Deciding Probabilistic Automata Weak Bisimulation in Polynomial Time

Holger Hermanns and Andrea Turrini

Saarland University – Computer Science, Saarbrücken, Germany

Abstract

Deciding in an efficient way weak probabilistic bisimulation in the context of probabilistic automata is an open problem for about a decade. In this work we close this problem by proposing a procedure that checks in polynomial time the existence of a weak combined transition satisfying the step condition of the bisimulation. This enables us to arrive at a polynomial time algorithm for deciding weak probabilistic bisimulation. We also present several extensions to interesting related problems setting the ground for the development of more effective and compositional analysis algorithms for probabilistic systems.

1998 ACM Subject Classification G.3: Probability and Statistics; F.2: Analysis Of Algorithms and Problem Complexity.

Keywords and phrases Probabilistic Automata, Weak probabilsitic bisimulation, Linear Programming problem, Polynomial decision algorithm.

Digital Object Identifier 10.4230/LIPIcs.FSTTCS.2012.435

1 Introduction

Probabilistic automata (*PA*) constitute a mathematical framework for the specification of probabilistic concurrent systems [4,22]. Probabilistic automata extend classical concurrency models in a simple yet conservative fashion. In probabilistic automata, there is no global notion of time, and probabilistic experiments can be performed inside a transition. This embodies a clear separation between probability and nondeterminism, and is represented by transitions of the form $s \xrightarrow{a} \mu$, where s is a state, a is an action label, and μ is a probability distribution on states. Labeled transition systems are instances of this model family, obtained by restricting to Dirac distributions (assigning full probability to single states). Thus, foundational concepts and results of standard concurrency theory are retained in full and extend smoothly to the model of probabilistic automata. The *PA* model is akin to Markov decision processes (*MDP*) [7], and its foundational beauty can be paired with powerful model checking techniques, as implemented for instance in the PRISM tool [16]. Variations of this model are Labeled Concurrent Markov Chains (*LCMC*) and alternating Models [11, 21, 27]. We refer the interested reader to [23] for a survey on *PA* and other models.

If facing a concrete probabilistic system, we can conceive several different *PA* models to reflect its behavior. For instance, we can use different state names, encode diverse information in the states, represent internal computations with different action labels, and so on. *Bisimulation relations* constitute a powerful tool allowing us to check whether two models describe essentially the same system. They are then called bisimilar. The bisimilarity of two systems can be viewed in terms of a game played between a challenger and a defender. In each step of the infinite bisimulation game, the challenger chooses one automaton, makes a step, and the defender matches it with a step of the other automaton. Depending on how we want to treat internal computations, this leads to *strong* and *weak* bisimulations: the

© H. Hermanns and A. Turrini;
licensed under Creative Commons License NC-ND

32nd Int'l Conference on Foundations of Software Technology and Theoretical Computer Science (FSTTCS 2012).
Editors: D. D'Souza, J. Radhakrishnan, and K. Telikepalli; pp. 435–447
Leibniz International Proceedings in Informatics
Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

former requires that each single step of the challenger automaton is matched by an equally labeled single step of the defender automaton, the latter allows the matching up to internal computation steps. On the other hand, depending on how nondeterminism is resolved, probabilistic bisimulation can be varied by allowing the defender to match the challenger's step by a convex combination of enabled probabilistic transitions. This results in a spectrum of four bisimulations: strong [11, 22, 27], strong probabilistic [22], weak [21, 22], and weak probabilistic [22] bisimulation.

Besides comparing automata, bisimulation relations allow us to reduce the size of an automaton without changing its properties (i.e., with respect to logic formulae satisfied by it). This is particularly useful to alleviate the state explosion problem notoriously encountered in model checking.

Polynomial decision algorithms for strong (probabilistic) bisimulation [3] and weak bisimulation [21] are known. However, *PA* weak bisimulation lacks in transitivity and this severely limits its usefulness. On the other hand weak probabilistic bisimulation is indeed transitive, while the only known algorithm for such bisimulation is exponential [3] in the size of the probabilistic automaton.

In this context, it is worth to note that *LCMC* weak bisimulation [21] and *PA* weak probabilistic bisimulation [22] coincide [24] when *LCMC* is seen as a *PA* with restrictions on the structure of the automaton and that restricted versions of *PA* weak probabilistic bisimulations, such as normed [1] and delay [25] bisimulation, can be decided in polynomial time. Following [24], an *LCMC* is just a *PA* where each state with outgoing transitions enables either labeled transitions each one leading to a single state, or a single transition leading to a probability distribution over states and this constraint on the structure of the automaton is enough to reduce the complexity of the decision procedure at the expense of the loss of using combined transitions and nondeterminism to simplify the automaton.

Lately, the model of *PA* has been enhanced with memoryless continuous time, integrated into the model of Markov automata [6, 8, 9]. This extension is also rooted in interactive Markov chains (*IMC*) [13], another model with a well-understood compositional theory. *IMC*s are applied in a large spectrum of practical applications, ranging from networked hardware on chips [5] to water treatment facilities [12] and ultra-modern satellite designs [10]. The standard analysis trajectory for *IMC* revolves around compositional applications of weak bisimulation minimization, a strategy that has been proven very effective [2, 5, 14], and is based on a polynomial time weak bisimulation decision algorithm [13, 28]. Owed to the unavailability of effective algorithms for *PA* weak probabilistic bisimulations, this compositional minimization strategy has thus far not been applied in the *PA* (or *MDP*) setting. We aim at making this possible, and furthermore, we intend to repeat and extend the successful applications of *IMC* in the extended Markov automata setting. For this, a polynomial time decision procedures for weak probabilistic bisimulation on *PA* is the essential building block.

In this paper we show that *PA* weak probabilistic bisimulation can be decided in polynomial time, thus just as all other bisimulations on *PA*. To arrive there, we provide a decision procedure that follows the standard partition refinement approach [3, 17, 19] and that is based on a Linear Programming (LP) problem. The crucial step is that we manage to generate and decide an LP problem that proves or disproves the existence of a weak step in time polynomial in the size of an automaton which in turn encodes a weak transition linear in its size. This enables us to decide in polynomial time whether the defender has a matching weak transition step - opposed to the exponential time required thus far [3] for this. Apart from this result, which closes successfully the open problem of [3], we show how our LP approach can be extended to hyper-transitions (weak transitions leaving a probability distribution instead of a single state) and to the novel concepts of allowed weak/hyper-transitions (weak/hyper-transitions involving only a restricted set of transitions) and of equivalence matching (given two states, check whether each one enables a weak transition matchable by the other). Hyper-transitions naturally occur in weak probabilistic bisimulation on Markov automata, and in the bisimulation formulation of probabilistic forward simulation [8, 22].

Organization of the paper. After the preliminaries in Section 2, we present in Section 3 the polynomial LP problem that models weak transitions together with several extensions that can be computed in polynomial time as well. Then, in Section 4, we recast the algorithm proposed in [3] that decides whether two probabilistic automata are weak probabilistic bisimilar and we show that the decision procedure is polynomial. We conclude the paper in Section 5 with some remarks. Due to space limitations, we refer the reader interested in detailed proofs to [15].

2 Mathematical Preliminaries

For a generic set X, denote by $\mathrm{Disc}(X)$ the set of discrete probability distributions over X, and by $\mathrm{SubDisc}(X)$ the set of discrete sub-probability distributions over X. Given $\rho \in \mathrm{SubDisc}(X)$, we denote by $\mathrm{Supp}(\rho)$ the set $\{\, x \in X \mid \rho(x) > 0 \,\}$, by $\rho(\bot)$ the value $1 - \rho(X)$ where $\bot \notin X$, and by δ_x the *Dirac* distribution such that $\rho(x) = 1$ for $x \in X \cup \{\bot\}$. For a sub-probability distribution ρ, we also write $\rho = \{\, p_x x \mid x \in X, p_x = \rho(x) \,\}$. The lifting $\mathcal{L}(\mathcal{R})$ [18] of a relation $\mathcal{R} \subseteq X \times Y$ is defined as follows: for $\rho_X \in \mathrm{Disc}(X)$ and $\rho_Y \in \mathrm{Disc}(Y)$, $\rho_X \mathrel{\mathcal{L}(\mathcal{R})} \rho_Y$ holds if there exists a *weighting function* $w \colon X \times Y \to [0,1]$ such that (1) $w(x,y) > 0$ implies $x \mathrel{\mathcal{R}} y$, (2) $\sum_{y \in Y} w(x,y) = \rho_X(x)$, and (3) $\sum_{x \in X} w(x,y) = \rho_Y(y)$. When $\mathcal{R}$ is an equivalence relation on a set X, $\rho_1 \mathrel{\mathcal{L}(\mathcal{R})} \rho_2$ holds if for each $\mathcal{C} \in X/\mathcal{R}$, $\rho_1(\mathcal{C}) = \rho_2(\mathcal{C})$.

A Probabilistic Automaton (*PA*) $\mathcal{A}$ is a tuple $(S, \bar{s}, \Sigma, D)$, where S is a set of *states*, $\bar{s} \in S$ is the *start state*, Σ is the set of *actions*, and $D \subseteq S \times \Sigma \times \mathrm{Disc}(S)$ is a *probabilistic transition relation*. The set Σ is parted in two sets H and E of internal (hidden) and external actions, respectively; we let s,t,u,v, and their variants with indices range over S, a, b range over actions, and τ range over hidden actions. In this work we consider only finite *PA*s, i.e., automata such that S and D are finite.

