Generic Forward and Backward Simulations III: Quantitative Simulations by Matrices

Natsuki Urabe and Ichiro Hasuo

University of Tokyo, Japan

Abstract. We introduce notions of simulation between semiring-weighted automata as models of quantitative systems. Our simulations are instances of the categorical/coalgebraic notions previously studied by Hasuo—hence soundness wrt. language inclusion comes for free—but are concretely presented as matrices that are subject to linear inequality constraints. Pervasiveness of these formalisms allows us to exploit existing algorithms in: searching for a simulation, and hence verifying quantitative correctness that is formulated as language inclusion. Transformations of automata that aid search for simulations are introduced, too. This verification workflow is implemented for the plus-times and max-plus semirings.

1 Introduction

Quantitative aspects of various systems are more and more emphasized in recent verification scenarios. Probabilities in randomized or fuzzy systems are a classic example; utility in economics and game theory is another. Furthermore, now that many computer systems are integrated into physical ambience—realizing so-called *cyber-physical systems*—physical quantities like energy consumption are necessarily taken into account.

Semiring-weighted automata It is standard in the concurrency community to model such quantitative systems by state-transition systems in which *weights* are assigned to their states and/or transitions. The semantics of such systems varies, however, depending on the interpretation of weights. If they are probabilities, they are accumulated by $\times$ along a path and summed across different paths; if weights are (worst-case) costs, they are summed up along a path and we would take $\max$ across different paths.

The algebraic structure of *semirings* then arises as a uniform mathematical language for different notions of "weight," as is widely acknowledged in the community. The subject of the current study is state-based systems with labeled transitions, in which each transition is assigned a weight from a prescribed semiring $\mathcal{S}$. We shall call them *$\mathcal{S}$-weighted automata*; and we are more specifically interested in the (weighted, finite) *language inclusion* problem and a *simulation-based* approach to it.

Language inclusion Let $\mathcal{A}$ be an $\mathcal{S}$-weighted automaton with labels from an alphabet Σ. It assigns to each word $w \in \Sigma^*$ a weight taken from $\mathcal{S}$—this is much like a (purely) probabilistic automaton assigns a probability to each word. Let us denote this function by $L(\mathcal{A})\colon \Sigma^* \to \mathcal{S}$ and call it the *(weighted) language* of $\mathcal{A}$ by analogy with classic automata theory. The *language inclusion* problem $L(\mathcal{A}) \sqsubseteq L(\mathcal{B})$ asks if: $L(\mathcal{A})(w) \sqsubseteq L(\mathcal{B})(w)$ for each word $w \in \Sigma^*$, where $\sqsubseteq$ is a natural order on the semiring $\mathcal{S}$.

It is not hard to see that language inclusion $L(\mathcal{A}) \sqsubseteq L(\mathcal{B})$ has numerous applications in verification. In a typical scenario, one of $\mathcal{A}$ and $\mathcal{B}$ is a model of a *system* and the other expresses *specification*; and $L(\mathcal{A}) \sqsubseteq L(\mathcal{B})$ gives the definition of "the system meeting the specification." More concrete examples are as follows.

- $\mathcal{S}$ represents probabilities; $\mathcal{A}$ models a system; and $\mathcal{B}$ expresses the specification that certain bad behaviors—identified with words—occur with a certain probability. Then $L(\mathcal{A}) \sqsubseteq L(\mathcal{B})$ is a *safety* statement: each bad behavior occurs in $\mathcal{A}$ at most as likely as in $\mathcal{B}$.
- $\mathcal{S}$ represents profit, $\mathcal{A}$ is a specification and $\mathcal{B}$ is a system. Then $L(\mathcal{A}) \sqsubseteq L(\mathcal{B})$ guarantees the minimal profit yielded by the system $\mathcal{B}$.
- There are other properties reduced to language inclusion in a less trivial manner. An example is *probable innocence* [26], a quantitative notion of anonymity. See [15].

Simulation Direct check of language inclusion is simply infeasible because there are infinitely many words $w \in \Sigma^*$. One finitary proof method—well-known for nondeterministic (i.e. possibilistic) systems—is by *(forward or backward) simulations*, whose systematic study is initiated in [23]. In the nondeterministic setting, a simulation R is a relation between states of $\mathcal{A}$ and $\mathcal{B}$ that witnesses "local language inclusion"; moreover, from the coinductive way in which it is defined, a simulation persistently witnesses local language inclusion—ultimately yielding (global) language inclusion. This property—existence of a simulation implies language inclusion—is called *soundness*.

Contribution: weighted forward/backward simulations by matrices In this paper we extend this simulation approach to language inclusion [23] to the quantitative setting of semiring-weighted automata. Our notions of (forward and backward) weighted simulation are not given by relations, but by *matrices* with entries from a semiring $\mathcal{S}$.

Use of matrices in automata theory is classic—in fact our framework instantiates to that in [23] when we take as $\mathcal{S}$ the Boolean semiring. This is not how we arrived here; conversely, the current results are obtained as instances of a more general theory of *coalgebraic simulations* [11, 12, 14]. There various systems are identified with a categorical construct of *coalgebras* in a Kleisli category; and fwd./bwd. simulations are characterized as lax/oplax morphisms between coalgebras. A generic soundness result (with respect to language/trace inclusion) is also proved in the general categorical terms.

This paper is devoted to concrete presentations of these categorical notions by matrices, and to their application to actual verification of quantitative systems. Presentation by matrices turns out to be an advantage: a simulation is now a matrix X that satisfies certain *linear inequalities*; and existence of such X—i.e. feasibility of linear inequalities—is so common a problem in many fields that there is a large body of existing work that is waiting to be applied. For example *linear programming (LP)* can be exploited for the plus-times semiring for probabilities; and there are algorithms proposed for other semirings such as the max-plus (tropical) one.

Our (mostly semiring-independent) workflow is as follows. A verification goal is formulated as language inclusion $L(\mathcal{A}) \sqsubseteq L(\mathcal{B})$, which we aim to establish by finding a fwd. or bwd. simulation from $\mathcal{A}$ to $\mathcal{B}$. Soundness of simulations follows from the general result in [11]. A simulation we seek for is a matrix subject to certain linear

inequalities, existence of which is checked by various algorithms that exist for different semirings. We implemented this workflow for the plus-times and max-plus semirings.

This simulation-based method is sound but not necessarily complete with respect to language inclusion. Therefore we introduce transformations of weighted automata—called *(forward/backward) partial execution*—that potentially create matrix simulations. Via our equivalence results between our matrix simulation and some known ones (including the one in [7]), the partial execution transformations potentially create those simulations, too.

Organization of the paper In §2 we define semiring-weighted automata, characterize them in coalgebraic terms and recap the coalgebraic theory in [11]. These are combined to yield the notion of simulation matrix in §3. In §4 partial execution transformations of automata are described and proved correct. The framework obtained so far is applied to the plus-times and max-plus semirings, in §5 and §6, respectively. There our proof-of-concept implementations (the code is found at the first author's webpage) and relationship to other known simulation notions are discussed, too. In §7 we conclude.

2 Preliminaries

We review the generic theory of traces and simulations in [11, 14] that is based on (T, F)-systems, which will eventually lead to the notion of simulation matrix in §3.

2.1 Semiring-Weighted Automata

The notion of semiring-weighted automaton is parametrized by a semiring $\mathcal{S}$. For our purpose of applying coalgebraic theory in [11, 14], we impose the following properties.

Definition 2.1 A *commutative cppo-semiring* is a tuple $\mathcal{S} = (S, +_{\mathcal{S}}, 0_{\mathcal{S}}, \times_{\mathcal{S}}, 1_{\mathcal{S}}, \sqsubseteq)$ that satisfies the following conditions.

- $(S, +_{\mathcal{S}}, 0_{\mathcal{S}}, \times_{\mathcal{S}}, 1_{\mathcal{S}})$ is a semiring in which $\times_{\mathcal{S}}$, in addition to $+_{\mathcal{S}}$, is commutative.
- A relation $\sqsubseteq$ is a partial order on S and $(S, \sqsubseteq)$ is ω-*complete*, i.e. an increasing chain $s_0 \sqsubseteq s_1 \sqsubseteq \cdots$ has a supremum.
- Any element $s \in S$ is *positive* in the sense that $0_{\mathcal{S}} \sqsubseteq s$.
- Addition $+_{\mathcal{S}}$ and multiplication $\times_{\mathcal{S}}$ are monotone with respect to $\sqsubseteq$.

It follows from positivity and ω-completeness that countable sum can be straightforwardly defined in a comm. cppo-semiring $\mathcal{S}$. We will use this fact throughout the paper.

Example 2.2 (Semirings $\mathcal{S}_{+,\times}, \mathcal{S}_{\max,+}, \mathcal{B}$) The *plus-times semiring* $\mathcal{S}_{+,\times} = ([0,\infty], +, 0, \times, 1, \leq)$ is a comm. cppo-semiring, where $+$ and $\times$ are usual addition and multiplication of real numbers. This is the semiring that we will use for modeling probabilistic branching. Specifically, probabilities of successive transitions are accumulated using $\times$, and those of different branches are combined with $+$.

The *max-plus semiring* $\mathcal{S}_{\max,+} = ([-\infty,\infty], \max, -\infty, +, 0, \leq)$—also sometimes called the *tropical semiring* [24]—is also a comm. cppo-semiring. Here a number $r \in [-\infty,\infty]$ can be understood as (best-case) *profit*: they are summed up along a path,

and an optimal one ($\max$) is chosen among different branches. Another possible understanding of r is as (worst-case) *cost*. The unit for the semiring addition $\max$ is given by $-\infty$; since it must also be a zero element of the semiring multiplication $+$, we define $(-\infty) + \infty = -\infty$. In the two examples $\mathcal{S}_{+,\times}$ and $\mathcal{S}_{\max,+}$ we added ∞ so that they become ω-complete.

Finally, the *Boolean semiring* $\mathcal{B} = (\{0,1\}, \vee, 0, \wedge, 1, \leq)$ is an example that is qualitative rather than quantitative.

Definition 2.3 ($\mathcal{S}$-weighted automaton, weighted language) Let $\mathcal{S} = (S, +_{\mathcal{S}}, 0_{\mathcal{S}}, \times_{\mathcal{S}}, 1_{\mathcal{S}}, \sqsubseteq)$ be a comm. cppo-semiring. An *$\mathcal{S}$-weighted automaton* $\mathcal{A} = (Q, \Sigma, M, \alpha, \beta)$ consists of a countable state space Q, a countable alphabet Σ, transition matrices $M(a) \in S^{Q \times Q}$ for all $a \in \Sigma$, an initial row vector $\alpha \in S^Q$ and a final column vector $\beta \in S^Q$.

Let $x, y \in Q$ and $a \in \Sigma$. We write α_x and β_x for the x-th entry of α and β, respectively, and $M(a)_{x,y}$ for the (x,y)-entry of the matrix $M(a)$. Note that these entries are all elements of the semiring $\mathcal{S}$.

An $\mathcal{S}$-weighted automaton $\mathcal{A} = (Q, \Sigma, M, \alpha, \beta)$ yields a *weighted language* $L(\mathcal{A})\colon \Sigma^* \to S$. It is given by the following multiplication of matrices and vectors.

$$L(\mathcal{A})(w) \;:=\; \alpha \cdot M(a_1) \cdot \;\cdots\; \cdot M(a_k) \cdot \beta \qquad \text{for each } w = a_1 \cdots a_k \in \Sigma^*. \quad (1)$$

We require a state space Q to be at most countably infinite. This is so that the matrix multiplications in (1)—by addition and multiplication of $\mathcal{S}$—are well-defined. Recall that $\mathcal{S}$ has countable sum given by supremums of suitable ω-chains.

Our interest is in establishing language inclusion between two weighted automata.

Definition 2.4 (language inclusion) We write $L(\mathcal{A}) \sqsubseteq L(\mathcal{B})$ if, for each $w \in \Sigma^*$, $L(\mathcal{A})(w) \sqsubseteq L(\mathcal{B})(w)$. The last $\sqsubseteq$ is the order of $\mathcal{S}$.

2.2 Coalgebraic Modeling of Semiring-Weighted Automata

Here we characterize semiring-weighted automata as instances of a generic coalgebraic model of branching systems—so-called (T,F)-*systems* with parameters T, F [11, 14].

Definition 2.5 ((T,F)-system) Let T be a monad and F be a functor, both on the category $\mathbf{Sets}$ of sets and functions. A (T,F)-*system* is a triple

$$\mathcal{X} \;=\; \big(\; X,\;\; s\colon \{\bullet\} \to TX,\;\; c\colon X \to TFX \;\big)$$

of a set X (the *state space*), and functions s (the *initial states*) and c (the *dynamics*).

This modeling is coalgebraic [17] in the sense that c is so-called a TF-coalgebra. In the definition we have two parameters T and F. Let us forget about their categorical structures (a *monad* or a *functor*) for a moment and think of them simply as constructions on sets. Intuitively speaking, T specifies what kind of *branching* the systems in question exhibit; and F specifies a type of *linear-time behaviors*. Here are some examples; in the example $F = 1 + \Sigma \times (\_)$ the only element of 1 is denoted by ✓ (i.e. $1 = \{\checkmark\}$).

