M})) > 0$.*

Proof. By assuming that for every occurrence s_c in $\mathsf{CT}(s, \sigma)$ we have $\mathcal{P}(\mathcal{B}_{(\sigma-c)}(s \stackrel{a}{\Longrightarrow}_{[s]_\mathcal{R}-s} \mathcal{M})) = 0$ we obtain that each path starting in the root s contains countably infinitely many different occurrences s_{c_i} and therefore it never reaches $\mathcal{M}$. Then, $\mathcal{B}_\sigma(s \stackrel{a}{\Longrightarrow}_{[s]_\mathcal{R}} \mathcal{M}) = \emptyset$ and therefore, $\mathcal{P}_{\max}(s \stackrel{a}{\Longrightarrow}_{[s]_\mathcal{R}} \mathcal{M}) = 0$. □

Corollary 17. *Let σ be as defined in Proposition* 16. *Then for any occurrence s_c in $\mathsf{CT}(s,\sigma)$, we find $\mathcal{B}_{(\sigma-c)}(s \stackrel{a}{\Longrightarrow}_{[s]_\mathcal{R}} \mathcal{M}) \neq \emptyset$ and hence $\mathcal{P}(\mathcal{B}_{(\sigma-c)}(s \stackrel{a}{\Longrightarrow}_{[s]_\mathcal{R}} \mathcal{M})) > 0$.*

3.2. Branching bisimulation using maximal probabilities

Using the concept of maximal probabilities as outlined in the previous section, we show that the definition of branching bisimulation can be rewritten to an equivalent definition in which we employ the notion of maximal probabilities. This is stated by the following theorem.

Theorem 18. *Let* x *and* y *be two graphs, and denote the set of their nodes by S. Let $S_{\mathsf{nil}} = S \cup \{\mathsf{nil}\}$. Then, a relation $\mathcal{R}$ on S_{nil} is a branching bisimulation relation iff the following two conditions are met for all nodes $s, t \in S_{\mathsf{nil}}$ satisfying $s\mathcal{R}t$:*
(1) $\mathcal{P}_{\max}(s \stackrel{a}{\Longrightarrow}_{[s]_\mathcal{R}} \mathcal{M}) = \mathcal{P}_{\max}(t \stackrel{a}{\Longrightarrow}_{[t]_\mathcal{R}} \mathcal{M})$ *for all* $a \in \mathrm{Act}$ *and* $\mathcal{M} \in S_{\mathsf{nil}/\mathcal{R}}$.
(2) $\mathcal{P}_{\max}(s \Longrightarrow_{[s]_\mathcal{R}} \mathcal{M}) = \mathcal{P}_{\max}(t \Longrightarrow_{[t]_\mathcal{R}} \mathcal{M})$ *for all* $\mathcal{M} \in S_{\mathsf{nil}/\mathcal{R}} \setminus \{[s]_\mathcal{R}\}$.

The remainder of this section is devoted to proving the above theorem. The two directions of the proof will be discussed separately. For the implication, we prove that branching bisimilar nodes have the same maximal probabilities of executing actions or reaching other equivalence classes. Due to the different way in which branching bisimulation treats the unobservable event τ and observable actions $a \in \mathrm{Act}$, we split the proof for our claim for these two classes of events. In Lemma 19 we deal with the τ transitions and in Lemma 21 we prove the claim for actions $a \in \mathrm{Act}$. Lemma 22 states that by requiring equal maximal probabilities we also obtain a branching bisimulation relation.

Fix two graphs x and y, and denote the set of their probabilistic nodes by P and the set of their non-deterministic nodes by N. We write $S = P \cup N$ and $S_{\mathsf{nil}} = S \cup \{\mathsf{nil}\}$.

Lemma 19. *Let $\mathcal{R}$ be a branching bisimulation on S_{nil} and $\mathcal{C} \in S_{\mathsf{nil}/\mathcal{R}}$.*
i. *If $s, t \in \mathcal{C}$, then $\mathcal{P}_{\max}(s \Longrightarrow_\mathcal{C} \mathcal{M}) = \mathcal{P}_{\max}(t \Longrightarrow_\mathcal{C} \mathcal{M})$, for all $\mathcal{M} \neq \mathcal{C}$.*
ii. *If $s \in P \cap \mathcal{C}$ and $\mu(s,\mathcal{C}) \neq 1$ then for all $\mathcal{M} \in S_{\mathsf{nil}/\mathcal{R}}$ with $\mathcal{M} \neq \mathcal{C}$, $\mathcal{P}_{\max}(s \Longrightarrow_\mathcal{C} \mathcal{M}) = \mu_\mathcal{C}(s,\mathcal{M})$.*

Proof. We first show that by employing claim (i), the second claim follows straightforwardly. We then proceed to prove claim (i).

(ii) We distinguish two cases. Suppose $\mu(s,\mathcal{C}) = 0$. In this case, the claim follows immediately. Now, suppose that $\mu(s,\mathcal{C}) \neq 0$, then using claim (i) we find:

$$\begin{aligned}\mathcal{P}_{\max}(s \Longrightarrow_\mathcal{C} \mathcal{M}) &= \mu(s,\mathcal{M}) + \sum_{s' \in \mathcal{C}} \mathsf{pr}(s,s') \cdot \mathcal{P}_{\max}(s' \Longrightarrow_\mathcal{C} \mathcal{M}) \\ &= \mu(s,\mathcal{M}) + \mathcal{P}_{\max}(s \Longrightarrow_\mathcal{C} \mathcal{M}) \cdot \sum_{s' \in \mathcal{C}} \mathsf{pr}(s,s') \\ &= \mu(s,\mathcal{M}) + \mathcal{P}_{\max}(s \Longrightarrow_\mathcal{C} \mathcal{M}) \cdot \mu(s,\mathcal{C})\end{aligned}$$

from which we obtain: $\mathcal{P}_{\max}(s \Longrightarrow_\mathcal{C} \mathcal{M}) = (\mu(s,\mathcal{M}))/(1-\mu(s,\mathcal{C})) = \mu_\mathcal{C}(s,\mathcal{M})$. We next focus on the proof of claim (i), which together with the above line of reasoning finishes the proof for claim (ii).

(i) Consider an arbitrary equivalence class $\mathcal{C} \in S_{\mathsf{nil}/\mathcal{R}}$. We observe that claim (i) follows immediately when $|\mathcal{C}| = 1$, so the interesting case is when $|\mathcal{C}| > 1$. So, assume that $|\mathcal{C}| > 1$.

We first focus on the non-deterministic nodes in class $\mathcal{C}$: assume that $n \in \mathcal{C} \cap N$. If $\mathcal{P}_{\max}(n \Longrightarrow_\mathcal{C} \mathcal{M}) \neq 0$, then either $n \stackrel{\tau}{\to} p$ for some $p \in \mathcal{M}$ or $n \stackrel{\tau}{\to} p$ for some $p \in \mathcal{C} \cap P$. In the first case, $\mathcal{P}_{\max}(n \Longrightarrow_\mathcal{C} \mathcal{M}) = 1$ and therefore (by definition of branching bisimulation) for all other $t \in \mathcal{C}$, $\mathcal{P}_{\max}(t \Longrightarrow_\mathcal{C} \mathcal{M}) = 1$, and the result follows. In the second case, $\mathcal{P}_{\max}(n \Longrightarrow_\mathcal{C} \mathcal{M}) = \max\{\, \mathcal{P}_{\max}(p \Longrightarrow_\mathcal{C} \mathcal{M}) \mid n \stackrel{\tau}{\to} p \text{ and } p\mathcal{R}n\}$. In other words, for some $p_n \in \mathcal{C} \cap P$ such that $n \stackrel{\tau}{\to} p_n$, we find:

$$\mathcal{P}_{\max}(n \Longrightarrow_\mathcal{C} \mathcal{M}) = \mathcal{P}_{\max}(p_n \Longrightarrow_\mathcal{C} \mathcal{M}). \tag{6}$$

Consequently, it suffices to investigate probabilistic nodes only. Let us assume that there is a node s in $\mathcal{C}$ with the highest maximal probability to reach $\mathcal{M}$ among the other nodes in $\mathcal{C}$ and that there is a node $t \in \mathcal{C}$ with the

strictly smaller maximal probability to reach $\mathcal{M}$. We will show that the assumption that such a node t exists leads to contradiction. Formally, let us assume that

$$\exists\, s \in \mathcal{C} \cap P \;:\; \forall s' \in \mathcal{C} : \mathcal{P}_{\max}(s \Longrightarrow_{\mathcal{C}} \mathcal{M}) \geqslant \mathcal{P}_{\max}(s' \Longrightarrow_{\mathcal{C}} \mathcal{M}) \wedge \;\; \exists t \in \mathcal{C} : \mathcal{P}_{\max}(s \Longrightarrow_{\mathcal{C}} \mathcal{M}) > \mathcal{P}_{\max}(t \Longrightarrow_{\mathcal{C}} \mathcal{M}). \tag{7}$$

Depending on the probability of s to leave the class $\mathcal{C}$ in one transition (i.e. the value of $\mu(s, \mathcal{C})$) we distinguish three cases. We show that each case leads to a contradiction of assumption (7).

($= 0$:) Assume that $\mu(s, \mathcal{C}) = 0$. From the definition of branching bisimulation we immediately obtain $\mathcal{P}_{\max}(s \Longrightarrow_{\mathcal{C}} \mathcal{M}) = \mu(s, \mathcal{M})$, since by all transitions emanating from s the class $\mathcal{C}$ is left. Since $t \in \mathcal{C}$, again by the definition of branching bisimulation there is a scheduler $\sigma_t \in \mathsf{Sched}(t)$ such that $\mathcal{P}(\mathcal{B}_{\sigma_t}(t \Longrightarrow_{\mathcal{C}} \mathcal{M})) = \mu(s, \mathcal{M})$, from which we obtain

$$\mathcal{P}_{\max}(t \Longrightarrow_{\mathcal{C}} \mathcal{M}) \geqslant \mathcal{P}_{\max}(s \Longrightarrow_{\mathcal{C}} \mathcal{M}).$$

But this leads to an immediate violation of assumption (7).

($\neq 0$:) Assume that $0 < \mu(s, \mathcal{C}) < 1$. Then:

$$\begin{aligned}\mathcal{P}_{\max}(s \Longrightarrow_{\mathcal{C}} \mathcal{M}) &= \mu(s, \mathcal{M}) + \sum_{s' \in \mathcal{C}} \mathsf{pr}(s, s') \cdot \mathcal{P}_{\max}(s' \Longrightarrow_{\mathcal{C}} \mathcal{M}) \\ &= \mu(s, \mathcal{M}) + \sum_{s' \in \mathcal{C}, \mathsf{pr}(s,s') > 0} \mathsf{pr}(s, s') \cdot \mathcal{P}_{\max}(s' \Longrightarrow_{\mathcal{C}} \mathcal{M}) \\ &\leqslant \mu(s, \mathcal{M}) + \mu(s, \mathcal{C}) \cdot \mathcal{P}_{\max}(s \Longrightarrow_{\mathcal{C}} \mathcal{M}).\end{aligned}$$

Hence, $\mathcal{P}_{\max}(s \Longrightarrow_{\mathcal{C}} \mathcal{M}) \leqslant (\mu(s, \mathcal{M}))/(1 - \mu(s, \mathcal{C})) = \mu_{\mathcal{C}}(s, \mathcal{M})$. But by the definition of branching bisimulation and the fact that $s \mathcal{R} t$ we find that there is a scheduler $\sigma_t \in \mathsf{Sched}(t)$ such that $\mathcal{P}(t \Longrightarrow_{\mathcal{C}} \mathcal{M}) = \mu_{\mathcal{C}}(s, \mathcal{M})$. Hence, we obtain

$$\mathcal{P}_{\max}(t \Longrightarrow_{\mathcal{C}} \mathcal{M}) \geqslant \mathcal{P}_{\max}(s \Longrightarrow_{\mathcal{C}} \mathcal{M}).$$

This is again in contradiction with assumption (7).

($= 1$:) Assume that $\mu(s, \mathcal{C}) = 1$. Then

$$\mathcal{P}_{\max}(s \Longrightarrow_{\mathcal{C}} \mathcal{M}) = \sum_{n : s \rightsquigarrow n} \mathsf{pr}(s, n) \cdot \mathcal{P}_{\max}(n \Longrightarrow_{\mathcal{C}} \mathcal{M})$$

Now, assume that there is a node $s' \in \mathcal{C}$ such that $s \rightsquigarrow s'$ and $\mathcal{P}_{\max}(s \Longrightarrow_{\mathcal{C}} \mathcal{M}) > \mathcal{P}_{\max}(s' \Longrightarrow_{\mathcal{C}} \mathcal{M})$. Together with assumption (7) and using the fact that $\sum_{n : s \rightsquigarrow n} \mathsf{pr}(s, n) = \mu(s, \mathcal{C}) = 1$ we immediately arrive at a contradiction:

$$\begin{aligned}\mathcal{P}_{\max}(s \Longrightarrow_{\mathcal{C}} \mathcal{M}) &= \sum_{n : s \rightsquigarrow n} \mathsf{pr}(s, n) \cdot \mathcal{P}_{\max}(n \Longrightarrow_{\mathcal{C}} \mathcal{M}) < \sum_{n : s \rightsquigarrow n} \mathsf{pr}(s, n) \cdot \mathcal{P}_{\max}(s \Longrightarrow_{\mathcal{C}} \mathcal{M}) \\ &= \mathcal{P}_{\max}(s \Longrightarrow_{\mathcal{C}} \mathcal{M}).\end{aligned}$$

Therefore, we have that for all $n \in \mathcal{C}$ such that $s \rightsquigarrow n$

$$\mathcal{P}_{\max}(s \Longrightarrow_{\mathcal{C}} \mathcal{M}) = \mathcal{P}_{\max}(n \Longrightarrow_{\mathcal{C}} \mathcal{M}). \tag{8}$$

We continue by assuming that $\sigma \in \mathsf{Sched}(s)$ is a scheduler that yields $\mathcal{P}_{\max}(s \Longrightarrow_{\mathcal{C}} \mathcal{M})$. We now consider the computation tree $\mathsf{CT}(s, \sigma)$. We will first show that all nodes from $\mathcal{C}$ that appear in $\mathsf{CT}(s, \sigma)$ have the same maximal probability as s to (silently) reach $\mathcal{M}$. Clearly, it is possible that not all nodes from $\mathcal{C}$ appear in $\mathsf{CT}(s, \sigma)$. In order to prove the claim for those nodes, we show that at least for one probabilistic node s' which appears in $\mathsf{CT}(s, \sigma)$, $\mu(s', \mathcal{C}) \neq 1$ holds. Then the result follows from the previous two cases that we considered.

