Generic Forward and Backward Simulations II: Probabilistic Simulation

Ichiro Hasuo

[1] RIMS, Kyoto University, Japan
[2] PRESTO Research Promotion Program, Japan Science and Technology Agency

Abstract. Jonsson and Larsen's notion of probabilistic simulation is studied from a coalgebraic perspective. The notion is compared with two generic coalgebraic definitions of simulation: Hughes and Jacobs' one, and the one introduced previously by the author. We show that the first almost coincides with the second, and that the second is a special case of the last. We investigate implications of this characterization; notably the Jonsson-Larsen simulation is shown to be *sound*, i.e. its existence implies trace inclusion.

1 Introduction

Use of probabilistic algorithms in distributed and concurrent applications is common practice. Consequently, modeling and verification techniques for probabilistic systems have been extensively developed. One fundamental branch therein is about probabilistic *(bi)simulation*: it gives an answer when a probabilistic system is "equivalent" to another, or when one "refines" another.

In this paper we focus on simulation notions for purely probabilistic systems.[3] For such systems it is standard to define a notion of simulation using *weight functions*. The idea is first devised by Jonsson and Larsen [12]; it has inspired a large body of work including [1]. Our aim in this paper is to shed fresh, mathematical light on the idea, from the viewpoint of *coalgebra*.

Coalgebra is a mathematical/categorical presentation of state-based systems. Its initial success was brought about by a generic, coalgebraic characterization of bisimulation that applies to a variety of systems, including probabilistic ones (see e.g. [11, 17, 19]). The theory has since been extended to include various aspects of concurrency theory—such as SOS and modal logic (see e.g. [13]). Simulation, as "one-sided bisimulation," is one of such aspects.

Two approaches have been presented towards a coalgebraic theory of simulation: Hughes and Jacobs' [10] and the current author's [5]. Both approaches are generic, applicable to non-deterministic systems like LTS as well as probabilistic ones. In this paper we restrict them to a purely probabilistic setting and conduct a comparative study. The comparison is among the *Jonsson-Larsen simulation*, the *Hughes-Jacobs simulation*, and the one in the author's previous work [5] which we call the *Kleisli simulation*.

Among the three, the notion of Kleisli simulation is the most distinguishable: it is given not as a relation but as a function $X \to \mathcal{D}Y$, where X and Y are the state spaces

[3] Unlike e.g. Segala's probabilistic automata [18], they do not feature non-determinism.

of the involved systems. Therefore it is not suitable as a candidate of a *refinement relation*, the original motivation for the Jonsson-Larsen one. The Kleisli simulation rather follows the spirit of Lynch and Vaandrager [16]: it is a powerful tool for showing *trace inclusion*. While a direct proof of trace inclusion involves transitions within arbitrary many steps, finding a simulation is a stepwise matter. Indeed the notion of Kleisli simulation is precisely a coalgebraic generalization of the one in [16]; the former comes with the *forward* and *backward* variations just like the latter. The theory in [5] has been successfully applied to verification of probabilistic anonymity in [9].

Our findings are as follows. The standard Jonsson-Larsen simulation, defined in §3 concretely for a specific kind of probabilistic systems (we describe them in §2), is identified with a slightly restricted variant of the Hughes-Jacobs simulation (§4). This allows us to remove the unnecessary restriction that was hidden in the original concrete definition (§4.4), as well as provides a guideline in transferring the definition to other kinds of probabilistic systems (§4.5). On another link in the triangle, we identify the Hughes-Jacobs simulation as a special case of the Kleisli simulation (§5). From the generic *soundness theorem* [5]—existence of a Kleisli simulation implies (finite) trace inclusion—we thus conclude soundness of the Hughes-Jacobs notion, hence of the Jonsson-Larsen one.

Our expedition will be in a leisurely pace. In particular, no categorical or coalgebraic prerequisites are assumed; they are introduced on our way, on a call-by-need basis.

Due to space limitation, most proofs are deferred to an extended version [7]. It has also a series of example systems for further comparison of different simulation notions,

Notations A square in a diagram which is not filled means that it *commutes*, that is, the equality symbol $=$ is implicit in it.

A probability (sub)distribution γ over a set X is often denoted like a table: $[\, x \mapsto \gamma(x) \,]_{x \in X}$. When an entry $x \in X$ is missing in the table, the probability 0 is assigned. Hence for example, when $x_0 \in X$ is a fixed element, $[\, x_0 \mapsto 1 \,]$ means the distribution γ such that $\gamma(x_0) = 1$ and $\gamma(x) = 0$ for $x \neq x_0$.

2 Probabilistic System

We will be mainly interested in two kinds of purely probabilistic systems—GPAs and DTMCs—which we now define formally.

Definition 2.1 (Generative probabilistic automaton, GPA) Let Ac be a fixed nonempty alphabet; we refer to its element as an *action*. A *generative probabilistic automaton (GPA)* over Ac is a triple $\mathcal{X} = (X, x_0, c)$ where

- X is a nonempty set of *states*;
- $x_0 \in X$ is a chosen state which is called the *initial* one; and
- $c : X \to \mathcal{D}(\{\checkmark\} + \mathsf{Ac} \times X)$ is a *transition function*. Here $\{\checkmark\}$ is a singleton; $+$ denotes the disjoint union; and $\mathcal{D}$ is the *subdistribution* operation such that for a set Y

$$\mathcal{D}Y \;=\; \{\gamma : Y \to [0,1] \mid \sum_{y \in Y} \gamma(y) \leq 1\} \;. \tag{1}$$

Such $d \in \mathcal{D}Y$ is called a *sub*-distribution since its values add up to not more than 1, instead of precisely 1.

The subdistribution $c(x)$ tells the probabilistic behavior of a state x. The value $c(x)(a, x')$ with $a \in \mathsf{Ac}$ and $x' \in X$ is the probability with which x makes the action a and moves to x'; that is, $c(x)(a, x') = \Pr[x \xrightarrow{a} x']$. We interpret the symbol ✓ as *successful termination*; thus with the probability $c(x)(\checkmark) = \Pr[x \to \checkmark]$ the state x is led to successful termination. The remaining probability $1 - c(x)(\checkmark) - \sum_{a,x'} c(x)(a, x')$—which may be more than 0 since $c(x)$ is a subdistribution—is understood as the probability with which x gets into *deadlock*.

GPAs are said to be *generative*, in contrast to *reactive* systems whose transition function is given as, say, $c : \mathsf{Ac} \times X \to \mathcal{D}(\{\checkmark\} + X)$. They differ in whether an action is chosen by the system or by the environment; see [4].

GPAs can be thought of as a probabilistic variant of labeled transition systems (LTSs). DTMCs, which we introduce shortly, are then probabilistic Kripke frames. The notion is standard, see e.g. [1, 14]. The definitions in the literature vary in details; the following one is adapted to fit the current context.

Definition 2.2 (Discrete-Time Markov Chain, DTMC) Let AP be a fixed set of *atomic propositions*. A *discrete-time Markov chain (DTMC)* over AP is a quadruple $\mathcal{X} = (X, x_0, l, p)$ where

- X is a nonempty set of *states*, among which $x_0 \in X$ is an *initial* state;
- $l : X \to \mathcal{P}(\mathsf{AP})$ is a *labeling function* where $\mathcal{P}$ denotes the powerset. This assigns to a state $x \in X$ the set $l(x)$ of atomic propositions that hold at x;
- $p : X \to \mathcal{D}X$ is a *transition function*, where $\mathcal{D}$ is the operation in (1).