T	"branching"
$\mathcal{P}$	non-deterministic
$\mathcal{D}$	probabilistic
$\mathcal{M}_{\mathcal{S}}$	$\mathcal{S}$-weighted

F	"linear-time behavior"
$1 + \Sigma \times (\_)$	$\to \checkmark$ or $\xrightarrow{a}$ (where $a \in \Sigma$)
$(\Sigma + (\_))^*$	words over terminals ($a \in \Sigma$)
	& nonterminals, suited for CFG [13]

The above examples of a monad T—the *powerset monad* $\mathcal{P}$, the *subdistribution monad* $\mathcal{D}$, and the $\mathcal{S}$*-multiset monad* $\mathcal{M}_{\mathcal{S}}$ for $\mathcal{S}$—are described as follows.

$$\begin{aligned} \mathcal{P}X &= \{X' \mid X' \subseteq X\} \qquad \mathcal{D}X = \{f : X \to [0,1] \mid \textstyle\sum_{x \in X} f(x) \leq 1\} \\ \mathcal{M}_{\mathcal{S}}X &= \{f : X \to S \mid \mathrm{supp}(f) \text{ is countable}\} \end{aligned} \tag{2}$$

Here $\mathrm{supp}(f) = \{x \in X \mid f(x) \neq 0_{\mathcal{S}}\}$. Countable support in $\mathcal{M}_{\mathcal{S}}$ is a technical requirement so that composition $\odot$ of Kleisli arrows is well-defined (Def. 2.7).

It should not be hard to see that a (T, F)-system models a state-based system with T-branching and F-linear-time behaviors. For example, when $T = \mathcal{P}$ and $F = 1 + \Sigma \times (\_)$, $s\colon \{\bullet\} \to \mathcal{P}X$ represents the set of initial states and $c\colon X \to \mathcal{P}(1 + \Sigma \times X)$ represents one-step transitions—that $\checkmark \in c(x)$ means x is accepting ($x \to \checkmark$), and $(a, x') \in c(x)$ means there is a transition $x \xrightarrow{a} x'$. Overall, a $(\mathcal{P}, 1 + \Sigma \times (\_))$-system is nothing but a nondeterministic automaton.

Analogously we obtain the following, by the definition of $\mathcal{M}_{\mathcal{S}}$ in (2).

Proposition 2.6 (weighted automata as (T, F)-systems) *Let $\mathcal{S}$ be a comm. cppo-semiring. There is a bijective correspondence between: 1) $\mathcal{S}$-weighted automata (Def. 2.3); and 2) $\big(\mathcal{M}_{\mathcal{S}}, 1 + \Sigma \times (\_)\big)$-systems whose state spaces are at most countably infinite.*

Concretely, an $\mathcal{S}$-weighted automaton $\mathcal{A} = (Q, \Sigma, M, \alpha, \beta)$ gives rise to an $\big(\mathcal{M}_{\mathcal{S}}, 1 + \Sigma \times (\_)\big)$-system $\mathcal{X}_{\mathcal{A}} = (Q, s_{\mathcal{A}}, c_{\mathcal{A}})$ defined as follows. $s_{\mathcal{A}}\colon \{\bullet\} \to \mathcal{M}_{\mathcal{S}}Q$ is given by $s_{\mathcal{A}}(\bullet)(x) = \alpha_x$; and $c_{\mathcal{A}}\colon Q \to \mathcal{M}_{\mathcal{S}}(1 + \Sigma \times Q)$ is given by $c_{\mathcal{A}}(x)(\checkmark) = \beta_x$ and $c_{\mathcal{A}}(x)(a, y) = M(a)_{x,y}$. □

2.3 Coalgebraic Theory of Traces and Simulations

We review the theory of traces and simulations in [11, 14] that is based on (T, F)-systems. In presentation we restrict to $T = \mathcal{M}_{\mathcal{S}}$ and $F = 1 + \Sigma \times (\_)$ for simplicity.

Kleisli Arrows One notable success of coalgebra was a uniform characterization, in terms of the same categorical diagram, of *bisimulations* for various kinds of systems (nondeterministic, probabilistic, etc.) [17]. This works quite well for branching-time process semantics. For linear-time semantics—i.e. trace semantics—it is noticed in [25] that so-called a *Kleisli category*, in place of the category $\mathbf{Sets}$, gives a suitable base category for coalgebraic treatment. This idea—replacing functions $X \to Y$ with *Kleisli arrows* $X \mathrel{+\!\!\to} Y$ and drawing the same diagrams—led to the development in [11, 12, 14] of an extensive theory of traces and simulations. The notion of Kleisli arrow is parametrized by a monad T: a T-Kleisli arrow $X \mathrel{+\!\!\to}_T Y$ (or simply $X \mathrel{+\!\!\to} Y$) is defined to be a function $X \to TY$, hence represents a "T-branching function from X to Y."

We restrict to $T = \mathcal{M}_{\mathcal{S}}$ for simplicity of presentation. An $\mathcal{M}_{\mathcal{S}}$-Kleisli arrow $f\colon X \mathrel{+\!\!\to} Y$ below is "an $\mathcal{S}$-weighted function from X to Y." In particular, for each $x \in X$ and $y \in Y$ it assigns a *weight* $f(x)(y) \in S$.

Definition 2.7 (Kleisli arrow) Let X, Y be sets. An $\mathcal{M}_{\mathcal{S}}$*-Kleisli arrow* (or simply a *Kleisli arrow*) from X to Y, denoted by $X \mathrel{+\!\!\to} Y$, is a function from X to $\mathcal{M}_{\mathcal{S}}Y$.

We list some special Kleisli arrows: η_X, $g \odot f$ and Jf.

- For each set X, the *unit arrow* $\eta_X\colon X \mathrel{+\!\!\to} X$ is given by: $\eta(x)(x) = 1_{\mathcal{S}}$; and $\eta(x)(x') = 0_{\mathcal{S}}$ for $x' \neq x$. Here $0_{\mathcal{S}}$ and $1_{\mathcal{S}}$ are units in the semiring $\mathcal{S}$.

– For consecutive Kleisli arrows $f : X \nrightarrow Y$ and $g : Y \nrightarrow Z$, their *composition* $g \odot f : X \nrightarrow Z$ is given as follows:

$$(g \odot f)(x)(z) \;:=\; \textstyle\sum_{y \in \mathrm{supp}(f(x))} f(x)(y) \times_{\mathcal{S}} g(y)(z) \;\; .$$

  Since $\mathrm{supp}(f(x))$ is countable, the above sum in a cppo-semiring $\mathcal{S}$ is well-defined.
– For a (usual) function $f : X \to Y$, its *lifting* to a Kleisli arrow $Jf : X \nrightarrow Y$ is given by $Jf = \eta_Y \circ f$. Here we identified $\eta_Y : Y \nrightarrow Y$ with a function $\eta_Y : Y \to \mathcal{M}_{\mathcal{S}} Y$.

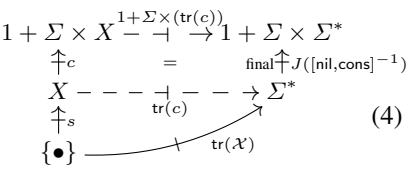
Categorically speaking: the first two (η and $\odot$) organize Kleisli arrows as a category (the *Kleisli category* $\mathcal{K}\ell(\mathcal{M}_{\mathcal{S}})$); and the third gives a functor $J \colon \mathbf{Sets} \to \mathcal{K}\ell(\mathcal{M}_{\mathcal{S}})$ that is identity on objects.

In Prop. 2.6 we characterized an $\mathcal{S}$-weighted automaton $\mathcal{A}$ in coalgebraic terms. Using Kleisli arrows it is presented as a triple

$$\mathcal{X}_{\mathcal{A}} \;=\; \big(\; Q,\;\; s_{\mathcal{A}} \colon \{\bullet\} \nrightarrow Q,\;\; c_{\mathcal{A}} \colon Q \nrightarrow 1 + \Sigma \times Q \;\big) \;\; . \tag{3}$$

Generic Trace Semantics In [14], for monads T with a suitable order, a final coalgebra in $\mathcal{K}\ell(T)$ is identified. It (somehow interestingly) coincides with an initial algebra in $\mathbf{Sets}$. Moreover, the universality of this final coalgebra is shown to capture natural notions of (finite) *trace semantics* for a variety of branching systems—i.e. for different T and F. What is important for the current work is the fact that the weighted language $L(A)$ in (1) is an instance of this generic trace semantics, as we will show in Thm. 2.10.

We shall state the results in [14] on coalgebraic traces, restricting again to $T = \mathcal{M}_{\mathcal{S}}$ and $F = 1 + \Sigma \times (\_)$ for simplicity. In the diagram (4) on the right, composition of Kleisli arrows are given by $\odot$ in Def. 2.7; J on the right is the lifting in Def. 2.7; and nil and cons are the obvious constructors of words in Σ^*. The top arrow $1 + \Sigma \times (\mathsf{tr}(c))$ is the functor $1 + \Sigma \times (\_)$ on $\mathbf{Sets}$, lifted to the Kleisli category $\mathcal{K}\ell(\mathcal{M}_{\mathcal{S}})$, and applied to the Kleisli arrow $\mathsf{tr}(c)$; its concrete description is found in Def. A.4.



(4)

Theorem 2.8 (final coalgebra in $\mathcal{K}\ell(\mathcal{M}_{\mathcal{S}})$) *Given any set X and any Kleisli arrow $c \colon X \nrightarrow 1 + \Sigma \times X$, there exists a unique Kleisli arrow $\mathsf{tr}(c)$ that makes the top square in the diagram (4) commute.* □

Definition 2.9 ($\mathsf{tr}(\mathcal{X})$) Given an $\big(\mathcal{M}_{\mathcal{S}}, 1 + \Sigma \times (\_)\big)$-system $\mathcal{X} = (X, s, c)$ (this is on the left in the diagram (4)), its component c induces an arrow $\mathsf{tr}(c) \colon X \nrightarrow \Sigma^*$ by Thm. 2.8. We define $\mathsf{tr}(\mathcal{X})$ to be the composite $\mathsf{tr}(c) \odot s$ (the bottom triangle in the diagram (4)), and call it the *trace semantics* of $\mathcal{X}$.

Theorem 2.10 (weighted language as trace semantics) *Let $\mathcal{A}$ be an $\mathcal{S}$-weighted automaton. For $\mathcal{X}_{\mathcal{A}} = (Q, s_{\mathcal{A}}, c_{\mathcal{A}})$ induced by $\mathcal{A}$ in (3), its trace semantics $\mathsf{tr}(\mathcal{X}_{\mathcal{A}}) \colon \{\bullet\} \nrightarrow \Sigma^*$—identified with a function $\{\bullet\} \to \mathcal{M}_{\mathcal{S}} \Sigma^*$, hence with a function $\Sigma^* \to S$—coincides with the weighted language $L(\mathcal{A}) \colon \Sigma^* \to S$ in (1).* □

In the last theorem we need that Σ^* is countable; this is why we assumed that Σ is countable in Def. 2.3. Henceforth we do not distinguish $L(\mathcal{A})$ and $\mathsf{tr}(\mathcal{X}_{\mathcal{A}}) \colon \{\bullet\} \nrightarrow \Sigma^*$.

Forward and Backward *Kleisli* simulations In [11], the classic results in [23] on forward and backward simulations—for (nondeterministic) labeled transition systems—are generalized to (T, F)-systems. Specifically, fwd./bwd. simulations are characterized as *lax/oplax coalgebra homomorphisms* in a Kleisli category; and *soundness*—their existence witnesses trace inclusion—is proved once for all in a general categorical setting.

As before, we present those notions and results in [11] restricting to $T = \mathcal{M}_{\mathcal{S}}$ and $F = 1 + \Sigma \times (\_)$. If $T = \mathcal{P}$ and $F = 1 + \Sigma \times (\_)$ they instantiate to the results in [23].

Definition 2.11 (Kleisli simulation) Let $\mathcal{X} = (X, s, c)$ and $\mathcal{Y} = (Y, t, d)$ be $(\mathcal{M}_{\mathcal{S}}, 1 + \Sigma \times (\_))$-systems (cf. Def. 2.5, Prop. 2.6 and (3)).

1. A *forward (Kleisli) simulation* from $\mathcal{X}$ to $\mathcal{Y}$ is a Kleisli arrow $f : Y \nrightarrow X$ such that $s \sqsubseteq f \odot t$ and $c \odot f \sqsubseteq (1 + \Sigma \times f) \odot d$. See Fig. 1.
2. A *backward simulation* from $\mathcal{X}$ to $\mathcal{Y}$ is a Kleisli arrow $b : X \nrightarrow Y$ such that $s \odot b \sqsubseteq t$ and $(1 + \Sigma \times b) \odot c \sqsubseteq d \odot b$.
3. A *forward-backward simulation* from $\mathcal{X}$ to $\mathcal{Y}$ consists of: a (T, F)-system $\mathcal{Z}$; a fwd. simulation f from $\mathcal{X}$ to $\mathcal{Z}$; and a bwd. simulation b from $\mathcal{Z}$ to $\mathcal{Y}$.
4. A *backward-forward simulation* from $\mathcal{X}$ to $\mathcal{Y}$ consists of: a (T, F)-system $\mathcal{Z}$; a bwd. simulation b from $\mathcal{X}$ to $\mathcal{Z}$; and a fwd. simulation f from $\mathcal{Z}$ to $\mathcal{Y}$.

