Available online at www.sciencedirect.com
SCIENCE DIRECT®
ELSEVIER
Electronic Notes in Theoretical Computer Science 128 (2005) 53–66
Electronic Notes in Theoretical Computer Science
www.elsevier.com/locate/entcs

Equivalences for Silent Transitions in Probabilistic Systems

(Extended Abstract)

S. Andova[a,3], T.A.C. Willemse[b,1,2]

[a] *Department of Computer Science, Twente University, P.O. Box 217, 7500 AE Enschede, The Netherlands*

[b] *Nijmegen Institute for Computing and Information Sciences (NIII), Radboud University Nijmegen, P.O. Box 9010, 6500 GL Nijmegen, The Netherlands*

Abstract

We address abstraction in the setting of probabilistic reactive systems, and study its formal underpinnings for the *strictly alternating model*. 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. Together they give a better understanding of branching bisimilarity. A crucial observation, and, in fact a major motivation for this work is that the notions of branching bisimilarity in the alternating and in the non-alternating model differ, and that the latter one discriminates between systems that are intuitively branching bisimilar.

Keywords: Process theory, abstraction, branching bisimulation, probabilistic systems, coloured trace equivalence

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

[1] Part of this research was conducted while the author was in the Formal Methods group at the Eindhoven University of Technology

[2] Corresponding author. Email: timw@cs.ru.nl

[3] Email: suzana@cs.utwente.nl

doi:10.1016/j.entcs.2004.11.028

purposes at hand, have been hidden. It is an invaluable tool when dealing with complex systems. Over the past decades, research has made great strides in coping with abstraction in theories that focus on functional behaviours of systems. However, when it comes to theories focusing on functional behaviours *and* extra-functional behaviours, we suddenly find that many issues are still unresolved.

This paper addresses abstraction in the setting of systems that combine non-determinism and probabilism, hereafter referred to as *probabilistic systems*. The model we use in this paper is that of *graphs* that adhere to the strictly alternating regime as studied by Hansson [8], rather than the non-alternating model [11,12] as proposed by Segala *et al.* We study the notion of *branching bisimilarity* for this model. The need for this particular equivalence relation is already convincingly argued by Van Glabbeek and Weijland [6] and Groote and Vaandrager [7]. Recall that branching bisimilarity for probabilistic systems has been defined earlier for the non-alternating model by Segala and Lynch [12] and a variation on that notion was defined by Stoelinga [13]. We stress that the differences in 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 an essential motivation for our investigation: while our branching bisimulation relation satisfies the properties one expects, the existing notions turn out to be too strict in their current phrasing (a more detailed account of this is given in our section on related work, see 5), and discriminate between systems that are intuitively branching bisimilar.

Van Glabbeek and Weijland [6] showed that a key property of branching bisimilarity is its preservation of *potentials* of a (non-probabilistic) system. Roughly speaking, these are the options the system has to branch and behave. They illustrated this property by defining a new equivalence, called *coloured trace equivalence*, which uses colours to code for the potentials. Subsequently, they showed that branching bisimilarity and this new equivalence coincide, and both are strictly finer than weak bisimilarity.

Although our setting is 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 a probabilistic counterpart of coloured trace equivalence, and show that it coincides with branching bisimilarity. A major advantage of 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 [10]), is the preservation of *maximal probabilities*. We show that branching bisimilarity can be rephrased in terms of such maximal probabilities.

Both alternatives to branching bisimilarity have their own merits and focus on orthogonal aspects. Together, they are instrumental in understanding branching bisimilarity and its properties for probabilistic systems.

This paper is structured as follows. In section 5, we discuss related work, and compare our notion with existing notions. In section 2, we introduce the semantic framework together with the notion of branching bisimilarity. In section 3, we prove that branching bisimilarity can be rephrased in terms of maximal probabilities. Section 4 focuses on coloured trace equivalence and we show that it coincides with branching bisimilarity. In section 5, we discuss related work and compare our notion with existing notions. In fact, section 5 also serves as the motivation for conducting this research, but it is postponed to the end of the paper for purposes of readability. We end with some closing remarks and directions for further research in section 6. Note that the full version of this paper will appear as [1], and contains several additional results and proofs.