First we observe that for every node p that appears in $\mathsf{CT}(s, \sigma)$ the scheduler σ induces at least one scheduler $\sigma_p \in \mathsf{Sched}(p)$ satisfying:

$$\mathcal{P}_{\max}(p \Longrightarrow_{\mathcal{C}} \mathcal{M}) = \mathcal{P}(\mathcal{B}_{\sigma_p}(p \Longrightarrow_{\mathcal{C}} \mathcal{M})). \tag{9}$$

Second, let p_n be a probabilistic node in $\mathsf{CT}(s,\sigma)$ with depth 3, that is, $s \rightsquigarrow n$ and $n \xrightarrow{\tau} p_n$ for some n. (Note that this means that for a path $s\ l\ n$, the scheduler σ schedules transition $n \xrightarrow{\tau} p_n$, that is, $\sigma(s\ l\ n) = n \xrightarrow{\tau} p_n$.) We identify the node p_n in the graph with the node $s\ l\ n\ \tau\ p_n$ in the computation tree $\mathsf{CT}(s,\sigma)$. Clearly, $\mathcal{P}(\mathcal{B}_{\sigma_n}(n \Longrightarrow_{\mathcal{C}} \mathcal{M})) = \mathcal{P}(\mathcal{B}_{\sigma_{p_n}}(p_n \Longrightarrow_{\mathcal{C}} \mathcal{M}))$. Moreover, from (9) it follows that $\mathcal{P}_{\max}(n \Longrightarrow_{\mathcal{C}} \mathcal{M}) = \mathcal{P}(\mathcal{B}_{\sigma_n}(n \Longrightarrow_{\mathcal{C}} \mathcal{M}))$ and $\mathcal{P}_{\max}(p_n \Longrightarrow_{\mathcal{C}} \mathcal{M}) = \mathcal{P}(\mathcal{B}_{\sigma_{p_n}}(p_n \Longrightarrow_{\mathcal{C}} \mathcal{M}))$. According to Eq. (8), $\mathcal{P}_{\max}(s \Longrightarrow_{\mathcal{C}} \mathcal{M}) = \mathcal{P}_{\max}(n \Longrightarrow_{\mathcal{C}} \mathcal{M})$. From this, we can conclude that $\mathcal{P}_{\max}(s \Longrightarrow_{\mathcal{C}} \mathcal{M}) = \mathcal{P}(\mathcal{B}_{\sigma_{p_n}}(p_n \Longrightarrow_{\mathcal{C}} \mathcal{M})) = \mathcal{P}_{\max}(p_n \Longrightarrow_{\mathcal{C}} \mathcal{M})$.Using induction on the depth of the probabilistic node in $\mathsf{CT}(s,\sigma)$ the claim that for every node s' that appears in $\mathsf{CT}(s,\sigma)$, $\mathcal{P}_{\max}(s' \Longrightarrow_{\mathcal{C}} \mathcal{M}) = \mathcal{P}_{\max}(s \Longrightarrow_{\mathcal{C}} \mathcal{M})$ can be shown to hold.

For the nodes that do not appear in $\mathsf{CT}(s,\sigma)$, we proceed as follows: Now, if we assume that for every node s' that appears in $\mathsf{CT}(s,\sigma)$, $\mu(s',\mathcal{C}) = 1$ holds, then we obtain that $\mathcal{P}_{\max}(s \Longrightarrow_{\mathcal{C}} \mathcal{M}) = 0$ which contradicts (7). Therefore, there is a node p that appears in $\mathsf{CT}(s,\sigma)$ with $\mu(p,\mathcal{C}) < 1$ and for which $\mathcal{P}_{\max}(p \Longrightarrow_{\mathcal{C}} \mathcal{M}) = \mathcal{P}_{\max}(s \Longrightarrow_{\mathcal{C}} \mathcal{M})$ as proven above. The conclusion follows from the previous analysis.

Summarising, we conclude that assumption (7) leads to a contradiction. Hence, we have proven that the following claim holds:

$$\forall s, s' \in \mathcal{C} \ :\ \mathcal{P}_{\max}(s \Longrightarrow_{\mathcal{C}} \mathcal{M}) = \mathcal{P}_{\max}(s' \Longrightarrow_{\mathcal{C}} \mathcal{M}). \quad \square \tag{10}$$

Lemma 19 shows that all nodes in one equivalence class have the same maximal probabilities to reach another class $\mathcal{M}$ via a set of τ-paths. Moreover, this maximal probability equals the normalised cumulative probability of reaching that class. Henceforth, we use the notation $\mathcal{P}_{\max}(s \Longrightarrow_{[s]_{\mathcal{R}}} \mathcal{M})$ and $\mu_{[s]_{\mathcal{R}}}([s]_{\mathcal{R}}, \mathcal{M})$ interchangeably for all $s \in S_{\mathsf{nil}}$.

Proposition 20. *Let $\mathcal{R}$ be a branching bisimulation relation on S_{nil} and let $\mathcal{C} \in S_{\mathsf{nil}/\mathcal{R}}$. If there is a node $n \in \mathcal{C}$ such that $n \xrightarrow{a} p$ for $a \neq \tau$, then $\mu_{\mathcal{C}}(\mathcal{C},\mathcal{M}) = 0$ for all equivalence classes $\mathcal{M} \in S_{\mathsf{nil}/\mathcal{R}}$ satisfying $\mathcal{M} \neq \mathcal{C}$.*

Proof. This follows from the definition of branching bisimulation. (Hint: we can conclude that for each $q \in P \cap \mathcal{C}$ if $q \rightsquigarrow n'$ then $n' \in \mathcal{C}$.) $\square$

From this proposition, we immediately find the following lemma, which together with Lemma 19 wraps up the proof for the implication part of Theorem 18.

Lemma 21. *Let $\mathcal{R}$ be a branching bisimulation relation on S_{nil}, and let $a \in \mathrm{Act}$. If $s\mathcal{R}t$, then $\mathcal{P}_{\max}(s \overset{a}{\Longrightarrow}_{[s]_{\mathcal{R}}} \mathcal{M}) = \mathcal{P}_{\max}(t \overset{a}{\Longrightarrow}_{[t]_{\mathcal{R}}} \mathcal{M})$.*

Proof. From Proposition 20 it follows that for every node s, all actions $a \in \mathrm{Act}$ and equivalence classes $\mathcal{M}$, $\mathcal{P}_{\max}(s \overset{a}{\Longrightarrow}_{[s]_{\mathcal{R}}} \mathcal{M}) = 0$ or $\mathcal{P}_{\max}(s \overset{a}{\Longrightarrow}_{[s]_{\mathcal{R}}} \mathcal{M}) = 1$. The claim then follows immediately. $\square$

Next, we focus on the proof of the contraposition of Theorem 18. We repeat this part of the theorem as Lemma 22.

Lemma 22. *Let $\mathcal{R}$ be an equivalence relation on S_{nil}. Then $\mathcal{R}$ is a branching bisimulation relation if for all nodes $s, t \in S_{\mathsf{nil}}$, for which $s\mathcal{R}t$ holds, the following two conditions are met*:

i. $\mathcal{P}_{\max}(s \overset{a}{\Longrightarrow}_{[s]_{\mathcal{R}}} \mathcal{M}) = \mathcal{P}_{\max}(t \overset{a}{\Longrightarrow}_{[t]_{\mathcal{R}}} \mathcal{M})$ *for all* $a \in \mathrm{Act}$ *and* $\mathcal{M} \in S_{\mathsf{nil}/\mathcal{R}}$.
ii. $\mathcal{P}_{\max}(s \Longrightarrow_{[s]_{\mathcal{R}}} \mathcal{M}) = \mathcal{P}_{\max}(t \Longrightarrow_{[t]_{\mathcal{R}}} \mathcal{M})$ *for all* $\mathcal{M} \in S_{\mathsf{nil}/\mathcal{R}} \setminus \{[s]_{\mathcal{R}}\}$.

Proof. Assume that $\mathcal{R}$ satisfies the conditions of the above lemma. We need to prove that $\mathcal{R}$ satisfies the two conditions of Definition 9.

a. Let $s\mathcal{R}t$, $s \in N$ and $s \xrightarrow{a} s'$. Hence, $\mathcal{P}_{\max}(s \overset{a}{\Longrightarrow}_{[s]_{\mathcal{R}}} [s']_{\mathcal{R}}) = 1$. From the first condition (i) it follows that $\mathcal{P}_{\max}(t \overset{a}{\Longrightarrow}_{[t]_{\mathcal{R}}} [s']_{\mathcal{R}}) = 1$ as well, which (directly) implies the correctness of the first condition of Definition 9.
b. Let $s \in P$ and $s\mathcal{R}t$. We distinguish two cases:
   b.1 Assume that $\mu(s,[s]_{\mathcal{R}}) = 1$. Then $\mu(s,\mathcal{M}) = 0$ for all $\mathcal{M} \in S_{\mathsf{nil}/\mathcal{R}} \setminus \{[s]_{\mathcal{R}}\}$. We define a scheduler $\sigma \in \mathsf{Sched}(t)$ as $\sigma(t) = \bot$. Then $\mathcal{P}(\mathcal{B}_{\sigma}(t \Longrightarrow_{[t]_{\mathcal{R}}} \mathcal{M})) = 0$ and therefore, $\mu_{[s]_{\mathcal{R}}}(s,\mathcal{M}) = \mathcal{P}(\mathcal{B}_{\sigma}(t \Longrightarrow_{[t]_{\mathcal{R}}} \mathcal{M}))$ for all $\mathcal{M} \in S_{\mathsf{nil}/\mathcal{R}} \setminus \{[s]_{\mathcal{R}}\}$.

b.2 Assume that $\mu(s, [s]_{\mathcal{R}}) < 1$ and $\mathcal{M} \in S_{\mathsf{nil}/\mathcal{R}} \setminus \{[s]_{\mathcal{R}}\}$. Then

$$\mathcal{P}_{\max}(s \Longrightarrow_{[s]_{\mathcal{R}}} \mathcal{M}) = \mu(s, \mathcal{M}) + \mu(s, [s]_{\mathcal{R}}) \cdot \mathcal{P}_{\max}(s \Longrightarrow_{[s]_{\mathcal{R}}} \mathcal{M})$$

from which we derive

$$\mathcal{P}_{\max}(s \Longrightarrow_{[s]_{\mathcal{R}}} \mathcal{M}) = \mu_{[s]_{\mathcal{R}}}(s, \mathcal{M}). \tag{11}$$

Subcase b.2.1: Assume that there is an $n \in [s]_{\mathcal{R}} \cap N$ such that $n \xrightarrow{\tau} \mathcal{M}$. Since $\mathcal{P}_{\max}(n \Longrightarrow_{[n]_{\mathcal{R}}} \mathcal{M}) = 1$ it follows that $\mathcal{P}_{\max}(s \Longrightarrow_{[s]_{\mathcal{R}}} \mathcal{M}) = 1$ as well. Moreover, from (11) we have that $\mu_{[s]_{\mathcal{R}}}(s, \mathcal{M}) = 1$. Therefore, $\mu_{[s]_{\mathcal{R}}}(s, \mathcal{M}') = 0$ for $\mathcal{M}' \neq \mathcal{M}$ and $\mathcal{M}' \neq [s]_{\mathcal{R}}$, since $\mu_{[s]_{\mathcal{R}}}(s, \cdot)$ is a probability mass function over set $S_{\mathsf{nil}} \setminus [s]_{\mathcal{R}}$.[3] Hence, using (11) we obtain $\mathcal{P}_{\max}(s \Longrightarrow_{[s]_{\mathcal{R}}} \mathcal{M}') = 0$ for every $\mathcal{M}' \in S_{\mathsf{nil}/\mathcal{R}} \setminus \{[s]_{\mathcal{R}}, \mathcal{M}\}$. Then $s\mathcal{R}t$ implies $\mathcal{P}_{\max}(t \Longrightarrow_{[t]_{\mathcal{R}}} \mathcal{M}) = 1$ and $\mathcal{P}_{\max}(t \Longrightarrow_{[t]_{\mathcal{R}}} \mathcal{M}') = 0$. We take a scheduler $\eta \in \mathsf{Sched}(t)$ such that $\mathcal{P}(\mathcal{B}_{\eta}(t \Longrightarrow_{[t]_{\mathcal{R}}} \mathcal{M})) = \mathcal{P}_{\max}(t \Longrightarrow_{[t]_{\mathcal{R}}} \mathcal{M})$ for which $\mathcal{P}(\mathcal{B}_{\eta}(t \Longrightarrow_{[t]_{\mathcal{R}}} \mathcal{M}')) = 0$, $\mathcal{M} \neq \mathcal{M}'$ and $\mathcal{M}' \neq [s]_{\mathcal{R}}$, since this holds for any scheduler in $\mathsf{Sched}(t)$. Thus η is the required scheduler in the second condition of Definition 9.

Subcase b.2.2: We next analyse the case in which for all $n \in [s]_{\mathcal{R}} \cap N$, $n \xrightarrow{\tau} s'$ implies $n\mathcal{R}s'$. We aim to show that all maximal probabilities $\mathcal{P}_{\max}(t \Longrightarrow_{[t]_{\mathcal{R}}} \mathcal{M})$, for $\mathcal{M} \in S_{\mathsf{nil}/\mathcal{R}} \setminus \{[s]_{\mathcal{R}}\}$, can be obtained by a single scheduler $\eta \in \mathsf{Sched}(t)$. This, together with (11), brings the proof to an end.

First, we sketch the approach we take. Given two arbitrary equivalence classes $\mathcal{M}_1$ ($\mathcal{M}_1 \neq [t]_{\mathcal{R}}$) and $\mathcal{M}_2$ ($\mathcal{M}_2 \neq [t]_{\mathcal{R}}$), we show that if one scheduler induces the maximal probability $\mathcal{P}_{\max}(t \Longrightarrow_{[t]_{\mathcal{R}}} \mathcal{M}_1)$ then the same scheduler also induces the maximal probability $\mathcal{P}_{\max}(t \Longrightarrow_{[t]_{\mathcal{R}}} \mathcal{M}_2)$. This procedure can be extended and generalised over all equivalence classes $\mathcal{M}_i \in S_{\mathsf{nil}/\mathcal{R}} \setminus \{[s]_{\mathcal{R}}\}$ for which $\mathcal{P}_{\max}(t \Longrightarrow_{[t]_{\mathcal{R}}} \mathcal{M}_i) > 0$.