A transition $tr = (s, a, \mu) \in D$, also denoted by $s \xrightarrow{a} \mu$, is said to *leave* from state s, to be *labeled* by a, and to *lead* to μ, also denoted by μ_{tr}. We denote by $src(tr)$ the *source* state s, by $act(tr)$ the *action* a, and by $trg(tr)$ the *target* distribution μ. We also say that s enables action a, that action a is enabled from s, and that (s, a, μ) is enabled from s. Finally, we denote by $D(s)$ the set of transitions enabled from s, i.e., $D(s) = \{\, tr \in D \mid src(tr) = s \,\}$, and similarly by $D(a)$ the set of transitions with action a, i.e., $D(a) = \{\, tr \in D \mid act(tr) = a \,\}$.

An *execution fragment* of a *PA* $\mathcal{A}$ is a finite or infinite sequence of alternating states and actions $\alpha = s_0 a_1 s_1 a_2 s_2 \dots$ starting from a state s_0, also denoted by $\mathit{first}(\alpha)$, and, if the sequence is finite, ending with a state, also denoted by $\mathit{last}(\alpha)$, such that for each $i > 0$ there exists a transition $(s_{i-1}, a_i, \mu_i) \in D$ such that $\mu_i(s_i) > 0$. The *length* of α, denoted by $|\alpha|$, is the number of occurrences of actions in α. If α is infinite, then $|\alpha| = \infty$. Denote by $\mathit{frags}(\mathcal{A})$ the set of execution fragments of $\mathcal{A}$ and by $\mathit{frags}^*(\mathcal{A})$ the set of finite execution fragments of $\mathcal{A}$. An execution fragment α is a *prefix* of an execution fragment α', denoted by $\alpha \leqslant \alpha'$, if the sequence α is a prefix of the sequence α'. The *trace* of α, denoted by $\mathit{trace}(\alpha)$, is the sub-sequence of external actions of α. For instance, for $a \in E$, $\mathit{trace}(s_0 a s_1) = \mathit{trace}(s_0 \tau s_1 \tau \dots \tau s_{n-1} a s_n) = a$, also denoted by $\mathit{trace}(a)$, and

$\mathit{trace}(s_0) = \mathit{trace}(s_0 \tau s_1 \tau \dots \tau s_n) = \varepsilon$, the empty sequence, also denoted by $\mathit{trace}(\tau)$.

A *scheduler* for a *PA* $\mathcal{A}$ is a function $\sigma \colon \mathit{frags}^*(\mathcal{A}) \to \mathrm{SubDisc}(D)$ such that for each finite execution fragment α, $\sigma(\alpha) \in \mathrm{SubDisc}(D(\mathit{last}(\alpha)))$. A scheduler is *determinate* [3] if for each pair of execution fragments α, α', if $\mathit{trace}(\alpha) = \mathit{trace}(\alpha')$ and $\mathit{last}(\alpha) = \mathit{last}(\alpha')$, then $\sigma(\alpha) = \sigma(\alpha')$. Given a scheduler σ and a finite execution fragment α, the distribution $\sigma(\alpha)$ describes how transitions are chosen to move on from $\mathit{last}(\alpha)$. A scheduler σ and a state s induce a probability distribution $\mu_{\sigma,s}$ over execution fragments as follows. The basic measurable events are the cones of finite execution fragments, where the cone of a finite execution fragment α, denoted by C_α, is the set $\{\, \alpha' \in \mathit{frags}^*(\mathcal{A}) \mid \alpha \leqslant \alpha' \,\}$. The probability $\mu_{\sigma,s}$ of a cone C_α is defined recursively as follows:

$$\mu_{\sigma,s}(C_\alpha) = \begin{cases} 0 & \text{if } \alpha = t \text{ for a state } t \neq s, \\ 1 & \text{if } \alpha = s, \\ \mu_{\sigma,s}(C_{\alpha'}) \cdot \sum_{tr \in D(a)} \sigma(\alpha')(tr) \cdot \mu_{tr}(t) & \text{if } \alpha = \alpha' a t. \end{cases}$$

Standard measure theoretical arguments ensure that $\mu_{\sigma,s}$ extends uniquely to the σ-field generated by cones. We call the measure $\mu_{\sigma,s}$ a *probabilistic execution fragment* of $\mathcal{A}$ and we say that it is generated by σ from s. Given a finite execution fragment α, we define $\mu_{\sigma,s}(\alpha)$ as $\mu_{\sigma,s}(\alpha) = \mu_{\sigma,s}(C_\alpha) \cdot \sigma(\alpha)(\bot)$, where $\sigma(\alpha)(\bot)$ is the probability of chosing no transitions, i.e., of terminating the computation after α has occurred.

We say that there is a *weak combined transition* from $s \in S$ to $\mu \in \mathrm{Disc}(S)$ labeled by $a \in \Sigma$ that is induced by σ, denoted by $s \xRightarrow{a}_{\mathrm{c}} \mu$, if there exists a scheduler σ such that the following holds for the induced probabilistic execution fragment $\mu_{\sigma,s}$: (1) $\mu_{\sigma,s}(\mathit{frags}^*(\mathcal{A})) = 1$; (2) for each $\alpha \in \mathit{frags}^*(\mathcal{A})$, if $\mu_{\sigma,s}(\alpha) > 0$ then $\mathit{trace}(\alpha) = \mathit{trace}(a)$; (3) for each state t, $\mu_{\sigma,s}(\{\, \alpha \in \mathit{frags}^*(\mathcal{A}) \mid \mathit{last}(\alpha) = t \,\}) = \mu(t)$. See [23] for more details on weak combined transitions.

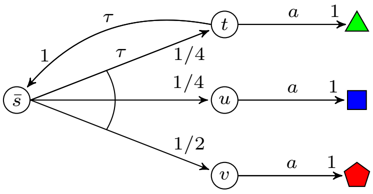


Figure 1 The *PA* $\mathcal{E}$

▶ **Example 1.** Consider the automaton $\mathcal{E}$ depicted in Figure 1 and denote by *tr* the only transition enabled by $\bar{s}$; $\mathcal{E}$ enables the weak combined transition $\bar{s} \xRightarrow{a}_{\mathrm{c}} \mu$ where $\mu = \{\frac{1}{16}\blacktriangle, \frac{5}{16}\blacksquare, \frac{10}{16}\text{⬟}\}$ via the scheduler σ defined as follows: $\sigma(\bar{s}) = \sigma(\bar{s}\tau t \tau \bar{s}) = \delta_{tr}$, $\sigma(\bar{s}\tau t) = \delta_{t \xrightarrow{\tau} \delta_{\bar{s}}}$, $\sigma(\bar{s}\tau u) = \sigma(\bar{s}\tau t \tau \bar{s} \tau u) = \delta_{u \xrightarrow{a} \delta_{\blacksquare}}$, $\sigma(\bar{s}\tau v) = \sigma(\bar{s}\tau t \tau \bar{s} \tau v) = \delta_{v \xrightarrow{a} \delta_{\text{⬟}}}$, $\sigma(\bar{s}\tau t \tau \bar{s} \tau t) = \delta_{t \xrightarrow{a} \delta_{\blacktriangle}}$, and $\sigma(\alpha) = \delta_\bot$ for each other $\alpha \in \mathit{frags}^*(\mathcal{E})$. For instance, state $\blacksquare$ is reached with probability $\mu_{\sigma,\bar{s}}(\{\, \alpha \in \mathit{frags}^*(\mathcal{E}) \mid \mathit{last}(\alpha) = \blacksquare \,\}) = \mu_{\sigma,\bar{s}}(\{\bar{s}\tau u a \blacksquare, \bar{s}\tau t \tau \bar{s} \tau u a \blacksquare\}) = 1 \cdot 1 \cdot \frac{1}{4} \cdot 1 \cdot 1 \cdot 1 + 1 \cdot 1 \cdot \frac{1}{4} \cdot 1 \cdot 1 \cdot 1 \cdot \frac{1}{4} \cdot 1 \cdot 1 \cdot 1 = \frac{5}{16} = \mu(\blacksquare)$, as required. As we will see in Sect. 3.4, the same weak combined transition is induced also by a determinate scheduler.

We say that there is a *hyper-transition* from $\rho \in \mathrm{Disc}(S)$ to $\mu \in \mathrm{Disc}(S)$ labeled by $a \in \Sigma$, denoted by $\rho \xRightarrow{a}_{\mathrm{c}} \mu$, if there exists a family of weak combined transitions $\{s \xRightarrow{a}_{\mathrm{c}} \mu_s\}_{s \in \mathrm{Supp}(\rho)}$ such that $\mu = \sum_{s \in \mathrm{Supp}(\rho)} \rho(s) \cdot \mu_s$, i.e., for each $t \in S$, $\mu(t) = \sum_{s \in \mathrm{Supp}(\rho)} \rho(s) \cdot \mu_s(t)$.

▶ **Definition 1.** Let $\mathcal{A}_1$, $\mathcal{A}_2$ be two probabilistic automata. An equivalence relation $\mathcal{R}$ on the disjoint union $S_1 \uplus S_2$ is a *weak probabilistic bisimulation* if, for each pair of states $s, t \in S_1 \uplus S_2$ such that $s \mathrel{\mathcal{R}} t$, if $s \xrightarrow{a} \mu_s$ for some probability distribution μ_s, then there exists a probability distribution μ_t such that $t \xRightarrow{a}_{\mathrm{c}} \mu_t$ and $\mu_s \mathrel{\mathcal{L}(\mathcal{R})} \mu_t$.