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

2 Branching Bisimilarity for the Alternating Model

We use *graphs*[4] to model probabilistic systems. The graphs we consider follow the strictly alternating regime of Hansson [8]. They can be used to describe systems with both non-deterministic and probabilistic traits. 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*, representing atomic activities of a system. The probabilistic transitions model the probabilistic information of a system. We assume 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 final node for all graphs.

Definition 2.1 *A* graph *is a 7-tuple* $\langle N, P, s, Act, \rightarrow, \rightsquigarrow, \mathsf{pr}\rangle$*, 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* $P \cup \{\mathsf{nil}\}$.

[4] The model we use is also known as *Labelled Concurrent Markov Chains*. We use the term *graph* to stay in line with [6].

- $s \in P$ *is the* initial node*, also called* root*.*
- *Act is a finite set of* action labels*. The special action* $\tau \in Act$ *represents* unobservable events*.*
- $\rightarrow \subseteq N \times Act \times P_{\mathsf{nil}}$ *is the* non-deterministic transition relation*. We require for all* $n \in N$*, that there is at least one* $(n, a, p) \in \rightarrow$*.*
- $\rightsquigarrow \subseteq P \times N$ *is a* probabilistic transition relation*.*
- $\mathsf{pr}\colon \rightsquigarrow \rightarrow (0, 1]$ *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 $N_{\mathsf{x}}, P_{\mathsf{x}}, s_{\mathsf{x}}$, *etc.* for the constituent parts 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 define the *cumulative probability* $\mu\colon S_{\mathsf{nil}} \times 2^{S_{\mathsf{nil}}} \rightarrow [0, 1]$, which yields the total probability of reaching a set of nodes via probabilistic transitions: $\mu(p, \mathcal{M}) = \sum_{n \in \mathcal{M} \cap N} \mathsf{pr}(p, n)$ if $p \in P$ and 0 otherwise.

2.1 Paths and Schedulers

The standard approach in analysing graphs combining non-determinism and probability is to decompose the graph in a set of *computation trees*, which are then the subject of further quantitative analysis. The decomposition requires all non-determinism in the graph to be resolved. This is achieved using a *scheduler* (also known as adversary or policy). We briefly repeat the basics. For a more in-depth explanation, we refer to [13,1].

Let x be a graph. A *path* starting in a node $s_0 \in S$ 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 $i \geq 1$, $s_i \in S_{\mathsf{nil}}$ and $l_i \in \mathrm{Act} \cup (0, 1]$. We require for all $s_i \in N$ $s_i \xrightarrow{l_{i+1}} s_{i+1}$ and for all $s_i \in P$ $s_i \rightsquigarrow s_{i+1}$ and $l_{i+1} = \mathsf{pr}(s_i, s_{i+1})$. 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* that occur in c. The set of *maximal paths* starting in a node s_0, denoted $\mathsf{Path}_m(s_0)$, consists of all infinite paths, and all finite paths ending in nil. The set of finite paths starting in s_0 is denoted $\mathsf{Path}_f(s_0)$.

A *scheduler* of paths starting in a node s_0 is a partial function $\sigma\colon \mathsf{Path}_f(s_0) \mapsto (\rightarrow \cup \{\bot\})$, (where $\bot$ represents "halt"). We require that for all $c \in \mathsf{Path}_m(s_0)$, $\sigma(c) = \bot$. Moreover, if, for some $c \in \mathsf{Path}_f(s_0)$, $\sigma(c)$ is defined we require:

(i) if $\mathsf{last}(c) \in N$ then $\sigma(c) = \bot$ or $\sigma(c) = \mathsf{last}(c) \xrightarrow{a} t$ for some a and t.

(ii) if $\mathsf{last}(c) \in P$ then $\sigma(c) = \bot$.