So, assume that $\mathcal{M}_1, \mathcal{M}_2 \in S_{\mathsf{nil}/\mathcal{R}} \setminus \{[t]_{\mathcal{R}}\}$ with $\mathcal{P}_{\max}(t \Longrightarrow_{[t]_{\mathcal{R}}} \mathcal{M}_1) > 0$ and $\mathcal{P}_{\max}(t \Longrightarrow_{[t]_{\mathcal{R}}} \mathcal{M}_2) > 0$. Let $\eta_1, \eta_2 \in \mathsf{Sched}(t)$ and $\mathcal{P}(\mathcal{B}_{\eta_1}(t \Longrightarrow_{[t]_{\mathcal{R}}} \mathcal{M}_1)) = \mathcal{P}_{\max}(t \Longrightarrow_{[t]_{\mathcal{R}}} \mathcal{M}_1)$ and $\mathcal{P}(\mathcal{B}_{\eta_2}(t \Longrightarrow_{[t]_{\mathcal{R}}} \mathcal{M}_2)) = \mathcal{P}_{\max}(t \Longrightarrow_{[t]_{\mathcal{R}}} \mathcal{M}_2)$. Now let us consider the computation trees $\mathsf{CT}(t, \eta_1)$ and $\mathsf{CT}(t, \eta_2)$. Note that both computation trees have t as a root. Assume that schedulers η_1 and η_2 schedule the same transitions up to a node with depth k. In addition, we assume that all nodes with depth k for which η_1 and η_2 schedule differently are ordered by $<_k$. Suppose that n_{k1} is the least node (by the ordering) with depth k for which $\eta_1(c) \neq \eta_2(c)$ where n_{k1} occurs in both $\mathsf{CT}(t, \eta_1)$ and $\mathsf{CT}(t, \eta_2)$ due to a node c in both computation trees (this node c is a path $c \in \mathsf{SPath}(t, \eta_1) \cap \mathsf{SPath}(t, \eta_2)$ and hence a node in both computation trees, because we have assumed that both schedulers η_1 and η_2 schedule in the same way for all prefixes of the path c). Clearly, $\mathsf{last}(c) = n_{k1}$. Moreover, n_{k1} is a nondeterministic node as η_1 and η_2 cannot schedule a probabilistic node differently! Let us assume $\eta_1(c) = n_{k1} \xrightarrow{\tau} p^1$ and $\eta_2(c) = n_{k1} \xrightarrow{\tau} p^2$. From our assumption we have that $p^1, p^2 \in [t]_{\mathcal{R}}$. Therefore,

$$\mathcal{P}_{\max}(p^1 \Longrightarrow_{[t]_{\mathcal{R}}} \mathcal{M}_2) = \mathcal{P}_{\max}(p^2 \Longrightarrow_{[t]_{\mathcal{R}}} \mathcal{M}_2) = \mathcal{P}_{\max}(t \Longrightarrow_{[t]_{\mathcal{R}}} \mathcal{M}_2).$$

Thus, there is a scheduler $\eta_2^{(1)} \in \mathsf{Sched}(p^1)$ for which we have $\mathcal{P}(\mathcal{B}_{\eta_2^{(1)}}(p^1 \Longrightarrow_{[t]_{\mathcal{R}}} \mathcal{M}_2)) = \mathcal{P}_{\max}(p^1 \Longrightarrow_{[t]_{\mathcal{R}}} \mathcal{M}_2)$. Now we have two schedulers: $(\eta_1 - c\tau p^1)$, which we denote by $\eta_1^{(1)}$, and $\eta_2^{(1)}$ in $\mathsf{Sched}(p^1)$. For these schedulers, we find $\mathcal{P}_{\max}(p^1 \Longrightarrow_{[t]_{\mathcal{R}}} \mathcal{M}_1) = \mathcal{P}(\mathcal{B}_{\eta_1^{(1)}}(p^1 \Longrightarrow_{[t]_{\mathcal{R}}} \mathcal{M}_1))$ and $\mathcal{P}_{\max}(p^1 \Longrightarrow_{[t]_{\mathcal{R}}} \mathcal{M}_2) = \mathcal{P}(\mathcal{B}_{\eta_2^{(1)}}(p^1 \Longrightarrow_{[t]_{\mathcal{R}}} \mathcal{M}_2))$ and $\eta_1^{(1)}(c) = \eta_2^{(1)}(c) = n_{k1} \xrightarrow{\tau} p^1$ (note that we have preferred the transition scheduled by η_1 over the transition scheduled by η_2). Moreover, p^1 is "closer" to $\mathcal{M}_1$ and $\mathcal{M}_2$ than the root (t) of $\mathsf{CT}(t, \eta_1)$ and $\mathsf{CT}(t, \eta_2)$ in the sense that all paths that start at t and reach $\mathcal{M}_1$ or $\mathcal{M}_2$ in $\mathsf{CT}(t, \eta_1)$ or $\mathsf{CT}(t, \eta_2)$, respectively are finite. Note that the set of infinite paths that are part of the computation trees have probability measure 0, and hence, we do not need to consider those. The procedure continues by comparing the schedulers $\eta_1^{(1)}$ and $\eta_2^{(1)}$ in the same way we have done it with η_1 and η_2. Remark that node p^1 cannot be processed further until all nondeterministic nodes with depth k have been investigated for η_1 and η_2.

[3] This probability mass function describes a discrete random variable X representing a node reached in one probabilistic transition from s under the condition that the class $[s]_{\mathcal{R}}$ is left.

If transitions scheduled by $\eta_1^{(m)}$, $m \geqslant 1$ (which are all basically induced by the scheduler η_1) are always chosen over transitions by any other scheduler in consideration, we obtain that η_1 induces the maximal probability $\mathcal{P}_{\max}(t \Longrightarrow_{[t]_\mathcal{R}} \mathcal{M}_2)$ as well. □

Proof of Theorem 18. Follows immediately from Lemmata 19 and 21. □

As a result of Theorem 18, we find the following corollary, which states that branching bisimilarity is an equivalence relation on the set of all graphs.

Corollary 23. *$\leftrightarrow_b$ is an equivalence relation on* **G**.

3.3. Deciding branching bisimulation

In this section, we extend the result that we obtained in the previous section. More concretely, we show that the alternative definition of branching bisimulation in terms of maximal probabilities is at the basis for deciding branching bisimulation. In line with the results obtained by Philippou et al. [18], we show that it suffices to consider a finite subset of all possible schedulers for a given graph. Whereas in [18], so-named determinate schedulers are introduced and used, we draw our attention to an even smaller class of schedulers, viz. the class of simple schedulers (see also Section 2.2). Remark that the computation tree under a simple scheduler can always be represented by a fully probabilistic graph, even when the computation tree itself may be of infinite size. This fact can be used to show that deciding branching bisimulation amounts to solving a linear optimisation problem.

We proceed as follows. First, the main theorem of this section is stated and proved, showing that among the schedulers that induce maximal probabilities, there is always at least one simple scheduler.

Theorem 24. *Let x be a graph. We denote the set of its nodes by S and $S_{\mathsf{nil}} = S \cup \{\mathsf{nil}\}$. Let $\mathcal{R}$ be a branching bisimulation relation on S_{nil}. Let $s \in S_{\mathsf{nil}}$, $a \in \mathrm{Act}_\tau$ and $\mathcal{M} \in S_{\mathsf{nil}/\mathcal{R}}$. Then, there is a simple scheduler σ' such that $\mathcal{P}_{\max}(s \stackrel{a}{\Longrightarrow}_{[s]_\mathcal{R}} \mathcal{M}) = \mathcal{P}(\mathcal{B}_{\sigma'}(s \stackrel{a}{\Longrightarrow}_{[s]_\mathcal{R}} \mathcal{M}))$ when $a \neq \tau$ or $\mathcal{M} \neq [s]_\mathcal{R}$.*

Proof. We show that from a given scheduler σ with $\mathcal{P}_{\max}(s \stackrel{a}{\Longrightarrow}_{[s]_\mathcal{R}} \mathcal{M}) = \mathcal{P}(\mathcal{B}_\sigma(s \stackrel{a}{\Longrightarrow}_{[s]_\mathcal{R}} \mathcal{M})) > 0$ a scheduler σ' can be derived such that $\mathcal{P}_{\max}(s \stackrel{a}{\Longrightarrow}_{[s]_\mathcal{R}} \mathcal{M}) = \mathcal{P}(\mathcal{B}_{\sigma'}(s \stackrel{a}{\Longrightarrow}_{[s]_\mathcal{R}} \mathcal{M}))$ and which for all paths that end in a node t (for some t) schedules the same transition. The case $\mathcal{P}_{\max}(s \stackrel{a}{\Longrightarrow}_{[s]_\mathcal{R}} \mathcal{M}) = 0$ is trivial, so we assume that we have $\mathcal{P}_{\max}(s \stackrel{a}{\Longrightarrow}_{[s]_\mathcal{R}} \mathcal{M}) > 0$.

Let $\mathcal{R}$ be a branching bisimulation relation on S_{nil}. Assume that scheduler $\sigma \in \mathsf{Sched}(s)$ is such that $\mathcal{P}_{\max}(s \stackrel{a}{\Longrightarrow}_{[s]_\mathcal{R}} \mathcal{M}) = \mathcal{P}(\mathcal{B}_\sigma(s \stackrel{a}{\Longrightarrow}_{[s]_\mathcal{R}} \mathcal{M}))$. Assume that node t appears in $\mathsf{CT}(s, \sigma)$. We distinguish between the case when t has finitely many occurrences and the case where t has infinitely many occurrences in the computation tree.

(1) Suppose t has finitely many occurrences in $\mathsf{CT}(s, \sigma)$: then there is an occurrence t_c (i.e. t is due to c) in $\mathsf{CT}(s, \sigma)$ such that the appearance of t in the subtree of $\mathsf{CT}(s, \sigma)$ with the root in c is only due to c. Or in terms of $\mathcal{B}$ set, $\mathcal{B}_{(\sigma-c)}(t \stackrel{a}{\Longrightarrow}_{[t]_\mathcal{R}} \mathcal{M}) = \mathcal{B}_{(\sigma-c)}(t \stackrel{a}{\Longrightarrow}_{[t]_\mathcal{R}-t} \mathcal{M})$ where $(\sigma-c)$ is the scheduler in $\mathsf{Sched}(t)$ induced by σ as described in Section 2.2. Clearly, $\mathsf{CT}(t, (\sigma-c))$ does not have any occurrence of t except its root. Now, we can define a scheduler $\sigma' \in \mathsf{Sched}(s)$ that schedules the same transitions to all paths that end at t.

$$\sigma'(c) = \begin{cases} \sigma(c) & \text{if } t \notin \mathsf{nodes}(c), \\ (\sigma-c)(c'') & \text{if } c \equiv c' \circ c'', \mathsf{first}(c'') = t \text{ and } t \notin \mathsf{nodes}(\mathsf{rest}(c'')). \end{cases} \tag{12}$$

Note that c' may be t in which case $c' \circ c'' = c''$.

(2) Suppose that t has infinitely many occurrences in $\mathsf{CT}(s, \sigma)$. If there is a subtree of $\mathsf{CT}(s, \sigma)$ with root in some occurrence of t which does not contain any other occurrences of t, then we proceed in the same way as in the previous case.

Now assume that there is no such a subtree of $\mathsf{CT}(s,\sigma)$. This means that for every occurrence of t in $\mathsf{CT}(s,\sigma)$ there is a path in $\mathsf{CT}(s,\sigma)$ starting in this occurrence of t that passes infinitely many times through t. Note that this does not mean that all paths starting in this occurrence of t have to pass infinitely many times through t. On the contrary, according to Proposition 16 there is an occurrence t_c in $\mathsf{CT}(s,\sigma)$, such that $\mathcal{P}(\mathcal{B}_{(\sigma-c)}(t \stackrel{a}{\Longrightarrow}_{[t]_{\mathcal{R}}-t} \mathcal{M})) > 0$.
Now we focus out attention on the tree $\mathsf{CT}(t,(\sigma-c))$. For short let us denote $(\sigma-c)$ by η. Let us enumerate all (different) occurrences t_{c_i}, $i > 0$ in this tree where t is due to c_i in $\mathsf{CT}(t,\eta)$. We define a function *t-depth* which to every occurrence t_{c_i} in $\mathsf{CT}(t,\eta)$ assigns the number of times c_i passes through t including the ending in t and excluding the starting in t (i.e. the root of $\mathsf{CT}(t,\eta)$). W.L.G. we can assume that $\mathsf{CT}(t,(\eta-c_i)) = \mathsf{CT}(t,(\eta-c_j))$ if $t\text{-}depth(t_i) = t\text{-}depth(t_j)$. Thus we start with the computation tree $\mathsf{CT}(t,\eta)$. Then

$$\begin{aligned}\mathcal{P}(\mathcal{B}_{\eta}(t \stackrel{a}{\Longrightarrow}_{[t]_{\mathcal{R}}} \mathcal{M})) &= \mathcal{P}(\mathcal{B}_{\eta}(t \stackrel{a}{\Longrightarrow}_{[t]_{\mathcal{R}}-t} \mathcal{M})) + \mathcal{P}(\mathcal{B}_{\eta}(t \stackrel{a}{\Longrightarrow}_{[t]_{\mathcal{R}}+t} \mathcal{M}))\\ &= \mathcal{P}(\mathcal{B}_{\eta}(t \stackrel{a}{\Longrightarrow}_{[t]_{\mathcal{R}}-t} \mathcal{M})) + \mathbf{P}(c_i)\cdot\mathcal{P}(\mathcal{B}_{(\eta-c_i)}(t \stackrel{a}{\Longrightarrow}_{[t]_{\mathcal{R}}} \mathcal{M})),\end{aligned}$$

where $(\eta-c_i)$ is the scheduler in $\mathsf{Sched}(t)$ induced by η as described in Section 2.2. Moreover, $\mathcal{P}(\mathcal{B}_{(\eta-c_i)}(t \stackrel{a}{\Longrightarrow}_{[t]_{\mathcal{R}}} \mathcal{M})) = \mathcal{P}_{\max}(t \stackrel{a}{\Longrightarrow}_{[t]_{\mathcal{R}}} \mathcal{M}) = \mathcal{P}(\mathcal{B}_{\eta}(t \stackrel{a}{\Longrightarrow}_{[t]_{\mathcal{R}}} \mathcal{M})) = \mathcal{P}(\mathcal{B}_{\sigma}(s \stackrel{a}{\Longrightarrow}_{[s]_{\mathcal{R}}} \mathcal{M}))$. Recall that c_i denotes the unique scheduled path from the root t to the occurrence t_{c_i} with $t-depth$ 1 in $\mathsf{CT}(t,\eta)$. (According to our assumption there is only one such an occurrence, otherwise the second summand would be $\sum_{t\text{-depth}}(c_i) = 1\,\mathbf{P}(c_i)\cdot\mathcal{P}(\mathcal{B}_{(\eta-c_i)}(t \stackrel{a}{\Longrightarrow}_{[t]_{\mathcal{R}}} \mathcal{M}))$.) Let us denote $\alpha = \mathcal{P}(\mathcal{B}_{(\sigma-c)}(t \stackrel{a}{\Longrightarrow}_{[t]_{\mathcal{R}}-t} \mathcal{M}))$ and $\beta = \mathbf{P}(c_i)$. Note that $\beta \neq 1$ since $\alpha > 0$. Now we obtain easily that $\mathcal{P}(\mathcal{B}_{(\sigma-c)}(t \stackrel{a}{\Longrightarrow}_{[t]_{\mathcal{R}}} \mathcal{M})) = \alpha/1-\beta$.
We proceed by defining a scheduler $\sigma' \in \mathsf{Sched}(t)$ (which can easily be extended to a scheduler starting in s) that schedules the same transitions to all paths that end in t:

$$\sigma'(c\circ t\circ c') \stackrel{\text{def}}{=} (\sigma-c)(t\circ c'), \quad \text{where } t \notin \mathsf{rest}(c'). \tag{13}$$