Two probabilistic automata $\mathcal{A}_1$ and $\mathcal{A}_2$ are weakly probabilistic bisimilar if there exists a weak probabilistic bisimulation $\mathcal{R}$ on $S_1 \uplus S_2$ such that $\bar{s}_1 \mathrel{\mathcal{R}} \bar{s}_2$. We denote the coarsest weak probabilistic bisimulation by $\approx$, and call it weak probabilistic bisimilarity.

This is the central definition around which the paper revolves. Weak probabilistic bisimilarity is an equivalence relation preserved by standard process algebraic composition operators on *PA* [20]. The complexity of deciding $\mathcal{A}_1 \approx \mathcal{A}_2$ strictly depends on finding $t \xRightarrow{a}_{\mathrm{c}} \mu_t$ for which determinate schedulers suffice (cf. [3, Prop. 1]): in Section 3 we will show how to find them in polynomial time. The definition of bisimulation can be reformulated as follows, by simple manipulation of quantifiers:

▶ **Definition 2.** Given two *PA*s $\mathcal{A}_1$, $\mathcal{A}_2$, an equivalence relation $\mathcal{R}$ on $S_1 \uplus S_2$ is a *weak probabilistic bisimulation* if, for each transition $(s, a, \mu_s) \in D_1 \uplus D_2$ and each state t such that $s \mathrel{\mathcal{R}} t$, there exists μ_t such that $t \xRightarrow{a}_{\mathrm{c}} \mu_t$ and $\mu_s \mathrel{\mathcal{L}(\mathcal{R})} \mu_t$.

3 Weak Transition Construction as a Linear Programming Problem

We now discuss key elements of a decision algorithm for weak probabilistic bisimilarity. As we will see in Section 4, the core ingredient - and the source of the exponential complexity of the decision algorithm of [3] - is the recurring need to verify the step condition, that is, given a challenging transition $s \xrightarrow{a} \mu$ and $(s, t) \in \mathcal{R}$, to check whether there exists $t \xRightarrow{a}_{\mathrm{c}} \mu_t$ such that $\mu \mathrel{\mathcal{L}(\mathcal{R})} \mu_t$.

With some inspiration from network flow problems, we will be able to see a transition $t \xRightarrow{a}_{\mathrm{c}} \mu_t$ of the *PA* $\mathcal{A}$ as a *flow* where the initial probability mass δ_t flows and splits along internal transitions (and exactly one transition with label a for each stream when $a \neq \tau$) accordingly to the transition target distributions and the resolution of the nondeterminism performed by the scheduler.

This will allow us to arrive at a polynomial time algorithm to verify or refute the existence of a weak combined transition $t \xRightarrow{a}_{\mathrm{c}} \mu_t$ such that $\mu \mathrel{\mathcal{L}(\mathcal{R})} \mu_t$.

3.1 Allowed Transitions

For the construction we are going to develop, we consider a more general case where we parametrize the scheduler so as to choose only specific, allowed, transitions when resolving the nondeterministic choices in a weak combined transition. This generalization will later be exploited by enabling us to generate tailored and thereby smaller LP-problems. For the intuition of this generalization, consider, for example, an automaton $\mathcal{C}$ that models a communication channel: it receives the information to transmit from the sender through an external action, then it performs an internal transition to represent the sending of the message on the communication channel, and finally it sends the transmitted information to the receiver. The communication channel is chosen nondeterministically between a reliable channel and an acknowledged lossy channel. If we want to check whether $\mathcal{C}$ always ensures the correct transmission of the received information, we can restrict the scheduler to choose only the lossy channel, i.e., we *allow* only the transitions relative to the lossy channel; if we impose this restriction and $\mathcal{C}$ is able to send eventually the transmitted information to the receiver with probability 1, then we can say that $\mathcal{C}$ always ensures the correct transmission of the received information.

▶ **Definition 3** (Allowed weak combined transition)**.** Given a *PA* $\mathcal{A}$ and a set of *allowed* transitions $A \subseteq D$, we say that there is an *allowed weak combined transition* from s to μ with label a respecting A, denoted by $s \xRightarrow{a}{}^{A}_{\mathrm{c}} \mu$, if there exists a scheduler σ that induces $s \xRightarrow{a}_{\mathrm{c}} \mu$ such that for each $\alpha \in \mathit{frags}^*(\mathcal{A})$, $\mathrm{Supp}(\sigma(\alpha)) \subseteq A$.

It is immediate to see that, when we consider every transition as allowed, i.e., $A = D$, the allowed weak combined transition $s \xRightarrow{a}{}_{\mathrm{c}}^{D} \mu$ is just the usual transition $s \xRightarrow{a}{}_{\mathrm{c}} \mu$.

▶ **Proposition 4.** *Given a PA $\mathcal{A}$, a state s, and action a, and a probability distribution $\mu \in \mathrm{Disc}(S)$, there exists a scheduler σ_D for $\mathcal{A}$ that induces $s \xRightarrow{a}{}_{\mathrm{c}}^{D} \mu$ if and only if there exists a scheduler σ for $\mathcal{A}$ that induces $s \xRightarrow{a}{}_{\mathrm{c}} \mu$.*

Similarly, we say that there is an *allowed hyper-transition* from a distribution over states ρ to a distribution over states μ labeled by a respecting A, denoted by $\rho \xRightarrow{a}{}_{\mathrm{c}}^{A} \mu$, if there exists a family of allowed weak combined transitions $\{s \xRightarrow{a}{}_{\mathrm{c}}^{A} \mu_s\}_{s \in \mathrm{Supp}(\rho)}$ such that $\mu = \sum \rho(s) \cdot \mu_s$.

An equivalent definition of allowed hyper-transition $\rho \xRightarrow{a}{}_{\mathrm{c}}^{A} \mu$ is the following: given a *PA* $\mathcal{A}$, we say that there is an *allowed hyper-transition* from a distribution over states ρ to a distribution over states μ labeled by a respecting A if there exists an allowed weak combined transition $h \xRightarrow{a}{}_{\mathrm{c}}^{A_h} \mu$ for the *PA* $\mathcal{A}_h = (S \cup \{h\}, \bar{s}, \Sigma, D \cup \{h \xrightarrow{\tau} \rho\})$ where $h \notin S$ and $A_h = A \cup \{h \xrightarrow{\tau} \rho\}$.

▶ **Proposition 5.** *Given a PA $\mathcal{A}$, $h \notin S$, $a \in \Sigma$, $A \subseteq D$, and $\rho, \mu \in \mathrm{Disc}(S)$, let $\mathcal{A}_h$ be the PA $\mathcal{A}_h = (S \cup \{h\}, \bar{s}, \Sigma, D \cup \{h \xrightarrow{\tau} \rho\})$ and A_h be $A \cup \{h \xrightarrow{\tau} \rho\}$.*

$\rho \xRightarrow{a}{}_{\mathrm{c}}^{A} \mu$ exists in $\mathcal{A}$ if and only if $h \xRightarrow{a}{}_{\mathrm{c}}^{A_h} \mu$ exists in $\mathcal{A}_h$.

▶ **Example 1 (cont.).** If we consider again the automaton $\mathcal{E}$ in Figure 1 and the set of allowed transitions $A = D \setminus \{t \xrightarrow{\tau} \delta_{\bar{s}}\}$, it is immediate to see that $\bar{s} \xRightarrow{a}{}_{\mathrm{c}} \mu$ with $\mu = \{\frac{1}{16}$▲, $\frac{5}{16}$■, $\frac{10}{16}$⬟$\}$ is not an allowed weak combined transition respecting A and that the only allowed weak combined transition with label a enabled by $\bar{s}$ is $\bar{s} \xRightarrow{a}{}_{\mathrm{c}}^{A} \rho$ having $\rho = \{\frac{1}{4}$▲, $\frac{1}{4}$■, $\frac{1}{2}$⬟$\}$ as target distribution.