The set of schedulers for a node s_0 is denoted $\mathsf{Sched}(s_0)$. For a scheduler σ on a graph x, we write $\mathsf{SPath}(s_0, \sigma)$ for the set of all finite and infinite paths $c \equiv s_0\ l_1\ s_1 \ldots$ with $\sigma(s_0\ l_1\ s_1\ \ldots\ s_i) = s_i \xrightarrow{l_{i+1}} s_{i+1}$ for every i (and $i < n$ for finite paths) with $s_i \in N$. We refer to them as $\sigma-$scheduled paths. Every scheduler σ on a graph x that starts in s_0 defines a graph $\mathsf{x}_{s_0,\sigma}$, called a *computation tree*, that contains no non-deterministic branching and whose nodes are finite paths in x. Using the probabilistic transition relation $\rightsquigarrow$ of x, a probability measure on the set of paths of $\mathsf{x}_{s_0,\sigma}$ is defined as follows. The set of maximal paths in $\mathsf{x}_{s_0,\sigma}$ (infinite paths and finite paths that end in a node scheduled to $\bot$), $\mathsf{SPath}_m(s_0, \sigma)$ is the sample space; the sigma-algebra is the smallest sigma-algebra on $\mathsf{SPath}_m(s, \sigma)$ that contains all basic cylinders $c\uparrow = \{c' \in \mathsf{SPath}_m(s_0, \sigma) \mid\ c \text{ is a prefix of } c'\}$ for c a finite $\sigma-$scheduled path; the measure $\mathcal{P}$ is the unique extension of the measure $\mathcal{P}$ (remark that here we overload the notation $\mathcal{P}$) defined over the basic cylinders: $\mathcal{P}(c\uparrow) = \mathbf{P}(c)$, where $\mathbf{P}(c)$ is the probability of c (for details, see e.g. [13,1]).

2.2 Branching Bisimilarity

As we show in this section, branching bisimilarity is an equivalence relation on graphs. It allows one to reason about systems using abstraction, and it enjoys several pleasing properties. For instance, in contrast to *weak bisimilarity* [10], it preserves the non-deterministic branching structure of graphs, in the sense that it "preserves computations together with the potentials in all intermediate states that are passed through, even if silent moves are involved" (quote from Groote and Vaandrager [7]). In the remainder of this section, we give a formal definition of branching bisimilarity.

We first fix some shorthand notation. Let c be a finite path. Then the path c satisfies a predicate ϕ, denoted by $c\ \mathsf{sat}\ \phi$ is defined as follows for the following predicates:

(i) $c\ \mathsf{sat}\ s \Longrightarrow_{\mathcal{M}} s'$ iff $\mathsf{first}(c) = s$, $\mathsf{last}(c) = s'$, $\mathsf{trace}(c) = \tau^*$ and $\mathsf{nodes}(c) \subseteq \mathcal{M}$.

(ii) $c\ \mathsf{sat}\ s \Longrightarrow_{\mathcal{M}} \cdot \xrightarrow{a} s'$ iff there is a path c' such that $c \equiv c'\ a\ s'$ and $c'\ \mathsf{sat}\ s \Longrightarrow_{\mathcal{M}} \mathsf{last}(c')$.

(iii) $c\ \mathsf{sat}\ s \Longrightarrow_{\mathcal{M}} \cdot \rightsquigarrow s'$ iff there is a path c' and probability $\pi \in (0, 1]$, such that $c \equiv c'\ \pi\ s'$ and $c'\ \mathsf{sat}\ s \Longrightarrow_{\mathcal{M}} \mathsf{last}(c')$.

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 σ-scheduled paths starting in s that silently traverse through a set of nodes $\mathcal{M}$ and reach a node in $\mathcal{M}'$ by executing an action a.

$$\begin{aligned}\mathcal{B}_\sigma(s \stackrel{a}{\Longrightarrow}_{\mathcal{M}} \mathcal{M}') = \{c \in \mathsf{SPath}(s,\sigma) \mid \sigma(c) = \bot \text{ and either} \\ c \text{ sat } s \Longrightarrow_{\mathcal{M}} \cdot \stackrel{a}{\longrightarrow} s', s' \in \mathcal{M}', \text{ or} \\ c \text{ sat } s \Longrightarrow_{\mathcal{M}} \cdot \rightsquigarrow s', s' \in \mathcal{M}', a = \tau, \text{ or} \\ c \equiv s, a = \tau, \mathcal{M} = \mathcal{M}'\}\end{aligned} \tag{1}$$