Then

$$\begin{aligned}\mathcal{P}(\mathcal{B}_{\sigma'}(t \stackrel{a}{\Longrightarrow}_{[t]_{\mathcal{R}}} \mathcal{M})) &= \mathcal{P}(\mathcal{B}_{\sigma'}(t \stackrel{a}{\Longrightarrow}_{[t]_{\mathcal{R}}-t} \mathcal{M})) + \mathcal{P}(\mathcal{B}_{\sigma'}(t \stackrel{a}{\Longrightarrow}_{[t]_{\mathcal{R}}+t} \mathcal{M}))\\ &= \mathcal{P}(\mathcal{B}_{(\sigma-c)}(t \stackrel{a}{\Longrightarrow}_{[t]_{\mathcal{R}}-t} \mathcal{M}) + \mathbf{P}(c_i)\cdot\mathcal{P}(\mathcal{B}_{\sigma'}(t \stackrel{a}{\Longrightarrow}_{[t]_{\mathcal{R}}} \mathcal{M})).\end{aligned}$$

from which $\mathcal{P}(\mathcal{B}_{\sigma'}(t \stackrel{a}{\Longrightarrow}_{[t]_{\mathcal{R}}} \mathcal{M})) = \alpha + \beta\cdot\mathcal{P}(\mathcal{B}_{\sigma'}(t \stackrel{a}{\Longrightarrow}_{[t]_{\mathcal{R}}} \mathcal{M}))$ and finally,

$$\mathcal{P}(\mathcal{B}_{\sigma'}(t \stackrel{a}{\Longrightarrow}_{[t]_{\mathcal{R}}} \mathcal{M})) = \frac{\alpha}{1-\beta}.$$

With this we have shown that $\mathcal{P}_{\max}(t \stackrel{a}{\Longrightarrow}_{[t]_{\mathcal{R}}} \mathcal{M}) = \mathcal{P}(\mathcal{B}_{\sigma'}(t \stackrel{a}{\Longrightarrow}_{[t]_{\mathcal{R}}} \mathcal{M}))$. □

As we already mentioned, the above result holds the key to the algorithm of polynomial time complexity for deciding branching bisimulation.

The algorithm for deciding branching bisimulation is similar to the algorithm for deciding weak bisimulation described in [18]. Since the reader can find many details in that paper, we will not elaborate on those details here.

The technique that is employed by the algorithm uses the well-known partitioning technique (which is also used in algorithms for deciding other bisimulation relations [12]). Starting from the trivial partition $\{S_{\mathsf{nil}}\}$, a sequence of partitions of S_{nil} is generated, each of them finer than any previous. The procedure is repeated until a partition that corresponds to a branching equivalence is obtained. A partition is refined by means of a *splitter*.

Definition 25. Let Π be a partition of S_{nil}. The tuple $(\mathcal{C},a,\mathcal{M})$, where $\mathcal{C} \in \Pi$, $a \in \mathrm{Act}_{\tau}$ and $\mathcal{M} \in \Pi$, is a *splitter* of Π if there are $s, s' \in \mathcal{C}$ for which $\mathcal{P}_{\max}(s \stackrel{a}{\Longrightarrow}_{\mathcal{C}} \mathcal{M}) \neq \mathcal{P}_{\max}(s' \stackrel{a}{\Longrightarrow}_{\mathcal{C}} \mathcal{M})$ where $a \neq \tau$ or $\mathcal{M} \neq \mathcal{C}$.

In other words, a splitter $(\mathcal{C},a,\mathcal{M})$ is found if the partition does not correspond to a branching bisimulation. Namely, the class $\mathcal{C}$ contains two nodes that do not have the same maximal probability to reach the class $\mathcal{M}$ by executing a.

Thus, $\mathcal{C}$ should be split into at least two classes in the next partition. The main algorithm is given below. It calls several procedures that we explain afterwards.

Input: $\langle N, P, s, \mathrm{Act}, \rightarrow, \rightsquigarrow, \mathsf{pr}\rangle$
Output: $S_{\mathsf{nil}/\ \leftrightarrows_b}$
Steps: $\Pi := \{S_{\mathsf{nil}}\}$;
$(\mathcal{C}, a, \mathcal{M}) := \mathit{FindSplit}(\Pi)$;
while $\mathcal{C} \neq \emptyset$ **do**
$\Pi := \mathit{Refine}(\Pi, \mathcal{C}, a, \mathcal{M})$
$(\mathcal{C}, a, \mathcal{M}) := \mathit{FindSplit}(\Pi)$
od
return Π

The procedure $\mathit{FindSplit}(\Pi)$ for partition Π finds and returns a splitter $(\mathcal{C}, a, \mathcal{M})$ if one exists and returns $(\emptyset, \varepsilon, \emptyset)$ otherwise.

Input: Π
Output: $(\mathcal{C}, a, \mathcal{M})$
Steps: **for** $a \in \mathrm{Act}_\tau$ **do**
for $\mathcal{C} \in \Pi$ **do**
for $\mathcal{M} \in \Pi$ **do**
for $s, s' \in \mathcal{C}$ **do**
$\max_s := \mathit{FindMax}(s, a, \mathcal{M})$;
$\max_{s'} := \mathit{FindMax}(s', a, \mathcal{M})$;
if $\max_s \neq \max_{s'}$ **return** $(\mathcal{C}, a, \mathcal{M})$
od
od
od
od
return $(\emptyset, \varepsilon, \emptyset)$

The function $\mathit{FindMax}(s, a, \mathcal{M})$ computes the maximal probability to reach $\mathcal{M}$ from s by executing a. To this end to each node s a variable X_s^τ is associated, and if $a \neq \tau$ a variable X_s^a as well. The variables are bound by the following system of equations:

$$X_s^a = \begin{cases} \sum_{s \overset{\pi}{\rightsquigarrow} t} \pi \cdot X_t^a, & s \in S_p, t \in [s]_\Pi, \\ 1, & s \in S_n, s \in \mathcal{M}, \\ \max\{X_t^a \mid s \xrightarrow{\tau} t\}, & s \in S_n, s \overset{a}{\not\longrightarrow} \mathcal{M}, t \in [s]_\Pi, \\ 0 & \text{otherwise}, \end{cases}$$

$$X_s^\tau = \begin{cases} 1, & s \in \mathcal{M}, \\ \sum_{s \overset{\pi}{\rightsquigarrow} t} \pi \cdot X_t^\tau, & s \in S_p, s \notin \mathcal{M}, t \in [s]_\Pi, \\ \max\{X_t^\tau \mid s \xrightarrow{\tau} t\}, & s \in S_n, s \notin \mathcal{M}, t \in [s]_\Pi, \\ 0 & \text{otherwise}. \end{cases}$$

As explained in [18] a solution of a system in such a form can be found by solving a linear optimisation programming problem. Namely, for all equations in the form $X = \max\{X_i \mid i \in I\}$ a set of inequalities $X \geqslant X_i$ is introduced and then the optimisation problem reduces to finding minimum of the function $\sum_{s \in S} X_s^\tau + X_s^a$. This problem can be solved in polynomial time in the number of variables that are involved.

4. Colours and blends

The focus of the previous section was on the understanding of the interplay between probabilities and functionality, i.e. we looked at the notion of branching bisimulation from a rather quantitative point of view. In the remaining sections of this paper, we investigate branching bisimulation from a different perspective, more focused on the qualitative aspects of branching bisimulation.

We claimed, in our introduction (and repeated this in Section 2), that one of the pleasing properties of branching bisimulation is that it preserves the potentials of a node, thereby preserving the non-deterministic branching structure of the system. In the next sections, we add weight to this claim: we show how we can employ *colours* to code for the potentials and prove that the observation of the colours of a node can be used to distinguish between inert transitions and non-inert transitions.

Before we commence, we provide the mathematical underpinnings and notations to facilitate mathematical reasoning about colours. Let $\mathcal{C}$ be a sufficiently large, but finite set of unique colours. A *raw blend* is a mix of colours in a particular ratio, i.e. a raw blend b is a bag of pairs $(c, \pi) \in \mathcal{C} \times (0, 1]$, with the sanity-condition $\sum_{(c,\pi)\in b} \pi = 1$. The set of all raw blends is denoted $\mathcal{B}_r$, i.e. $\mathcal{B}_r$ is a set of bags. In short, raw blends are built from fractions of colours, that together add up to 1. Raw blends are necessarily represented by *bags* rather than *sets*, since we want to consider blends in which the same quantity of a colour appears more than once (e.g. for a colour $c \in \mathcal{C}$, we want to allow the raw blend $\{(c, \frac{1}{2}), (c, \frac{1}{2})\}$). Note that we use ordinary set notation for bags, as, from the context it is always clear whether we are dealing with bags or sets.

The function $\mathsf{probe} : \mathcal{B}_r \times \mathcal{C} \to [0, 1]$, defined as $b\mathsf{probe}\ c = \sum_{(c,\pi)\in b} \pi$, yields the "weight" a colour c has in the raw blend b. To test whether a colour actually occurs in a blend, we introduce the predicate $b \lfloor c$, which holds iff $b\mathsf{probe}\ c > 0$. Thus, for a raw blend b and a colour c, the predicate $b \lfloor c$ is true iff the colour c occurs with a positive weight in blend b.

In the remainder of this paper, we use a subset of raw blends, simply called *blends*. A raw blend is a blend b iff for all colours c, $b \lfloor c$ implies $(c, b\mathsf{probe}\ c) \in b$. In other words, a colour occurs only once in a blend. Alternatively, a blend can be seen as a partial function with domain $\mathcal{C}$ and co-domain $(0, 1]$, thus representing a *distribution* of colours. Let $\mathcal{B}$ be the set of blends. We have $\mathcal{B} \subset \mathcal{B}_r$. Raw blends can be turned into blends using the operator $\circlearrowleft : \mathcal{B}_r \to \mathcal{B}$. For a raw blend b, the blend $\circlearrowleft(b)$ is given by the set $\circlearrowleft(b) \stackrel{\text{def}}{=} \{(c, b\mathsf{probe}\ c) \mid \text{for all } c \text{ satisfying } b \lfloor c\}$.

For reasons of convenience, we freely interpret a blend consisting of a single element as a colour (i.e. we write $b \in \mathcal{C}$ iff $|b| = 1$), and a colour is interpreted as a blend (i.e. we think of the colour c as the blend $\{(c, 1)\}$).

5. Concrete coloured traces

Information that can be obtained from any (reactive) system is *trace* information. By this, we mean a sequence of actions that are observed during execution of the system.

Definition 26. A *concrete trace*, starting in a node s of a graph x is a finite sequence of actions $a_1\ a_2 \ldots a_n$, $(a_i \in \mathrm{Act}_\tau)$ for which there exists a finite path c, with $\mathsf{first}(c) = s$ and $\mathsf{trace}(c) = a_1\ a_2 \ldots a_n$.

Note that both the probabilistic information and the non-deterministic branching structure are lost in such traces. Hence, it may come as no surprise that an equivalence that is based on the comparison of the sets of concrete traces of two systems is necessarily coarser (i.e. less discriminative) than strong bisimilarity.

We show that we can use *colours* and *blends* to recapture this information, and obtain a "decorated trace equivalence" (in the sense of e.g. [3,8,14]) that coincides with strong bisimilarity. The colours can be used to encode the potentials of the system in a node, while the blends can be used to encode the probabilistic information. Graphs that are endowed with a colouring of their nodes are referred to as *coloured graphs*.

Definition 27. A *coloured graph* is a tuple $\langle \mathsf{x}, \gamma \rangle$, where x is a graph and γ is a labelling function, assigning blends or colours to the nodes of x.

We next consider "decorated traces" of a coloured graph. We assume that we can observe the colours and blends of the nodes (but not the probabilistic and the non-deterministic branching structure of the graph). In other words, by

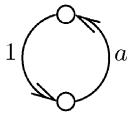

Fig. 4. Graph x.

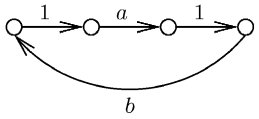

Fig. 5. Graph y.

executing the system, we can observe sequences of blends, colours and actions. We refer to such decorated traces as *concrete coloured traces*.

Definition 28. Let $\langle \mathsf{x}, \gamma \rangle$ be a coloured graph. A *concrete coloured trace*, starting in a node $s \in S_{\mathsf{nil}}$ is a sequence of one of the following forms:
(1) $\gamma(s)$ is a concrete coloured trace for $s = \mathsf{nil}$.
(2) $b'_0\ a_1\ b_1\ b'_1 \ldots a_{m+1}\ b_{m+1}$ when $s \in N$ and there is at least one path $c \equiv n_0\ a_1\ p_1 \ldots n_m a_{m+1}\ p_{m+1}$ with $\mathsf{trace}(c) = a_1 \ldots a_{m+1}$, $\mathsf{first}(c) = s = n_0$ and for all $1 \leqslant i \leqslant m+1$, $\gamma(p_i) = b_i$ and $\gamma(n_{i-1}) = b'_{i-1}$.
(3) $\gamma(s)\ u$ when $s \in P$, $s \rightsquigarrow n$ for some $n \in N$ and u is a concrete coloured trace starting in n.

In our coloured graphs, we use colours as an indication for the potentials of a node. This suggests that we should distinguish between *informative* colourings and *non-informative* colourings. We make the following observations:
(1) In the non-probabilistic case colours suffice (see e.g. [2,11]) to code for the potentials of a node.
(2) For each node $p \in P$, the cumulative probability $\mu(p, \mathcal{M})$ can be seen as a function that assigns values to each partition of the set of nodes. This roughly corresponds to the notion of a blend.
This leads us to consider a subset of coloured graphs in which non-deterministic nodes are labelled with colours and probabilistic nodes are labelled with blends that encode the probability distributions over successor nodes.