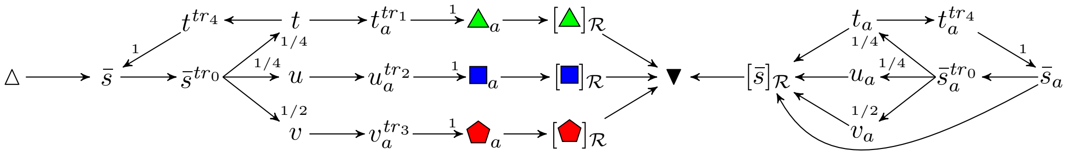
3.2 A Linear Programming Problem

We now assume we are given the *PA* $\mathcal{A}$, the set of allowed transitions $A \subseteq D$, the state t, the action a, the probability distribution μ, and the equivalence relation $\mathcal{R}$ on S. We intend to verify or refute the existence of a weak combined transition $t \xRightarrow{a}{}_{\mathrm{c}}^{A} \mu_t$ of $\mathcal{A}$ satisfying $\mu \, \mathcal{L}(\mathcal{R}) \, \mu_t$ via the construction of a flow through the network graph $G(t, a, \mu, A, \mathcal{R}) = (V, E)$ defined as follows: for $a \neq \tau$, the set of vertices is $V = \{\vartriangle, \blacktriangledown\} \cup S \cup S^{tr} \cup S_a \cup S_a^{tr} \cup (S/\mathcal{R})$ where $S^{tr} = \{\, v^{tr} \mid tr = v \xrightarrow{b} \rho \in A, b \in \{a, \tau\} \,\}$, $S_a = \{\, v_a \mid v \in S \,\}$ and $S_a^{tr} = \{\, v_a^{tr} \mid v^{tr} \in S^{tr} \,\}$ and the set of arcs is $E = \{(\vartriangle, t)\} \cup \{\, (v_a, \mathcal{C}), (\mathcal{C}, \blacktriangledown) \mid \mathcal{C} \in S/\mathcal{R}, v \in \mathcal{C} \,\} \cup \{\, (v, v^{tr}), (v^{tr}, v'), (v_a, v_a^{tr}), (v_a^{tr}, v'_a) \mid tr = v \xrightarrow{\tau} \rho \in A, v' \in \mathrm{Supp}(\rho) \,\} \cup \{\, (v, v_a^{tr}), (v_a^{tr}, v'_a) \mid tr = v \xrightarrow{a} \rho \in A, v' \in \mathrm{Supp}(\rho) \,\}$. For $a = \tau$ the definition is similar: $V = \{\vartriangle, \blacktriangledown\} \cup S \cup S^{tr} \cup (S/\mathcal{R})$ and $E = \{(\vartriangle, t)\} \cup \{\, (v, \mathcal{C}), (\mathcal{C}, \blacktriangledown) \mid \mathcal{C} \in S/\mathcal{R}, v \in \mathcal{C} \,\} \cup \{\, (v, v^{tr}), (v^{tr}, v') \mid tr = v \xrightarrow{\tau} \rho \in A, v' \in \mathrm{Supp}(\rho) \,\}$.

$\vartriangle$ and $\blacktriangledown$ are two vertices that represent the source and the sink of the network, respectively. The graph encodes possible sequences of internal transitions, keeping track of which transition has happened by means of the vertices superscripted with tr, for this the set S^{tr} contains vertices that model the transitions of the automaton. The subsets of vertices subscripted by a are used to record that action a has happened already. Notably, not every vertex is used for defining arcs: the vertices v^{tr} where $tr = v \xrightarrow{b} \rho \in A$ and $b = a \neq \tau$ are used only to define the corresponding vertices v_a^{tr} that are actually involved in the definition of the set E of arcs. We could have removed these vertices from S^{tr} but this reduces the readability of the definition of S_a^{tr} without giving us a valuable effect on the computational complexity of the proposed solution.

▶ **Example 1 (cont.).** Consider the automaton $\mathcal{E}$ in Figure 1 and suppose that we want to check whether there exists an allowed weak combined transition $\bar{s} \xRightarrow{a}{}_{\mathrm{c}}^{D} \rho$ such that $\rho \, \mathcal{L}(\mathcal{R}) \, \mu$

where $\mu = \{\frac{1}{16}▲, \frac{5}{16}■, \frac{10}{16}⬟\}$ and the classes induced by $\mathcal{R}$ are $\{\{\bar{s}, t, u, v\}, \{▲\}, \{■\}, \{⬟\}\}$. Let $tr_0 = \bar{s} \xrightarrow{\tau} \{\frac{1}{4}t, \frac{1}{4}u, \frac{1}{2}v\}$, $tr_1 = t \xrightarrow{a} \delta_{▲}$, $tr_2 = u \xrightarrow{a} \delta_{■}$, $tr_3 = v \xrightarrow{a} \delta_{⬟}$, and $tr_4 = t \xrightarrow{\tau} \delta_{\bar{s}}$. The network $G(\bar{s}, a, \mu, D, \mathcal{R})$ is as follows, where we omit vertices ▲, ■, and ⬟ since they are not involved in any arc. Numbers attached to arcs indicate probabilities, and are not part of the graph.



Our intention is to use the network $G(t, a, \mu, A, \mathcal{R})$, in a maximum flow problem, since solving the latter has polynomial complexity. Unfortunately, the resulting problem does not model an allowed weak combined transition because probabilities are as such not necessarily respected: in ordinary flow problems we can not enforce a proportional balancing between the flows out of a given vertex. Instead, the entire incoming flow might be sent over a single outgoing arc, provided that the arc capacity is respected, while zero flow is sent over other arcs. In particular, we have no way to force the flow to split proportionally to the target probability distribution of a transition when the flow is less than 1. Apart from that, there is no obvious way to assign arc capacities since imposing capacity 1 to arcs is not always correct even if this is the maximum value for a probability. This problem is specifically caused by cycles of internal transitions. For self loops like $s \xrightarrow{\tau} \rho$ with $\rho(s) > 0$, one might after some reflection come up with a capacity $1/(1-p)$ where $p = \rho(s)$, but this does not extend to arbitrary τ-connected components.

For these reasons, we have to proceed differently: since any maximum flow problem can be expressed as a Linear Programming (LP) problem, we follow this path, but then refine the LP problem further, in order to eventually define a maximization problem whose solution is indeed equivalent to an allowed weak combined transition, as we will show in Section 3.4. For this, we use the above transformation of the automaton into a network graph as the starting point for generating an LP problem, which is afterwards enriched with additional constraints: we adopt the same notation of the max flow problem so we use $f_{u,v}$ to denote the "flow" through the arc from u to v. The *balancing factor* is a new concept we introduce to model a probabilistic choice and to ensure a balancing between flows that leave a vertex representing a probabilistic choice, i.e., leaving a vertex $v \in S^{tr} \cup S_a^{tr}$.

▶ **Definition 6** (The $t \xRightarrow{a}{}_{\mathrm{c}}^{A} \diamond \mathcal{L}(\mathcal{R})\ \mu$ LP problem)**.** For $a \neq \tau$ we define the LP problem $t \xRightarrow{a}{}_{\mathrm{c}}^{A} \diamond \mathcal{L}(\mathcal{R})\ \mu$ associated to the network graph $(V, E) = G(t, a, \mu, A, \mathcal{R})$ as follows.

$$\max \sum\nolimits_{(x,y)\in E} -f_{x,y}$$

subject to

$f_{u,v} \geq 0$	for each $(u, v) \in E$
$f_{\triangle,t} = 1$	
$f_{\mathcal{C},▼} = \mu(\mathcal{C})$	for each $\mathcal{C} \in S/\mathcal{R}$
$\sum_{u\in\{\,x\mid(x,v)\in E\,\}} f_{u,v} - \sum_{u\in\{\,y\mid(v,y)\in E\,\}} f_{v,u} = 0$	for each $v \in V \setminus \{\triangle, ▼\}$
$f_{v^{tr},v'} - \rho(v') \cdot f_{v,v^{tr}} = 0$	for each $tr = v \xrightarrow{\tau} \rho \in A$ and $v' \in \mathrm{Supp}(\rho)$
$f_{v_a^{tr},v'_a} - \rho(v') \cdot f_{v_a,v_a^{tr}} = 0$	for each $tr = v \xrightarrow{\tau} \rho \in A$ and $v' \in \mathrm{Supp}(\rho)$
$f_{v_a^{tr},v'_a} - \rho(v') \cdot f_{v,v_a^{tr}} = 0$	for each $tr = v \xrightarrow{a} \rho \in A$ and $v' \in \mathrm{Supp}(\rho)$

When a is τ, the LP problem $t \xRightarrow{\tau}{}_{\mathrm{c}}^{A} \diamond \mathcal{L}(\mathcal{R})\ \mu$ associated to $G(t, \tau, \mu, A, \mathcal{R})$ is defined as above without the last two groups of constraints. Note that $t \xRightarrow{a}{}_{\mathrm{c}}^{A} \diamond \mathcal{L}(\mathcal{R})\ \mu$ constraints define a system of linear equations extended with the non-negativity of variables $f_{u,v}$ and this rules out solutions where some variable $f_{x,y}$ has an infinite value. Moreover this may be used to improve the actual implementation of the solver. It is worthwhile to point out that the objective function $\max \sum_{(x,y)\in E} -f_{x,y}$ is actually equivalent to $\min \sum_{(x,y)\in E} f_{x,y}$, i.e., a weak transition can also be seen as a minimum cost flow problem plus balancing constraints.

We can define the objective function in several ways but this does not affect the equivalence of $t \overset{a}{\Longrightarrow}{}_{\mathrm{c}}^{A} \diamond \mathcal{L}(\mathcal{R})\ \mu$ and allowed weak combined transitions: in fact, the equivalence is based on variables $f_{v_a,[v]_{\mathcal{R}}}$ and $f_{\mathcal{C},\blacktriangledown}$ (where $v \in S$ and $\mathcal{C} \in S/\mathcal{R}$) that represent the probability to reach each state v (and then stopping) and each equivalence class $\mathcal{C}$, respectively; by definition of $t \overset{a}{\Longrightarrow}{}_{\mathrm{c}}^{A} \diamond \mathcal{L}(\mathcal{R})\ \mu$ we have that $\sum_{v\in\mathcal{C}} f_{v_a,\mathcal{C}} = f_{\mathcal{C},\blacktriangledown}$ and $f_{\mathcal{C},\blacktriangledown} = \mu(\mathcal{C})$, thus their value does not strictly depend on the objective function. When $a = \tau$, we have the same result, just replacing v_a with v.