When $a = \tau$, we generally omit it and write $\mathcal{B}_\sigma(s \Longrightarrow_{\mathcal{M}} \mathcal{M}')$ instead of $\mathcal{B}_\sigma(s \stackrel{\tau}{\Longrightarrow}_{\mathcal{M}} \mathcal{M})$. 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 $p \in \mathcal{M}$, the function $\mu_{\mathcal{M}}(p, \mathcal{M}')$ is used to denote the conditional probability of entering $\mathcal{M}'$ from p (in one step), under condition that a step leaving $\mathcal{M}$ is taken. Formally, we have

$$\mu_{\mathcal{M}}(p, \mathcal{M}') = \begin{cases} \dfrac{\mu(p, \mathcal{M}')}{1 - \mu(p, \mathcal{M})} & \text{if } p \in P \text{ and } \mu(p, \mathcal{M}) \neq 1 \\ 0 & \text{otherwise} \end{cases} \tag{2}$$

Using the above definitions, we are in a position to formally define branching bisimilarity.

Definition 2.2 *Let* x *and* y *be graphs and denote* $S = S_{\mathsf{x}} \cup S_{\mathsf{y}}$ *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 nodes* s *and* t *for which* $s\mathcal{R}t$ *holds, we have*

(i) *if* $s \in N$ *and* $s \stackrel{a}{\longrightarrow} s'$, *then there is a scheduler* σ, *such that* $\mathcal{P}(\mathcal{B}_\sigma(t \stackrel{a}{\Longrightarrow}_{[t]_{\mathcal{R}}} [s']_{\mathcal{R}})) = 1$.

(ii) *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* $\mathsf{x} \leftrightarrows_b \mathsf{y}$ *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 action transitions (i.e. also the "inert" τ transitions: τ transitions that do not change the potentials of a system) emanating from nodes in an equivalence class to be schedulable (with probability 1) from *all* nodes in that class. This means that all nodes in the same equivalence class have the same observable potentials. The second condition requires that a single scheduler of one node can be used to simulate the normalised cumulative probability of a probabilistic bisimilar node. Such a scheduler can schedule any finite number of silent steps (possibly zero) within

its own class before reaching the corresponding class.

Example 2.3 Consider the graphs of Figure 1. Applying the definition of branching bisimulation, it is clear that the two graphs are branching bisimilar. Nodes in the same equivalence class are coloured with the same colour. The annotation "Node p" is not of importance at this point, but is there to facilitate an argument in section 5.

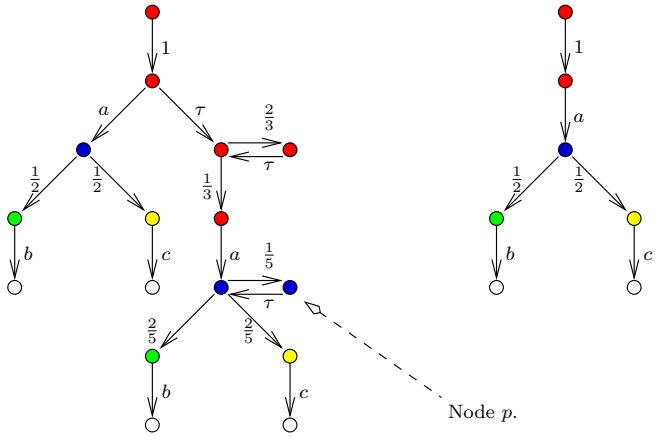

Fig. 1. Branching bisimilar graphs

Proposition 2.4 *$\leftrightarrow_b$ is a conservative extension of branching bisimilarity for non-probabilistic graphs as defined in [6].*

3 Branching Bisimilarity and Maximal Probabilities

Philippou *et al.* [10] showed that weak bisimilarity for the alternating model can be rephrased in terms of maximal probabilities. In this section, we show that also branching bisimilarity admits an alternative characterisation in terms of *maximal probabilities*. This means that it suffices to check for the branching bisimulation conditions using schedulers that induce maximal probabilities. As a result, an algorithm for deciding branching bisimilarity can be defined.