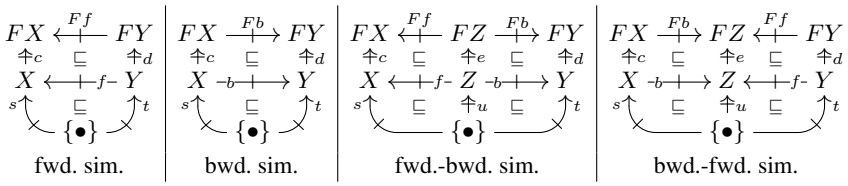


Fig. 1. Kleisli simulations (here $F = 1 + \Sigma \times (\_)$)

We write $\mathcal{X} \sqsubseteq_{\mathbf{F}} \mathcal{Y}$, $\mathcal{X} \sqsubseteq_{\mathbf{B}} \mathcal{Y}$, $\mathcal{X} \sqsubseteq_{\mathbf{FB}} \mathcal{Y}$ or $\mathcal{X} \sqsubseteq_{\mathbf{FB}} \mathcal{Y}$ if there exists a forward, backward, forward-backward, or backward-forward simulation, respectively.

(Generic) soundness is proved using the maximality of $\mathsf{tr}(c)$ in (4) among (op)lax coalgebra homomorphisms, arguing in the language of enriched category theory [11].

Theorem 2.12 (soundness) *Let $\mathcal{X}$ and $\mathcal{Y}$ be $(\mathcal{M}_{\mathcal{S}}, 1 + \Sigma \times (\_))$-systems. Each of the following yields $\mathsf{tr}(\mathcal{X}) \sqsubseteq \mathsf{tr}(\mathcal{Y}) \colon \{\bullet\} \nrightarrow \Sigma^*$ (cf. Def. 2.9).*
1. $\mathcal{X} \sqsubseteq_{\mathbf{F}} \mathcal{Y}$ 2. $\mathcal{X} \sqsubseteq_{\mathbf{B}} \mathcal{Y}$ 3. $\mathcal{X} \sqsubseteq_{\mathbf{FB}} \mathcal{Y}$ 4. $\mathcal{X} \sqsubseteq_{\mathbf{BF}} \mathcal{Y}$ □

Theorem 2.13 (completeness) *The converse of soundness holds for backward-forward simulations. That is: $\mathsf{tr}(\mathcal{X}) \sqsubseteq \mathsf{tr}(\mathcal{Y})$ implies $\mathcal{X} \sqsubseteq_{\mathbf{BF}} \mathcal{Y}$.* □

3 Simulation Matrices for Semiring-Weighted Automata

In this section we fix parameters $T = \mathcal{M}_{\mathcal{S}}$ and $F = 1 + \Sigma \times (\_)$ in the generic theory in §2.3 and rephrase the coalgebraic framework in terms of matrices (whose entries are taken from $\mathcal{S}$). Specifically: Kleisli arrows become matrices; and Kleisli simulations become matrices subject to certain linear inequalities. Such matrix representations ease implementation, a feature we will exploit in later sections.

Recall that a Kleisli arrow $A \nrightarrow B$ is a function $A \to \mathcal{M}_{\mathcal{S}} B$ (Def. 2.7).

Definition 3.1 (matrix representation M_f) Given a Kleisli arrow $f \colon A \nrightarrow B$, its *matrix representation* $M_f \in S^{A \times B}$ is given by $(M_f)_{x,y} = f(x)(y)$.

In what follows we shall use the notations f and M_f interchangeably.

Lemma 3.2 *Let $f, f' \colon A \nrightarrow B$ and $g \colon B \nrightarrow C$ be Kleisli arrows.*

1. *$f \sqsubseteq f'$ if and only if $M_f \sqsubseteq M_{f'}$. Here the former $\sqsubseteq$ is between $\mathcal{M}_{\mathcal{S}}$-Kleisli arrows, and the latter order $\sqsubseteq$ is between matrices, defined entrywise.*
2. *$M_{g \odot f} = M_f M_g$, computed by matrix multiplication.* □

The correspondence from $A \overset{f}{\nrightarrow} B$ to $1 + \Sigma \times A \overset{1+\Sigma\times f}{\nrightarrow} 1 + \Sigma \times B$—used in (4) and in Fig. 1—can be described using matrices, too. Details are in Appendix A.2.

Lemma 3.3 *Let $f : A \nrightarrow B$ be a Kleisli arrow and M_f be its matrix representation. Then the matrix representation $M_{1+\Sigma\times f}$ is given by*

$$I_1 \oplus (I_\Sigma \otimes M_f) \;\in\; S^{(1+\Sigma\times A)\times(1+\Sigma\times B)} \;,$$

where $\oplus$ and $\otimes$ denote coproduct *and* the Kronecker product *of matrices:*

$$X \oplus Y \;=\; \begin{pmatrix} \boxed{X} & O \\ O & \boxed{Y} \end{pmatrix} \;, \qquad \begin{pmatrix} \vdots \\ \cdots\; x_{i,j} \;\cdots \\ \vdots \end{pmatrix} \otimes \boxed{Y} \;=\; \begin{pmatrix} \vdots \\ \cdots\; \boxed{x_{i,j} Y} \;\cdots \\ \vdots \end{pmatrix} \;.$$

□

This description of M_{Ff} generalizes from $F = 1 + \Sigma \times (\_)$ to any polynomial functor F, inductively on the construction of F. In this paper the generality is not needed.

Using Lem. 3.2–3.3, we can present Kleisli simulations (Def. 2.11) as matrices. Recall that a state space of a weighted automaton is assumed to be countable (Def. 2.3); hence all the matrix multiplications in the definition below make sense.

Definition 3.4 (forward/backward simulation matrix) Let $\mathcal{A} = (Q_{\mathcal{A}}, \Sigma, M_{\mathcal{A}}, \alpha_{\mathcal{A}}, \beta_{\mathcal{A}})$ and $\mathcal{B} = (Q_{\mathcal{B}}, \Sigma, M_{\mathcal{B}}, \alpha_{\mathcal{B}}, \beta_{\mathcal{B}})$ be $\mathcal{S}$-weighted automata.

- A matrix $X \in S^{Q_{\mathcal{B}} \times Q_{\mathcal{A}}}$ is a *forward simulation matrix* from $\mathcal{A}$ to $\mathcal{B}$ if
  $\alpha_{\mathcal{A}} \sqsubseteq \alpha_{\mathcal{B}} X \;,\quad X \cdot M_{\mathcal{A}}(a) \sqsubseteq M_{\mathcal{B}}(a) \cdot X \quad (\forall a \in \Sigma) \;,\quad$ and $\quad X \beta_{\mathcal{A}} \sqsubseteq \beta_{\mathcal{B}} \;.$
- A matrix $X \in S^{Q_{\mathcal{A}} \times Q_{\mathcal{B}}}$ is a *backward simulation matrix* from $\mathcal{A}$ to $\mathcal{B}$ if
  $\alpha_{\mathcal{A}} X \sqsubseteq \alpha_{\mathcal{B}} \;,\quad M_{\mathcal{A}}(a) \cdot X \sqsubseteq X \cdot M_{\mathcal{B}}(a) \quad (\forall a \in \Sigma) \;,\quad$ and $\quad \beta_{\mathcal{A}} \sqsubseteq X \beta_{\mathcal{B}} \;.$

The requirements on X are obtained by first translating Fig. 1 into matrices, and then breaking them up into smaller matrices using Lem. 3.3. It is notable that the requirements are given in the form of *linear inequalities*, a format often used in constraint solvers. Solving them is a topic of extensive research efforts that include [2, 6]. This fact becomes an advantage in implementing search algorithms, as we see later.

We also note that *forward* and *backward* simulation matrices have different dimensions. This difference comes from the different directions of arrows in Fig. 1.

Theorem 3.5 *Let $\mathcal{A}$ and $\mathcal{B}$ be $\mathcal{S}$-weighted automata. There is a bijective correspondence between: 1) forward simulation matrices from $\mathcal{A}$ to $\mathcal{B}$; and 2) forward Kleisli simulations from $\mathcal{X}_{\mathcal{A}}$ to $\mathcal{X}_{\mathcal{B}}$. The same holds for the backward variants.* □

In what follows we write $\sqsubseteq_{\mathbf{F}}, \sqsubseteq_{\mathbf{B}}$ also between $\mathcal{S}$-weighted automata. Thm. 3.5 yields: $\mathcal{A} \sqsubseteq_{\mathbf{F}} \mathcal{B}$ if and only if there is a forward simulation matrix.

Here is our core result; the rest of the paper is devoted to its application.

Corollary 3.6 (soundness of simulation matrices) *Let $\mathcal{A}$ and $\mathcal{B}$ be $\mathcal{S}$-weighted automata. Existence of a forward (or backward) simulation matrix from $\mathcal{A}$ to $\mathcal{B}$—i.e. $\mathcal{A} \sqsubseteq_{\mathbf{F}} \mathcal{B}$ or $\mathcal{A} \sqsubseteq_{\mathbf{B}} \mathcal{B}$—witnesses language inclusion $L(\mathcal{A}) \sqsubseteq L(\mathcal{B})$.*

Proof. $\exists$ (fwd./bwd. simulation matrix from $\mathcal{A}$ to $\mathcal{B}$)
$\overset{\text{Thm. 3.5}}{\iff}$ $\exists$ (fwd./bwd. Kleisli simulation from $\mathcal{X}_{\mathcal{A}}$ to $\mathcal{X}_{\mathcal{B}}$)
$\overset{\text{Thm. 2.12}}{\implies}$ $\mathsf{tr}(\mathcal{X}_{\mathcal{A}}) \sqsubseteq \mathsf{tr}(\mathcal{X}_{\mathcal{B}})$ $\overset{\text{Thm. 2.10}}{\iff}$ $L(\mathcal{A}) \sqsubseteq L(\mathcal{B})$. □

It is classic to represent nondeterministic automata by Boolean matrices. This corresponds to the special case $\mathcal{S} = \mathcal{B}$ (the Boolean semiring) of the current framework; and a simulation matrix becomes the same thing as a (relational) simulation in [23].

Remark 3.7 The *opposite* of an $\mathcal{S}$-weighted automaton $\mathcal{A} = (Q, \Sigma, M, \alpha, \beta)$—obtained by reversing transitions and swapping initial/final states—can be naturally defined by matrix transpose, that is, ${}^t\mathcal{A} := (Q, \Sigma, {}^tM, {}^t\beta, {}^t\alpha)$. It is easy to see that: if X is a fwd. simulation matrix from $\mathcal{A}$ to $\mathcal{B}$, then ${}^tX$ is a bwd. simulation matrix from ${}^t\mathcal{A}$ to ${}^t\mathcal{B}$.

4 Forward and Backward Partial Execution

We have four different notions of simulation (Def. 2.11): fwd., bwd., fwd.-bwd., and bwd.-fwd. Our view on these is as (possibly finitary) witnesses of language inclusion.

The combined ones (fwd.-bwd. and bwd.-fwd.) subsume the one-direction ones (fwd. and bwd.)—simply take the identity arrow as one of the two simulations required. Moreover, bwd.-fwd. is complete (Thm. 2.13). Despite these theoretical advantages, the combined simulations are generally harder to find: in addition to two simulations, we have to find an intermediate system too ($\mathcal{Z}$ in Def. 2.11). Furthermore, since language inclusion for finite $\mathcal{S}_{+,\times}$-weighted automata—models of probabilistic systems—is known to be undecidable [5], existence of a bwd.-fwd. simulation is undecidable too.

Therefore in what follows we focus on the one-directional (i.e. fwd. or bwd.) simulations as proof methods for language inclusion. They have convenient matrix presentations, too, as we saw in §3. However fwd. or bwd. simulations are not necessarily complete, by a counterexample (Example A.1) or by complexity arguments (§5.1).

In this section we introduce for semiring-weighted automata their transformations—called *forward* and *backward partial execution*—that increase the number of fwd./bwd. simulation matrices. We also prove some correctness results.

Definition 4.1 (FPE, BPE) *Forward partial execution (FPE)* is a transformation of a weighted automaton that "replaces some states with their forward one-step behaviors." Concretely, given an $\mathcal{S}$-weighted automaton $\mathcal{A} = (Q, \Sigma, M, \alpha, \beta)$ and a parameter $P \subseteq Q$, the resulting automaton $\mathcal{A}_{\mathsf{FPE},P} = (Q', \Sigma, M', \alpha', \beta')$ has a state space

$$Q' = \left\{\checkmark \,\middle|\, \exists x \in P.\, \beta_x \neq 0_{\mathcal{S}}\right\} + \left\{(a,y) \,\middle|\, \exists x \in P.\, M(a)_{x,y} \neq 0_{\mathcal{S}}\right\} + (Q \setminus P) \ , \quad (5)$$

replacing each $x \in P$ with its forward one-step behaviors—$\checkmark$ or (a, y)—as new states. The other data M', α', β' are suitably defined following the above intuition; see Appendix A.3. Possible patterns of transformations are illustrated in Fig. 2.