The objective function we use allows us to rule out trivial self-loops: suppose that there exists a transition $tr = x \xrightarrow{\tau} \delta_x \in A$ that we model by arcs (x, x^{tr}) and (x^{tr}, x). The balancing constraint for such arcs is $f_{x^{tr},x} - 1 \cdot f_{x,x^{tr}} = 0$ that is satisfied for each value of $f_{x^{tr},x} = f_{x,x^{tr}}$; however, the maximum for the objective function can be reached only when $f_{x,x^{tr}} = 0$, that is, the self-loop is not used. Similarly, we obtain that the value of the flow involving vertices that can not be reached from the vertex t is null as well as when such vertices may be reached from t but the solution of the problem requires that the flow from the vertex t to them is null.

▶ **Example 1 (cont.).** Consider again the automaton $\mathcal{E}$ in Figure 1 and suppose that we want to check whether there exists an allowed weak combined transition $\bar{s} \overset{a}{\Longrightarrow}{}_{\mathrm{c}}^{D} \rho$ such that $\rho\ \mathcal{L}(\mathcal{R})\ \mu$ where $\mu = \{\frac{1}{16}\blacktriangle, \frac{5}{16}\blacksquare, \frac{10}{16}\text{⬟}\}$ and the only non-singleton class of $\mathcal{R}$ is $\{\bar{s}, t, u, v\}$. Let $tr_0 = \bar{s} \xrightarrow{\tau} \{\frac{1}{4}t, \frac{1}{4}u, \frac{1}{2}v\}$, $tr_1 = t \xrightarrow{a} \delta_{\blacktriangle}$, $tr_2 = u \xrightarrow{a} \delta_{\blacksquare}$, $tr_3 = v \xrightarrow{a} \delta_{\text{⬟}}$, and $tr_4 = t \xrightarrow{\tau} \delta_{\bar{s}}$.

Besides other constraints, the LP problem $\bar{s} \overset{a}{\Longrightarrow}{}_{\mathrm{c}}^{D} \diamond \mathcal{L}(\mathcal{R})\ \mu$ has the following constraints:

$$\begin{array}{lll}
f_{\triangle,\bar{s}} = 1 & f_{[\blacktriangle]_{\mathcal{R}},\blacktriangledown} = 1/16 & f_{[\blacksquare]_{\mathcal{R}},\blacktriangledown} = 5/16 \\
f_{[\text{⬟}]_{\mathcal{R}},\blacktriangledown} = 10/16 & f_{\bar{s},\bar{s}^{tr_0}} - f_{\bar{s}^{tr_0},t} - f_{\bar{s}^{tr_0},u} - f_{\bar{s}^{tr_0},v} = 0 & f_{\triangle,\bar{s}} + f_{t^{tr_4},\bar{s}} - f_{\bar{s},\bar{s}^{tr_0}} = 0 \\
f_{\bar{s}^{tr_0},t} - f_{t,t_a^{tr_1}} - f_{t,t^{tr_4}} = 0 & f_{\bar{s}^{tr_0},u} - f_{u,u_a^{tr_2}} = 0 & f_{\bar{s}^{tr_0},v} - f_{v,v_a^{tr_3}} = 0 \\
f_{t,t_a^{tr_1}} - f_{t_a^{tr_1},\blacktriangle_a} = 0 & f_{u,u_a^{tr_2}} - f_{u_a^{tr_2},\blacksquare_a} = 0 & f_{v,v_a^{tr_3}} - f_{v_a^{tr_3},\text{⬟}_a} = 0 \\
f_{t,t^{tr_4}} - f_{t^{tr_4},\bar{s}} = 0 & f_{t_a^{tr_1},\blacktriangle_a} - f_{\blacktriangle_a,[\blacktriangle]_{\mathcal{R}}} = 0 & f_{u_a^{tr_2},\blacksquare_a} - f_{\blacksquare_a,[\blacksquare]_{\mathcal{R}}} = 0 \\
f_{v_a^{tr_3},\text{⬟}_a} - f_{\text{⬟}_a,[\text{⬟}]_{\mathcal{R}}} = 0 & f_{\blacktriangle_a,[\blacktriangle]_{\mathcal{R}}} - f_{[\blacktriangle]_{\mathcal{R}},\blacktriangledown} = 0 & f_{\blacksquare_a,[\blacksquare]_{\mathcal{R}}} - f_{[\blacksquare]_{\mathcal{R}},\blacktriangledown} = 0 \\
f_{\text{⬟}_a,[\text{⬟}]_{\mathcal{R}}} - f_{[\text{⬟}]_{\mathcal{R}},\blacktriangledown} = 0 & f_{\bar{s}^{tr_0},t} - 1/4 f_{\bar{s},\bar{s}^{tr_0}} = 0 & f_{\bar{s}^{tr_0},u} - 1/4 f_{\bar{s},\bar{s}^{tr_0}} = 0 \\
f_{\bar{s}^{tr_0},v} - 1/2 f_{\bar{s},\bar{s}^{tr_0}} = 0 & f_{t_a^{tr_1},\blacktriangle_a} - 1 f_{t,t_a^{tr_1}} = 0 & f_{u_a^{tr_2},\blacksquare_a} - 1 f_{u,u_a^{tr_2}} = 0 \\
f_{v_a^{tr_3},\text{⬟}_a} - 1 f_{v,v_a^{tr_3}} = 0 & f_{t^{tr_4},\bar{s}} - 1 f_{t,t^{tr_4}} = 0 &
\end{array}$$

A solution that maximizes the objective function sets all variables to value 0 except for

$$\begin{array}{llllllll}
f_{\triangle,\bar{s}} & = 16/16 & f_{[\blacktriangle]_{\mathcal{R}},\blacktriangledown} & = 1/16 & f_{[\blacksquare]_{\mathcal{R}},\blacktriangledown} & = 5/16 & f_{[\text{⬟}]_{\mathcal{R}},\blacktriangledown} & = 10/16 \\
f_{\bar{s},\bar{s}^{tr_0}} & = 20/16 & f_{\bar{s}^{tr_0},t} & = 5/16 & f_{\bar{s}^{tr_0},u} & = 5/16 & f_{\bar{s}^{tr_0},v} & = 10/16 \\
f_{t,t_a^{tr_1}} & = 1/16 & f_{t,t^{tr_4}} & = 4/16 & f_{u,u_a^{tr_2}} & = 5/16 & f_{v,v_a^{tr_3}} & = 10/16 \\
f_{t_a^{tr_1},\blacktriangle_a} & = 1/16 & f_{t^{tr_4},\bar{s}} & = 4/16 & f_{u_a^{tr_2},\blacksquare_a} & = 5/16 & f_{v_a^{tr_3},\text{⬟}_a} & = 10/16 \\
f_{\blacktriangle_a,[\blacktriangle]_{\mathcal{R}}} & = 1/16 & f_{\blacksquare_a,[\blacksquare]_{\mathcal{R}}} & = 5/16 & f_{\text{⬟}_a,[\text{⬟}]_{\mathcal{R}}} & = 10/16 & &
\end{array}$$

The variable $f_{\bar{s},\bar{s}^{tr_0}} = 20/16$ is part of a cycle and its value is greater than 1, confirming that 1, the maximum probability, in general is not a proper value for arc capacities.

3.3 Complexity of the LP Problem

We analyze the complexity of $t \overset{a}{\Longrightarrow}{}_{\mathrm{c}}^{A} \diamond \mathcal{L}(\mathcal{R})\ \mu$ for $a \neq \tau$ since $t \overset{\tau}{\Longrightarrow}{}_{\mathrm{c}}^{A} \diamond \mathcal{L}(\mathcal{R})\ \mu$ is just a special case of $t \overset{a}{\Longrightarrow}{}_{\mathrm{c}}^{A} \diamond \mathcal{L}(\mathcal{R})\ \mu$.

Given the automaton $\mathcal{A}$ and the set $A \subseteq D$ of allowed transitions, let $N_S = |S|$, $N_A = |A|$, and $N = \max\{N_S, N_A\}$. Suppose that $a \neq \tau$ and consider the network graph $G(t, a, \mu, A, \mathcal{R}) = (V, E)$. The cardinality of V is: $|V| \leq 2+N_S+N_A+N_S+N_A+N_S \in \mathcal{O}(N)$ and the cardinality of E is: $|E| \leq 1+2N_S+2(N_S+1)N_A+(N_S+1)N_A \in \mathcal{O}(N^2)$. Note that this is also the cost of generating the $G(t, a, \mu, A, \mathcal{R})$ network graph from the automaton $\mathcal{A}$.

Now, consider the $t \overset{a}{\Longrightarrow}{}^{A}_{\mathrm{c}} \diamond \mathcal{L}(\mathcal{R})\ \mu$ LP problem: the number of variables is $|\{\, f_{u,v} \mid (u, v) \in E \,\}| = |E| \in \mathcal{O}(N^2)$ and the number of constraints is $|E| + 1 + N_S + N_S N_A + N_S N_A + N_S N_A + |V| - 2 \in \mathcal{O}(N^2)$, so generating $t \overset{a}{\Longrightarrow}{}^{A}_{\mathrm{c}} \diamond \mathcal{L}(\mathcal{R})\ \mu$ is polynomial in N. Since there exist polynomial algorithms for solving LP problems [26], solving the $t \overset{a}{\Longrightarrow}{}^{A}_{\mathrm{c}} \diamond \mathcal{L}(\mathcal{R})\ \mu$ problem is polynomial in N. The implementation of $t \overset{a}{\Longrightarrow}{}^{A}_{\mathrm{c}} \diamond \mathcal{L}(\mathcal{R})\ \mu$ can be optimized in several ways (cf. [15, Sec. 3.4]) without changing the resulting complexity class.