Given a graph x, $a \in \mathrm{Act}$ and $\mathcal{M} \subseteq S_{\mathsf{nil}}$ we introduce the following notation for the maximal probability, over all schedulers in $\mathsf{Sched}(s)$, to reach $\mathcal{M}$ by executing a while internal transitions to states related to s are allowed.

$$\mathcal{P}_{\max}(s \stackrel{a}{\Longrightarrow}_{[s]_{\mathcal{R}}} \mathcal{M}) = \max_{\sigma \in \mathsf{Sched}(s)} \mathcal{P}(\mathcal{B}_\sigma(s \stackrel{a}{\Longrightarrow}_{[s]_{\mathcal{R}}} \mathcal{M})) \tag{3}$$

If $a = \tau$ we omit a and we simply write $\mathcal{P}_{\max}(s \Longrightarrow_{[s]_{\mathcal{R}}} \mathcal{M})$.

The main result in this section is split in two lemmata. Combined, they show that it suffices to prove the branching bisimulation conditions for schedulers that induce maximal probabilities. The first lemma expresses that branching bisimilarity implies equal maximal probabilities for related nodes.

Lemma 3.1 *Let $\mathcal{R}$ be a branching bisimulation relation on S_{nil}.*

(i) *If $s\mathcal{R}t$, then $\mathcal{P}_{max}(s \Longrightarrow_{[s]_\mathcal{R}} \mathcal{M}) = \mathcal{P}_{max}(t \Longrightarrow_{[t]_\mathcal{R}} \mathcal{M})$, for $\mathcal{M} \neq [s]_\mathcal{R}$.*

(ii) *If $s \in P$ and $\mu(s, [s]_\mathcal{R}) \neq 1$ then for all $\mathcal{M} \in S_{\mathsf{nil}/\mathcal{R}} \backslash \{[s]_\mathcal{R}\}$, $\mathcal{P}_{max}(s \Longrightarrow_{[s]_\mathcal{R}} \mathcal{M}) = \mu_{[s]_\mathcal{R}}(s, \mathcal{M})$.*

(iii) *If $s\mathcal{R}t$, then $\mathcal{P}_{max}(s \overset{a}{\Longrightarrow}_{[s]_\mathcal{R}} \mathcal{M}) = \mathcal{P}_{max}(t \overset{a}{\Longrightarrow}_{[t]_\mathcal{R}} \mathcal{M})$, for $\mathcal{M} \neq [s]_R$ or $a \neq \tau$.*

The second lemma gives the other direction, namely that branching bisimulation can be defined only in terms of maximal probabilities.

Lemma 3.2 *Let x and y be graphs and denote $S = S_x \cup S_y$ and $S_{\mathsf{nil}} = S \cup \{\mathsf{nil}\}$. Let $\mathcal{R}$ be an equivalence relation on S_{nil} such that for nodes s and t for which $s\mathcal{R}t$ holds, we have:*

(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 $a \in Act$ 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 $\mathcal{M} \in S_{\mathsf{nil}/\mathcal{R}} \setminus \{[s]_\mathcal{R}\}$.*

Then $\mathcal{R}$ is a branching bisimulation relation.

In [10] an algorithm for deciding weak bisimilarity for probabilistic systems is given. A finite set of schedulers, so-named determinate schedulers, is introduced and these are shown to suffice for computing maximal probabilities. For branching bisimilarity we have shown [1] in a similar manner that only a finite subset of the set of all schedulers of the graph under consideration needs to be investigated. Namely, we can show that *simple schedulers* are sufficient for computing the maximal probabilities. A scheduler is simple if for all finite paths ending in a same node it schedules a unique transition. Remark that the computation tree under a simple scheduler can be represented by a finite state fully-probabilistic graph. Thus, the problem of deciding branching bisimulation amounts to solving a linear optimisation problem, giving rise to an algorithm of polynomial complexity. It is worth mentioning that for branching bisimilarity for the non-alternating model no algorithm has been defined, while, so far, only an exponential algorithm deciding weak bisimilarity for this model has been proposed.

4 Colours, Blends and Coloured Trace Equivalence

In our introduction we claimed that one of the pleasing properties of branching bisimilarity (which also sets it apart from weak bisimilarity) was that it can distinguish between nodes with different potentials. We now add weight to this claim: we show how colours can be used to code for these potentials and prove that the observation of the colours of a node can be used to distinguish between truly inert transitions and non-inert transitions. The full version of this paper also provides a *concrete coloured trace equivalence* characterisation of *strong bisimilarity* [9], which further supports the view of using colours for potentials.