Definition 29. A *properly coloured graph* is a coloured graph $\langle \mathsf{x}, \gamma \rangle$ where γ satisfies:
(1) all nodes $n \in N_{\mathsf{nil}}$ are labelled with a colour $\gamma(n) \in \mathcal{C}$.
(2) all nodes $p \in P$ are labelled with the blend $\circlearrowleft(\{(\gamma(n), \mathsf{pr}(p, n)) \mid p \rightsquigarrow n\})$.

We say that the colouring of a coloured graph is *proper* to indicate that we are in fact dealing with a properly coloured graph.

The assumption that we can use colours to code for the potentials in a graph is not immediately vindicated. For instance, assigning the same colour to nodes from which different actions are possible conflicts with the idea that colours code for the potentials of a node. To rule out such situations, we distinguish between colourings that respect our assumption and those that violate our assumption. Colourings that respect our assumption are referred to as *consistent*. Formally, given a set of graphs, we say that the colouring of their nodes is *consistent* iff non-deterministic nodes have the same colour and probabilistic nodes have the same blend *only if* they have the same concrete coloured trace sets.

Example 30. The graph $\mathsf{x} = \langle \{n\}, \{p\}, p, \{a\}, n \xrightarrow{a} p, p \rightsquigarrow n, \mathsf{pr}(p, n) = 1 \rangle$, depicted in Fig. 4 can have many consistent colourings.

For instance, the colouring γ that assigns the colour *blue* to all nodes is consistent *and* proper. The colouring γ that assigns the colour *blue* to node n and the "blend" *yellow* to p is consistent but *not* proper. Generalising, a coloured graph that is coloured using a trivial colouring, i.e. a colouring that assigns different colours to each node, is consistently coloured (but almost never properly coloured). The graph $\mathsf{y} = \langle \{n, m\}, \{p, q\}, p, \{a, b\}, \{n \xrightarrow{a} q, m \xrightarrow{b} p\}, \{p \rightsquigarrow n, q \rightsquigarrow m\}, \{\mathsf{pr}(p, n) = \mathsf{pr}(q, m) = 1\} \rangle$, depicted in Fig. 5 has a non-proper and non-consistent colouring γ, assigning *blue* to all

nodes. The same graph also admits a proper and consistent colouring γ. For instance, take γ such that it assigns *blue* to nodes p and n, and *yellow* to nodes q and m.

Definition 31. Graphs x and y are *concrete coloured trace equivalent*, notation $\mathsf{x} \equiv_{cc} \mathsf{y}$ if for some consistent, proper colouring γ, $\langle \mathsf{x}, \gamma \rangle$ and $\langle \mathsf{y}, \gamma \rangle$ have the same concrete coloured traces, or, equivalently, their root nodes have the same colour or blend.

Concrete coloured trace equivalence is an equivalence relation on graphs. In fact, we next establish a firm relation between concrete coloured trace equivalence and strong bisimilarity. First, we show that concrete coloured trace equivalence is at least as discriminating as strong bisimilarity, i.e. graphs that are strong bisimilar are also concrete coloured trace equivalent.

Lemma 32. *For all* x *and* y, $\mathsf{x} \leftrightarrows \mathsf{y}$ *implies* $\mathsf{x} \equiv_{cc} \mathsf{y}$.

Proof. Let x and y be graphs. We denote the union of their nodes by S, the union of their non-deterministic nodes by N and the union of their probabilistic nodes by P. We denote the union of S and the special termination node nil by S_{nil}.

Assume $\mathsf{x} \leftrightarrows \mathsf{y}$. Let $\mathcal{R}$ be the largest strong bisimulation relation on S_{nil} that only relates probabilistic nodes to probabilistic nodes and non-deterministic nodes to non-deterministic nodes; nil is related to itself. Let $\Gamma : S_{\mathsf{nil}/\mathcal{R}} \to \mathcal{B}$ be a total, injective mapping with the following two characteristics:

(1) $\Gamma(\mathcal{M}) \in \mathcal{C}$ when $\mathcal{M} \subseteq N_{\mathsf{nil}}$,

(2) $\Gamma(\mathcal{M}) = \circlearrowleft(\{(\Gamma(\mathcal{M}'), \mu(\mathcal{M}, \mathcal{M}')) \mid \mu(\mathcal{M}, \mathcal{M}') \neq 0\})$ when $\mathcal{M} \subseteq P$.

This mapping is well-defined. Now, consider the coloured graphs x and y that are obtained by colouring all nodes with the colour of their equivalence classes. Formally, we define the coloured graphs $\langle \mathsf{x}, \gamma \rangle$, $\langle \mathsf{y}, \gamma \rangle$ where γ is defined as $\gamma(s) = \Gamma([s]_{\mathcal{R}})$. By definition of Γ, γ yields properly coloured graphs. By construction, the root nodes of $\langle \mathsf{x}, \gamma \rangle$ and $\langle \mathsf{y}, \gamma \rangle$ have the same colour. Hence, it suffices to show that γ is a consistent colouring. We distinguish two cases.

(1) First, we show that non-deterministic nodes that have the same colour, also have the same sets of concrete coloured traces. Let $n_0, n_1 \in N$ be two arbitrary nodes with $\gamma(n_0) = \gamma(n_1)$. Then, by definition of γ and injectivity of Γ, we know that $n_0 \mathcal{R} n_1$. Let $b'_0\ a_1\ b_1\ b'_1 \ldots a_m\ b_m$ be a concrete coloured trace starting in n_0. Since $b'_0\ a_1\ b_1$ is a concrete coloured subtrace of t, we know there is a $p_0 \in P$ with $\gamma(p_0) = b_1$ such that we have $n_0 \xrightarrow{a_1} p_0$. By strong bisimilarity, we then also have $n_1 \xrightarrow{a_1} p_1$ for some p_1 with $p_0 \mathcal{R} p_1$. Thus, $\gamma(p_0) = \gamma(p_1)$. Thus, $b'_0\ a_1\ b_1$ is also a concrete coloured subtrace that starts in n_1. Hence, it remains to show that when probabilistic nodes have the same colour, they also have the same sets of concrete coloured traces.

(2) Next, we show that two probabilistic nodes with the same blend (or colour) also have the same sets of concrete coloured traces. Let $p_0, p_1 \in P$ be arbitrary nodes with $\gamma(p_0) = \gamma(p_1)$. Then, by definition of γ and injectivity of Γ, we know that $p_0 \mathcal{R} p_1$. Let $b_0\ b'_0\ a_1\ b_1\ b'_1 \ldots a_m\ b_m$ be a concrete coloured trace starting in p_0. Since $b_0 \downharpoonright b'_0$ (which follows from the definition of Γ), we know that there is a node n_0 with $\gamma(n_0) = b'_0$, such that $\mu(p_0, [n_0]_{\mathcal{R}}) > 0$. Since $p_0 \mathcal{R} p_1$, it then follows that $\mu(p_0, [n_0]_{\mathcal{R}}) = \mu(p_1, [n_0]_{\mathcal{R}})$, and hence, $b_0\ b'_0$ is also a concrete coloured subtrace that starts in p_1. By case 1, we then also know that $b_0\ b'_0\ a_1\ b_1$ is a concrete coloured trace starting in p_1. Repeating the above arguments for m times, we find that also $b_0\ b'_0\ a_1\ b_1\ b'_1 \ldots a_m\ b_m$ is a concrete coloured trace starting in p_0.

Hence, we can conclude that the colouring γ is both proper and consistent. This means that we have $\mathsf{x} \equiv_{cc} \mathsf{y}$. □

Second, we show that strong bisimilarity is at least as discriminating as concrete coloured trace equivalence, i.e. graphs that are concrete coloured trace equivalent are also strong bisimilar.

Lemma 33. *For all graphs* x, y, $\mathsf{x} \equiv_{cc} \mathsf{y}$ implies $\mathsf{x} \leftrightarrows \mathsf{y}$.

Proof. Let x and y be graphs. We denote the union of their nodes by S, the union of their non-deterministic nodes by N and the union of their probabilistic nodes by P. We write S_{nil} for $S \cup \{\mathsf{nil}\}$.

Let γ be a consistent colouring of the nodes S_{nil}, such that the graphs $\langle \mathsf{x}, \gamma \rangle$ and $\langle \mathsf{y}, \gamma \rangle$ are properly coloured graphs. Assume that $\gamma(s_{\mathsf{x}}) = \gamma(s_{\mathsf{y}})$. Define the relation $\mathcal{R}$ on S_{nil} as $s \mathcal{R} t$ iff $\gamma(s) = \gamma(t)$ for all $(s, t) \in N^2_{\mathsf{nil}}$ and all $(s, t) \in P^2$.

By definition, we have $s_{\mathsf{x}} \mathcal{R} s_{\mathsf{y}}$. Thus, it suffices to show that $\mathcal{R}$ is a strong bisimulation relation. We proceed by showing that $\mathcal{R}$ satisfies the two conditions of Definition 2.

(1) Let $n_0 \in N$ and $n_1 \in N$ such that $n_0 \mathcal{R} n_1$. Assume that $n_0 \xrightarrow{a} p_0$ for some p_0. Then we know that $\gamma(n_0)\ a\gamma(p_0)$ is a concrete coloured trace starting in n_0. Since γ is consistent, we know that $\gamma(n_0)\ a\gamma(p_0)$ is also a concrete coloured trace starting in n_1. In turn, this means that there is a node p_1 with $\gamma(p_1) = \gamma(p_0)$, such that $n_1 \xrightarrow{a} p_1$. Hence, by definition of $\mathcal{R}$, also $p_0 \mathcal{R} p_1$.

(2) Let $p_0 \in P_{\mathsf{x}}$ and $p_1 \in P_{\mathsf{y}}$ such that $p_0 \mathcal{R} p_1$. Let $\mathcal{M} \in S_{\mathsf{nil}/\mathcal{R}}$. Suppose that $\mathcal{M} \subseteq P$. Then, immediately we obtain $\mu(p_0, \mathcal{M}) = \mu(p_1, \mathcal{M}) = 0$. Thus, without loss of generality, we assume that $\mathcal{M} \subseteq N_{\mathsf{nil}}$. The properness of γ implies $\gamma(\mathcal{M}) \in \mathcal{C}$. Because $p_0 \mathcal{R} p_1$, we also have $\gamma(p_0) = \gamma(p_1)$, from which we immediately find that $\gamma(p_0)\mathsf{probe}\ \gamma(\mathcal{M}) = \gamma(p_1)\mathsf{probe}\ \gamma(\mathcal{M})$. Since γ is proper, we have $\gamma(p_0)\mathsf{probe}\ \gamma(\mathcal{M}) = \circlearrowleft(\{(\gamma(n), \mathsf{pr}(p_0, n)) \mid p \rightsquigarrow n\})\mathsf{probe}\ \gamma(\mathcal{M}) = \mu(p_0, \mathcal{M})$. Likewise, we derive $\gamma(p_1)\mathsf{probe}\gamma(\mathcal{M}) = \mu(p_1, \mathcal{M})$. Thus, we have $\mu(p_0, \mathcal{M}) = \mu(p_1, \mathcal{M})$.

Hence, $\mathcal{R}$ is a strong bisimulation relation. □

The following theorem, stating that strong bisimilarity and concrete coloured trace equivalence are equi-discriminating, is an immediate consequence of Lemmata 32 and 33. This means that strong bisimilarity and concrete coloured trace equivalence both preserve potentials and probabilistic information. Moreover, this also proves that colours and blends can be used to code for the potentials of a system, a result that we can reuse in the setting with abstraction.

Theorem 34. *For all graphs* x *and* y, $\mathsf{x} \leftrightarrows \mathsf{y}$ *iff* $\mathsf{x} \equiv_{cc} \mathsf{y}$.

6. Coloured traces

In the previous section, we showed that colours and blends can fill in the missing information in concrete traces, allowing us to define a trace-based equivalence that coincides with strong bisimulation. A natural question is whether this feat can be repeated in a setting with abstraction. The results in this section answer this question positively.

We start by making the following observation, which is crucial for our further reasonings: abstraction obscures the strict separation between probabilistic nodes and non-deterministic nodes. This is because unobservable events allow us to move between the two without notice.

Consider again the notion of coloured graphs of Section 5 and the concrete coloured traces in such graphs. To facilitate the comparison of potentials of non-deterministic nodes and probabilistic nodes, we consider a variation on the concrete coloured traces of Section 5, which we call *pre-coloured traces*.

Definition 35. Let $\langle \mathsf{x}, \gamma \rangle$ be a coloured graph. A concrete coloured trace, starting in a node $s \in P_{\mathsf{nil}}$ is also a *pre-coloured trace* starting in s. If t is a concrete coloured trace starting in a node $s \in N$ then $\gamma(s)\ t$ is also a *pre-coloured trace* starting in s.

Note that a pre-coloured trace starting in a non-deterministic node n always starts with two occurrences of the colour (or blend) of node n. This puts us in the position to compare decorated traces starting in probabilistic nodes with those starting in non-deterministic nodes.

Pre-coloured traces still contain τ actions, which are intended to be unobservable. As we already argued in Section 2.3, we cannot bluntly remove all τ actions from such pre-coloured traces without affecting the potentials (and thereby the behaviours) of a system. Intuitively, the idea of using colours (or blends) for coding for these potentials indicates that by removing *only those* τ *actions in a pre-coloured trace that are in between nodes with the same colour* (*or blend*), we leave the potentials of the system unaffected. Pre-coloured traces from which these inert τ actions have been removed are called *coloured traces*.

Definition 36. A coloured trace starting in a node s is a finite sequence $b_0\ b_0'\ a_1 \ldots a_m\ b_m$, not ending with a subsequence $b\ \tau\ b$,[4] that is obtained from a pre-coloured trace starting in node s in which all subsequences of the

[4] Remark that the condition that a coloured trace does not end with the subsequence $b\ \tau\ b$ is required to ensure that the coloured trace does not end with a potentially inert τ step. If the τ step is not inert, then there must also be an extension of the coloured trace in which it appears.

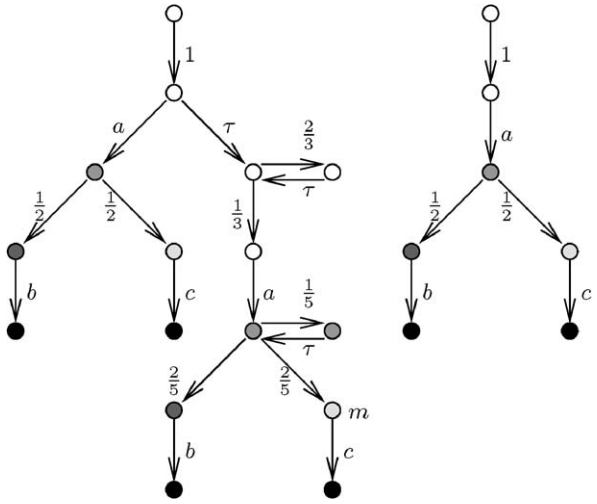

Fig. 6. Two coloured graphs.

form $b\ (b\ \tau\ b)^+$ and $(b\ \tau\ b)^+\ b$ have been replaced with b. When $m = 0$, we require that b_0 is the colour of the node nil.