Backward partial execution (BPE) in contrast "replaces states in a parameter $P \subseteq Q$ with their backward one-step behaviors." For the same $\mathcal{A}$ as above, the resulting automaton $\mathcal{A}_{\mathsf{BPE},P} = (Q', \Sigma, M', \alpha', \beta')$ has a state space

$$Q' = \left\{\bullet \,\middle|\, \exists x \in P.\, \alpha_x \neq 0_{\mathcal{S}}\right\} + \left\{(a,y) \,\middle|\, \exists x \in P.\, M(a)_{y,x} \neq 0_{\mathcal{S}}\right\} + (Q \setminus P) \ ,$$

replacing each $p \in P$ with its backward one-step behaviors—(a, y) with $y \xrightarrow{a} x$, and $\bullet$ if x is initial—as new states. M', α', β' are defined in Appendix A.3. See also Fig. 2.

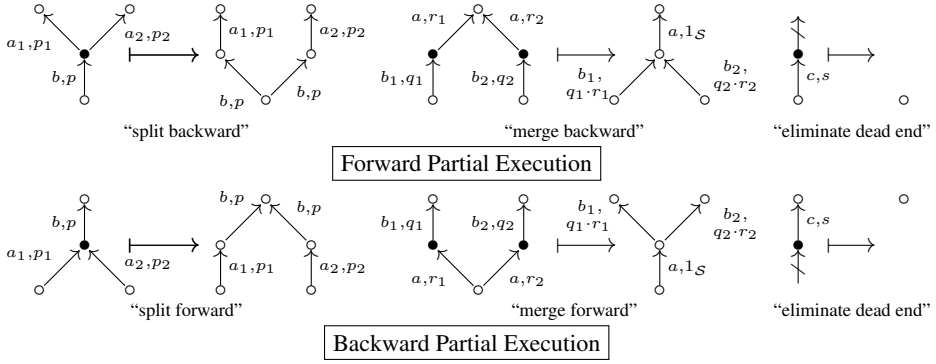


Fig. 2. Fwd./bwd. partial execution (FPE, BPE), pictorially. Black nodes need to be in P

Roughly speaking, FPE replaces a *concrete* state $p \in P$ with an *abstract* state, such as (a, q) in Q' of (5) that is thought of as a description "a state that makes an a-transition to q." The idea comes from *partial evaluation* of a program; hence the name.

The use of FPE/BPE is as follows: we aim to establish $L(\mathcal{A}) \sqsubseteq L(\mathcal{B})$; depending on whether we search for a forward or backward simulation matrix, we apply one of FPE and BPE to each of $\mathcal{A}$ and $\mathcal{B}$, according to the above table.

goal: $L(\mathcal{A}) \sqsubseteq L(\mathcal{B})$	$\mathcal{A}$	$\mathcal{B}$
by $\sqsubseteq_{\mathbf{F}}$	FPE	BPE
by $\sqsubseteq_{\mathbf{B}}$	BPE	FPE

We shall now state correctness properties of this strategy; proofs are in Appendix A.4. *Soundness* means that discovery of a simulation after transformation indeed witnesses the language inclusion for the *original* automata. The second property—we call it *adequacy*—states that simulations that are already there are preserved by partial execution.

Theorem 4.2 (soundness of FPE/BPE) *Let P, P' be arbitrary subsets. Each of the following implies $L(\mathcal{A}) \sqsubseteq L(\mathcal{B})$.*
1. $\mathcal{A}_{\mathsf{FPE},P} \sqsubseteq_{\mathbf{F}} \mathcal{B}_{\mathsf{BPE},P'}$ 2. $\mathcal{A}_{\mathsf{BPE},P} \sqsubseteq_{\mathbf{B}} \mathcal{B}_{\mathsf{FPE},P'}$ □

Theorem 4.3 (adequacy of FPE/BPE) *Let P, P' be arbitrary subsets. We have:*
1. $\mathcal{A} \sqsubseteq_{\mathbf{F}} \mathcal{B} \Rightarrow \mathcal{A}_{\mathsf{FPE},P} \sqsubseteq_{\mathbf{F}} \mathcal{B}_{\mathsf{BPE},P'}$ 2. $\mathcal{A} \sqsubseteq_{\mathbf{B}} \mathcal{B} \Rightarrow \mathcal{A}_{\mathsf{BPE},P} \sqsubseteq_{\mathbf{B}} \mathcal{B}_{\mathsf{FPE},P'}$. □

We also show that a bigger parameter P yields a greater number of simulations. In implementation, however, a bigger P generally gives us a bigger state space which slows down search for a simulation, resulting in a trade-off situation.

Proposition 4.4 (monotonicity) *Assume $P_1 \subseteq P_1'$ and $P_2 \subseteq P_2'$. We have:*
1. $\mathcal{A}_{\mathsf{FPE},P_1} \sqsubseteq_{\mathbf{F}} \mathcal{B}_{\mathsf{BPE},P_2} \Rightarrow \mathcal{A}_{\mathsf{FPE},P_1'} \sqsubseteq_{\mathbf{F}} \mathcal{B}_{\mathsf{BPE},P_2'}$,
2. $\mathcal{A}_{\mathsf{BPE},P_1} \sqsubseteq_{\mathbf{B}} \mathcal{B}_{\mathsf{FPE},P_2} \Rightarrow \mathcal{A}_{\mathsf{BPE},P_1'} \sqsubseteq_{\mathbf{B}} \mathcal{B}_{\mathsf{FPE},P_2'}$. □

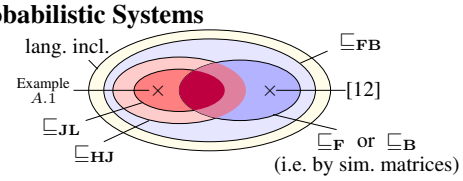
In fact we have a coalgebraic characterization of FPE, too, as a partial application of the functor $1 + \Sigma \times (\_)$. This characterization generalizes to a large class of (T, F)-systems, and the above correctness results can be proved generally by categorical arguments. See Appendix A.5 for details. Capturing BPE categorically is still open—it seems that BPE exists somewhat coincidentally, for the specific functor $F = 1 + \Sigma \times (\_)$ for which an opposite automaton is canonically defined (cf. Rem. 3.7).

For $\mathcal{S} = \mathcal{S}_{+,\times}$ or $\mathcal{S}_{\max,+}$, we can easily see that the complement problem of language inclusion between finite $\mathcal{S}$-weighted automata is semi-decidable. Since language inclusion itself is undecidable [5, 22], language inclusion is not semidecidable either. Because existence of a simulation matrix is decidable, it can be the case that however many times we apply FPE or BPE, simulation matrices do not exist while language inclusion holds. A concrete example is found in Example A.2.

5 Simulation Matrices for Probabilistic Systems by $\mathcal{S} = \mathcal{S}_{+,\times}$

In §5 we focus on $\mathcal{S}_{+,\times}$-weighted automata which we identify as (purely) probabilistic automata (cf. Example 2.2). In §5.1 our method by simulation matrices is compared with other notions of probabilistic simulation; in §5.2 we discuss our implementation.

5.1 Other Simulation Notions for Probabilistic Systems



Various simulation notions have been introduced for probabilistic systems, either as a behavioral order by itself or as a proof method for language inclusion. Jonsson and Larsen's one [18] (denoted by $\sqsubseteq_{\mathbf{JL}}$) is well-known; it is shown in [12] to be a special case of Hughes and Jacobs' coalgebraic notion of simulation [16] ($\sqsubseteq_{\mathbf{HJ}}$), which in turn is a special case of fwd.-bwd. (Kleisli) simulation ($\sqsubseteq_{\mathbf{FB}}$, Def. 2.11). Comparison of all these notions (observed in [12]) is as depicted above; it follows from Thm. 2.12 that all these simulation notions are sound with respect to language inclusion.

We note that language inclusion between finite $\mathcal{S}_{+,\times}$-weighted automata is undecidable [5] while language equivalence can be determined in polynomial time [19]. The former result can account for the fact that there seem to be not many proof methods for probabilistic/quantitative language inclusion. For example, *probabilistic simulation* in [3] is possibilistic simulation between systems with both probabilistic and nondeterministic choice and not a quantitative notion like in the current study.

We also note that given finite-state $\mathcal{S}_{+,\times}$-weighted automata $\mathcal{A}$ and $\mathcal{B}$, if $\mathcal{A} \sqsubseteq_{\mathbf{F}} \mathcal{B}$ or not is decidable: existence of a solution X of the linear constraints in Def. 3.4 can be reduced to linear programming (LP) problems, and the latter are known to be decidable. The same applies to $\sqsubseteq_{\mathbf{B}}$ too.

Probabilistic systems are commonly modeled using the monad $\mathcal{D}$ (see (2))—with an explicit *normalization* condition $\sum_x d(x) \leq 1$—instead of $\mathcal{M}_{\mathcal{S}_{+,\times}}$. However there is no need to impose normalization on simulations: sometimes only "non-normalized" simulation matrices are found (Example A.3) and they are still sound.

5.2 Implementation, Experiments and Discussions

Our implementation consists of two components: $+\times$-sim and $+\times$-PE.

- The program $+\times$-sim (implemented in C++) computes if a forward or backward simulation matrix X between $\mathcal{S}_{+,\times}$-weighted automata exists, and returns X if it does exist. It first combines the constraints in Def. 3.4 into a single linear inequality $A\boldsymbol{x} \leq \boldsymbol{b}$ and solves it with a linear programming solver *glpk* [1]. We note that the matrix A is sparse, having $n + anm + m$ rows, nm columns and at most $2nm + a(n^2m + nm^2)$ non-zero entries.

– The program $+\times$-PE (implemented in OCaml) takes an automaton $\mathcal{A}$ and $d \in \mathbb{N}$ as input, and returns $\mathcal{A}_{\mathsf{FPE},P}$ (or $\mathcal{A}_{\mathsf{BPE},P}$, by choice) where P is chosen, by heuristics, to be $P = \{x \mid x \overbrace{\to \cdots \to}^{d} \checkmark\}$ (or $P = \{x \mid \bullet \overbrace{\to \cdots \to}^{d} x\}$, respectively).

The two programs are alternately applied to given automata, for $d = 1, 2, \ldots$, each time incrementing the parameter d for $+\times$-PE. The experiments were on an Ubuntu Linux laptop with a Core i5 2.53 GHz processor (4 cores) and 4 GB RAM.

Grades Protocol The *grades protocol* is introduced in [19] and is used there as a benchmark: the protocol and its specification are expressed as probabilistic programs P and S; they are then translated into (purely) probabilistic automata $\mathcal{A}_\mathsf{P}$ and $\mathcal{A}_\mathsf{S}$ by a game semantics-based tool APEX [20]. By establishing $L(\mathcal{A}_\mathsf{P}) = L(\mathcal{A}_\mathsf{S})$, the protocol is shown to exhibit the same behaviors as the specification—hence is verified. The protocol has two parameters G and S.