4.1 Colours and Blends

Let $\mathcal{C}$ be a finite (but sufficiently large) set of colours. A *raw blend* is a mix of colours in a particular ratio, i.e. a raw blend b is a subset of $\mathcal{C} \times (0,1]$, with the sanity-condition $\sum_{(c,\pi)\in b} \pi = 1$. The set of all raw blends is denoted $\mathcal{B}_r$.

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 a raw blend b. To test whether a colour actually occurs in a blend, we introduce the predicate $b\lfloor c$, which holds iff $b\,\mathsf{probe}\,c > 0$.

In the remainder, we shall use a subset of raw blends simply called *blends*. A blend is a raw blend b iff for all colours c, $b\lfloor c$ implies $(c, b\,\mathsf{probe}\,c) \in b$. In other words, a colour appears only once in a blend. 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 $\circlearrowleft(b) = \{(c, b\,\mathsf{probe}\,c) \mid \text{for all } c \text{ satisfying } b\lfloor c\}$

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 $\{(c,1)\}$).

4.2 Coloured Trace Equivalence

Trace equivalence in general is too weak to characterise a branching-time equivalence. However, we show that by augmenting the graphs of section 2 with colour codings, we obtain a decorated trace equivalence that is equi-discriminating as branching bisimilarity. The graphs that are endowed with a colouring of their nodes are referred to as *coloured graphs*.

Definition 4.1 *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, and we assume that we can observe the blends of the nodes and the labels on the non-deterministic

transitions. Such runs are called *pre-coloured traces*.

Definition 4.2 *Let $\langle \mathsf{x}, \gamma \rangle$ be a coloured graph. A* pre-coloured trace*, starting in a node s is a sequence of one of the following forms:*

(i) $\gamma(\mathsf{nil})$ *is a pre-coloured trace.*

(ii) $b'_0\ b'_0\ a_1\ b_1\ b'_1 \ldots a_{m+1}\ b_{m+1}$ *when $s \in N$ and there is at least one path $c \equiv n_0\ a_1\ p_1\ \pi_1 \ldots a_{m+1}\ p_{m+1}$ with $\mathsf{trace}(c) = a_1 \ldots a_{m+1}$, $\mathsf{first}(c) = s = n_0$ and for all $2 \leq i \leq m+1$, $\gamma(p_i) = b_i$ and for all $1 \leq j \leq m+1$ $\gamma(n_{j-1}) = b'_{j-1}$.*

(iii) $b_0\ b'_0\ a_1\ b_1\ b'_1 \ldots a_{m+1}\ b_{m+1}$ *when $s \in P$ and there is at least one path $c \equiv p_0\ \pi_0\ n_0\ a_1\ p_1 \pi_1 \ldots a_{m+1}\ p_{m+1}$ with $\mathsf{trace}(c) = a_1 \ldots a_{m+1}$, $\mathsf{first}(c) = s = p_0$ and for all $0 \leq i \leq m+1$, $\gamma(p_i) = b_i$ and for all $1 \leq j \leq m+1$ $\gamma(n_{j-1}) = b'_{j-1}$.*

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 allows us to compare decorated traces starting in probabilistic nodes with those starting in non-deterministic nodes.

The idea behind branching bisimilarity is that it preserves the potentials of a system; τ actions that can be removed without changing the potentials are called *inert*. In our coloured graphs, we use the blends as an indication for the potentials of the node. Intuitively, this means that by removing *only those τ actions in a pre-coloured trace that are in between nodes with the same 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 4.3 *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 $b\ \tau\ b$* [5]*, that is obtained from a pre-coloured trace starting in node s in which all subsequences of the form $b\ (b\ \tau\ b)^+$ and $(b\ \tau\ b)^+\ b$ have been replaced with b.*

Thus far, we have considered arbitrary coloured graphs. Before we continue, we make the following two observation:

(i) in the non-probabilistic case, blends are not needed to code for potentials: plain colours suffice (see e.g. [6]).

(ii) the distinction between probabilistic and non-deterministic nodes is obscured by the unobservable actions.

[5] 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 potential inert τ step. If the τ step is not inert, it will appear in some extension of the coloured trace

This leads us to consider a subset of coloured graphs in which non-deterministic nodes are labelled with a blend, *only when we cannot distinguish them from probabilistic nodes*. Blends are reserved for coding the probability distributions over successor states. This leads to the following definition.