A DTMC has labels on its states, while a GPA has labels on its transitions.

3 Jonsson-Larsen Simulation

For DTMC and its variants, a standard definition of simulation [12] uses *weight functions*. Here we present the definition for DTMC taken from [1]. This family of simulation notions—based on weight functions—will be called *Jonsson-Larsen simulation*.

Definition 3.1 (JL-simulation for DTMC) Let $\mathcal{X} = (X, x_0, l, p)$ and $\mathcal{Y} = (Y, y_0, m, q)$ be DTMCs. A *Jonsson-Larsen simulation (JL-simulation)* from $\mathcal{X}$ to $\mathcal{Y}$ is a relation $R \subseteq X \times Y$ which satisfies the following.

1. The initial states are related, that is, $x_0 R y_0$.
2. Related states satisfy the same atomic propositions: xRy implies $l(x) = m(y)$.
3. For each $x \in X$ and $y \in Y$ such that xRy, there exists a *weight function*

$$\Delta_{x,y} \;:\; \big(\{\bot\} + X\big) \times \big(\{\bot\} + Y\big) \longrightarrow [0, 1] \qquad \text{such that}$$

   (a) $\Delta_{x,y}(u, v) > 0$ implies either
   - $u = \bot$, or

– $u = x' \in X$, $v = y' \in Y$ and $x' R y'$;

(b) $\Delta_{x,y}(\bot,\bot) + \sum_{y' \in Y} \Delta_{x,y}(\bot, y') = 1 - \sum_{x' \in X} p(x)(x')$;

(c) for each $x' \in X$: $\Delta_{x,y}(x',\bot) + \sum_{y' \in Y} \Delta_{x,y}(x', y') = p(x)(x')$;

(d) $\Delta_{x,y}(\bot,\bot) + \sum_{x' \in X} \Delta_{x,y}(x',\bot) = 1 - \sum_{y' \in Y} q(y)(y')$;

(e) for each $y' \in Y$: $\Delta_{x,y}(\bot, y') + \sum_{x' \in X} \Delta_{x,y}(x', y') = q(y)(y')$.

Although illustration of the previous definition is found e.g. in [1, Ex. 14], the definition hardly seems as "canonical" or "intuitive" as other notions such as (bi)simulation for ordinary LTS. For example, the only asymmetry between $\mathcal{X}$ and $\mathcal{Y}$ is found in Cond. 3.(a) where u, not v, is allowed to be $\bot$. One might wonder if it is possible to weaken this condition. Weakening $=$ into $\subseteq$ in Cond. 2 looks like another possibility. It is not clear either how to adapt this definition to GPA. Even less clear is whether the adapted notion satisfies soundness—existence of a simulation implies trace inclusion—which is a natural property to expect.

What we do in the rest of the paper is to put the above definition in a coalgebraic context. First it will be identified with a restriction of Hughes and Jacobs' simulation *(HJ-simulation)* [10]. From this we immediately obtain natural generalizations of the original definition, which are hinted above. The generic theory in [10] can be used to conduct some "sanity checks" for the generalized definitions. Adaptation to GPA comes for free, too. After that we will identify HJ-simulation with a certain subclass of *Kleisli* simulation from [5]. Soundness of JL-simulation for GPA is its corollary.

4 Hughes-Jacobs Simulation

4.1 Coalgebraic Modeling

In the Hughes-Jacobs theory of coalgebraic simulation, a system is modeled as a B-*coalgebra*, which is a function c of the type on the right. The set X (which is arbitrary) is the system's *state space*; the operation B specifies the kind of transitional behavior exhibited by the system; and the function c determines the system's dynamic behavior. We now elaborate on the operation B, which takes a set X and returns another set BX.

$$\begin{array}{c} BX \\ {\scriptstyle c}\uparrow \\ X \end{array}$$

Roughly speaking, it is the operation B which determines what kind of systems we are talking about. One choice of B makes a B-coalgebra an LTS; another choice of B is for a deterministic automaton (DA); and so on. Specifically,

B	$\mathcal{P}(\mathsf{Ac} \times \_)$	$2 \times (\_)^{\mathsf{Ac}}$	$(\mathsf{Ac}_{\mathsf{out}} \times \_)^{\mathsf{Ac}_{\mathsf{in}}}$	$\mathcal{D}\big(\{\checkmark\} + \mathsf{Ac} \times \_\big)$	$\mathcal{P}(\mathsf{AP}) \times \mathcal{D}(\_)$
B-coalg.	LTS	DA	Mealy mach.	GPA	DTMC

When B is $\mathcal{D}\big(\{\checkmark\} + \mathsf{Ac} \times \_\big)$, a B-coalgebra is a function $c : X \to \mathcal{D}\big(\{\checkmark\} + \mathsf{Ac} \times X\big)$; this is precisely a GPA (Def. 2.1) without an explicit initial state. For $B = \mathcal{P}(\mathsf{AP}) \times \mathcal{D}(\_)$, a B-coalgebra is a function $c : X \to \mathcal{P}(\mathsf{AP}) \times \mathcal{D}X$, which is identified with a DTMC (without an initial state) via the following bijective projection-tupling correspondence.

$$\frac{X \longrightarrow \mathcal{P}(\mathsf{AP}) \times \mathcal{D}X}{X \longrightarrow \mathcal{P}(\mathsf{AP}) \qquad X \longrightarrow \mathcal{D}X} \qquad \begin{array}{c} c \\ {\scriptstyle \pi_1 \circ c,\ \pi_2 \circ c}\downarrow\cong \\ (\pi_1 \circ c,\ \pi_2 \circ c) \end{array} \qquad \begin{array}{c} \langle l, p\rangle := \lambda x.\,(\,l(x),\ p(x)\,) \\ \uparrow\cong \\ (l, p) \end{array}$$

Here π_i denotes the i-th projection; $\langle l, p\rangle$ denotes the *tupling* of l and p.

To develop a "theory of systems" on top of this modeling, an operation B needs to be a *functor*. Leaving its detailed treatment to literature like [11], what it means is that the operation B not only applies to sets (i.e. $X \mapsto BX$) but also to functions. That is,

$$B \ : \quad (X \xrightarrow{f} Y) \quad \longmapsto \quad (BX \xrightarrow{Bf} BY) \ .$$

Note the domain and the codomain of the resulting function Bf.

The previous examples of B have natural action on functions. For example, given a function $f : X \to Y$,

$$\mathcal{P}(\mathsf{Ac} \times X) \xrightarrow{\mathcal{P}(\mathsf{Ac}\times f)} \mathcal{P}(\mathsf{Ac} \times Y) \ , \quad u \longmapsto \big\{\, (a, f(x)) \,\big|\, (a, x) \in u \,\big\} \ ;$$

$$\mathcal{D}\big(\{\checkmark\} + \mathsf{Ac} \times X\big) \xrightarrow{\mathcal{D}(\{\checkmark\}+\mathsf{Ac}\times f)} \mathcal{D}\big(\{\checkmark\} + \mathsf{Ac} \times Y\big) \ ,$$
$$\gamma \longmapsto \big[\, \checkmark \mapsto \gamma(\checkmark) \ , \quad (a, y) \mapsto \textstyle\sum_{x\in f^{-1}(\{y\})} \gamma(a, x) \,\big] \ ;$$

$$\mathcal{P}(\mathsf{AP}) \times \mathcal{D}X \xrightarrow{\mathcal{P}(\mathsf{AP})\times\mathcal{D}f} \mathcal{P}(\mathsf{AP}) \times \mathcal{D}Y \ , \quad (u, \gamma) \longmapsto \big(\, u, \, \big[y \mapsto \textstyle\sum_{x\in f^{-1}(\{y\})} \gamma(x)\big] \,\big) \ .$$

To be precise, such B is a functor of the type $\mathbf{Sets} \to \mathbf{Sets}$, from the category $\mathbf{Sets}$ of sets and functions to itself. We make a formal definition for the record.