In our experiment we proved $L(\mathcal{A}_\mathsf{P}) = L(\mathcal{A}_\mathsf{S})$ by establishing two-way language inclusion ($\sqsubseteq$ and $\sqsupseteq$). The results are shown in Table. 1. For all the choices of parameters G and S, our program $+\times$-sim was able to establish, without applying $+\times$-PE: $\mathcal{A}_\mathsf{P} \sqsubseteq_\mathbf{F} \mathcal{A}_\mathsf{S}$ (but not $\sqsubseteq_\mathbf{B}$) for the $\sqsubseteq$ direction; and $\mathcal{A}_\mathsf{P} \sqsupseteq_\mathbf{B} \mathcal{A}_\mathsf{S}$ (but not $\sqsupseteq_\mathbf{F}$) for the $\sqsupseteq$ direction. In the table, #st. and #tr. denote the numbers of states and transitions, respectively, and $|\Sigma|$ is the size of the alphabet. All these numbers are determined by APEX .

param. G	param. S	$\mathcal{A}_\mathsf{P}$ #st.	$\mathcal{A}_\mathsf{P}$ #tr.	$\mathcal{A}_\mathsf{S}$ #st.	$\mathcal{A}_\mathsf{S}$ #tr.	$\lvert\Sigma\rvert$	direction, fwd./bwd.	time (sec)	space (GB)
2	8	578	1522	130	642	11	$\mathcal{A}_\mathsf{P} \sqsubseteq_\mathbf{F} \mathcal{A}_\mathsf{S}$	1.77	1.21
							$\mathcal{A}_\mathsf{P} \sqsupseteq_\mathbf{B} \mathcal{A}_\mathsf{S}$	1.72	1.22
2	10	1102	2982	202	1202	13	$\mathcal{A}_\mathsf{P} \sqsubseteq_\mathbf{F} \mathcal{A}_\mathsf{S}$	9.42	4.05
							$\mathcal{A}_\mathsf{P} \sqsupseteq_\mathbf{B} \mathcal{A}_\mathsf{S}$	9.25	4.09
2	12	1874	5162	290	2018	15	$\mathcal{A}_\mathsf{P} \sqsubseteq_\mathbf{F} \mathcal{A}_\mathsf{S}$	38.60	11.51
							$\mathcal{A}_\mathsf{P} \sqsupseteq_\mathbf{B} \mathcal{A}_\mathsf{S}$	38.34	11.63
3	8	1923	7107	243	2163	20	$\mathcal{A}_\mathsf{P} \sqsubseteq_\mathbf{F} \mathcal{A}_\mathsf{S}$	44.43	12.26
							$\mathcal{A}_\mathsf{P} \sqsupseteq_\mathbf{B} \mathcal{A}_\mathsf{S}$	44.11	12.64
4	6	1636	7468	196	1924	23	$\mathcal{A}_\mathsf{P} \sqsubseteq_\mathbf{F} \mathcal{A}_\mathsf{S}$	30.28	10.39
							$\mathcal{A}_\mathsf{P} \sqsupseteq_\mathbf{B} \mathcal{A}_\mathsf{S}$	29.94	10.49

Table 1. Results for the grades protocol [19]

The table indicates that space is a bigger problem for our approach than time. In [19] four algorithms for checking language *equivalence* between $\mathcal{S}_{+,\times}$-weighted automata are implemented and compared: two are deterministic [9, 27] and the other two are randomized [19]. These algorithms can process bigger problem instances (e.g. $G = 2, S = 100$ in ca. 10 sec) and, in comparison, the results in Table 1 are far from impressive. Note however that our algorithm is for language *inclusion*—an undecidable problem, unlike language *equivalence* that is in $\mathbf{P}$, see §5.1—and hence is more general.

Crowds Protocol Our second experiment calls for checking language *inclusion*, making the algorithms studied in [19] unapplicable. We verified some instances of the *Crowds protocol* [26] against a quantitative anonymity specification called *probable innocence* [21]. We used a general trace-based verification method in [15] for probable innocence: language inclusion $L(\mathcal{A}_\mathsf{P}) \sqsubseteq L(\mathcal{A}_\mathsf{S})$, from the model $\mathcal{A}_\mathsf{P}$ of a protocol in question to $\mathcal{A}_\mathsf{P}$'s suitable modification $\mathcal{A}_\mathsf{S}$, guarantees probable innocence.

param. n	c	p_f	$\mathcal{A}_\mathsf{P}$ #st.	$\mathcal{A}_\mathsf{P}$ #tr.	$\mathcal{A}_\mathsf{S}$ #st.	$\mathcal{A}_\mathsf{S}$ #tr.	$\lvert\Sigma\rvert$	direction fwd./bwd.	time (sec)	space (GB)	d
5	1	$\frac{9}{10}$	7	44	7	56	18	$\mathcal{A}_\mathsf{P} \sqsubseteq_\mathbf{F} \mathcal{A}_\mathsf{S}$	52.48	0.01	2
								$\mathcal{A}_\mathsf{P} \sqsubseteq_\mathbf{B} \mathcal{A}_\mathsf{S}$	0.01	0.01	2
7	1	$\frac{3}{4}$	9	88	9	118	26	$\mathcal{A}_\mathsf{P} \sqsubseteq_\mathbf{F} \mathcal{A}_\mathsf{S}$	0.15	0.03	2
								$\mathcal{A}_\mathsf{P} \sqsubseteq_\mathbf{B} \mathcal{A}_\mathsf{S}$	0.02	0.01	2
10	2	$\frac{4}{5}$	12	224	12	280	54	$\mathcal{A}_\mathsf{P} \sqsubseteq_\mathbf{F} \mathcal{A}_\mathsf{S}$	802.47	0.35	2
								$\mathcal{A}_\mathsf{P} \sqsubseteq_\mathbf{B} \mathcal{A}_\mathsf{S}$	0.05	0.03	2
20	6	$\frac{4}{5}$	22	1514	22	1696	238	$\mathcal{A}_\mathsf{P} \sqsubseteq_\mathbf{F} \mathcal{A}_\mathsf{S}$	T/O		2
								$\mathcal{A}_\mathsf{P} \sqsubseteq_\mathbf{B} \mathcal{A}_\mathsf{S}$	1.32	0.78	2
30	6	$\frac{4}{5}$	32	4732	32	5112	550	$\mathcal{A}_\mathsf{P} \sqsubseteq_\mathbf{F} \mathcal{A}_\mathsf{S}$	S/F		
								$\mathcal{A}_\mathsf{P} \sqsubseteq_\mathbf{B} \mathcal{A}_\mathsf{S}$	11.84	5.99	2

Table 2. Results for the Crowds protocol

The Crowds protocol has parameters n, c and p_f. In fact, for this specific protocol, a sufficient condition for probable innocence is known [26] (namely $n \geq \frac{p_f}{p_f - 1/2}(c+1)$); we used parameters that satisfy this condition. We implemented a small program that takes a choice of n, c, p_f and generates an automaton $\mathcal{A}_{\mathsf{P}}$; it is then passed to another program that generates $\mathcal{A}_{\mathsf{S}}$.

The results are in Table. 2. For each problem instance we tried both $\sqsubseteq_{\mathbf{F}}$ and $\sqsubseteq_{\mathbf{B}}$. The last column shows the final value of the parameter d for $+\times$-PE —i.e. how many times partial execution (§4) was applied.

The entry "S/F" designates that $+\times$-PE was killed because of segmentation fault caused by an oversized automaton. "T/O" means that alternate application of $+\times$-sim and $+\times$-PE did not terminate within a time limit (one hour).

We observe that backward simulation matrices were much faster to be found than forward ones. This seems to result from the shapes of the automata for this specific problem; after all it is an advantage of our fwd./bwd. approach that we can try two different directions and use the faster one. Space consumption seems again serious.

6 Simulation Matrices for $\mathcal{S}_{\mathbf{max},+}$-Weighted Automata

In §6 we discuss $\mathcal{S}_{\max,+}$-weighted automata, in which weights are understood as (best-case) profit or (worst-case) cost (see Example 2.2). Such automata are studied in [7] (called Sum-*automata* there). In fact we observe that their notion of simulation—formulated in game-theoretic terms and hence called *G-simulation* here—coincides with fwd. simulation matrix. This observation is in §6.1; in §6.2 our implementation is presented.

6.1 G-Simulation by Forward Simulation Matrices

In this section we restrict to finite-state automata, in which case we can also dispose of the weight ∞. What we shall call *G-simulation* is introduced in [7], and its soundness with respect to weighted languages over *infinite-length words* $\Sigma^\omega \to [-\infty, \infty)$ is proved there. Here we adapt their definition to the current setting of finite-length words.

Definition 6.1 ($\sqsubseteq_{\mathbf{G}}$) Let $\mathcal{A} = (Q_{\mathcal{A}}, \Sigma, M_{\mathcal{A}}, \alpha_{\mathcal{A}}, \beta_{\mathcal{A}})$ and $\mathcal{B} = (Q_{\mathcal{B}}, \Sigma, M_{\mathcal{B}}, \alpha_{\mathcal{B}}, \beta_{\mathcal{B}})$ be finite-state $\mathcal{S}_{\max,+}$-weighted automata. A *finite simulation game* from $\mathcal{A}$ to $\mathcal{B}$ is played by two players called *Challenger* and *Simulator*: a *strategy for Challenger* is a pair $(\rho_1 : 1 \to Q_{\mathcal{A}}, \tau_1 : (Q_{\mathcal{A}} \times Q_{\mathcal{B}})^+ \to 1 + \Sigma \times Q_{\mathcal{A}})$ of functions; a *strategy for Simulator* is a pair $(\rho_2 : Q_{\mathcal{A}} \to Q_{\mathcal{B}}, \tau_2 : (Q_{\mathcal{A}} \times Q_{\mathcal{B}})^+ \times \Sigma \times Q_{\mathcal{A}} \to Q_{\mathcal{B}})$.

A pair $(p_0 a_1 \dots a_n p_n, q_0 a_1 \dots a_n q_n)$ of runs on $\mathcal{A}$ and $\mathcal{B}$ is called the *outcome of strategies* (ρ_1, τ_1) *and* (ρ_2, τ_2) if:

- $\rho_1(\bullet) = p_0$, $\rho_2(p_0) = q_0$ and $\tau_1((p_0, q_0) \dots (p_n, q_n)) = \checkmark$.
- $\tau_1\big((p_0, q_0) \dots (p_i, q_i)\big) = (a_{i+1}, p_{i+1})$ and $\tau_2\big((p_0, q_0) \dots (p_i, q_i), (a_{i+1}, p_{i+1})\big) = q_{i+1}$, for each $i \in [0, n-1]$.

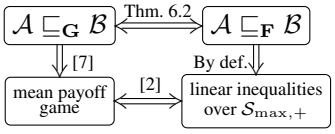
A strategy (ρ_1, τ_1) for Challenger is *winning* if for any strategy (ρ_2, τ_2) for Simulator, their outcome (r_1, r_2) satisfies $L(\mathcal{A})(r_1) > L(\mathcal{B})(r_2)$. Here the weight $L(\mathcal{A})(r)$ of a run r is defined in the obvious way.

Finally, we write $\mathcal{A} \sqsubseteq_{\mathbf{G}} \mathcal{B}$ if there is no winning strategy for Challenger.

Theorem 6.2 *Let $\mathcal{A}$ and $\mathcal{B}$ be finite-state $\mathcal{S}_{\max,+}$-weighted automata. Assume that $\mathcal{A}$ has no trap states, that is, every state has a path to $\checkmark$ whose weight is not $-\infty$. Then, $\mathcal{A} \sqsubseteq_{\mathbf{F}} \mathcal{B}$ if and only if $\mathcal{A} \sqsubseteq_{\mathbf{G}} \mathcal{B}$.* □

The extra assumption can be easily enforced by eliminating trap states through backward reachability check. This does not change the (finite) weighted language.

Now the situation is as shown on the right. It follows from [7] that the question if $\mathcal{A} \sqsubseteq_{\mathbf{G}} \mathcal{B}$ holds or not is reduced to a *mean payoff game* [10], whose decision problem is in $\mathbf{NP} \cap \mathbf{co\text{-}NP}$ and has a pseudo polynomial-time algorithm [28]. Moreover it is known that the decision problem of mean payoff games is equivalent to the feasibility problem of linear inequalities over $\mathcal{S}_{\max,+}$ [2]. For the latter an algorithm is proposed in [6] that is shown in [4] to be superpolynomial.



Similarly to $\mathcal{S}_{+,\times}$-weighted automata, language inclusion between $\mathcal{S}_{\max,+}$-weighted automata is known to be undecidable [22]. We note that, by Thm. 6.2, applying FPE or BPE (§4) increases the likelihood of $\sqsubseteq_{\mathbf{G}}$ in the sense of Thm. 4.3. We additionally note that, by exploiting symmetry of fwd. and bwd. simulation matrices (Rem. 3.7), we can define "backward G-simulation" as a variation of Def. 6.1.

6.2 Implementation, Experiments and Discussions

We implemented two programs: $\max+$-sim and $\max+$-PE.

- We have seen that finding simulation matrices can be reduced to some problems that have known algorithms. Since we did not find actual software available, we implemented (in C++) the algorithm in [6] as the program $\max+$-sim. It transforms the constraints in Def. 3.4 into an inequality $A\boldsymbol{x} \leq B\boldsymbol{x}$, which in turn is made into a linear equality $A'\boldsymbol{x'} = B'\boldsymbol{x'}$ by adding slack variables. The last equality is solved by the algorithm in [6].
- $\max+$-PE is as in §5. It simply uses the whole state space as the parameter P.

Experiments were done on an Ubuntu Linux laptop with a Core 2 duo processor (1.40 GHz, 2 cores) and 2 GB RAM. There we faced a difficulty of finding a benchmark example: although small examples are not hard to come up with by human efforts, we could not find a good example that has parameters (like G, S in Table 1) and allows for experiments with problem instances of a varying size.

We therefore ran $\max+$-sim for:

- the problem if $\mathcal{A} \sqsubseteq_{\mathbf{F}} \mathcal{A}$ for randomly generated $\mathcal{A}$, and
- the problem if $\mathcal{A} \sqsubseteq_{\mathbf{F}} \mathcal{B}$ for randomly generated $\mathcal{A}, \mathcal{B}$,

and measured time and memory consumption. Although the answers are known by construction (positive for the former, and almost surely negative for the latter), actual calculation via linear inequality constraints gives us an idea about resource consumption of our simulation-based method when it is applied to real-world problems.