Definition 4.4 *A* properly coloured graph *is a coloured graph* $\langle \mathsf{x}, \gamma \rangle$ *where* γ *satisfies:*

(i) *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* $a \in Act$ *and* $p \in P_{\mathsf{nil}}$, $n \xrightarrow{a} p$ *implies* $a = \tau$ *and* $\gamma(n) = \gamma(p)$.

(ii) *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 introduce the notion of *consistency*, basically capturing that two nodes can only be coloured with the same colour when they have the same behaviours. We say that for a set of coloured graphs 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 4.5 *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.*

We next relate branching bisimilarity and coloured trace equivalence.

Theorem 4.6 *For all* x *and* y, $\mathsf{x} \leftrightarrows_b \mathsf{y}$ *iff* $\mathsf{x} \equiv_c \mathsf{y}$.

5 Related Work

Two approaches in modelling probabilistic systems (i.e. systems with both probabilism and non-determinism) can be distinguished. The first approach is the model of *probabilistic (simple) automata* (often called the *non-alternating model*), which was introduced in [12,11]. The second approach, based on the Concurrent Markov Chains of [14], is that of the *alternating model*, which was introduced in [8] by Hansson.

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 [10]) and weak bisimilarity for the

non-alternating model (as defined in [11,12]) are incomparable.

Alternating vs. non-alternating

Comparing our notion of branching bisimulation with the notion of branching bisimulation in the non-alternating setting, as defined by Segala and Lynch [11,12] we find that their notion is too restrictive. This is illustrated by the following example. Consider the two graphs of Figure 1 (see section 2.2, page 7). 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: p appears as a state in the "non-alternating" counterpart of the left graph and it cannot be related to any state in the "non-alternating" counterpart of the right graph. The same phenomenon is also present in a variation of branching bisimulation, called *delay branching bisimulation*, which is defined by Stoelinga [13].

In this paper, we show 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 [6] in the non-probabilistic case, see section 3 and section 4). We therefore believe that the definition of branching bisimulation in the non-alternating setting is incomplete and requires further research.

Note that the so-named *combined* version of branching bisimulation in [12] relates processes that are not related by our branching bisimulation (but still not the ones from Figure 1). 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.

Branching bisimilarity vs. weak bisimilarity

Comparing our definition of branching bisimilarity with weak bisimilarity as defined by Philippou *et al.* [10], we find that branching bisimilarity is strictly finer (although there is a big overlap in systems that are both, such as for *fully probabilistic systems* [2]). 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. As an example, consider the graphs of figure 2, which are weak bisimilar, but not branching bisimilar.

Decidability

Finally, we find that no extensive study on the decidability and complexity of branching bisimulation has been conducted. To this date, no algorithm

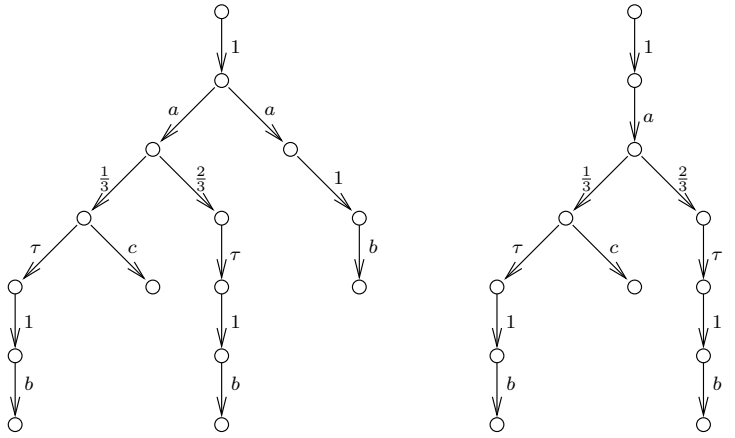

Fig. 2. Graphs that are weak bisimilar but not branching bisimilar

for deciding branching bisimilarity (in the non-alternating model) has been defined, whereas our notion can be decided in polynomial time [1]. Deciding weak bisimilarity in the alternating setting can be achieved in polynomial time [10], whereas the best known algorithm for deciding weak bisimilarity in the non-alternating model as defined in [12] is exponential [5]. Only a finer variant of weak bisimulation (for the non-alternating model), called weak delay bisimulation [13,3] is decidable in polynomial time [3].