▶ **Theorem 7.** *Given a PA $\mathcal{A}$, an equivalence relation $\mathcal{R}$ on S, an action a, a probability distribution $\mu \in \mathrm{Disc}(S)$, a set of allowed transitions $A \subseteq D$, and a state $t \in S$, consider the problem $t \overset{a}{\Longrightarrow}{}^{A}_{\mathrm{c}} \diamond \mathcal{L}(\mathcal{R})\ \mu$ as defined above. Let $N = \max\{|S|, |A|\}$.*

Generating and checking the existence of a valid solution of the $t \overset{a}{\Longrightarrow}{}^{A}_{\mathrm{c}} \diamond \mathcal{L}(\mathcal{R})\ \mu$ LP problem is polynomial in N.

3.4 Equivalence of LP Problems and Weak Transitions

In this section we present the main theorem that equates $t \overset{a}{\Longrightarrow}{}^{A}_{\mathrm{c}} \diamond \mathcal{L}(\mathcal{R})\ \mu$ with an allowed weak combined transition, that is, $t \overset{a}{\Longrightarrow}{}^{A}_{\mathrm{c}} \diamond \mathcal{L}(\mathcal{R})\ \mu$ has a solution if and only if there exists a scheduler σ for $\mathcal{A}$ that induces $t \overset{a}{\Longrightarrow}{}^{A}_{\mathrm{c}} \mu_t$ such that $\mu\ \mathcal{L}(\mathcal{R})\ \mu_t$. This result easily extends to ordinary weak combined transitions and hyper-transitions.

▶ **Theorem 8.** *Given a PA $\mathcal{A}$, an equivalence relation $\mathcal{R}$ on S, an action a, a probability distribution $\mu \in \mathrm{Disc}(S)$, a set of allowed transitions $A \subseteq D$, and a state $t \in S$, consider the problem $t \overset{a}{\Longrightarrow}{}^{A}_{\mathrm{c}} \diamond \mathcal{L}(\mathcal{R})\ \mu$ as defined above.*

$t \overset{a}{\Longrightarrow}{}^{A}_{\mathrm{c}} \diamond \mathcal{L}(\mathcal{R})\ \mu$ has a solution f^* such that $f^*_{\mathcal{C},\blacktriangledown} = \mu(\mathcal{C})$ for each $\mathcal{C} \in S/\mathcal{R}$ if and only if there exists a scheduler σ for $\mathcal{A}$ that induces $t \overset{a}{\Longrightarrow}{}^{A}_{\mathrm{c}} \mu_t$ such that $\mu\ \mathcal{L}(\mathcal{R})\ \mu_t$.

Proof outline. The scheduler σ we define in the proof for the "only if" direction assigns to each execution fragment α with $last(\alpha) = v$ the sub-probability distribution over transitions defined, for each transition $tr \in A$ such that $src(tr) = v$, as the ratio $f^*_{v_{\mathtt{t}}, v^{tr}_{\bar{\mathtt{t}}}} / \vec{f}^*_{v_{\mathtt{t}}}$, given that $\vec{f}^*_{v_{\mathtt{t}}} > 0$, where $\vec{f}^*_v$ is the total flow incoming v, $\mathtt{t} = trace(\alpha)$, and $\bar{\mathtt{t}}$ is the concatenation of $trace(\alpha)$ and of $trace(act(tr))$. The remaining probability of stopping in the state v is exactly $f^*_{v_{\mathtt{t}}, [v]_{\mathcal{R}}} / \vec{f}^*_{v_{\mathtt{t}}}$. The way we generate the network $G(t, a, \mu, A, \mathcal{R})$ ensures that $f^*_{v_{\mathtt{t}}, v^{tr}_{\bar{\mathtt{t}}}} = 0$ when $\mathtt{t} \notin \{\varepsilon, trace(a)\}$ and that $f^*_{v_{\mathtt{t}}, [v]_{\mathcal{R}}} / \vec{f}^*_{v_{\mathtt{t}}} = 0$ when $\mathtt{t} \neq trace(a)$. The proof for the "if" direction is the dual, that is, we define a feasible solution f^* as the sum of the probabilities of the cones of execution fragments, i.e., $\vec{f}^*_{v_b} = \sum_{\alpha \in \{\, \phi \in frags^*(\mathcal{A}) \mid trace(\phi) = b \wedge last(\phi) = v \,\}} \mu_{\sigma,t}(C_\alpha)$; then the existence of such feasible solution is enough to prove that there exists a (possibly different) solution f^o that maximizes the objective function while preserving the property that for each $\mathcal{C} \in S/\mathcal{R}$, $f^o_{\mathcal{C},\blacktriangledown} = \mu(\mathcal{C})$. ◀

It is worth to observe that the resulting scheduler is a determinate scheduler and an immediate corollary of this theorem confirming and improving Proposition 3 of [3] is that each scheduler inducing $t \overset{a}{\Longrightarrow}{}^{A}_{\mathrm{c}} \mu_t$ can be replaced by a determinate scheduler inducing $t \overset{a}{\Longrightarrow}{}^{A}_{\mathrm{c}} \mu_t$ as well.

▶ **Example 1 (cont.).** We already know from the first part of this example that there exists a non-determinate scheduler σ inducing $\bar{s} \overset{a}{\Longrightarrow}{}^{D}_{\mathrm{c}} \mu$ where $\mu = \{\frac{1}{16}$▲, $\frac{5}{16}$■, $\frac{10}{16}$⬟$\}$. So, let again $tr_0 = \bar{s} \xrightarrow{\tau} \{\frac{1}{4}t, \frac{1}{4}u, \frac{1}{2}v\}$, $tr_1 = t \xrightarrow{a} \delta_{\blacktriangle}$, $tr_2 = u \xrightarrow{a} \delta_{\blacksquare}$, $tr_3 = v \xrightarrow{a} \delta_{\text{⬟}}$, and $tr_4 = t \xrightarrow{\tau} \delta_{\bar{s}}$. Theorem 8 ensures that there exists a scheduler σ', possibly different from σ, that induces $\bar{s} \overset{a}{\Longrightarrow}{}_{\mathrm{c}}^{D} \mu$; in particular, σ' is the determinate scheduler defined as follows.

$$\sigma'(\alpha) = \begin{cases} \delta_{tr_0} & \text{if } \mathit{trace}(\alpha) = \varepsilon \text{ and } \mathit{last}(\alpha) = \bar{s}; \\ \{\frac{1}{5}tr_1, \frac{4}{5}tr_4\} & \text{if } \mathit{trace}(\alpha) = \varepsilon \text{ and } \mathit{last}(\alpha) = t; \\ \delta_{tr_2} & \text{if } \mathit{trace}(\alpha) = \varepsilon \text{ and } \mathit{last}(\alpha) = u; \\ \delta_{tr_3} & \text{if } \mathit{trace}(\alpha) = \varepsilon \text{ and } \mathit{last}(\alpha) = v; \\ \delta_{\perp} & \text{otherwise.} \end{cases}$$

It is straightforward to check that σ' actually induces $\bar{s} \overset{a}{\Longrightarrow}{}_{\mathrm{c}}^{D} \mu$. For instance, state $\blacktriangle$ is reached with probability $\mu_{\sigma',\bar{s}}(\{\, \alpha \in \mathit{frags}^*(\mathcal{E}) \mid \mathit{last}(\alpha) = \blacktriangle \,\}) = \mu_{\sigma',\bar{s}}(\{\, \bar{s}\tau t(\tau\bar{s}\tau t)^n a \blacktriangle \mid n \in \mathbb{N} \,\}) = 1 \cdot \frac{1}{4} \cdot \sum_{n \in \mathbb{N}} (\frac{4}{5} \cdot 1 \cdot 1 \cdot \frac{1}{4})^n \cdot \frac{1}{5} \cdot 1 \cdot 1 = \frac{1}{4} \cdot \frac{1}{5} \cdot (1 - \frac{1}{5})^{-1} = \frac{1}{4} \cdot \frac{1}{5} \cdot \frac{5}{4} = \mu(\blacktriangle)$, as required.