Definition 4.1 (Functor, coalgebra) A *functor* $B : \mathbf{Sets} \to \mathbf{Sets}$ consists of its action on sets $X \longmapsto BX$ and on functions $(X \xrightarrow{f} Y) \longmapsto (BX \xrightarrow{Bf} BY)$, for each X and f. This is subject to the following conditions:

$$B(X \xrightarrow{\mathrm{id}_X} X) = (BX \xrightarrow{\mathrm{id}_{BX}} BX) \ ; \qquad B(X \xrightarrow{f} Y \xrightarrow{g} U) = (BX \xrightarrow{Bf} BY \xrightarrow{Bg} BU) \ .$$

A *B-coalgebra* is a pair $(X, c : X \to BX)$ of a set and a function; we shall simply denote it by $X \xrightarrow{c} BX$.

The *functoriality* of B is crucial in the following definition of coalgebraic bisimulation (notice use of $B\pi_i$). The definition subsumes many known notions of bisimulation.

Definition 4.2 Let $B : \mathbf{Sets} \to \mathbf{Sets}$ be a functor and $c : X \to BX$ and $d : Y \to BY$ be B-coalgebras. A *coalgebraic bisimulation* is a relation $R \subseteq X \times Y$ such that: there exists a function $r : R \to BR$ that makes the diagram on the right commute. Here π_1 and π_2 are obvious projections.

$$\begin{array}{ccccc} BX & \xleftarrow{B\pi_1} & BR & \xrightarrow{B\pi_2} & BY \\ {\scriptstyle c}\uparrow & & {\scriptstyle r}\uparrow & & \uparrow{\scriptstyle d} \\ X & \xleftarrow[\pi_1]{} & R & \xrightarrow[\pi_2]{} & Y \end{array} \tag{2}$$

When B represents purely probabilistic systems such as DTMCs, the above coalgebraic bisimulation instantiates to the one that uses a weight function. It coincides with the more common formulation via equivalence classes [15]. The coincidence proof is implicit in [12, Thm. 4.6] and is much more systematically conducted in [19].

4.2 Hughes-Jacobs Simulation

Roughly speaking, simulation is "one-sided" bisimulation. When R is a simulation and xRy, we require y to exhibit "at least as much" behavior as x does, that is,

$$(x\text{'s behavior}) \ \sqsubseteq \ (y\text{'s behavior})$$

in terms of a suitable preorder $\sqsubseteq$ of "behavior inclusion." Hughes and Jacobs [10] used this intuition and defined generic simulation as a variant of Def. 4.2. In order to do so, a functor $B : \mathbf{Sets} \to \mathbf{Sets}$ needs to come with a "behavior-inclusion" preorder $\sqsubseteq$.

Definition 4.3 (Functor with preorder) A *functor with preorder* consists of a functor $B : \mathbf{Sets} \to \mathbf{Sets}$ and a class of preorders $\{\sqsubseteq_{BX}\}_X$ for each set X, where $\sqsubseteq_{BX}$ is on the set BX. Further, given a function $f : X \to Y$, its action $Bf : BX \to BY$ is required to be a monotone function. We often suppress the subscript in $\sqsubseteq_{BX}$.

Example 4.4 Let $B = \mathcal{P}(\mathsf{Ac} \times \_)$, for which B-coalgebras are LTSs. It is a functor with preorder, with a natural choice of $\sqsubseteq_{BX}$ being the inclusion order.

Let $B = \mathcal{D}\big(\{\checkmark\} + \mathsf{Ac} \times \_\big)$; a B-coalgebra is a GPA. For $\gamma, \delta \in BX$ we define

$$\gamma \sqsubseteq_{BX} \delta \quad \overset{\text{def.}}{\iff} \quad \gamma(\checkmark) \leq \delta(\checkmark) \quad \text{and} \quad \gamma(a, x) \leq \delta(a, x) \quad \text{for each } a \text{ and } x.$$

Note that $\sqsubseteq_{BX}$ need not be reduced to the equality, since γ and δ are *sub*distributions.

Let $B = \mathcal{P}(\mathsf{AP}) \times \mathcal{D}(\_)$; a B-coalgebra is then a DTMC. There are a few natural candidates for the preorder $\sqsubseteq_{BX}$. One is:

$$(u, \gamma) \sqsubseteq^{=}_{BX} (v, \delta) \quad \overset{\text{def.}}{\iff} \quad u = v \quad \text{and} \quad \gamma(x) \leq \delta(x) \quad \text{for each } x.$$

We denote this order by $\sqsubseteq^{=}$. Noting that u and v are subsets of AP, we could replace the condition $u = v$ by $u \subseteq v$, for example. The resulting order will be denoted by $\sqsubseteq^{\subseteq}$.

Definition 4.5 (HJ-simulation) Let $(B, \sqsubseteq)$ be a functor with preorder, and $X \xrightarrow{c} BX, Y \xrightarrow{d} BY$ be B-coalgebras. A *Hughes-Jacobs simulation (HJ-simulation)* from c to d is a relation $R \subseteq X \times Y$ such that: there exists a function $r : R \to BR$ which makes the inequalities on the right hold.

$$\begin{array}{ccccc} BX & \xleftarrow{B\pi_1} & BR & \xrightarrow{B\pi_2} & BY \\ {\scriptstyle c}\uparrow & \sqsubseteq & {\scriptstyle r}\uparrow & \sqsubseteq & \uparrow{\scriptstyle d} \\ X & \xleftarrow[\pi_1]{} & R & \xrightarrow[\pi_2]{} & Y \end{array} \tag{3}$$

That is to be precise: for each $(x, y) \in R$

$$(c \circ \pi_1)(x, y) \sqsubseteq_{BX} (B\pi_1 \circ r)(x, y) \quad \text{and} \quad (B\pi_2 \circ r)(x, y) \sqsubseteq_{BY} (d \circ \pi_2)(x, y) \ .$$

The formulation is different from the original one [10] where a lax relation lifting is used. The equivalence is proved in [7, Appendix A.1].

4.3 Jonsson-Larsen Simulation as Hughes-Jacobs Simulation

Here is the first main observation in this paper: JL is HJ. The order $\sqsubseteq^{=}$ is from Ex. 4.4.