Example 37. Consider the two coloured graphs of Fig. 6 (the colours and blends in this graph are indicated by distinct patterns). Note that, as we saw in Section 2.3, these two graphs are in fact branching bisimilar. The set of concrete coloured traces of the right graph is given by the following set:

$$C_1 = \{\bullet,\ \bullet\!\bullet\, b\, \bullet,\ \circ\!\circ\, c\, \bullet,\ \circ\ a\ \bullet,\ \bullet\!\bullet\, b\, \bullet,\ \bullet\!\circ\, c\, \bullet,\ \circ\ \circ\ a\ \bullet,\ \circ\ a\ \bullet\!\bullet\, b\, \bullet,$$
$$\circ\ a\ \bullet\!\circ\, c\, \bullet,\ \circ\ \circ\ a\ \bullet\!\bullet\, b\, \bullet,\ \circ\ \circ\ a\ \bullet\!\circ\, c\, \bullet\}$$

From this set, we obtain the set of pre-coloured traces:

$$C_2 = \{\bullet,\ \bullet\!\bullet\, b\, \bullet,\ \circ\!\circ\, c\, \bullet,\ \circ\ \circ\ a\ \bullet,\ \bullet\!\bullet\, b\, \bullet,\ \bullet\!\circ\, c\, \bullet,\ \circ\ \circ\ a\ \bullet\!\bullet\, b\, \bullet,\ \circ\ \circ\ a\ \bullet\!\circ\, c\, \bullet\}$$

Similarly, we can obtain (an infinite) set of pre-coloured traces for the left graph:

$$C_3 = \{\bullet,\ \bullet\!\bullet\, b\, \bullet,\ \circ\!\circ\, c\, \bullet,\ \bullet\!\bullet\, b\, \bullet,\ \bullet\!\circ\, c\, \bullet,\ \circ\ (\circ\ \tau\ \circ)^i,$$
$$\circ\ (\circ\ \tau\ \circ)^j\ \circ\ a\ \bullet\ (\bullet\ \tau\ \bullet)^k,\ \circ\ (\circ\ \tau\ \circ)^j\ \circ\ a\ \bullet\ (\bullet\ \tau\ \bullet)^k\ \bullet\, b\, \bullet,$$
$$\circ\ (\circ\ \tau\ \circ)^j\ \circ\ a\ \bullet\ (\bullet\ \tau\ \bullet)^k\ \circ\, c\ \bullet\ \mid i > 0,\ j, k \geqslant 0\}$$

Finally, we can derive the set of coloured traces from C_2 and C_3. Since there are no sequences of the form $b\ (b\ \tau\ b)^+$ or $(b\ \tau\ b)^+ b$ in the set C_2 of pre-coloured traces, the set C_2 is also a set of coloured traces. Replacing the sequences of the form $(b\ \tau\ b)^+$ by b in the set of pre-coloured traces C_3, we also obtain the set C_2 of coloured traces. Thus, the set of coloured traces that we can obtain from C_3 and C_2 is the same. Note that the coloured traces $\circ\ (\circ\ \tau\ \circ)^i$, for $i > 0$ are of the form $b\ (b\ \tau\ b)^+$. However, we cannot reduce this to $\circ$, as this colour is different from the colour of node nil. Therefore, this entire subset of pre-coloured traces does not contribute to any of the coloured traces.

As we observed earlier, the strict distinction between probabilistic nodes and non-deterministic nodes is obscured. We suggested that this might happen when we can move silently from a non-deterministic node to a probabilistic node. Now, recall the definition of a proper colouring of Section 5. It requires that non-deterministic nodes are coloured with colours and probabilistic nodes can be coloured with blends. Since branching bisimulation allows us to move between both types of nodes without notice, we can no longer assume that this strict colouring regime is sufficient for our purposes. This means that the definition of a proper colouring, given in Section 5 is too strict: the requirement that all non-deterministic nodes are labelled with real colours must be weakened to cope with unobservable transitions.

Definition 38. A *properly coloured graph* is a coloured graph $\langle \mathsf{x}, \gamma\rangle$ where γ satisfies:
(1) A node $n \in N_{\mathsf{nil}}$ is labelled with a blend $\gamma(n) \notin \mathcal{C}$ only if
  (a) $n \xrightarrow{\tau} p$ for some p.
  (b) for all $a \in \mathrm{Act}$ and $p \in P_{\mathsf{nil}}$, $n \xrightarrow{a} p$ implies $a = \tau$ and $\gamma(n) = \gamma(p)$.
(2) All nodes $p \in P$ are labelled with the blend $\circlearrowleft(\{(c, \mathsf{pr}(p, n) \cdot (\gamma(n)\mathsf{probe}\ c)) \mid p \rightsquigarrow n \text{ and } \gamma(n)\lfloor c\})$.

We say that the colouring of a coloured graph is *proper* to indicate that we are dealing with a properly coloured graph. Next, we overload the notion of consistency as defined in Section 5 as follows. For a set of coloured graphs we say that the colouring that is used to colour the nodes of the graphs is *consistent* whenever two nodes have the same colour (or blend) *only when* they have the same coloured trace sets.

Definition 39. Graphs x and y are *coloured trace equivalent*, notation $\mathsf{x} \equiv_c \mathsf{y}$ iff for some consistent, proper colouring γ, $\langle \mathsf{x}, \gamma\rangle$ and $\langle \mathsf{y}, \gamma\rangle$ have the same coloured traces, or, equivalently, their root nodes have the same blend or colour.

Example 40. The graphs in Fig. 6 are coloured trace equivalent. In example 37, we showed that the set of coloured traces match. Moreover, it is easy to see that the graph is consistent and properly coloured if we assume that $\bullet = \circlearrowleft(\{(\circ, \frac{1}{2}), (\bullet, \frac{1}{2})\})$ and $\circ, \circ, \bullet \in \mathcal{C}$.

Coloured trace equivalence is an equivalence relation on graphs. The following theorem states that two graphs are branching bisimilar if and only if they are coloured trace equivalent. First, we prove several propositions and two lemmata that together form the basis for this theorem.

For the remainder of this section, we consider two arbitrary graphs x and y. We denote the union of their nodes by S, the union of their non-deterministic nodes by N and the union of their probabilistic nodes by P. We write S_{nil} for $S \cup \{\mathsf{nil}\}$. When we assume $\mathsf{x} \leftrightarrows_b \mathsf{y}$, we take $\mathcal{R}$ to be the largest branching bisimulation relation relating (the nodes of) x and y. Let $\Gamma : S_{\mathsf{nil}/\mathcal{R}} \to \mathcal{B}$ be an injective, total function satisfying:
(1) $\Gamma(\mathcal{M}) \in \mathcal{C}$ when for all classes $\mathcal{M}' \neq \mathcal{M}$, we have $\mu_{\mathcal{M}}(\mathcal{M}, \mathcal{M}') = 0$.
(2) $\Gamma(\mathcal{M}) = \circlearrowleft(\{(c, w \cdot (\Gamma(\mathcal{M}')\mathsf{probe}\ c)) \mid \text{for all } \mathcal{M}' \neq \mathcal{M} \text{ with } \Gamma(\mathcal{M}')\lfloor c \text{ and } w = \mu_{\mathcal{M}}(\mathcal{M}, \mathcal{M}') > 0\}$.
We refer to Γ as an *equivalence class colour-coding* for the branching bisimilar graphs x and y.

Proposition 41. *The equivalence class colour-coding function Γ is well-defined.*

Proof. Showing that each equivalence class colour-coding function Γ is well-defined requires showing that its recursive definition has a unique solution. We make the following observations.
(1) There is at least one equivalence class to which we can assign a colour. [5]
(2) Each equivalence class to which a blend is assigned depends on a finite number of other classes that are either assigned blends or colours. Given that there are only finitely many classes, we can represent Γ by a dependency matrix. This matrix can be interpreted as a Markov Chain with absorbing states [15] (which correspond to the colours that have been assigned). The absorption probabilities for ending up in a particular absorbing state then correspond to the weight a particular colour has in the blend.
We formalise the above observations in some detail. Let $\mathcal{N}$ be an equivalence class for which we want to assign a blend. We construct a Markov Chain $\mathsf{MC}_{\mathcal{N}}$ for the equivalence class $\mathcal{N}$ as follows:
(1) For each equivalence class $\mathcal{M} \in S_{\mathsf{nil}/\mathcal{R}}$, there is a state $s_{\mathcal{M}}$ in the Markov Chain $\mathsf{MC}_{\mathcal{N}}$. The state $s_{\mathcal{N}}$ is the *initial state*.
(2) A state $s_{\mathcal{M}}$ is a *absorbing state* when its corresponding class $\mathcal{M}$ satisfies $\mu_{\mathcal{M}}(\mathcal{M}, \mathcal{M}') = 0$ for all classes $\mathcal{M}' \neq \mathcal{M}$.
(3) A state $s_{\mathcal{M}}$ is a *transient state* when its corresponding class $\mathcal{M}$ satisfies $\mu_{\mathcal{M}}(\mathcal{M}, \mathcal{M}') > 0$ for some classes $\mathcal{M}' \neq \mathcal{M}$.

---

[5] The reason for this is as follows: if there is no node $n \in N$, with $n \xrightarrow{a}$ for some $a \in \mathrm{Act}$, then the entire graph is branching bisimilar to nil. Yet, if there is an $n \in N$ with $n \xrightarrow{a} p$ (for some $p \in P_{\mathsf{nil}}$ and $a \in \mathrm{Act}$), then we have $\mu_{[n]_{\mathcal{R}}}([n]_{\mathcal{R}}, [p]_{\mathcal{R}}) = 0$ (when $[n]_{\mathcal{R}} \neq [p]_{\mathcal{R}}$) or $\mu_{[n]_{\mathcal{R}}}([n]_{\mathcal{R}}, [p]_{\mathcal{R}}) = 1$ (when $[n]_{\mathcal{R}} = [p]_{\mathcal{R}}$). In both cases, the equivalence class colour-coding will assign a colour to the class $[n]_{\mathcal{R}}$.

(4) For each transient state $s_{\mathcal{M}}$, we have a transition $s_{\mathcal{M}} \to s_{\mathcal{M}'}$ iff there is a probabilistic node $p \in \mathcal{M}$, such that $p \rightsquigarrow n$ for some node $n \in \mathcal{M}'$. The probability assigned to this transition is $\mu_{\mathcal{M}}(\mathcal{M}, \mathcal{M}')$.
(5) Every state not reachable from $s_{\mathcal{N}}$ is removed, together with their outgoing transitions.

Note that when $s_{\mathcal{M}}$ is a transient state in $\mathsf{MC}_{\mathcal{N}}$, then $\mathsf{MC}_{\mathcal{M}}$ is a "sub-chain" of $\mathsf{MC}_{\mathcal{N}}$. Moreover, if in $\mathsf{MC}_{\mathcal{N}}$, the state $s_{\mathcal{N}}$ is reachable from the state $s_{\mathcal{M}}$ then the Markov Chains $\mathsf{MC}_{\mathcal{N}}$ and $\mathsf{MC}_{\mathcal{M}}$ are the same.

The Markov Chain $\mathsf{MC}_{\mathcal{N}}$ is a finite Markov Chain with absorbing states. Finding the absorption probabilities for such a Markov Chain boils down to solving a system of linear equations. As already observed, these absorption probabilities are exactly the weights a colour has in a blend. In [15] it is shown that these absorption probabilities can always be computed for finite Markov Chains. All absorption probabilities together make up the blends. □

Define the colouring $\gamma : S_{\mathsf{nil}} \to \mathcal{B}$ as $\gamma(s) = \Gamma(\mathcal{M})$ iff $s \in \mathcal{M}$. Henceforth, we refer to this colouring γ as an *equivalence class colouring*.

Proposition 42. *The equivalence class colouring γ induces properly coloured graphs $\langle \mathsf{x}, \gamma \rangle$ and $\langle \mathsf{y}, \gamma \rangle$.*

Proof. To show that γ is a proper colouring, we proceed as follows.

(1) Let $n \in N_{\mathsf{nil}}$ be a non-deterministic node or nil, and suppose $\gamma(n) \notin \mathcal{C}$.
   (a) By definition of γ, we find that there is a class $\mathcal{M}' \neq [n]_{\mathcal{R}}$, for which $\mu_{[n]_{\mathcal{R}}}([n]_{\mathcal{R}}, \mathcal{M}') > 0$. Since $n \in N_{\mathsf{nil}}$, this can only be the case when $n \xrightarrow{\tau} p$ for some $p \in [n]_{\mathcal{R}}$.
   (b) Let $a \in \mathrm{Act}_{\tau}$ and $p \in P_{\mathsf{nil}}$, and assume $n \xrightarrow{a} p$. Let $\mathcal{M} \neq [n]_{\mathcal{R}}$ be a class for which $\mu_{[n]_{\mathcal{R}}}([n]_{\mathcal{R}}, \mathcal{M}) > 0$. By definition of γ and Γ, at least two such classes exist, hence also $\mu_{[n]_{\mathcal{R}}}([n]_{\mathcal{R}}, \mathcal{M}) < 1$. This means that there must be a node $p' \in [n]_{\mathcal{R}}$, such that $1 > \mathsf{pr}(p', \mathcal{M}) > 0$. Since $\mathcal{R}$ is a branching bisimulation relation, this means that $n \xrightarrow{a} p$ can be mimicked by p' with probability 1. But this is only possible by a scheduler σ that schedules $\sigma(p') = \bot$, and when $a = \tau$ and $p \in [n]_{\mathcal{R}}$, i.e. $\gamma(n) = \gamma(p)$.
(2) Let $p \in P$, and let $(c, \pi) \in \gamma(p)$ be a part of the blend or colour of p. This means that $\pi > 0$. By definition of the equivalence class colour-coding Γ and the operator $\circlearrowleft$, we find the following relation between c, π and p:

$$\pi = \sum_{\mathcal{M} \neq [p]_{\mathcal{R}}} \mu_{[p]_{\mathcal{R}}}([p]_{\mathcal{R}}, \mathcal{M}) \cdot (\Gamma(\mathcal{M})\mathsf{probe}\ c). \tag{14}$$