▶ **Corollary 9.** *Given a PA $\mathcal{A}$, $t \in S$ and $h \notin S$, $a \in \Sigma$, $\rho, \mu, \mu_t \in \mathrm{Disc}(S)$, $A \subseteq D$, an equivalence relation $\mathcal{R}$ on S, a transition $h \xrightarrow{\tau} \rho$, $A_h = A \cup \{h \xrightarrow{\tau} \rho\}$, $D_h = D \cup \{h \xrightarrow{\tau} \rho\}$, and the PA $\mathcal{A}_h = (S \cup \{h\}, \bar{s}, \Sigma, D_h)$, the following holds:*

1. *$t \overset{a}{\Longrightarrow}{}_{\mathrm{c}}^{D} \diamond \mathcal{L}(\mathcal{R})\ \mu$ has a solution f^* such that $f^*_{\mathcal{C},\blacktriangledown} = \mu(\mathcal{C})$ for each $\mathcal{C} \in S/\mathcal{R}$ if and only if there exists a scheduler σ for $\mathcal{A}$ inducing $t \overset{a}{\Longrightarrow}_{\mathrm{c}} \mu_t$ such that $\mu\ \mathcal{L}(\mathcal{R})\ \mu_t$;*
2. *$h \overset{a}{\Longrightarrow}{}_{\mathrm{c}}^{A_h} \diamond \mathcal{L}(\mathcal{R})\ \mu$ ($h \overset{a}{\Longrightarrow}{}_{\mathrm{c}}^{D_h} \diamond \mathcal{L}(\mathcal{R})\ \mu$) relative to $\mathcal{A}_h$ has a solution f^* such that $f^*_{\mathcal{C},\blacktriangledown} = \mu(\mathcal{C})$ for each $\mathcal{C} \in S/\mathcal{R}$ if and only if there exists a scheduler σ for $\mathcal{A}$ inducing $\rho \overset{a}{\Longrightarrow}{}_{\mathrm{c}}^{A} \mu_t$ ($\rho \overset{a}{\Longrightarrow}_{\mathrm{c}} \mu_t$, respectively) such that $\mu_t\ \mathcal{L}(\mathcal{R})\ \mu$.*

When $\mathcal{R}$ is the identity relation $\mathcal{I}$, $\mu\ \mathcal{L}(\mathcal{I})\ \mu_t$ implies $\mu_t = \mu$.

Proof outline. The corollary follows directly from a combination of Theorem 8 for the equivalence between the LP problem and allowed weak combined transition, Proposition 4 for weak combined transitions, and Proposition 5 for hyper-transitions. ◀

It is worth to note also that, when we consider an *MDP* as a *PA*, the $t \overset{a}{\Longrightarrow}{}_{\mathrm{c}}^{A} \diamond \mathcal{L}(\mathcal{R})\ \mu$ construction allows us to solve the multi-objective reachability problem on *MDP*s, that is, given an *MDP* and a collection of disjoint target sets $E_1, \ldots, E_k$ and goal probabilities $p_1, \ldots, p_k$, check whether there exists a scheduler that reaches E_i with probability exactly p_i.

3.5 Equivalence Matching

Theorem 8 and its corollary allow us to check in polynomial time whether it is possible to reach a given probability distribution μ from a state t or a probability distribution ρ. We now consider a more general case where, given a *PA* $\mathcal{A}$, two distributions $\rho_1, \rho_2 \in \mathrm{Disc}(S)$, two actions $a_1, a_2 \in \Sigma$, two sets $A_1, A_2 \subseteq D$ of allowed transitions, and an equivalence relation $\mathcal{R}$ on S, we want to check in polynomial time whether there exist $\mu_1, \mu_2 \in \mathrm{Disc}(S)$ such that $\rho_1 \overset{a_1}{\Longrightarrow}{}_{\mathrm{c}}^{A_1} \mu_1$, $\rho_2 \overset{a_2}{\Longrightarrow}{}_{\mathrm{c}}^{A_2} \mu_2$, and $\mu_1\ \mathcal{L}(\mathcal{R})\ \mu_2$. In order to find μ_1 and μ_2, we can consider a family $\{p_{\mathcal{C}}\}_{\mathcal{C} \in S/\mathcal{R}}$ of non-negative values such that $\sum_{\mathcal{C} \in S/\mathcal{R}} p_{\mathcal{C}} = 1$ and a probability distribution $\bar{\mu}$ satisfying $\bar{\mu}(\mathcal{C}) = p_{\mathcal{C}}$ for each $\mathcal{C} \in S/\mathcal{R}$ and then solve $\rho_1 \overset{a_1}{\Longrightarrow}{}_{\mathrm{c}}^{A_1} \diamond \mathcal{L}(\mathcal{R})\ \bar{\mu}$ and $\rho_2 \overset{a_2}{\Longrightarrow}{}_{\mathrm{c}}^{A_2} \diamond \mathcal{L}(\mathcal{R})\ \bar{\mu}$ where $\rho \overset{a}{\Longrightarrow}{}_{\mathrm{c}}^{A} \diamond \mathcal{L}(\mathcal{R})\ \mu$ is the problem $h \overset{a}{\Longrightarrow}{}_{\mathrm{c}}^{A_h} \diamond \mathcal{L}(\mathcal{R})\ \mu$ relative to $\mathcal{A}_h = (S \cup \{h\}, \bar{s}, \Sigma, D \cup \{h \xrightarrow{\tau} \rho\})$ with $h \notin S$ and $A_h = A \cup \{h \xrightarrow{\tau} \rho\}$. The main problem of this approach is to find a good family of values $p_{\mathcal{C}}$; since we do not care about actual values, we consider $p_{\mathcal{C}}$ as variables satisfying $p_{\mathcal{C}} \geq 0$ and $\sum_{\mathcal{C} \in S/\mathcal{R}} p_{\mathcal{C}} = 1$ and we define the LP problem $P_{1,2}$ derived from $P_1 = \rho_1 \overset{a_1}{\Longrightarrow}{}_{\mathrm{c}}^{A_1} \diamond \mathcal{L}(\mathcal{R})\ \bar{\mu}$ and $P_2 = \rho_2 \overset{a_2}{\Longrightarrow}{}_{\mathrm{c}}^{A_2} \diamond \mathcal{L}(\mathcal{R})\ \bar{\mu}$ as follows (after renaming of P_2 variables to avoid collisions): the objective function of $P_{1,2}$ is the sum of the objective functions of P_1 and P_2; the set of constraints of $P_{1,2}$ is $\sum_{\mathcal{C} \in S/\mathcal{R}} p_{\mathcal{C}} = 1$ together with $p_{\mathcal{C}} \geq 0$ for $\mathcal{C} \in S/\mathcal{R}$ and the union of the sets of constraints of P_1 and P_2 where each occurrence of $\bar{\mu}(\mathcal{C})$ is replaced by $p_{\mathcal{C}}$.

It is quite easy to verify that $P_{1,2}$ has a solution if and only if both P_1 and P_2 have a solution (with respect to the same $\bar{\mu}$) and thus, by Corollary 9(2), if and only if ρ_1 and ρ_2 enable an allowed hyper-transition to μ_1 and μ_2, respectively, such that $\mu_1\ \mathcal{L}(\mathcal{R})\ \mu_2$, as required. It is immediate to see that $P_{1,2}$ can still be solved in polynomial time, since it is just the union of P_1 and P_2 extended with at most $|S|$ variables and $2|S|$ constraints.

▶ **Proposition 10.** *Given a PA $\mathcal{A}$, $\rho_1, \rho_2 \in \mathrm{Disc}(S)$, $a_1, a_2 \in \Sigma$, two sets $A_1, A_2 \subseteq D$ of allowed transitions, and an equivalence relation $\mathcal{R}$ on S, the existence of $\mu_1, \mu_2 \in \mathrm{Disc}(S)$ such that $\rho_1 \xRightarrow{a_1}{}^{A_1}_{\mathrm{c}} \mu_1$, $\rho_2 \xRightarrow{a_2}{}^{A_2}_{\mathrm{c}} \mu_2$, and $\mu_1\ \mathcal{L}(\mathcal{R})\ \mu_2$ can be checked in polynomial time.*

The above proposition easily extends, by Corollary 9, to each combination of weak combined transitions, allowed hyper-transitions, and allowed weak combined transitions as well as to exact matching as induced by the identity relation $\mathcal{I}$.

4 Decision Procedure

In this section, we recast the decision procedure of [3] that decides whether two probabilistic automata $\mathcal{A}_1$ and $\mathcal{A}_2$ are bisimilar according to $\approx$, that is, whether $\mathcal{A}_1 \approx \mathcal{A}_2$, following the standard partition refinement approach [3, 17, 19, 21]. More precisely, procedure Quotient iteratively constructs the set $S/{\approx}$, the set of equivalence classes of states $S = S_1 \uplus S_2$ under $\approx$, starting with the partitioning $\mathcal{W} = \{S\}$ and refining it until $\mathcal{W}$ satisfies the definition of weak probabilistic bisimulation and thus the resulting partitioning is the coarsest one, i.e., we compute the weak probabilistic bisimilarity.

```
Quotient($\mathcal{A}_1, \mathcal{A}_2$)
$\mathcal{W} = \{S_1 \uplus S_2\}$;
$(\mathcal{C}, a, \mu) = \mathrm{FindSplit}(\mathcal{W})$;
while $\mathcal{C} \neq \emptyset$ do
    $\mathcal{W} = \mathrm{Refine}(\mathcal{W}, (\mathcal{C}, a, \mu))$;
    $(\mathcal{C}, a, \mu) = \mathrm{FindSplit}(\mathcal{W})$;
return $\mathcal{W}$
```

Deciding whether two automata are bisimilar then reduces to check whether their start states belong to the same class. In the following, we treat $\mathcal{W}$ both as a set of partitions and as an equivalence relation without further mentioning.

```
FindSplit($\mathcal{W}$)
1: for all $(s, a, \mu) \in D = D_1 \uplus D_2$ do
2:     for all $t \in [s]_{\mathcal{W}}$ do
3:         if $t \xRightarrow{a}{}^D_{c} \diamond \mathcal{L}(\mathcal{W})\ \mu$ has no solution
4:             return $([s]_{\mathcal{W}}, a, \mu)$
5: return $(\emptyset, \tau, \delta_{\bar{s}})$
```

The partitioning is refined by procedure Refine into a finer partitioning as long as there is a partition containing two states that violate the bisimulation condition, which is checked for in procedure FindSplit. Procedure Refine, that we do not provide explicitly as in [3], splits partition $\mathcal{C}$ into two new partitions according to the discriminating information $(\mathcal{C}, a, \mu)$ identified by FindSplit before. So far, the procedure is as the *DecideBisim*$(\mathcal{A}_1, \mathcal{A}_2)$ procedure proposed in [3].