Theorem 4.6 *Let $\mathcal{X} = (X, x_0, l, p)$ and $\mathcal{Y} = (Y, y_0, m, q)$ be DTMCs, and let R be a JL-simulation from $\mathcal{X}$ to $\mathcal{Y}$. Then R is a HJ-simulation from the coalgebra $\langle l, p\rangle$ to $\langle m, q\rangle$: there exists r that makes the following (in)equalities hold.*

$$\begin{array}{ccccc} \mathcal{P}(\mathsf{AP}) \times \mathcal{D}X & \xleftarrow{\mathcal{P}(\mathsf{AP}) \times \mathcal{D}\pi_1} & \mathcal{P}(\mathsf{AP}) \times \mathcal{D}R & \xrightarrow{\mathcal{P}(\mathsf{AP}) \times \mathcal{D}\pi_2} & \mathcal{P}(\mathsf{AP}) \times \mathcal{D}Y \\ {\scriptstyle \langle l,p\rangle}\uparrow & = & {\scriptstyle r}\uparrow & \sqsubseteq^{=} & \uparrow{\scriptstyle \langle m,q\rangle} \\ X & \xleftarrow[\pi_1]{} & R & \xrightarrow[\pi_2]{} & Y \end{array}$$

□

We include initial states (Cond. 1, Def. 3.1) and obtain the following characterization.

Theorem 4.7 (JL is HJ) *Let $\mathcal{X}$ and $\mathcal{Y}$ be DTMCs in Thm. 4.6. A relation $R \subseteq X \times Y$ is a JL-simulation if and only if there exist functions r and r_0 that make the following (in)equalities hold. The set $\{*\}$ is a singleton.*

$$\begin{array}{ccccc}
\mathcal{P}(\mathsf{AP}) \times \mathcal{D}X & \xleftarrow{\mathcal{P}(\mathsf{AP}) \times \mathcal{D}\pi_1} & \mathcal{P}(\mathsf{AP}) \times \mathcal{D}R & \xrightarrow{\mathcal{P}(\mathsf{AP}) \times \mathcal{D}\pi_2} & \mathcal{P}(\mathsf{AP}) \times \mathcal{D}Y \\
{\scriptstyle \langle l,p \rangle}\uparrow & = & {\scriptstyle r}\uparrow & \sqsubseteq^{=} & \uparrow{\scriptstyle \langle m,q \rangle} \\
X & \xleftarrow[\pi_1 \;=]{} & R & \xrightarrow[= \;\pi_2]{} & Y \\
\uparrow{\scriptstyle x_0} & & {\scriptstyle r_0}\uparrow & & \uparrow{\scriptstyle y_0} \\
 & & \{*\} & &
\end{array} \tag{4}$$

Note that a function $x_0 : \{*\} \to X$ can be identified with an element $x_0 \in X$. □

4.4 Generalized Jonsson-Larsen Simulation

Thm. 4.7 shows that JL-simulation does not reach the full generality of HJ-simulation: the top-left square is an equality in (4), which is not necessary. Translating HJ-simulation into the Jonsson-Larsen style concrete terms, we are led to the following definition.

Definition 4.8 (JL'-simulation for DTMC) A *JL'-simulation* is the same thing as a JL-simulation (Def. 3.1) except for the following.

- A weight function is of the type $\Delta_{x,y} : \big(\{\bot\} + X\big) \times \big(\{\bot\} + Y\big) \longrightarrow [-1, 1]$.
- Cond. 3.(a) is weakened: the value of $\Delta_{x,y}(u, v)$ must lie in the following range, according to u and v:
  - when $u = x' \in X$ and $v = y' \in Y$, if $x'Ry'$ then $\Delta_{x,y}(x', y') \geq 0$; if $(x', y') \notin R$ then $\Delta_{x,y}(x', y') = 0$;
  - $\Delta_{x,y}(\bot, y') \geq 0$ for each $y' \in Y$;
  - $\Delta_{x,y}(x', \bot) \leq 0$ for each $x' \in X$;
  - $\Delta_{x,y}(\bot, \bot)$ can be positive, zero or negative.

$u \backslash v$	$\bot$	$\cdots\ y'\ \cdots$
$\bot$	$\lesseqgtr 0$	≥ 0
$\vdots$		
x'	≤ 0	$\begin{cases} \geq 0 & (x'Ry') \\ 0 & (\text{o.w.}) \end{cases}$
$\vdots$		

- Cond. 3.(b) and 3.(d) are dropped.

Now a weight function can take negative values. Cond. 3.(b) and 3.(d) played no role in Thm. 4.6, hence are dropped. Similarly to JL-simulation, finding a weight function is filling in the matrix above on the right, in such a way that its rows and columns add up to the right values like $p(x)(x')$ or $q(y)(y')$. The task is easier with JL'-simulation because each entry can be picked from a broadened domain.

One can further generalize the previous definition by replacing $\sqsubseteq^{=}$ by $\sqsubseteq^{\subseteq}$ (Ex. 4.4): in this case the system $\mathcal{X}$ to be simulated satisfies no more atomic propositions than $\mathcal{Y}$ does. This generalization is useful e.g. when we are interested in safety properties, and atomic propositions represent systems' actions.

Definition 4.9 (JL"-simulation for DTMC) A *JL"-simulation* is the same as a JL'-simulation (Def. 4.8), except that Cond. 2 is replaced by

2. xRy implies $l(x) \subseteq m(y)$.

Proposition 4.10 *Let $\mathcal{X}$ and $\mathcal{Y}$ be DTMCs as in Thm. 4.6, and $R \subseteq X \times Y$.*

1. *The relation R is a JL'-simulation if and only if there exist r and r_0 that validate the (in)equalities in (4), with the top-left equality replaced by $\sqsubseteq^{=}$.*
2. *The relation R is a JL"-simulation if and only if there exist r and r_0 that validate the diagram (4), with the top two squares filled with $\sqsubseteq^{\subseteq}$.* □

Let us do some sanity checks. The following holds for JL" instead of JL' too; also for the conventional notion of JL simulation (see [1]).

Proposition 4.11 *Let $\mathcal{X} = (X, x_0, l, p)$ and $\mathcal{Y} = (Y, y_0, m, q)$ be DTMCs.*

1. *If $R \subseteq X \times Y$ is a bisimulation, then R and R^{op} are both JL'-simulations.*
2. *The family of JL'-simulations from $\mathcal{X}$ to $\mathcal{Y}$ is closed under arbitrary unions. Therefore there is the largest JL'-simulation $\lesssim_{JL'}$, called* JL'-similarity.
3. *JL'-simulations are closed under composition. Hence $\lesssim_{JL'}$ is transitive.*

Proof. We apply Lem. 4.2 and Prop. 5.4 of [10]. This involves checking a technical condition of *stability* of orders. See [7]. □

4.5 Jonsson-Larsen Simulation for GPA

Another implication of Thm. 4.7 is adaptation of JL-simulation for other kinds of probabilistic systems, via HJ-simulation which is general by definition.