The outcome is as shown in Fig. 3. The parameter p is the probability with which an a-transition exists given a source state, a target state, and a character $a \in \Sigma$. Its weight is chosen from $\{0, 1, \ldots, 16\}$ subject to the uniform distribution. "Same" means checking $\mathcal{A} \sqsubseteq_{\mathbf{F}} \mathcal{A}$ and "difference" means checking $\mathcal{A} \sqsubseteq_{\mathbf{F}} \mathcal{B}$ (see above). The two problem settings resulted in comparable performance.

We observe that space consumption is not so big a problem as in the $\mathcal{S}_{+,\times}$ case (§5.2). Somehow unexpectedly, there is no big performance gap between the sparse case ($p = 0.1$) and the dense case ($p = 0.9$); in fact the sparse case consumes slightly more

memory. Consumption of both time and space grows faster than linearly, which poses a question about the scalability of our approach. That said, our current implementation of the algorithm in [6] leaves a lot of room for further optimization: one possibility is use of dynamic programming (DP). After all, it is an advantage of our approach that a simulation problem is reduced to linear inequality constraints, a subject of extensive research efforts (cf. §5.1 and §6.1).

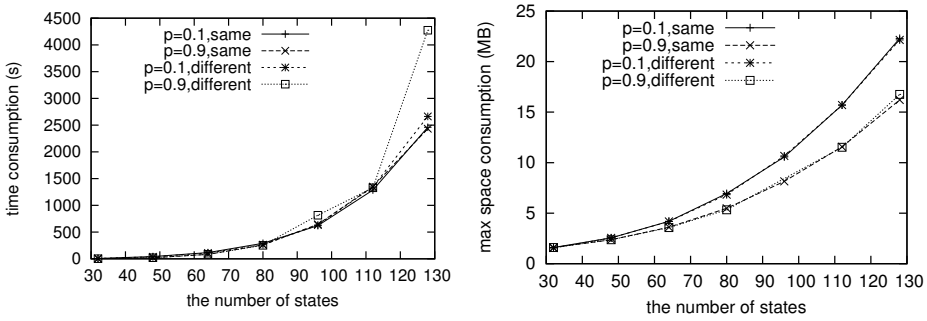


Fig. 3. Time and max space consumption for $\max+$-sim

7 Conclusions and Future Work

We introduced simulation matrices for weighted automata. While they are instances of (categorical) Kleisli simulations, their concrete presentation by matrices and linear inequalities yields concrete algorithms for simulation-based quantitative verification.

There are some directions in which the current matrix-based simulation framework can be further generalized. Generalizing F from $1 + \Sigma \times (\_)$ to any polynomial functor is mostly straightforward, as we noted after Lem. 3.3. We have not done that mainly for the space reason. Generalizing T from $\mathcal{M}_{\mathcal{S}}$ (for semiring-weighted branching) to others seems more challenging. For example, in [7] weights can be given from an algebraic structure $\mathcal{S}$ in which an additive unit $0_{\mathcal{S}}$ does not satisfy $0_{\mathcal{S}} \times_{\mathcal{S}} x = 0_{\mathcal{S}}$. In this case operations like matrix multiplication becomes hard to define.

Another direction is to incorporate infinite traces, which is done in [7]. In fact our current coalgebraic backend [11] fails to do so; the work [8] will be useful here. Finally, further optimization of our implementation is obvious future work.

Acknowledgments Thanks are due to Shota Nakagawa and the anonymous referees for useful discussions and comments. The authors are supported by Grants-in-Aid for Young Scientists (A) No. 24680001, JSPS.

References

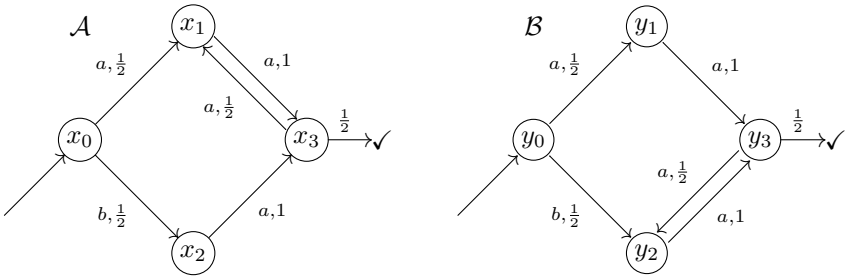
1. The GNU linear programming kit, http://www.gnu.org/software/glpk
2. Akian, M., Gaubert, S., Guterman, A.E.: Tropical polyhedra are equivalent to mean payoff games. International Journal of Algebra and Computation 22(1) (2012)
3. Baier, C., Hermanns, H., Katoen, J.P.: Probabilistic weak simulation is decidable in polynomial time. Inf. Process. Lett. 89(3), 123–130 (2004)
4. Bezem, M., Nieuwenhuis, R., Rodríguez-Carbonell, E.: Exponential behaviour of the butkovic-zimmermann algorithm for solving two-sided linear systems in max-algebra. Discrete Applied Mathematics 156(18), 3506–3509 (2008)

5. Blondel, V.D., Canterini, V.: Undecidable problems for probabilistic automata of fixed dimension. Theory Comput. Syst. 36(3), 231–245 (2003)
6. Butkovic, P., Zimmermann, K.: A strongly polynomial algorithm for solving two-sided linear systems in max-algebra. Discrete Applied Math. 154(3), 437–446 (2006)
7. Chatterjee, K., Doyen, L., Henzinger, T.A.: Expressiveness and closure properties for quantitative languages. Logical Methods in Computer Science 6(3) (2010)
8. Cîrstea, C.: Maximal traces and path-based coalgebraic temporal logics. Theor. Comput. Sci. 412(38), 5025–5042 (2011)
9. Doyen, L., Henzinger, T.A., Raskin, J.F.: Equivalence of labeled markov chains. Int. J. Found. Comput. Sci. 19(3), 549–563 (2008)
10. Ehrenfeucht, A., Mycielski, J.: Positional strategies for mean payoff games. International Journal of Game Theory 8(2), 109–113 (1979)
11. Hasuo, I.: Generic forward and backward simulations. In: Baier, C., Hermanns, H. (eds.) CONCUR. Lect. Notes Comp. Sci., vol. 4137, pp. 406–420. Springer (2006)
12. Hasuo, I.: Generic forward and backward simulations II: Probabilistic simulation. In: Gastin, P., Laroussinie, F. (eds.) CONCUR. LNCS, vol. 6269, pp. 447–461. Springer (2010)
13. Hasuo, I., Jacobs, B.: Context-free languages via coalgebraic trace semantics. In: Fiadeiro, J.L., Harman, N., Roggenbach, M., Rutten, J.J.M.M. (eds.) CALCO. Lecture Notes in Computer Science, vol. 3629, pp. 213–231. Springer (2005)
14. Hasuo, I., Jacobs, B., Sokolova, A.: Generic trace semantics via coinduction. Logical Methods in Computer Science 3(4) (2007)
15. Hasuo, I., Kawabe, Y., Sakurada, H.: Probabilistic anonymity via coalgebraic simulations. Theor. Comput. Sci. 411(22-24), 2239–2259 (2010)
16. Hughes, J., Jacobs, B.: Simulations in coalgebra. Theor. Comput. Sci. 327(1-2), 71–108 (2004)
17. Jacobs, B.: Introduction to coalgebra. Towards mathematics of states and observations (2012), Draft of a book (ver. 2.0), available online
18. Jonsson, B., Larsen, K.G.: Specification and refinement of probabilistic processes. In: LICS. pp. 266–277. IEEE Computer Society (1991)
19. Kiefer, S., Murawski, A.S., Ouaknine, J., Wachter, B., Worrell, J.: Language equivalence for probabilistic automata. In: Gopalakrishnan, G., Qadeer, S. (eds.) CAV. Lecture Notes in Computer Science, vol. 6806, pp. 526–540. Springer (2011)
20. Kiefer, S., Murawski, A.S., Ouaknine, J., Wachter, B., Worrell, J.: Algorithmic probabilistic game semantics—playing games with automata. Formal Methods in System Design 43(2), 285–312 (2013)
21. Konstantinos, C., Catuscia, P.: Probable innocence revisited. Theoretical Computer Science 367(1), 123–138 (2006)
22. Krob, D.: The equality problem for rational series with multiplicities in the tropical semiring is undecidable. In: Kuich, W. (ed.) ICALP. Lecture Notes in Computer Science, vol. 623, pp. 101–112. Springer (1992)
23. Lynch, N.A., Vaandrager, F.W.: Forward and backward simulations: I. Untimed systems. Inf. Comput. 121(2), 214–233 (1995)
24. Pin, J.E.: Tropical semirings. Idempotency (Bristol, 1994) pp. 50–69 (1998)
25. Power, J., Turi, D.: A coalgebraic foundation for linear time semantics. Electr. Notes Theor. Comput. Sci. 29, 259–274 (1999)
26. Reiter, M.K., Rubin, A.D.: Crowds: Anonymity for web transactions. ACM Trans. Inf. Syst. Secur. 1(1), 66–92 (1998)
27. Tzeng, W.G.: A polynomial-time algorithm for the equivalence of probabilistic automata. SIAM J. Comput. 21(2), 216–227 (1992)
28. Zwick, U., Paterson, M.: The complexity of mean payoff games on graphs. Theor. Comput. Sci. 158(1&2), 343–359 (1996)

A Appendix

A.1 Examples and Counterexamples

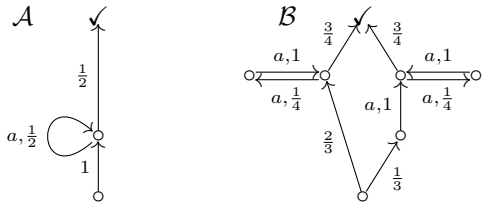
Example A.1 The following $\mathcal{S}_{+,\times}$-weighted automata exhibit $L(\mathcal{A}) \sqsubseteq L(\mathcal{B})$, but there is no forward or backward simulation from $\mathcal{A}$ to $\mathcal{B}$—as is shown by the fact that there is no X that satisfies the requirements in Def. 3.4. Hence this pair is a counterexample for the completeness of $\sqsubseteq_{\mathbf{F}}$ and that of $\sqsubseteq_{\mathbf{B}}$. In contrast, there exists a simulation by Jonsson and Larsen [18] between them.



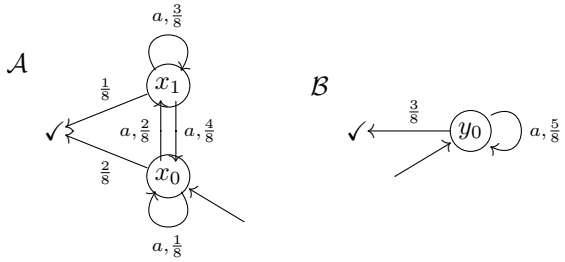
Language inclusion (in fact language equivalence) is shown as follows. For each word $w \in \Sigma^*$,

$$L(\mathcal{A})(w) = L(\mathcal{B})(w) = \begin{cases} \frac{1}{4}\left(\frac{1}{2}\right)^n & (w = aaa^{2n} \text{ or } baa^{2n}) \\ 0 & (\text{otherwise}) \ . \end{cases}$$

Example A.2 The following $\mathcal{S}_{+,\times}$-weighted automata exhibit $L(\mathcal{A}) \sqsubseteq L(\mathcal{B})$, but a forward simulation does not exist no matter how many times FPE is applied to $\mathcal{A}$.



Example A.3 The following $\mathcal{S}_{+,\times}$-weighted automata exhibit $L(\mathcal{A}) \sqsubseteq L(\mathcal{B})$. Neither forward nor backward Kleisli simulation exists between them if we represent them as $(\mathcal{D}, 1+\Sigma\times(\_))$-systems while both can be found if we represent as $(\mathcal{M}_{\mathcal{S}}, 1+\Sigma\times(\_))$-systems.



The only X satisfying the inequalities in Def. 3.4.1 is $X = \begin{pmatrix}1 & 1\end{pmatrix}$, and the only X satisfying the inequalities in Def. 3.4.2 is $X = \begin{pmatrix}1\\2\end{pmatrix}$.

A.2 Action of $1 + \Sigma \times (\_)$ on Kleisli arrows

We have used an operation $1 + \Sigma \times (\_)$ acting on Kleisli arrows. It is defined as the functor $1+\Sigma\times(\_)$ on $\mathbf{Sets}$ lifted to the Kleisli category $\mathcal{K}\ell(\mathcal{M}_{\mathcal{S}})$; see [14] for details. Here we describe it in concrete terms.

Definition A.4 For a Kleisli arrow $f\colon X \nrightarrow Y$, its lifting $1 + \Sigma \times f\colon 1 + \Sigma \times X \nrightarrow 1 + \Sigma \times Y$ is defined as follows:

$$(1+\Sigma\times f)(*)(t) = \begin{cases} 1_{\mathcal{S}} & (t = *) \\ 0_{\mathcal{S}} & (\text{otherwise}) \end{cases} \qquad (1+\Sigma\times f)(a,x)(t) = \begin{cases} f(x)(y) & (t = (a,y)) \\ 0_{\mathcal{S}} & (\text{otherwise}) \end{cases}$$

Lem. 3.3 follows immediately from this concrete description.

A.3 Formal Definitions of Forward/Backward Partial Execution

We formally describe Def. 4.1.