The formula $\mu_{[p]_{\mathcal{R}}}([p]_{\mathcal{R}}, \mathcal{M})$ represents the maximal probability of reaching $\mathcal{M}$ via a node in $[p]_{\mathcal{R}}$ using silent transitions only (see also Section 3, where we established a correspondence between the normalised cumulative probability and the schedulers inducing maximal probabilities). It can be defined in terms of the maximal probability of reaching $\mathcal{M}$ from all nodes n that can be reached from p in one step. We use this observation to rewrite Eq. (14) to:

$$\pi = \sum_{\mathcal{M} \neq [p]_{\mathcal{R}}} \left(\left(\sum_{p \rightsquigarrow n, [n]_{\mathcal{R}} = [p]_{\mathcal{R}}} \mathsf{pr}(p, n) \cdot \mu_{[n]_{\mathcal{R}}}([n]_{\mathcal{R}}, \mathcal{M}) \cdot (\Gamma(\mathcal{M})\mathsf{probe}\ c) \right) + \left(\sum_{p \rightsquigarrow n, [n]_{\mathcal{R}} = \mathcal{M}} \mathsf{pr}(p, n) \cdot (\Gamma(\mathcal{M})\mathsf{probe}\ c) \right) \right). \tag{15}$$

Reordering the quantification over $\mathcal{M}$, and substituting $\mathcal{M}$ for $[n]_{\mathcal{R}}$ in the second part of Eq. (15), we find the following equivalence:

$$\pi = \sum_{p \rightsquigarrow n, [n]_{\mathcal{R}} = [p]_{\mathcal{R}}} \mathsf{pr}(p, n) \cdot \sum_{\mathcal{M} \neq [n]_{\mathcal{R}}} \mu_{[n]_{\mathcal{R}}}([n]_{\mathcal{R}}, \mathcal{M}) \cdot (\Gamma(\mathcal{M})\mathsf{probe}\ c) + \sum_{p \rightsquigarrow n, [n]_{\mathcal{R}} \neq [p]_{\mathcal{R}}} \mathsf{pr}(p, n) \cdot (\Gamma([n]_{\mathcal{R}})\mathsf{probe}\ c).$$

From this, we immediately find that

$$\pi = \sum_{p \rightsquigarrow n, [n]_{\mathcal{R}} = [p]_{\mathcal{R}}} \mathsf{pr}(p, n) \cdot (\gamma(n)\mathsf{probe}\ c) + \sum_{p \rightsquigarrow n, [n]_{\mathcal{R}} \neq [p]_{\mathcal{R}}} \mathsf{pr}(p, n) \cdot (\gamma(n)\mathsf{probe}\ c) \tag{16}$$

Simplifying Eq. (16) we find $\pi = \sum_{p \rightsquigarrow n} \mathsf{pr}(p,n) \cdot (\gamma(n)\mathsf{probe}\ c)$, which means that the probabilistic node p is coloured properly. □

Proposition 43. *All equivalence class colourings γ for branching bisimilar graphs* x *and* y *are consistent.*

Proof. We must show that for all nodes $s, t \in S_{\mathsf{nil}}$ satisfying $\gamma(s) = \gamma(t)$ the sets of coloured traces of s and t are the same.

So, let $s, t \in S_{\mathsf{nil}}$ be arbitrary nodes. Suppose that $\gamma(s) = \gamma(t)$. Let $\eta \equiv b_0\, b'_0\, a_1 \ldots a_m\, b_m$ be a coloured trace starting in s. This coloured trace must come from a pre-coloured trace

$$\zeta \equiv b_0\ (b_0\ \tau\ b_0)^{k_0}\ (b'_0\ \tau\ b'_0)^{l_0}\ b'_0\ a_1 \cdots a_m\ b_m\ (b_m\ \tau\ b_m)^{l_m}.$$

At this point, we must distinguish between the case when $s \in P$ and $s \notin P$. We only investigate the former. The latter case can be treated similarly. Assume $s \in P$. Then the pre-coloured trace ζ must come from a path c for which we have

$$c\ \mathsf{sat}\ s_0 \Longrightarrow_{[s_0]_{\mathcal{R}}} s'_0 \rightsquigarrow s''_0 \Longrightarrow_{[s''_0]} s'''_0 \xrightarrow{a_1} s_1 \cdots s'''_{m-1} \xrightarrow{a_m} s_m$$

for some s_i, s'_i, s''_i and s'''_i satisfying $\gamma(s_i) = \gamma(s'_i) = b_i$ and $\gamma(s''_i) = \gamma(s'''_i) = b'_i$ for all $i \leqslant m$. Since $\gamma(s) = \gamma(t)$, we find that $s\mathcal{R}t$. By repeatedly applying the definition of branching bisimulation, we also find that there must be a path c' for which we have either:

$$c'\ \mathsf{sat}\ t_0 \Longrightarrow_{[t_0]_{\mathcal{R}}} t'_0 \rightsquigarrow t''_0 \Longrightarrow_{[t''_0]} t'''_0 \xrightarrow{a_1} t_1 \ldots t'''_{m-1} \xrightarrow{a_m} t_m$$

or (only when $b_0 = b'_0$):

$$c'\ \mathsf{sat}\ t_0 \Longrightarrow_{[t_0]_{\mathcal{R}}} t'''_0 \xrightarrow{a_1} t_1 \Longrightarrow_{[t_1]_{\mathcal{R}}} t'_1 \rightsquigarrow t''_1 \ldots t'''_{m-1} \xrightarrow{a_m} t_m$$

for some t_i, t'_i, t''_i and t'''_i satisfying $s_i\mathcal{R}t_i\mathcal{R}t'_i$ and $s'_i\mathcal{R}t'_i\mathcal{R}t'''_i$. This means that also ζ is a pre-coloured trace starting in t, and thus η is a coloured trace starting in t. □

Lemma 44. *For all graphs* x *and* y, $\mathsf{x} \leftrightarrows_b \mathsf{y}$ *implies* $\mathsf{x} \equiv_c \mathsf{y}$.

Proof. Let x and y be graphs. Assume that $\mathsf{x} \leftrightarrows_b \mathsf{y}$. Then, using propositions 41, 42 and 43, we find that there is a proper and consistent colouring γ for the graphs x and y, such that $\langle \mathsf{x}, \gamma\rangle$ and $\langle \mathsf{y}, \gamma\rangle$ have the same sets of coloured traces. □

Let x be a graph. Let $\mathcal{M} \subseteq \mathcal{M}' \subseteq S_{\mathsf{nil}}$ be two non-empty sets of nodes. We define the distance function $|s|^{\mathcal{M}'}_{\mathcal{M}}$, which yields the minimal number of steps (probabilistic transitions and non-deterministic τ-transitions) that is required to reach a node in $\mathcal{M}$ via nodes in $\mathcal{M}'$, starting in $s \in \mathcal{M}'$.

$$|s|^{\mathcal{M}'}_{\mathcal{M}} = \begin{cases} 0 & \text{if } s \in \mathcal{M}, \\ \infty & \text{if } s \notin \mathcal{M}', \\ 1 + \min\limits_{s \rightsquigarrow n} |n|^{\mathcal{M}'}_{\mathcal{M}} & \text{if } s \in (P \cap \mathcal{M}') \setminus \mathcal{M}, \\ 1 + \min\limits_{s \xrightarrow{\tau} p} |p|^{\mathcal{M}'}_{\mathcal{M}} & \text{if } s \in (N \cap \mathcal{M}') \setminus \mathcal{M}. \end{cases} \tag{17}$$

Note that we take as a convention that when there is no $s \rightsquigarrow n$ (and, analogously, when there is no $s \xrightarrow{\tau} p$), then $\min_{s \rightsquigarrow n} |n|^{\mathcal{M}'}_{\mathcal{M}}$ yields ∞.

Lemma 45. *For all graphs* x *and* y, $\mathsf{x} \equiv_c \mathsf{y}$ *implies* $\mathsf{x} \leftrightarrows_b \mathsf{y}$.

Proof. Assume $\mathsf{x} \equiv_c \mathsf{y}$. Then, there is a consistent and proper colouring γ of x and y, such that x and y have the same set of coloured traces. We show that there is a branching bisimulation relation $\mathcal{R}$, such that $s_{\mathsf{x}}\mathcal{R}s_{\mathsf{y}}$. Define $\mathcal{R}$ as $s\mathcal{R}t$ iff $\gamma(s) = \gamma(t)$. By definition of coloured trace equivalence, we have $s_{\mathsf{x}}\mathcal{R}s_{\mathsf{y}}$. It thus suffices to show that $\mathcal{R}$ satisfies the requirements for a branching bisimulation relation.

(1) Let $s \in N$ be a non-deterministic node and assume that for some $t \in S_{\mathsf{nil}}$, we have $s\mathcal{R}t$. Suppose $s \xrightarrow{a} s'$ for some a and s'. We distinguish two cases.
   (a) Suppose $a = \tau$ and $s' \in [s]_{\mathcal{R}}$. It suffices to show that there is a scheduler σ, such that $\mathcal{P}(\mathcal{B}_\sigma(t \Longrightarrow_{[t]_{\mathcal{R}}} [t]_{\mathcal{R}})) = 1$. This is readily achieved by the scheduler $\sigma(t) = \bot$.
   (b) Suppose $a \neq \tau$ or $s' \notin [s]_{\mathcal{R}}$. It suffices to show that there is a scheduler σ, such that $\mathcal{P}(\mathcal{B}_\sigma(t \xrightarrow{a}_{[t]_{\mathcal{R}}} [s']_{\mathcal{R}})) = 1$. Let $\mathcal{M}$ be the set of nodes $\{t'' \mid t''\mathcal{R}t, t'' \xrightarrow{a} t', t'\mathcal{R}s'\}$. Obviously, $\mathcal{M} \subseteq [s]_{\mathcal{R}}$. Next, let $\sigma \in \mathsf{Sched}(t)$ be a scheduler that satisfies the following conditions.

$$\sigma(c) = \begin{cases} \mathsf{last}(c) \xrightarrow{a} t' & \text{if } \gamma(\mathsf{last}(c)) = \gamma(t) \text{ and } \gamma(t') = \gamma(s'), \\ \mathsf{last}(c) \xrightarrow{\tau} u & \text{if } \gamma(\mathsf{last}(c)) = \gamma(t) \text{ and } \gamma(u) = \gamma(t) \text{ and} \\ & \text{for all } t'' \text{ with } \mathsf{last}(c) \xrightarrow{\tau} t'' \text{ we require} |t''|_{\mathcal{M}}^{[s]_{\mathcal{R}}} \geqslant |u|_{\mathcal{M}}^{[s]_{\mathcal{R}}} \text{ and } \gamma(t'') \neq \gamma(s'), \\ \bot & \text{if } \gamma(\mathsf{last}(c)) \neq \gamma(t). \end{cases}$$

   Since γ is proper, we find that $\gamma(s) \in \mathcal{C}$ (this follows immediately from $a \neq \tau$ or $s' \notin [s]_{\mathcal{R}}$). Using the consistency of γ, we know that there is a pre-coloured trace $\gamma(s)\ (\gamma(s)\ \tau\ \gamma(s))^k a\ \gamma(s')$ starting in t, and, hence, there is a path from t through $[t]_{\mathcal{R}}$ to a node in $\mathcal{M}$. Therefore, $\mathcal{P}(\mathcal{B}_\sigma(t \xRightarrow{a}_{[t]_{\mathcal{R}}} [s'])) > 0$. It suffices to prove that also $\mathcal{P}(\mathcal{B}_\sigma(t \xRightarrow{a}_{[t]_{\mathcal{R}}} [s']_{\mathcal{R}})) = 1$. But this follows immediately from the fact that all probabilistic nodes p in paths $c \in \mathcal{B}_\sigma(t \xRightarrow{a}_{[t]_{\mathcal{R}}} [s']_{\mathcal{R}})$ are coloured with $\gamma(s) \in \mathcal{C}$. The properness of γ ensures that we then stay in the class $[s]_{\mathcal{R}}$ $(= [t]_{\mathcal{R}})$ with probability 1 before executing a and entering a class with colour $\gamma(s')$.

(2) Let $s \in P$ be a probabilistic node and assume that for some $t \in S_{\mathsf{nil}}$, we have $s\mathcal{R}t$. We distinguish three cases.
   (a) Suppose $\mu_{[s]_{\mathcal{R}}}(s, \mathcal{M}) = 0$ for all $\mathcal{M} \neq [s]_{\mathcal{R}}$. It suffices to show that there is a scheduler $\sigma \in \mathsf{Sched}(t)$, such that $\mathcal{P}(\mathcal{B}_\sigma(t \Longrightarrow_{[t]_{\mathcal{R}}} \mathcal{M})) = \mu_{[s]_{\mathcal{R}}}(s, \mathcal{M})$ for all $\mathcal{M} \neq [s]_{\mathcal{R}}$. This is readily achieved by the scheduler $\sigma(t) = \bot$.
   (b) Suppose there is a unique class $\mathcal{M} \neq [s]_{\mathcal{R}}$, such that $\mu_{[s]_{\mathcal{R}}}(s, \mathcal{M}) = 1$. Since γ is a proper colouring, this implies that $\gamma(s) = \gamma(s')$ for all nodes $s' \in \mathcal{M}$. But this cannot be since this implies that $s'\mathcal{R}s$, which contradicts $\mathcal{M} \neq [s]_{\mathcal{R}}$. Hence, there cannot be a class $\mathcal{M} \neq [s]_{\mathcal{R}}$, for which $\mu_{[s]_{\mathcal{R}}}(s, \mathcal{M}) = 1$.
   (c) Suppose there is a class $\mathcal{M} \neq [s]_{\mathcal{R}}$, such that $1 > \mu_{[s]_{\mathcal{R}}}(s, \mathcal{M}) > 0$. Let $\mathcal{N} \subseteq [t]_{\mathcal{R}}$ be the set of nodes for which there is a probabilistic transition leaving class $[t]_{\mathcal{R}}$ in one step, i.e. $\mathcal{N} = \{t'' \mid t'' \rightsquigarrow t', t''\mathcal{R}t, t' \notin [t]_{\mathcal{R}}\}$. Let $|u|_{\mathcal{N}}^{[s]_{\mathcal{R}}}$ again denote the minimal distance from node u to a node in $\mathcal{N}$. Let $\sigma \in \mathsf{Sched}(t)$ be a scheduler that satisfies the following conditions:

$$\sigma(c) = \begin{cases} \mathsf{last}(c) \xrightarrow{\tau} u & \text{if } \gamma(\mathsf{last}(c)) = \gamma(t) \text{ and } \gamma(u) = \gamma(t) \text{ and} \\ & \text{for all } t'' \text{ such that } \mathsf{last}(c) \xrightarrow{\tau} t'', \text{ we require } |t''|_{\mathcal{N}}^{[t]_{\mathcal{R}}} \geqslant |u|_{\mathcal{N}}^{[t]_{\mathcal{R}}}, \\ \bot & \text{if } \gamma(\mathsf{last}(c)) \neq \gamma(t). \end{cases}$$

   Note that there is in general not a single scheduler that is determined by the above conditions, as there may at some point be more than one node that has a minimal distance from a node in $\mathcal{N}$. However each σ is such that $\mu_{[s]_{\mathcal{R}}}(s, \mathcal{M}) = \mathcal{P}(\mathcal{B}_\sigma(t \Longrightarrow_{[t]_{\mathcal{R}}} \mathcal{M}))$ for all $\mathcal{M} \neq [s]_{\mathcal{R}}$, due to the properness of the colouring γ.