Definition 4.12 (JL-simulation for GPA) Let $\mathcal{X} = (X, x_0, c)$ and $\mathcal{Y} = (Y, y_0, d)$ be GPAs. A *JL-simulation* from $\mathcal{X}$ to $\mathcal{Y}$ is a relation $R \subseteq X \times Y$ such that:

1. The initial states are related, that is, $x_0 R y_0$.
2. For each pair $(x, y) \in R$, there exists a weight function

   $\Delta_{x,y} \;:\; \big(\{\bot\}+\{\checkmark\}+\mathsf{Ac}\times X\big)\times\big(\{\bot\}+\{\checkmark\}+\mathsf{Ac}\times Y\big) \longrightarrow [-1,1]$ such that

   (a) $\Delta_{x,y}(u,v)$ lies in the range on the right. In particular, $\Delta_{x,y}((a,x'),(a',y')) > 0$ only if $a = a'$ and $x' R y'$;

$u \backslash v$	$\bot$	$\checkmark$	$\cdots\ (a_1, y_1')\ \cdots$	$\cdots\ (a_2, y_2')\ \cdots$
$\bot$	≤ 0 ≥ 0	≥ 0	≥ 0	≥ 0
$\checkmark$	≤ 0	≥ 0	0	0
$\vdots$ (a_1, x_1') $\vdots$	≤ 0	0	$\begin{cases}\geq 0 & (x_1' R y_1')\\ 0 & \text{(o.w.)}\end{cases}$	0
$\vdots$ (a_2, x_2') $\vdots$	≤ 0	0	0	$\begin{cases}\geq 0 & (x_2' R y_2')\\ 0 & \text{(o.w.)}\end{cases}$

   (b) $c(x)(\checkmark) = \Delta_{x,y}(\checkmark,\bot) + \Delta_{x,y}(\checkmark,\checkmark)$;
   (c) $c(x)(a,x') = \Delta_{x,y}((a,x'),\bot) + \sum_{y'} \Delta_{x,y}((a,x'),(a,y'))$ for each a and x';
   (d) $d(x)(\checkmark) = \Delta_{x,y}(\bot,\checkmark) + \Delta_{x,y}(\checkmark,\checkmark)$;
   (e) $d(y)(a,y') = \Delta_{x,y}(\bot,(a,y')) + \sum_{x'} \Delta_{x,y}((a,x'),(a,y'))$ for each a and y'.

This definition seems to appear for the first time. It coincides with HJ-simulation for $B = \mathcal{D}(\{\checkmark\}+\mathsf{Ac}\times(\_))$, with B equipped with the order in Ex. 4.4 (the proof is easy). Properties like in Prop. 4.11 hold as well. Remaining is the issue of *soundness*; it is not obvious at all from the above complicated definition. One of our main contributions is the soundness proof later in §5.7, which uses the generic theory in [5].

5 Kleisli Forward and Backward Simulation

We now describe the third kind of simulation from [5]. We shall refer to this family as *Kleisli simulation*, for the reason that is explained shortly. Kleisli simulation consists of four subclasses: *forward*, *backward*, and two *hybrid* ones, like in [16]. The most notable difference from JL- and HJ-notions is that a Kleisli simulation is itself not a relation.

5.1 Kleisli Arrow

First we fix our domain of discourse—*Kleisli arrows*. They are arrows in a Kleisli category, a standard categorical construct. Our description is however in concrete terms.

Definition 5.1 (Kleisli arrow) Let X and Y be arbitrary sets. A *Kleisli arrow* from X to Y, denoted by $f : X \nrightarrow Y$, is a function $f : X \to \mathcal{D}Y$. A few typical Kleisli arrows:

- The Kleisli arrow $\eta_X : X \nrightarrow X$, for each X, is the function $\eta_X : X \to \mathcal{D}X$ that carries $x \in X$ to $[x \mapsto 1]$.
- Given consecutive Kleisli arrows $X \overset{f}{\nrightarrow} Y$ and $Y \overset{g}{\nrightarrow} U$, we have $g \odot f : X \nrightarrow U$ by

$$g \odot f \ : \ X \longrightarrow \mathcal{D}U \ \ , \quad x \longmapsto \lambda u.\ \textstyle\sum_{y \in Y} g(y)(u) \cdot f(x)(y) \ \ .$$

- For each (ordinary) function $f : X \to Y$, we have $Jf : X \nrightarrow Y$ defined by $X \xrightarrow{f} Y \xrightarrow{\eta_Y} \mathcal{D}Y$. That is, $(Jf)(x) = [f(x) \mapsto 1]$. This generalizes η_X by: $\eta_X = J(\mathrm{id}_X)$.

The following are straightforward; they say that Kleisli arrows form a category.

Proposition 5.2 *1. Composition of Kleisli arrows is associative: for three consecutive Kleisli arrows $X \overset{f}{\nrightarrow} Y \overset{g}{\nrightarrow} U \overset{h}{\nrightarrow} V$, we have $h \odot (g \odot f) = (h \odot g) \odot f$.*

2. η is the unit of composition: for $X \overset{f}{\nrightarrow} Y$ we have $\eta_Y \odot f = f = f \odot \eta_X$. □

One can think of a Kleisli arrow $f : X \nrightarrow Y$ as a "function from X to Y, with implicit probabilistic branching"; or as a "probabilistic computation of input type X and output type Y." The operator $\odot$ realizes natural composition of such probabilistic computations. The embedding Jf of an ordinary function endows f with trivial branching.

There is a natural order between parallel Kleisli arrows.

Definition 5.3 Between a parallel pair of Kleisli arrows $f, g : X \nrightarrow Y$, we define an order $f \sqsubseteq g$ if: $f(x)(y) \leq g(x)(y)$ for each $x \in X$ and $y \in Y$.

5.2 Probabilistic Systems as Kleisli Coalgebras

A GPA $\mathcal{X} = (X, x_0, c)$ (Def. 2.1) can be presented by two Kleisli arrows:

$$\{*\} \overset{Jx_0}{\nrightarrow} X \overset{c}{\nrightarrow} \{\checkmark\} + \mathsf{Ac} \times X \ \ . \tag{5}$$

This is a prototype of the kind of systems on which we define Kleisli simulation. First we parametrize the '$\{\checkmark\} + \mathsf{Ac} \times (\_)$' part in the above.

Definition 5.4 (Polynomial functor) A *polynomial functor* is a functor $F : \mathbf{Sets} \to \mathbf{Sets}$ which is constructed

- from the identity functor $(\_)$ and the constant functor C for each set C,
- using finite products and arbitrary disjoint union (i.e. coproduct).

In the BNF notation: $F ::= (\_) \mid C \mid F_1 \times F_2 \mid \coprod_{i\in I} F_i$.

The functor $\{\checkmark\} + \mathsf{Ac} \times (\_)$ is polynomial; so is e.g. $(\mathsf{Ac} + \_)^* = \coprod_{n<\omega}(\mathsf{Ac} + \_)^n$.

Lemma 5.5 *A polynomial functor F has canonical action on Kleisli arrows, carrying $X \overset{f}{\nrightarrow} Y$ to $FX \overset{Ff}{\nrightarrow} FY$.*

Proof. A general categorical proof is found in [8, §2.2]; one can also define such action concretely by induction on the construction of F. □

In most cases F's action on Kleisli arrows is obvious. For $F = \{\checkmark\} + \mathsf{Ac} \times (\_)$ and $f : X \nrightarrow Y$, the Kleisli arrow $Ff : FX \nrightarrow FY$ is given by the function $\{\checkmark\} + \mathsf{Ac} \times X \longrightarrow \mathcal{D}(\{\checkmark\} + \mathsf{Ac} \times Y)$, defined by

$$\checkmark \longmapsto [\, \checkmark \mapsto 1 \,] \ , \quad (a, x) \longmapsto [\, (a, y) \mapsto f(x)(y) \,]_{y\in Y} \ .$$

Definition 5.6 (Probabilistic F-system) Let F be a polynomial functor. A *probabilistic F-system* (or simply *F-system*) is a triple $\mathcal{X} = (X, s, c)$, where X is an arbitrary set and $\{*\} \overset{s}{\nrightarrow} X \overset{c}{\nrightarrow} FX$ are two Kleisli arrows. Recall that probabilistic branching is implicit in Kleisli arrows.