Definition A.5 (FPE, BPE) *Forward partial execution (FPE)* is a transformation of a weighted automaton that "replaces some states with their one-step behaviors." Precisely, given an $\mathcal{S}$-weighted automaton $\mathcal{A} = (Q, \Sigma, M, \alpha, \beta)$ and a parameter $P \subseteq Q$, the resulting automaton $\mathcal{A}_{\mathsf{FPE},P} = (Q', \Sigma, M', \alpha', \beta')$ has a state space

$$Q' \;=\; \big\{\checkmark \,\big|\, \exists x \in P.\, \beta_x \neq 0_{\mathcal{S}}\big\} + \big\{(a,y) \,\big|\, \exists x \in P.\, M(a)_{x,y} \neq 0_{\mathcal{S}}\big\} + (Q \setminus P) \;\;,$$

replacing each $p \in P$ with its one-step behaviors ($\checkmark$ or (a,q)) as new states. The other data M', α', β' are defined as follows.

$$\begin{array}{ccccc} M'(a)_{(a,x),\checkmark} = \beta_x & & M'(a)_{(a,x),(a',y)} = M(a')_{x,y} & & M'(a)_{(a,x),x} = 1_{\mathcal{S}} \\ M'(a)_{x,\checkmark} = \big(M(a)\beta\big)_x & & M'(a)_{x,(a',y)} = (M(a)M(a'))_{x,y} & & \\ M'(a)_{x,y} = M(a)_{x,y} & & M(a)_{u,v} = 0_{\mathcal{S}} \quad (\text{otherwise}) & & \end{array}$$

$$\alpha'_{\checkmark} = \alpha\beta \qquad \alpha'_{(a,x)} = (\alpha M(a))_x \qquad \alpha'_x = \alpha_x$$

$$\beta'_{\checkmark} = 1_{\mathcal{S}} \qquad \beta_{(a,x)} = 0_{\mathcal{S}} \qquad \beta'_x = \beta_x$$

Backward partial execution (BPE) in contrast "replaces states in a parameter $P \subseteq Q$ with their backward one-step behaviors." For the same $\mathcal{A}$ as above, the resulting automaton $\mathcal{A}_{\mathsf{BPE},P} = (Q', \Sigma, M', \alpha', \beta')$ has a state space

$$Q' \;=\; \big\{\bullet \,\big|\, \exists x \in P.\, \alpha_x \neq 0_{\mathcal{S}}\big\} + \big\{(a,y) \,\big|\, \exists x \in P.\, M(a)_{y,x} \neq 0_{\mathcal{S}}\big\} + (Q \setminus P) \;\;,$$

replacing each $p \in P$ with its backward one-step behaviors—(a, q) with $q \xrightarrow{a} p$, and $\bullet$ if p is initial—as new states. The definitions of M', α', β' are as follows.

$$\begin{array}{lll} M'(a)_{\bullet,(a,y)} = \beta_y & M'(a)_{(a',x),(a,y)} = M(a')_{x,y} & M'(a)_{x,(a,x)} = 1_{\mathcal{S}} \\ M'(a)_{\bullet,y} = \big(\alpha M(a)\big)_y & M'(a)_{(a',x),y} = (M(a')M(a))_{x,y} & \\ M'(a)_{x,y} = M(a)_{x,y} & M(a)_{u,v} = 0_{\mathcal{S}} \quad \text{(otherwise)} & \end{array}$$

$$\alpha'_{\bullet} = 1_{\mathcal{S}} \qquad \alpha_{(a,x)} = 0_{\mathcal{S}} \qquad \alpha'_x = \alpha_x$$

$$\beta'_{\bullet} = \alpha\beta \qquad \beta'_{(a,x)} = (M(a)\beta)_x \qquad \beta'_x = \beta_x$$

A.4 Correctness of Forward/Backward Partial Execution

Lemma A.6 *For each subset P, we have the following.*
1. $\mathcal{A} \sqsubseteq_{\mathbf{B}} \mathcal{A}_{\mathsf{FPE},P}$ *2.* $\mathcal{A}_{\mathsf{FPE},P} \sqsubseteq_{\mathbf{F}} \mathcal{A}$ *3.* $\mathcal{A} \sqsubseteq_{\mathbf{F}} \mathcal{A}_{\mathsf{BPE},P}$ *4.* $\mathcal{A}_{\mathsf{BPE},P} \sqsubseteq_{\mathbf{B}} \mathcal{A}$

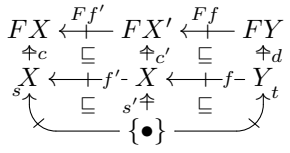
Proof. 1. Let $\mathcal{A} = (Q, \Sigma, M, \alpha, \beta)$ and $\mathcal{A}_{\mathsf{FPE},P} = (Q', \Sigma, M', \alpha', \beta')$. We define $X \in S^{Q \times Q'}$ as follows:

$$\begin{array}{ll} X_{x,\checkmark} = \beta_x \ (x \in P), & X_{x,(y,a)} = M(a)_{x,y} \ (x \in P), \\ X_{x,x} = 1_{\mathcal{S}} \ (x \notin P), & X_{u,v} = 0_{\mathcal{S}} \qquad \text{(otherwise)} \end{array}$$

Then, this X is a backward simulation matrix from $\mathcal{A}$ to $\mathcal{A}_{\mathsf{FPE},P}$
The items 2. , 3. and 4. can be proved similarly. □

Proof of Thm. 4.2 1. From Prop. A.6.1, $\mathcal{A} \sqsubseteq_{\mathbf{B}} \mathcal{A}_{\mathsf{FPE},P} \sqsubseteq_{\mathbf{F}} \mathcal{B}_{\mathsf{BPE},P'} \sqsubseteq_{\mathbf{B}} \mathcal{B}$. Hence from Cor. 3.6, $L(\mathcal{A}) \sqsubseteq L(\mathcal{A}_{\mathsf{FPE},P}) \sqsubseteq L(\mathcal{B}_{\mathsf{BPE},P'}) \sqsubseteq L(\mathcal{B})$ holds.
The item 2. can be proved similarly. □

Proof of Thm. 4.3 1. From Prop. A.6.2, $\mathcal{A}_{\mathsf{FPE},P} \sqsubseteq_{\mathbf{F}} \mathcal{A} \sqsubseteq_{\mathbf{F}} \mathcal{B} \sqsubseteq_{\mathbf{F}} \mathcal{B}_{\mathsf{BPE},P'}$. Because $\sqsubseteq_{\mathbf{F}}$ satisfies transitivity rule (see the diagram on the right where $F = 1 + \Sigma \times (\_)$), this implies $\mathcal{A}_{\mathsf{FPE},P} \sqsubseteq_{\mathbf{F}} \mathcal{B}_{\mathsf{BPE},P'}$.
The item 2 can be proved similarly. □



A.5 Categorical Forward Partial Execution

Here we describe FPE in §4 in categorical terms of (T, F)-systems, and prove its correctness results (namely soundness and adequacy) in this general setting.

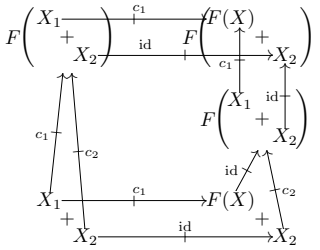
Definition A.7 *Forward partial execution* (FPE) for (T, F)-systems is a transformation that takes a (T, F)-system $\mathcal{X} = (X, s, c)$ and a parameter $X_1 \subseteq X$ as inputs and returns a (T, F)-system $\mathcal{X}_{\mathsf{FPE},X_1} = (F(X) + X_2, (c_1 + \mathrm{id}) \odot s, \overline{F}(c_1 + \mathrm{id}) \odot [\mathrm{id}, c_2])$. ($X = X_1 + X_2$, $c_1 = c \odot \kappa_1$, and $c_2 = c \odot \kappa_2$ where κ_1 and κ_2 are injections. Moreover, $\overline{F}\colon \mathcal{K}\ell(T) \to \mathcal{K}\ell(T)$ is a lifted functor of $F : \mathbf{Sets} \to \mathbf{Sets}$ (see [17] for concrete definition)).

Lem A.8, Thm. A.9, Thm. A.10, and Prop. A.11 are categorical analogies of Lem. A.6, Thm. 4.2, Thm. 4.3, and Prop. 4.4 respectively.

Proposition A.8 *For each subset X_1, we have the following.*
1. $\mathcal{X} \sqsubseteq_{\mathbf{B}} \mathcal{X}_{\mathsf{FPE},X_1}$. *2.* $\mathcal{X}_{\mathsf{FPE},X_1} \sqsubseteq_{\mathbf{F}} \mathcal{X}$.

Proof. 1. We define $g : X_1 + X_2 \nrightarrow F(X) + X_2$ as $g = c_1 + \mathrm{id}$. Then,

$$\begin{aligned}\overline{F}(g) \odot c &= \overline{F}(c_1 + \mathrm{id}) \odot [c_1, c_2] \\ &= \overline{F}(c_1 + \mathrm{id}) \odot [\mathrm{id}, c_2] \odot (c_1 + \mathrm{id}) \\ &= (\overline{F}(c_1 + \mathrm{id}) \odot [\mathrm{id}, c_2]) \odot g \\ g \odot s &= (c_1 + \mathrm{id}) \odot s \ .\end{aligned}$$



Hence g is a backward simulation from $\mathcal{X}$ to $\mathcal{X}_{\mathsf{FPE}}$.

The item 2. can be proved similarly (g in the proof of the item 1. is also a forward simulation from $\mathcal{X}_{\mathsf{FPE}}$ to $\mathcal{X}$). □

Theorem A.9 (soundness of FPE, categorically) *Let X_1, Y_1 be arbitrary subsets. Each of the following implies* $\mathsf{tr}(\mathcal{X}) \sqsubseteq \mathsf{tr}(\mathcal{Y})$.
1. $\mathcal{A}_{\mathsf{FPE},X_1} \sqsubseteq_{\mathbf{F}} \mathcal{Y}$ *2.* $\mathcal{A} \sqsubseteq_{\mathbf{B}} \mathcal{B}_{\mathsf{FPE},Y_1}$ □

Theorem A.10 (adequacy of FPE, categorically) *Let X_1, Y_1 be arbitrary subsets. We have:*
1. $\mathcal{X} \sqsubseteq_{\mathbf{F}} \mathcal{Y} \Rightarrow \mathcal{X}_{\mathsf{FPE},X_1} \sqsubseteq_{\mathbf{F}} \mathcal{Y}$ *2.* $\mathcal{X} \sqsubseteq_{\mathbf{B}} \mathcal{Y} \Rightarrow \mathcal{X} \sqsubseteq_{\mathbf{B}} \mathcal{B}_{\mathsf{FPE},Y_1}$. □

Proposition A.11 (monotonicity, categorically) *Assume $X_1 \subseteq X_1'$ and $X_2 \subseteq X_2'$. We have:*
1. $\mathcal{X}_{\mathsf{FPE},X_1} \sqsubseteq_{\mathbf{F}} \mathcal{Y} \Rightarrow \mathcal{X}_{\mathsf{FPE},X_1'} \sqsubseteq_{\mathbf{F}} \mathcal{Y}$, *2.* $\mathcal{X} \sqsubseteq_{\mathbf{B}} \mathcal{Y}_{\mathsf{FPE},Y_2} \Rightarrow \mathcal{X} \sqsubseteq_{\mathbf{B}} \mathcal{Y}_{\mathsf{FPE},Y_2'}$. □

A.6 Proof of Thm. 6.2

Before proving Thm.6.2, we review the notions of Limavg *automaton* [7] and *mean payoff game* [10], which is used in the proof of Thm. 6.2.

Definition A.12 A Limavg *automaton* $\mathcal{A}^{\mathsf{Limavg}} = (Q, \Sigma, M, q_0)$ consists of a finite state space Q, a finite alphabet Σ, transition matrices $M : \Sigma \to [-\infty, \infty)^{Q \times Q}$, and the initial state $q_0 \in Q$. For an infinite word $w = a_0 a_1 \cdots \in \Sigma^\omega$, $\mathcal{A}^{\mathsf{Limavg}}$ assigns a value $L(\mathcal{A}^{\mathsf{Limavg}})(w) := \sup_{q_1 q_2 \cdots \in Q^\infty} \mathrm{liminf}_{N \to \infty} \frac{1}{N} \sum_{i=0}^{N} M(a_i)_{q_i, q_{i+1}}$

G-simulation for Limavg automaton can be defined similarly as Def. 6.1.