The difference arises inside the procedure FindSplit, where we check directly the step condition by solving for each transition $s \xrightarrow{a} \mu$ the LP problem $t \xRightarrow{a}{}^D_{c} \diamond \mathcal{L}(\mathcal{W})\ \mu$ that has a solution, according to Corollary 9(1), if and only if there exists $t \xRightarrow{a}_{\mathrm{c}} \mu_t$ such that $\mu\ \mathcal{L}(\mathcal{W})\ \mu_t$.

4.1 Complexity Analysis of the Procedure

Given two PAs $\mathcal{A}_1$ and $\mathcal{A}_2$, let $S = S_1 \uplus S_2$, $D = D_1 \uplus D_2$, and $N = \max\{|S|, |D|\}$.

In the worst case (that occurs when the current $\mathcal{W}$ satisfies the step condition), the **for** at line 1 of procedure FindSplit is performed at most N times as well as the inner

for, so $t \xRightarrow{a}{}_{\mathrm{c}}^{D} \diamond \mathcal{L}(\mathcal{W})\ \mu$ is generated and solved at most N^2 times. Since by Theorem 7 generating and checking the existence of a valid solution for $t \xRightarrow{a}{}_{\mathrm{c}}^{D} \diamond \mathcal{L}(\mathcal{W})\ \mu$ is polynomial in N, this implies that also FindSplit is polynomial in N; more precisely, denoted by $p(N)$ the complexity of $t \xRightarrow{a}{}_{\mathrm{c}}^{D} \diamond \mathcal{L}(\mathcal{W})\ \mu$, FindSplit $\in \mathcal{O}(N^2 p(N))$. Note that we can improve the running time required to solve the $t \xRightarrow{a}{}_{\mathrm{c}}^{D} \diamond \mathcal{L}(\mathcal{W})\ \mu$ LP problem by replacing D with D' at line 3 of FindSplit where D' contains only transitions with label τ or a enabled by states reachable from t.

The **while** loop in the procedure Quotient can be performed at most N times; this happens when in each loop the procedure FindSplit returns $(\mathcal{C}, a, \mu)$ where $\mathcal{C} \neq \emptyset$, that is, not every pair of states in $\mathcal{C}$ satisfies the step condition. Since in each loop the procedure Refine splits such class $\mathcal{C}$ in two classes $\mathcal{C}_1$ and $\mathcal{C}_2$, after at most N loops every class contains a single state and the procedure FindSplit returns $(\emptyset, \tau, \delta_{\bar{s}})$ since each transition $s \xrightarrow{a} \mu_s$ is obviously matched by s itself. Since Refine and FindSplit are polynomial in N, also Quotient is polynomial in N, thus checking $\mathcal{A}_1 \approx \mathcal{A}_2$ is polynomial in N.

▶ **Theorem 11.** *Given two PAs $\mathcal{A}_1$ and $\mathcal{A}_2$, let $N = \max\{|S_1 \uplus S_2|, |D_1 \uplus D_2|\}$. Checking $\mathcal{A}_1 \approx \mathcal{A}_2$ is polynomial in N.*

It is as yet open whether checking $\mathcal{A}_1 \approx \mathcal{A}_2$ can be done in strong-polynomial time, while this is known for strong bisimulation.

5 Concluding Remarks

This paper has established a polynomial time decision algorithm for *PA* weak probabilistic bisimulation, closing the quest for an effective decision algorithm coined in [3]. The core innovation is a novel characterization of weak combined transitions as an LP problem, enabling us to check the existence of a weak combined transition in polynomial time. The algorithm can be exploited in an effective compositional minimization strategy for *PA* (or *MDP*) and potentially also for Markov automata. Furthermore, the LP approach we developed is readily extensible to related problems requiring to find a specific weak transition. Another area of immediate applicability concerns cost-related problems where transition costs may relate to power or resource consumption in *PA* or *MDP*.

Acknowledgments. The authors are grateful to Christian Eisentraut (Saarland University) for insightful discussions. This work has been supported by the DFG as part of the SFB/TR 14 “Automatic Verification and Analysis of Complex Systems” (AVACS), by the DFG/NWO Bilateral Research Programme ROCKS, and by the European Union Seventh Framework Programme under grant agreement no. 295261 (MEALS). Andrea Turrini has received support from the Cluster of Excellence “Multimodal Computing and Interaction” (MMCI), part of the German Excellence Initiative.

References

1 C. Baier and M. Stoelinga. Norm functions for probabilistic bisimulations with delays. In *FOSSACS*, vol. 1784 of *LNCS*, pages 1–16, 2000.

2 E. Böde, M. Herbstritt, H. Hermanns, S. Johr, T. Peikenkamp, R. Pulungan, J. Rakow, R. Wimmer, and B. Becker. Compositional dependability evaluation for STATEMATE. *IEEE Transactions on Software Engineering*, 35(2):274–292, 2009.

3 S. Cattani and R. Segala. Decision algorithms for probabilistic bisimulation. In *CONCUR*, vol. 2421 of *LNCS*, pages 371–385, 2002.

4 L. Cheung, M. Stoelinga, and F. W. Vaandrager. A testing scenario for probabilistic processes. *Journal of the ACM*, 54(6), 2007.

5 N. Coste, H. Hermanns, E. Lantreibecq, and W. Serwe. Towards performance prediction of compositional models in industrial GALS designs. In *CAV*, vol. 5643 of *LNCS*, pages 204–218, 2009.

6 Y. Deng and M. Hennessy. On the semantics of Markov automata. In *ICALP*, vol. 6756 of *LNCS*, pages 307–318, 2011.

7 C. Derman. *Finite State Markovian Decision Processes*. Academic Press, Inc., 1970.

8 C. Eisentraut, H. Hermanns, and L. Zhang. Concurrency and composition in a stochastic world. In *CONCUR*, vol. 6269 of *LNCS*, pages 21–39, 2010.

9 C. Eisentraut, H. Hermanns, and L. Zhang. On probabilistic automata in continuous time. In *LICS*, pages 342–351, 2010.

10 M.-A. Esteve, J.-P. Katoen, V. Y. Nguyen, B. Postma, and Y. Yushtein. Formal correctness, safety, dependability and performance analysis of a satellite. In *ICSE*, pages 1022–1031, 2012.

11 H. A. Hansson. *Time and Probability in Formal Design of Distributed Systems*. PhD thesis, Department of Computer Systems, Uppsala University, 1991.

12 B. R. Haverkort, M. Kuntz, A. Remke, S. Roolvink, and M. Stoelinga. Evaluating repair strategies for a water-treatment facility using Arcade. In *DSN*, pages 419–424, 2010.

13 H. Hermanns. *Interactive Markov Chains: The Quest for Quantified Quality*, vol. 2428 of *LNCS*. Springer Verlang, 2002.

14 H. Hermanns and J.-P. Katoen. Automated compositional Markov chain generation for a plain-old telephone system. *Science of Computer Programming*, 36(1):97–127, 2000.

15 H. Hermanns and A. Turrini. Deciding Probabilistic Automata weak bisimulation in polynomial time. Available at `http://arxiv.org/abs/1205.0376`.

16 A. Hinton, M. Z. Kwiatkowska, G. Norman, and D. Parker. PRISM: A tool for automatic verification of probabilistic systems. In *TACAS*, vol. 3920 of *LNCS*, pages 441–444, 2006.

17 P. C. Kanellakis and S. A. Smolka. CCS expressions, finite state processes, and three problems of equivalence. *Information and Computation*, 86(1):43–68, 1990.

18 K. G. Larsen and A. Skou. Bisimulation through probabilistic testing (preliminary report). In *POPL*, pages 344–352, 1989.

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

20 A. Parma and R. Segala. Axiomatization of trace semantics for stochastic nondeterministic processes. In QEST, pages 294–303, 2004.

21 A. Philippou, I. Lee, and O. Sokolsky. Weak bisimulation for probabilistic systems. In *CONCUR*, vol. 1877 of *LNCS*, pages 334–349, 2000.

22 R. Segala. *Modeling and Verification of Randomized Distributed Real-Time Systems*. PhD thesis, MIT, 1995.

23 R. Segala. Probability and nondeterminism in operational models of concurrency. In *CONCUR*, vol. 4137 of *LNCS*, pages 64–78, 2006.

24 R. Segala and A. Turrini. Comparative analysis of bisimulation relations on alternating and non-alternating probabilistic models. In *QEST*, pages 44–53, 2005.

25 M. Stoelinga. *Alea Jacta Est: Verification of Probabilistic, Real-Time and Parametric Systems*. PhD thesis, University of Nijmegen, the Netherlands, 2002.

26 M. J. Todd. The many facets of linear programming. *Math. Progr.*, 91(3):417–436, 2002.

27 M. Y. Vardi. Automatic verification of probabilistic concurrent finite-state programs. In *Foundations of Computer Science*, pages 327–338, 1985.

28 R. Wimmer, M. Herbstritt, H. Hermanns, K. Strampp, and B. Becker. Sigref - A symbolic bisimulation tool box. In *ATVA*, volume 4218 of LNCS, pages 477–492, 2006.