Example 5.7 A GPA induces an F-system, with $F = \{\checkmark\} + \mathsf{Ac} \times (\_)$; see (5). F-system is more general than GPA since the former allows a subdistribution on initial states (i.e. $s \in \mathcal{D}X$) rather than a single initial state. This additional generality is however not important.

A DTMC cannot be seen as an F-system as it is: its dynamics is given by a function $X \overset{\langle l,p\rangle}{\longrightarrow} \mathcal{P}(\mathsf{AP}) \times \mathcal{D}X$ which cannot be understood as a Kleisli arrow. We can fix it by moving "state labels" into "transition labels." Let us define a function $c_{l,p}$ by

$$c_{l,p} \ : \ X \longrightarrow \mathcal{D}\big(\, \mathcal{P}(\mathsf{AP}) \times X \,\big) \ , \quad x \longmapsto \big[\, (l(x), x') \ \mapsto \ p(x)(x') \,\big]_{x'\in X} \ ;$$

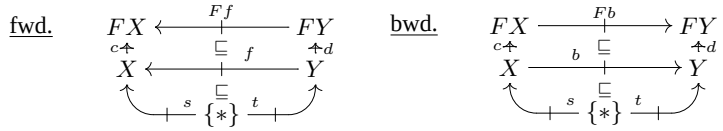
then the F-system $\{*\} \overset{Jx_0}{\nrightarrow} X \overset{c_{l,p}}{\nrightarrow} \mathcal{P}(\mathsf{AP}) \times X$ represents a DTMC (X, x_0, l, p).

The notion of (probabilistic) F-system is essentially a *Kleisli F-coalgebra* $X \overset{c}{\nrightarrow} FX$ equipped with an explicit initial state $\{*\} \overset{s}{\nrightarrow} X$. In coalgebraic studies it is usually unnecessary to speak about explicit initial states; we however need that in this paper for formulating the soundness result (Thm. 5.20). See [6, §3.2.4].

Let us compare the current *Kleisli coalgebraic modeling* of GPAs (Ex. 5.7) with the modeling in §4.1. They are the same in that the dynamics of a GPA is represented by a function $X \to \mathcal{D}(\{\checkmark\} + \mathsf{Ac} \times X)$. In the Kleisli modeling, the functor $B = \mathcal{D}(\{\checkmark\} + \mathsf{Ac} \times (\_))$ is divided into $\mathcal{D}$ (*branching* part) and $F = \{\checkmark\} + \mathsf{Ac} \times (\_)$ (*transition/action* part); the former is then "thrown under the rug" using Kleisli arrows.

5.3 Kleisli Simulation

Definition 5.8 (Kleisli simulation) Let F be a polynomial functor and $\mathcal{X} = (X, s, c)$ and $\mathcal{Y} = (Y, t, d)$ be F-systems. A *forward Kleisli simulation* from $\mathcal{X}$ to $\mathcal{Y}$ is a Kleisli arrow $f : Y \nrightarrow X$ such that $c \odot f \sqsubseteq (Ff) \odot d$ and $s \sqsubseteq f \odot t$ (see below left). Note the direction of f. It is also called simply a *forward simulation*.



A *backward (Kleisli) simulation* is a Kleisli arrow $b : X \nrightarrow Y$ such that $(Fb) \odot c \sqsubseteq d \odot b$ and $b \odot s \sqsubseteq t$ (see above right). Here the order $\sqsubseteq$ refers to the one in Def. 5.3.

In fact, the last definition is an instance of *generic forward and backward simulation* in [5,6]. The general definition has an extra parameter T that specifies a *branching type*. It is fixed to $T = \mathcal{D}$ in this paper, representing probabilistic branching. Another main example is $T = \mathcal{P}$, the powerset operation, for *non-deterministic* branching.

This extra parameter T is used in the definition of Kleisli arrow. Namely, $f : X \nrightarrow Y$ is defined to be a function $f : X \to TY$. When $T = \mathcal{P}$, a Kleisli arrow $f : X \nrightarrow Y$ can be identified with a *binary relation* $R_f \subseteq X \times Y$: xR_fy if and only if $y \in f(x)$. In this case, if moreover $F = \mathsf{Ac} \times (\_)$ for which F-systems are ordinary LTSs, Kleisli simulation (Def. 5.8) coincides with the standard notions of forward and backward simulation for LTS (see e.g. [16]). To summarize: probabilistic Kleisli simulation (Def. 5.8) is a natural generalization of non-deterministic simulation in [16].

5.4 Kleisli Simulation for GPA

We further instantiate the definition to GPA, i.e. $F = \{\checkmark\} + \mathsf{Ac} \times (\_)$. It demonstrates Kleisli simulation's affinity to the conventional simulation notions for LTS.

Notation 5.9 A forward simulation is a function $f : Y \to \mathcal{D}X$; we write $\Pr[y \dashrightarrow x]$ for the value $f(y)(x)$. We let $\Pr[x \to \checkmark]$ and $\Pr[x \xrightarrow{a} x']$ have their obvious meanings. We also compose events; for example

$$\Pr[y \dashrightarrow x \xrightarrow{a} x'] \;:=\; \Pr[y \dashrightarrow x] \cdot \Pr[x \xrightarrow{a} x'] \;=\; f(y)(x) \cdot c(x)(a, x') \;\; .$$

For a backward simulation, we write $\Pr[x \dashrightarrow y]$ for $b(x)(y)$.

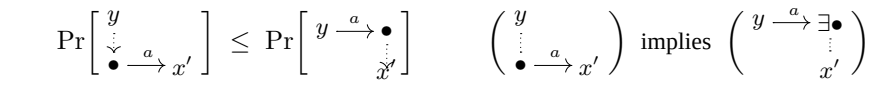
Definition 5.10 (Forward simulation for GPA) Let $\mathcal{X} = (X, x_0, c)$ and $\mathcal{Y} = (Y, y_0, d)$ be GPAs. A *forward (Kleisli) simulation* from $\mathcal{X}$ to $\mathcal{Y}$ is a function $f : Y \to \mathcal{D}X$ which satisfies the following (in)equalities.

$$\Pr[y_0 \dashrightarrow x_0] = 1 \qquad \text{(INIT)}$$

$$\textstyle\sum_{x \in X} \Pr[y \dashrightarrow x \to \checkmark] \le \Pr[y \to \checkmark] \qquad \text{for each } y \in Y \qquad \text{(TERM)}$$

$$\textstyle\sum_{x \in X} \Pr[y \dashrightarrow x \xrightarrow{a} x'] \le \sum_{y' \in Y} \Pr[y \xrightarrow{a} y' \dashrightarrow x'] \qquad \text{for each } y \in Y,\ a \in \mathsf{Ac} \text{ and } x' \in X \qquad \text{(ACT)}$$

The condition (ACT) is illuminating. It can be depicted as the below left, which bears a clear affinity to the standard non-deterministic condition shown on the right.