Definition A.13 A *mean payoff game* is a game played by two players Min and Max on a parity graph $\mathcal{G} = (Q_{\mathrm{Min}}, Q_{\mathrm{Max}}, q_I, E, \gamma)$ consisting of *states of* Min Q_{Min}, *states of* Max Q_{Max}, the initial state $q_I \in Q_{\mathrm{Min}}$, edges $E = Q_{\mathrm{Min}} \times Q_{\mathrm{Max}} + Q_{\mathrm{Max}} \times Q_{\mathrm{Min}}$, and a weight function $\gamma : E \to \mathbb{R}$. A *strategy* for Min and Max are functions $\tau_{\mathrm{Min}} : (Q_{\mathrm{Min}} \times Q_{\mathrm{Max}})^* \times Q_{\mathrm{Min}} \to Q_{\mathrm{Max}}$ and $\tau_{\mathrm{Max}} : (Q_{\mathrm{Min}} \times Q_{\mathrm{Max}})^+ \to Q_{\mathrm{Min}}$ respectively. An infinite run $p_0 q_0 p_1 q_1 \ldots$ on $\mathcal{G}$ is the *outcome* of τ_{Min} and τ_{Max} if $p_0 = q_I$, $p_0, p_1, \cdots \in Q_{\mathrm{Min}}$, $q_0, q_1, \cdots \in Q_{\mathrm{Max}}$, and for any $i \geq 1$, $\tau_{\mathrm{Min}}(p_0 q_0 \ldots q_{i-1} p_i) = q_i$ and $\tau_{\mathrm{Min}}(p_0 q_0 \ldots p_i q_i) = p_{i+1}$. A strategy τ_{Max} for Max is *winning* if for any strategy τ_{Min} for Min, their outcome $r_0 r_1 r_2 r_3 \ldots$ satisfies $\mathrm{liminf}_{N \to \infty} \frac{1}{N} \sum_{i=0}^{N} \big(\gamma(r_i, r_{i+1})\big) \geq 0$.

Using these notions, Thm. 6.2 is derived as follows.

1. There exists a linear inequality $A\boldsymbol{x} \leq B\boldsymbol{x}$ in $\mathcal{S}_{\max,+}$ such that forward Kleisli simulation from $\mathcal{A}$ to $\mathcal{B}$ exists if and only if $A\boldsymbol{x} \leq B\boldsymbol{x}$ has a non-trivial (i.e. $\boldsymbol{x} \neq -\infty$) solution.
2. ([2]) There exists a mean payoff game $\mathcal{MP}$ such that existence of a non-trivial solution in $A\boldsymbol{x} \leq B\boldsymbol{x}$ is equivalent to the existence of winning strategy for Max in $\mathcal{MP}$.
3. There exists a Limavg automaton $\mathcal{A}^{\mathsf{Limavg}}$ and $\mathcal{B}^{\mathsf{Limavg}}$ such that G-simulation from $\mathcal{A}$ to $\mathcal{B}$ exists if and only if G-simulation from $\mathcal{A}^{\mathsf{Limavg}}$ to $\mathcal{B}^{\mathsf{Limavg}}$ exits.
4. ([7]) G-simulation from $\mathcal{A}^{\mathsf{Limavg}}$ to $\mathcal{B}^{\mathsf{Limavg}}$ exists if and only if there exists a winning strategy for Max in $\mathcal{MP}$.

Proof of Thm. 6.2 Let $\mathcal{A} = (Q_{\mathcal{A}}, \Sigma, M_{\mathcal{A}}, \alpha_{\mathcal{A}}, \beta_{\mathcal{A}})$ and $\mathcal{B} = (Q_{\mathcal{B}}, \Sigma, M_{\mathcal{B}}, \alpha_{\mathcal{B}}, \beta_{\mathcal{B}})$.
1 : By definition, forward simulation matrix is a matrix X that satisfies below:

$$\alpha_{\mathcal{A}} \leq \alpha_{\mathcal{B}} X \ \wedge\ \forall a \in \Sigma.\, X M_{\mathcal{A}}(a) \leq M_{\mathcal{B}}(a) X \ \wedge\ X\beta_{\mathcal{A}} \leq \beta_{\mathcal{B}}\,. \tag{6}$$

Adding a new variable $x_{\star,\star}$, such X exists iff there exist non-trivial X' and z satisfying

$$\alpha_{\mathcal{A}} + x_{\star,\star} \leq \alpha_{\mathcal{B}} X' \ \wedge\ \forall a \in \Sigma.\, X' M_{\mathcal{A}}(a) \leq M_{\mathcal{B}}(a) X' \ \wedge\ X'\beta_{\mathcal{A}} \leq \beta_{\mathcal{B}} + x_{\star,\star}\,. \tag{7}$$

(Only if part is easy. Assume non-trivial X' and z satisfying (7) exist. If $x_{\star,\star} \neq -\infty$, then $X = X' - x_{\star,\star}$ satisfies (6). If $x_{\star,\star} = -\infty$, by the assumption that $\mathcal{A}$ has no trap state, all entries of X' are $-\infty$, hence X' and $x_{\star,\star}$ are trivial solution. This contradicts.)
2 : From [2] and that $x_{\star,\star} \neq -\infty$ if (7) has a non-trivial solution, a non-trivial solution of (7) exists if and only if there is a winning strategy for player Max in a mean payoff game played on a graph $\mathcal{G} = (Q_{\mathrm{Min}}, Q_{\mathrm{Max}}, q_I, E, \gamma)$ such that

$$\begin{aligned}
Q_{\mathrm{Min}} &= \{x_{\star,\star}\} + \{x_{j,i} \mid j \in Q_{\mathcal{B}}, i \in Q_{\mathcal{A}}\}\,,\\
Q_{\mathrm{Max}} &= Q_{\mathcal{A}} + \Sigma \times Q_{\mathcal{B}} \times Q_{\mathcal{A}} + Q_{\mathcal{B}}\,,\\
q_I &= x_{\star,\star}\\
E &= E_1 + E_2 \text{ s.t. } E_1 \subseteq Q_{\mathrm{Min}} \times Q_{\mathrm{Max}}, E_2 \subseteq Q_{\mathrm{Max}} \times Q_{\mathrm{Min}} \text{ and}\\
&\quad E_1 = \{(x_{\star,\star}, i) \mid (\alpha_{\mathcal{A}})_i \neq -\infty\}\} + \{(x_{j,i}, (a,j,i')) \mid M_{\mathcal{A}}(a)_{i,i'} \neq -\infty\}\\
&\qquad + \{(x_{j,i}, j) \mid (\beta_{\mathcal{A}})_i \neq -\infty\}\\
&\quad E_2 = \{(i, x_{j,i}) \mid (\alpha_{\mathcal{B}})_j \neq -\infty\} + \{((a,j,i), x_{j',i}) \mid M_{\mathcal{B}}(a)_{j,j'} \neq -\infty\}\\
&\qquad + \{(j, x_{\star,\star}) \mid (\beta_{\mathcal{B}})_j \neq -\infty\}\,,
\end{aligned}$$

$$\begin{aligned}
&\gamma(x_{\star,\star}, i) = -(\alpha_{\mathcal{A}})_i, \qquad \gamma(x_{j,i}, (a,j,i')) = -M_{\mathcal{A}}(a)_{i,i'}, \qquad \gamma(x_{j,i}, j) = -(\beta_{\mathcal{A}})_i\\
&\gamma(i, x_{j,i}) = (\alpha_{\mathcal{B}})_j, \qquad \gamma((a,j,i), x_{j',i}) = M_{\mathcal{B}}(a)_{j,j'}, \qquad \gamma(j, x_{\star,\star}) = (\beta_{\mathcal{B}})_j\,.
\end{aligned}$$

where $i \in Q_{\mathcal{A}}, j \in Q_{\mathcal{B}}$. This game is equal to a mean payoff game $\mathcal{MP}$ played on a graph $\mathcal{G}' = (Q'_{\mathrm{Min}}, Q'_{\mathrm{Max}}, q'_I, E', \gamma')$ such that

$$
\begin{aligned}
Q'_{\mathrm{Min}} &= (\{\star\} + Q_{\mathcal{A}}) \times (\{\star\} + Q_{\mathcal{B}}) \,, \\
Q'_{\mathrm{Max}} &= (\{\star\} + Q_{\mathcal{A}}) \times (\{\star\} + Q_{\mathcal{B}}) \times (\{?\} + \{!\} + \Sigma) \,, \\
q'_I &= (\star, \star) \,, \\
E' &= E_1 + E_2 \text{ s.t. } E_1 \subseteq Q_{\mathrm{Min}} \times Q_{\mathrm{Max}}, E_2 \subseteq Q_{\mathrm{Max}} \times Q_{\mathrm{Min}} \text{ and} \\
&\quad E_1 = \{((\star,\star),(i,\star,?)) \mid (\alpha_{\mathcal{A}})_i \neq -\infty\} + \{((i,j),(i',j,a)) \mid M_{\mathcal{A}}(a)_{i,i'} \neq -\infty\} \\
&\qquad + \{((i,j),(\star,j,!)) \mid (\beta_{\mathcal{A}})_i \neq -\infty\} \\
&\quad E_2 = \{((i,\star,?),(i,j)) \mid (\alpha_{\mathcal{B}})_j \neq -\infty\} + \{((i,j,a),(i,j')) \mid M_{\mathcal{B}}(a)_{j,j'} \neq -\infty\} \\
&\qquad + \{((\star,j,!),(\star,\star)) \mid (\beta_{\mathcal{B}})_j \neq -\infty\} \,,
\end{aligned}
$$

$$
\gamma((\star,\star),(i,\star,?)) = -(\alpha_{\mathcal{A}})_i, \gamma((i,j),(i',j,a)) = -M_{\mathcal{A}}(a)_{i,i'}, \gamma((i,j),(\star,j,!)) = -(\beta_{\mathcal{A}})_i
$$
$$
\gamma((i,\star,?),(i,j)) = (\alpha_{\mathcal{B}})_j, \quad \gamma((i,j,a),(i,j')) = M_{\mathcal{B}}(a)_{j,j'}, \quad \gamma((\star,j,!),(\star,\star)) = (\beta_{\mathcal{B}})_j \,.
$$

3 : Let $\mathcal{A}^{\mathsf{Limavg}}$ and $\mathcal{B}^{\mathsf{Limavg}}$ be Limavg automata such that $\mathcal{A}^{\mathsf{Limavg}} = (\{\star\} + Q_{\mathcal{A}}, \{?\} + \{!\} + \Sigma, M'_{\mathcal{A}}, \star)$ and $\mathcal{B}^{\mathsf{Limavg}} = (\{\star\} + Q_{\mathcal{B}}, \{?\} + \{!\} + \Sigma, M'_{\mathcal{B}}, \star)$. (Here $M'_{\mathcal{A}}(a)_{x,y} = M_{\mathcal{A}}(a)_{x,y}$ $(a \in \Sigma, x, y \neq \star)$, $M'_{\mathcal{A}}(?)_{\star,y} = (\alpha_{\mathcal{A}})_y$, $M'_{\mathcal{A}}(!)_{x,\star} = (\beta_{\mathcal{A}})_x$ and $(M'_{\mathcal{A}}(a))_{x,y} = -\infty$ otherwise. $M'_{\mathcal{B}}$ is defined similarly.) We prove that a winning strategy for Challenger exists in a game played on $\mathcal{A}$ and $\mathcal{B}$ if and only if it exists in a game played on $\mathcal{A}^{\mathsf{Limavg}}$ and $\mathcal{B}^{\mathsf{Limavg}}$. Only if part is easy: by repeating winning strategy on $\mathcal{A}$ and $\mathcal{B}$,challenger can also win the game on $\mathcal{A}^{\mathsf{Limavg}}$ and $\mathcal{B}^{\mathsf{Limavg}}$. Assume that there exists a winning strategy τ for Challenger in the game played on $\mathcal{A}^{\mathsf{Limavg}}$ and $\mathcal{B}^{\mathsf{Limavg}}$. From [10], we can assume that τ is positional (i.e. $\tau((p_0,q_0)\dots(p_i,q_i))$ only depends on (p_i,q_i)). Because $\mathcal{A}$ has no trap state and $\mathcal{A}$ and $\mathcal{B}$ are finite, there exists $r \in \mathbb{R}$ such that for each $q \in Q_{\mathcal{A}}$, there is a path π_q to the final state in $\mathcal{A}$ such that for all path π from an arbitrary state to the final state in $\mathcal{B}$ on the same word, difference between the summation of weights on π_q and π is less than r. Using this r, we define strategy (ρ_1, τ_1) for Challenger. Because τ is a winning strategy, p_0 such that $\tau(\star,\star) = (?, p_0)$ exists. We define $\rho_1(\bullet)$ as p_0. Furthermore, we define τ_1 as follows: $\tau_1((p_0,q_0)\dots(p_i,q_i)) = \tau((\star,\star)(p_0,q_0)\dots(p_i,q_i))$ until difference between the summation of weights on $p_0 \dots p_i$ and $q_0 \dots q_i$ exceeds r, and after that, go π_q according to the state q where Challenger is at that time. Strategy τ constructed in a such manner is a winning strategy for Challenger in the game played on $\mathcal{A}^{\mathsf{Limavg}}$ and $\mathcal{B}^{\mathsf{Limavg}}$), and taking a path π_i according to the state i where Challenger is at that time.
4 : From [7], G-simulation from $\mathcal{A}^{\mathsf{Limavg}}$ to $\mathcal{B}^{\mathsf{Limavg}}$ exists if and only if there exists a winning strategy for Max in a mean payoff game $\mathcal{MP}$. □