6 Closing Remarks

We defined the notion of branching bisimilarity for the strictly alternating model. We showed that it preserves the branching structure of a system by defining an alternative equivalence, called *coloured trace equivalence*, that clearly satisfies this property (cf. Van Glabbeek and Weijland did in the non-probabilistic setting [6]), and subsequently showing that the two equivalences coincide. Coloured trace equivalence is easily understood without knowledge of probability measures, schedulers, *etcetera*. We pose it as an open problem whether coloured trace equivalence gives rise to a different type of algorithm for deciding branching bisimilarity than the ones that are based on schedulers.

Furthermore, we showed that the branching bisimulation conditions can be rephrased to conditions that use schedulers which induce maximal probabilities, thereby giving rise to a decision procedure for the equivalence (see [1]).

The two alternative characterisations strengthen our belief that our notion of branching bisimilarity is correct.

In contrast to the existing notions of branching bisimilarity (defined for the non-alternating model), our notion relates exactly those processes that

are intuitively branching bisimilar, whereas the other notions identify less (see section 5). This means that additional research is required to mend this situation in the non-alternating model.

References

[1] Andova, S., and T.A.C. Willemse, *Equivalences for silent transitions in probabilistic systems*, Computer Science Report, University of Twente, *to appear*.

[2] Baier, C., and H. Hermanns, *Weak bisimulation for fully probabilistic processes*, Proc. CAV'97, O. Grumberg, ed., LNCS 1254, pp. 119-130, 1997.

[3] Baier, C., and M. Stoelinga, *Norm function for probabilistic bisimulations with delays*, Proc. FOSSACS'00, J. Tiuryn, ed., LNCS 1784, pp. 1-16, Berlin, Germany, 2000.

[4] Bandini, E., and R. Segala, *Axiomatizations for probabilistic bisimulation*, Proc. ICALP'01, F. Orejas, P.G. Spirakis, J. van Leeuwen, eds., Crete, Greece, LNCS 2076, Springer Verlag, pp. 370-381, 2001.

[5] Cattani, S., and R. Segala, *Decision algorithms for probabilistic bisimulation*, Proc. CONCUR'02, L. Brim, P. Janar, M. Katínský, A. Kuera, eds., Brno, Czech Republic, LNCS 2421, Springer Verlag, pp. 371-385, 2002.

[6] van Glabbeek, R.J., and W. P. Weijland, *Branching time and abstraction in bisimulation semantics*, Journal of ACM, Vol. **43-3** (1996), pp. 555-600.

[7] Groote, J.F., and F. Vaandrager, *An Efficient Algorithm for Branching Bisimulation and Stuttering Equivalence*, Proceedings 17th ICALP, Warwick, M.S. Paterson, ed., LNCS 443, Springer Verlag, pp. 626-638, 1990.

[8] Hansson, H. "Time and probability in formal design of distributed systems," Ph.D. thesis, DoCS 91/27, University of Uppsala, 1991.

[9] Larsen, K.G., and A. Skou, *Bisimulation through probabilistic testing*, Information and Computation, **94** (1991), pp. 1-28.

[10] Philippou, A., and I. Lee, O. Sokolsky, *Weak bisimulation for probabilistic systems*, Proc. CONCUR'00, C. Palamidessi, ed., LNCS 1877, Springer Verlag, University Park, PA, USA, pp. 334-349, 2000.

[11] Segala, R. "Modeling and verification of randomized distributed real-time systems," Ph.D. thesis, Massachusetts Institute of Technology, 1995.

[12] Segala, R., and N.A. Lynch, *Probabilistic simulations for probabilistic processes*, Nordic Journal of Computing, **2(2)** (1995), pp. 250-273.

[13] Stoelinga, M. "Alea jacta est: Verification of probabilistic , real-time and parametric systems," Ph.D. thesis, Katholieke Universiteit Nijmegen, The Netherlands, 2002.

[14] Vardi, M.Y. *Automatic verification of probabilistic concurrent finite state programs*, Proc. of 26th Symp. on Foundations of Com. Sc., IEEE Comp. Soc. Press, pp. 327-338, 1985.