Definition 5.11 (Backward simulation for GPA) A *backward (Kleisli) simulation* from $\mathcal{X}$ to $\mathcal{Y}$ is a function $b : X \to \mathcal{D}Y$ which satisfies the following inequalities.

$$\Pr[x_0 \dashrightarrow y_0] \leq 1 \qquad \text{(INIT)}$$

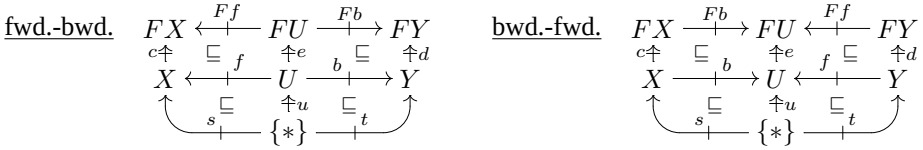
$$\Pr[x \to \checkmark] \leq \textstyle\sum_{y \in X} \Pr[x \dashrightarrow y \to \checkmark] \quad \text{for each } x \in X \qquad \text{(TERM)}$$

$$\textstyle\sum_{x' \in X} \Pr[x \xrightarrow{a} x' \dashrightarrow y'] \leq \sum_{y \in Y} \Pr[x \dashrightarrow y \xrightarrow{a} y']$$
$$\text{for each } x \in X,\ a \in \mathsf{Ac} \text{ and } y' \in Y \qquad \text{(ACT)}$$

5.5 Hughes-Jacobs Simulation as Hybrid Kleisli Simulation

Definition 5.12 (Hybrid simulation) Let $\mathcal{X} = (X, s, c)$ and $\mathcal{Y} = (Y, t, d)$ be F-systems. A *forward-backward (Kleisli) simulation* is a triple $(\mathcal{U}, f, b)$ where

- $\mathcal{U} = (U, u, e)$ is an F-system called the *intermediate system*;
- f is a forward simulation from $\mathcal{X}$ to $\mathcal{U}$, and
- b is a backward simulation from $\mathcal{U}$ to $\mathcal{Y}$. See below on the left.



Similarly, a *backward-forward (Kleisli) simulation* is a triple $(\mathcal{U}, b, f)$ of an intermediate system $\mathcal{U}$, a backward simulation b from $\mathcal{X}$ to $\mathcal{U}$, and a forward simulation f from $\mathcal{U}$ to $\mathcal{Y}$. See above on the right.

Proposition 5.13 *Let $\mathcal{X}, \mathcal{Y}$ be F-systems. If there is a non-hybrid simulation from $\mathcal{X}$ to $\mathcal{Y}$, then there are both fwd.-bwd. and bwd.-fwd. ones from $\mathcal{X}$ to $\mathcal{Y}$.*

Proof. A forward simulation f from $\mathcal{X}$ to $\mathcal{Y}$ induces a backward-forward simulation $(\mathcal{X}, J(\mathrm{id}), f)$; it has $\mathcal{X}$ itself as an intermediate system. The other cases are similar. □

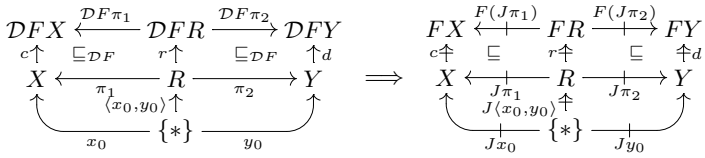
Lemma 5.14 *Let F be a polynomial functor. Then the functor $\mathcal{D}F$ has the following natural order. This makes $(\mathcal{D}F, \sqsubseteq_{\mathcal{D}F})$ a functor with preorder (Def. 4.3).*

$$\gamma \sqsubseteq_{\mathcal{D}FX} \delta \quad \overset{\text{def.}}{\iff} \quad \gamma(u) \leq \delta(u) \quad \textit{for all } u \in FX. \qquad \square$$

When $F = \{\checkmark\} + \mathsf{Ac} \times (\_)$ and $B = \mathcal{D}F$, both B-coalgebras and probabilistic F-systems represent GPAs. In this case, the order on B in the previous definition coincides with the one in Ex. 4.4.

Here comes our second main observation.

Theorem 5.15 (HJ is Kleisli) *Let $X \xrightarrow{c} \mathcal{D}FX$ and $Y \xrightarrow{d} \mathcal{D}FY$ be $\mathcal{D}F$-coalgebras, $x_0 \in X$ and $y_0 \in Y$ be chosen (initial) states, and $R \subseteq X \times Y$ be a relation. Assume that there exists a function r that validates the inequalities in the diagram on the left, that is, that R is a HJ-simulation from c to d such that $x_0 R y_0$.*



(6)

Then we have a fwd.-bwd. simulation from the F-system (X, Jx_0, c) to (Y, Jy_0, d), shown above on the right. Note the order $\sqsubseteq$ therein refers to the one in Def. 5.3. □

In short: a HJ-simulation between $\mathcal{D}F$-coalgebras induces a fwd.-bwd. simulation between the corresponding F-systems. The proof is found in [7].

Fwd.-bwd. simulation instantiates to GPA, like in §5.4. Thm. 4.7 yields:

Corollary 5.16 (JL is Kleisli) *Let $\mathcal{X}$ and $\mathcal{Y}$ be GPAs. A JL-simulation R from $\mathcal{X}$ to $\mathcal{Y}$ induces a fwd.-bwd. (Kleisli) simulation from $\mathcal{X}$ to $\mathcal{Y}$.* □

5.6 Generic Trace Semantics

Like in [16], the principal aim of Kleisli simulation is to show *trace inclusion*—a refinement relation with respect to (linear time) *trace semantics* which is the coarsest in the spectrum of [3]. Our use of the generic notion of Kleisli coalgebra calls for a generic definition of trace semantics too. We employ the theory in [8]; here is its quick recap.

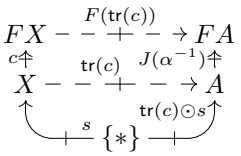
A polynomial functor F always has an *initial algebra* $\alpha : FA \xrightarrow{\cong} A$. The intuition is: F represents a set of datatype constructors; and A is the induced inductive datatype. The algebraic structure α always becomes an invertible function.

Example 5.17 The functor $F = \{\checkmark\} + \mathsf{Ac} \times (\_)$ is thought of as: a nullary constructor $\checkmark$ and a family of unary constructors $a(\_)$, for each $a \in \mathsf{Ac}$. The induced inductive datatype is the set $\mathsf{Ac}^* = \{a_1 a_2 \cdots a_n \checkmark \mid n < \omega, a_i \in \mathsf{Ac}\}$ of (finite) lists over Ac. This set indeed carries an initial algebra: there is a canonical algebraic structure $\alpha : \{\checkmark\} + \mathsf{Ac} \times \mathsf{Ac}^* \xrightarrow{\cong} \mathsf{Ac}^*$, namely

$$\checkmark \longmapsto \checkmark \text{ (the empty sequence)}, \qquad (a, a_1 \cdots a_n \checkmark) \longmapsto a a_1 \cdots a_n \checkmark \ .$$

The following is the main result in [8], adapted to the current context.