Hence, relation $\mathcal{R}$ satisfies the condition for branching bisimulation, and we have $s_{\mathsf{x}}\mathcal{R}s_{\mathsf{y}}$. Thus, we find $\mathsf{x} \leftrightarrows_b \mathsf{y}$. □

Theorem 46. *For all* x *and* y, $\mathsf{x} \leftrightarrows_b \mathsf{y}$ *iff* $\mathsf{x} \equiv_c \mathsf{y}$.

Proof. Follows immediately from Lemmas 44 and 45. □

7. Related work

The literature reports on two approaches for modelling reactive probabilistic system. The first approach is the model of *probabilistic* (*simple*) *automata* (often called the *non-alternating model*), which was introduced in [20,19]. The second approach, based on the Concurrent Markov Chains of [22], is that of the *alternating model*, which was introduced in [13] by Hansson. The theory outlined in this paper is based on this latter model.

One might argue that the differences between both models are fairly insignificant, and, up to a certain point, this is true: as shown in [4], the two models do not differ up to strong bisimulation. However, when we consider equivalence relations that are sensitive to internal activities, this picture suddenly changes. For instance, in [4], Segala and Bandini show that weak bisimilarity for the alternating model (defined in [18]) and weak bisimilarity for the non-alternating model (as defined in [19,20]) are incomparable. We briefly review the relevant literature and place our contribution and motivation in perspective.

7.1. Alternating model vs. non-alternating model

Comparing our notion of branching bisimulation with the notion of branching bisimulation in the non-alternating setting, as defined by Segala and Lynch [19,20] we find that their notion is more restrictive. This is illustrated by the following example. Consider the two graphs of Fig. 3 (see Section 2.3, p. 9). In contrast with our notion of branching bisimulation, we find that these two graphs are not related by branching bisimulation in the non-alternating model. The reason is obvious: k appears as a node in the "non-alternating" counterpart of the left graph and it cannot be related to any node in the "non-alternating" counterpart of the right graph. A variation of branching bisimulation called *delay branching bisimulation*, which is defined by Stoelinga [21], exhibits the same phenomenon.

In this paper, we showed that our definition of branching bisimilarity satisfies the properties originally attributed to it (by following the approach as laid out by van Glabbeek and Weijland [11] in the non-probabilistic case, see Sections 5 and 6). We therefore believe that the definition of branching bisimulation in the non-alternating setting may be incomplete and further research is required to solve this issue.

Note that the so-named *combined* version of branching bisimulation in [20] relates processes that are not related by our branching bisimulation (but still not the ones from Fig. 3). This means that our branching bisimulation and the combined version of branching bisimulation are incomparable. Further investigations along the lines of [4] are needed to fully explore all differences. This, however, is beyond the scope of this paper.

7.2. Branching bisimilarity vs. weak bisimilarity

When we compare our definition of branching bisimilarity with weak bisimilarity as defined by Philippou et al. [18], we find that branching bisimilarity is strictly finer (although there is a big overlap in systems that are both, and for some classes of systems such as *fully probabilistic systems* [6], it is known that both equivalences coincide). This is due to the fact that branching bisimilarity preserves the (non-deterministic) branching structure of a system, whereas weak bisimilarity does not, which is also the case in the non-probabilistic setting. Note that in [10] a logic in the PCTL* style is defined and the soundness and completeness properties of the equivalence relation induced by the logic are proven with respect to the weak bisimulation of [18]. Having in mind the results in the non-probabilistic setting saying that CTL* without the next operator corresponds to branching bisimulation (e.g. [17]), the result in [10] may suggest that weak and branching bisimulation for the alternating model (for systems with both probabilistic and non-deterministic behaviour) *do* coincide. However, this is not the case: the soundness and completeness results in [10] are due to the "non-standard" semantics given to the PCTL*-like operators. Namely, the path formulas are not interpreted on paths but on *behaviours*—informally, a behaviour is the observable part of one path. Clearly, with this interpretation the logic cannot make it possible to see the change of the potentials, which is the essential point that distinguishes weak and branching bisimulation.

Below, we give two examples to illustrate the differences between weak and branching bisimulation. The first example shows that two non-probabilistic systems, encoded as graphs are weak bisimilar but not branching bisimilar (other examples of this can also be found in van Glabbeek and Weijland [11]).

Example 47. Consider the two graphs of Fig. 7. These graphs encode two non-probabilistic systems (i.e. only the trivial probability 1 appears). Using the definition of weak bisimulation [18], one can easily check that both graphs are weak bisimilar. However, the graphs are not branching bisimilar, or, equivalently, there is no proper consistent colouring of the two graphs such that both have the same set of coloured traces.

The next example shows that weak bisimilarity and branching bisimilarity do not only differ for non-probabilistic systems, but that they also differ for real probabilistic systems.

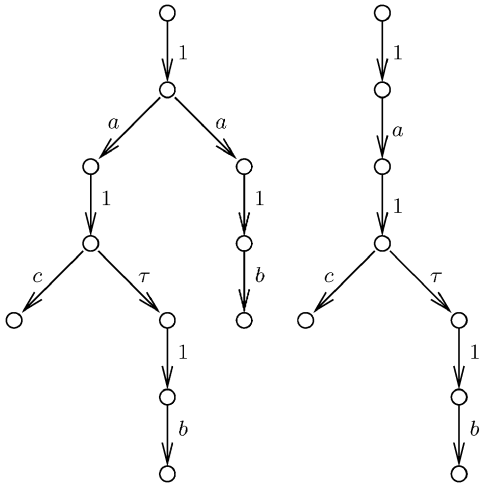

Fig. 7. Two weak bisimilar graphs (representing non-probabilistic systems) that are not branching bisimilar.

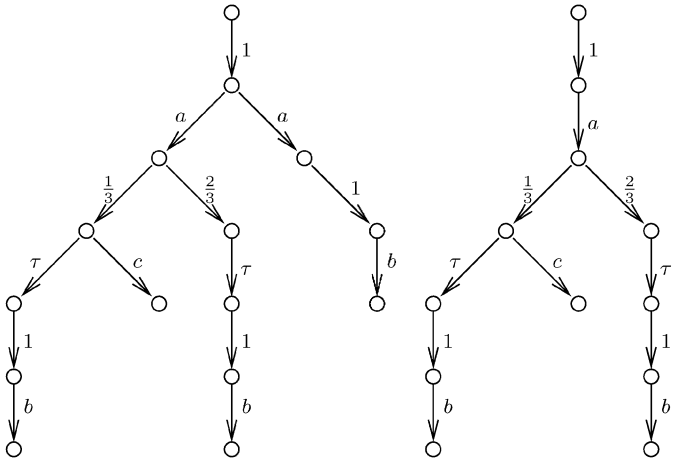

Fig. 8. Two weak bisimilar graphs (representing probabilistic systems) that are not branching bisimilar.

Example 48. Consider the two graphs of Fig. 8. Using the definition of weak bisimulation [18], it easily follows that the two graphs are weak bisimilar. Based on our definition of branching bisimulation, we find that the two graphs are not equivalent. This is because in the right graph, after executing action a it is always possible to execute action c, unlike in the right branch of the left graph.

7.3. Decidability

Finally, we find that no extensive study on the decidability and complexity of branching bisimulation has been conducted. To this date, no algorithm for deciding branching bisimilarity (in the non-alternating model) has been defined, whereas our notion can be decided in polynomial time (see Section 3.3). Deciding weak bisimilarity in the alternating setting can be achieved in polynomial time [18], whereas the best known algorithm for deciding weak bisimilarity in the non-alternating model as defined in [20] is exponential [9]. Only a finer variant of weak bisimulation (for the non-alternating model), called weak delay bisimulation [21,7] is decidable in polynomial time [7].

8. Summary

We defined the notion of branching bisimulation for the strictly alternating model of Hansson [13] for probabilistic systems. We showed that it preserves the non-deterministic branching structure of a system by defining an alternative equivalence, called *coloured trace equivalence*, that clearly satisfies this property, and we subsequently showed that the two equivalences coincide. Furthermore, we showed that the branching bisimulation conditions can be rephrased to conditions that use schedulers which induce maximal probabilities.

The alternative characterisations (in terms of colours and in terms of maximal probabilities) each have their own merits. Coloured trace equivalence is easily understood without knowledge of probability theory, schedulers, etc. It moreover clearly illustrates the fundamental property of branching bisimulation: the preservation of *potentials* and *computations* (see Section 6). The result that indicates that it suffices to use schedulers which induce maximal probabilities is at the basis for the decision procedure with polynomial time complexity that we give in Section 3.3.

We find that the two alternative characterisations add to the understanding of branching bisimulation in the alternating model, and to the correctness of the definition. Moreover, we find that it also can be used to validate the existing notion of branching bisimulation in another setting, i.e. the non-alternating model. A brief comparison of both notions already indicates that there are fundamental differences between the two (see Section 7). These differences provide compelling evidence that the notion of branching bisimulation in the non-alternating model may not live up to its name in its current phrasing: we find that processes that are intuitively branching bisimilar (and can be proven to be branching bisimilar in our setting) cannot be related in the non-alternating setting. However, more research (possibly along the lines of [4]) is required to compare the two notions in more detail. This is beyond the scope of this paper.

We pose two open problems. The first open problem is whether coloured trace equivalence gives rise to a different type of algorithm for deciding branching bisimilarity than the ones that are based on schedulers. The second open problem is to find an answer to whether the branching bisimulation relation of [20] admits a characterisation in terms of an equivalence based on colours. Apart from these problems, we are in the process of giving a complete and sound axiomatisation of branching bisimulation for the basic operators.

Acknowledgements

Thanks are due to Jos Baeten, Christel Baier, Holger Hermanns, Joost-Pieter Katoen, Ana Sokolova and Frits Vaandrager for fruitful discussions and useful comments on the topics addressed in this paper.

References

[1] S. Andova, J. Baeten, Abstraction in probabilistic process algebra, in: Proc. Tools and Algorithms for the Construction and Analysis of Systems, Seventh Internat. Conf., TACAS 2001, Lecture Notes in Computer Science, Vol. 2031, Springer, Berlin, 2001, pp. 204–219.
[2] J. Baeten, W. Weijland, Process Algebra, Cambridge University Press, Cambridge, MA, 1990.
[3] J. Baeten, J. Bergstra, J. Klop, Ready-trace semantics for concrete process algebra with the priority operator, Comput. J. 30 (6) (1987) 498–506.
[4] E. Bandini, R. Segala, Axiomatizations for probabilistic bisimulation, in: F. Orejas, P. Spirakis, J. van Leeuwen (Eds.), Proc. ICALP'01, Lecture Notes in Computer Science, Vol. 2076, Springer, Berlin, 2001, pp. 370–381.
[5] C. Baier, On algorithmic verification methods for probabilistic systems, Habilitation thesis, University of Mannheim, 1998.
[6] C. Baier, H. Hermanns, Weak bisimulation for fully probabilistic processes, in: O. Grumberg (Ed.), Proc. CAV'97, Lecture Notes in Computer Science, Vol. 1254, Springer, Berlin, 1997, pp. 119–130.
[7] C. Baier, M. Stoelinga, Norm function for probabilistic bisimulations with delays, in: J. Tiuryn (Ed.), Proc. FOSSACS'00, Lecture Notes in Computer Science, Vol. 1784, Springer, Berlin, 2000, pp. 1–16.
[8] B. Bloom, S. Istrail, A. Meyer, Bisimulation can't be traced, J. Assoc. Comput. Math. 42 (1) (1995) 232–268.
[9] S. Cattani, R. Segala, Decision algorithms for probabilistic bisimulation, in: L. Brim, P. Janar, M. Katínský, A. Kuera (Eds.), Proc. CONCUR'02, Lecture Notes in Computer Science, Vol. 2421, Springer, Berlin, 2002, pp. 371–385.
[10] J. Desharnais, V. Gupta, R. Jagadeesan, P. Panangaden, Weak bisimulation is sound and complete for PCTL*, in: L. Brim, P. Janar, M. Katínský, A. Kuera (Eds.), Proc. CONCUR'02, Lecture Notes in Computer Science, Vol. 2421, Springer, Berlin, 2002, pp. 355–370.
[11] R. van Glabbeek, W. Weijland, Branching time and abstraction in bisimulation semantics, J. Assoc. Comput. Math. 43 (3) (1996) 555–600.
[12] J. Groote, F. Vaandrager, An efficient algorithm for branching bisimulation and stuttering equivalence, in: M. Paterson (Ed.), Proc. 17th ICALP, Warwick, Lecture Notes in Computer Science, Vol. 443, Springer, Berlin, 1990, pp. 626–638.
[13] H. Hansson, Time and probability in formal design of distributed systems, Ph.D. thesis, University of Uppsala, 1991.

[14] C. Hoare, Communicating Sequential Processes, Prentice-Hall, London, 1985.
[15] V. Kulkarni, Modeling and Analysis of Stochastic Systems, Chapman & Hall, London, 1996.
[16] K. Larsen, A. Skou, Bisimulation through probabilistic testing, Inform. Comput. 94 (1991) 1–28.
[17] R. De Nicola, F. Vaandrager, Three logics for branching bisimulation, J. Assoc. Comput. Math. 42 (2) (1995) 458–487.
[18] A. Philippou, I. Lee, O. Sokolsky, Weak bisimulation for probabilistic systems, in: C. Palamidessi (Ed.), Proc. CONCUR'00, Lecture Notes in Computer Science, Vol. 1877, Springer, Berlin, 2000, pp. 334–349.
[19] R. Segala, Modeling and verification of randomized distributed real-time systems, Ph.D. Thesis, Massachusetts Institute of Technology, 1995.
[20] R. Segala, N. Lynch, Probabilistic simulations for probabilistic processes, Nordic J. Comput. 2 (2) (1995) 250–273.
[21] M. Stoelinga, Alea jacta est: Verification of probabilistic, real-time and parametric systems, Ph.D. Thesis, Katholieke Universiteit Nijmegen, The Netherlands, 2002.
[22] M. Vardi, Automatic verification of probabilistic concurrent finite state programs, in: Proc. of 26th Symp. on Foundations of Computer Science, IEEE Computer Soc. Press, Silver Spring, MD, 1985, pp. 327–338.