Theorem 5.18 (Generic (finite) trace semantics) *Let $\alpha : FA \to A$ be an initial algebra. Given any F-system $\mathcal{X} = (X, s, c)$, there is a unique Kleisli arrow $\mathsf{tr}(c)$ that makes the diagram on the right commute. In particular, the Kleisli coalgebra $J(\alpha^{-1})$ is a* final *coalgebra.*



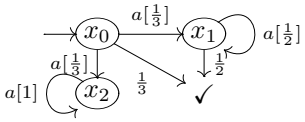
(7)

We set $\mathsf{tr}(\mathcal{X}) := \mathsf{tr}(c) \odot s$. It is $\mathsf{tr}(\mathcal{X}) : \{*\} \to \mathcal{D}A$ as a function, hence is a subdistribution over A. This $\mathsf{tr}(\mathcal{X})$ is referred to as the *(finite) trace semantics* of $\mathcal{X}$. To summarize: the *action type* F determines the set A of *linear-time behavior*; $\mathcal{X}$'s trace semantics $\mathsf{tr}(\mathcal{X})$ tells us which linear-time behavior is exhibited with how much likelihood.

Example 5.19 (Trace semantics for GPA) Let $F = \{\checkmark\} + \mathsf{Ac} \times (\_)$. The diagram (7) translates into the following conditions, where σ ranges over Ac^*.

$$\begin{aligned} \mathsf{tr}(c)(x)(\checkmark) &= \Pr[x \to \checkmark] \ , \\ \mathsf{tr}(c)(x)(a\sigma) &= \textstyle\sum_{x' \in X} \Pr[x \xrightarrow{a} x'] \cdot \mathsf{tr}(c)(x')(\sigma) \ ; \quad \text{and} \\ \mathsf{tr}(\mathcal{X})(\sigma) &= \mathsf{tr}(c)(x_0)(\sigma) \quad \text{when } s(*) = [x_0 \mapsto 1]. \end{aligned}$$

This is a reasonable definition of a "trace semantics" for GPA; resulting is a subdistribution $\mathsf{tr}(\mathcal{X})$ over lists on Ac. For example, let $\mathcal{X}$ be the GPA below on the left; then its trace semantics is as on the right.



$$\mathsf{tr}(\mathcal{X}) = \begin{bmatrix} \checkmark \mapsto \frac{1}{3},\ a\checkmark \mapsto \frac{1}{3} \cdot \frac{1}{2},\ a^2\checkmark \mapsto \frac{1}{3} \cdot \frac{1}{2} \cdot \frac{1}{2},\ \cdots \\ a^n\checkmark \mapsto \frac{1}{3} \cdot \left(\frac{1}{2}\right)^n,\ \cdots \end{bmatrix}$$

Note that our trace semantics only captures *finite* behavior; infinite sequences like a^ω are not in its domain Ac^*. With infinite behavior included we no longer have a clean characterization like in Thm. 5.18.

Like the definition of Kleisli simulation, the generic trace semantics (Thm. 5.18) also applies to other kinds of branching such as non-determinism. See [8].

5.7 Soundness Theorems

We recall the soundness result [5] for Kleisli simulation, via which soundness of JL- and HJ-simulation immediately follows. Its short proof in [5] makes use of order-theoretic properties of the diagram (7).

Theorem 5.20 (Soundness of Kleisli) *Let $\mathcal{X}, \mathcal{Y}$ be F-systems. Existence of a Kleisli simulation from $\mathcal{X}$ to $\mathcal{Y}$ implies trace inclusion: $\mathsf{tr}(\mathcal{X}) \sqsubseteq \mathsf{tr}(\mathcal{Y})$. Here a Kleisli simulation can be any of forward, backward, or hybrid.* □

Using Thm. 5.15 and Cor. 5.16, we immediately obtain soundness of JL-simulation (Def. 4.12). This is new to the best of the author's knowledge. Therefore the notion of JL-simulation can also be used for proving trace inclusion between GPAs, a use that has not been investigated much in the literature. The same applies to JL- and JL'-simulation for DTMCs; we postpone detailed treatment to another venue.

6 Conclusions and Future Work

We have showed that JL-simulation is a special case of HJ-simulation, which is further a special case of Kleisli simulation. This allows to transfer general results for a latter notion to a former one, most notably soundness.

Finding a Kleisli simulation is reduced to solving a family of linear inequalities. Its algorithmic aspect is to be investigated. We also aim to exploit acquired genericity and apply our results to other kinds of systems. We are interested in *stochastic context-free grammars* which have their application in modeling the secondary structure of RNA [2].

References

1. C. Baier, J.P. Katoen, H. Hermanns and V. Wolf. Comparative branching-time semantics for markov chains. *Inf. & Comp.*, 200(2):149–214, 2005.
2. R. Durbin, S.R. Eddy, A. Krogh and G. Mitchison. *Biological Sequence Analysis*. Cambridge Univ. Press, 1998.
3. R.J. van Glabbeek. The linear time–branching time spectrum I; the semantics of concrete, sequential processes. In J.A. Bergstra, A. Ponse and S.A. Smolka, editors, *Handbook of Process Algebra*, chap. 1, pp. 3–99. Elsevier, 2001.
4. R.J. van Glabbeek, S.A. Smolka and B. Steffen. Reactive, generative, and stratified models of probabilistic processes. *Inf. & Comp.*, 121:59–80, 1995.
5. I. Hasuo. Generic forward and backward simulations. In C. Baier and H. Hermanns, editors, *International Conference on Concurrency Theory (CONCUR 2006)*, vol. 4137 of *Lect. Notes Comp. Sci.*, pp. 406–420. Springer, Berlin, 2006.
6. I. Hasuo. *Tracing Anonymity with Coalgebras*. PhD thesis, Radboud Univ. Nijmegen, 2008.
7. I. Hasuo. Generic forward and backward simulations II: Probabilistic simulations. Extended version, to appear in RIMS Preprints, June 2010.
8. I. Hasuo, B. Jacobs and A. Sokolova. Generic trace semantics via coinduction. *Logical Methods in Comp. Sci.*, 3(4:11), 2007.
9. I. Hasuo, Y. Kawabe and H. Sakurada. Probabilistic anonymity via coalgebraic simulations. *Theor. Comp. Sci.*, 411(22–24):2239–2259, 2010.
10. J. Hughes and B. Jacobs. Simulations in coalgebra. *Theor. Comp. Sci.*, 327(1-2):71–108, 2004.
11. B. Jacobs. Introduction to coalgebra. Towards mathematics of states and observations, 2005. Draft of a book, `www.cs.ru.nl/B.Jacobs/PAPERS`.
12. B. Jonsson and K.G. Larsen. Specification and refinement of probabilistic processes. In *LICS*, pp. 266–277. IEEE Computer Society, 1991.
13. B. Klin. Bialgebraic methods and modal logic in structural operational semantics. *Inf. & Comp.*, 207(2):237–257, 2009.
14. V.G. Kulkarni. *Modeling and Analysis of Stochastic Systems*. Chapman & Hall, 1995.
15. K.G. Larsen and A. Skou. Bisimulation through probabilistic testing. *Inf. & Comp.*, 94(1):1–28, 1991.
16. N. Lynch and F. Vaandrager. Forward and backward simulations. I. Untimed systems. *Inf. & Comp.*, 121(2):214–233, 1995.
17. J.J.M.M. Rutten. Universal coalgebra: a theory of systems. *Theor. Comp. Sci.*, 249:3–80, 2000.
18. R. Segala. *Modeling and verification of randomized distributed real-time systems*. PhD thesis, MIT, 1995.
19. A. Sokolova. *Coalgebraic Analysis of Probabilistic Systems*. PhD thesis, Techn. Univ. Eindhoven, 2005.