Generic Forward and Backward Simulations*

蓮尾 一郎† バート・ヤコブス‡ アナ・ソコロバ
Ichiro Hasuo Bart Jacobs Ana Sokolova

ナイメーヘン・ラドバウド大学 計算機科学科 （オランダ）
Institute for Computing and Information Sciences, Radboud University Nijmegen, the Netherlands
http://www.cs.ru.nl/~{ichiro, bart, anas}

The technique of forward/backward simulations has been applied successfully in many distributed and concurrent applications. In this paper, however, we claim that the technique can actually have more genericity and mathematical clarity. We do so by identifying forward/backward simulations as lax/oplax morphisms of coalgebras. Starting from this observation, we present a systematic study of this generic notion of simulations. It is meant to be a generic version of the study by Lynch and Vaandrager, covering both non-deterministic and probabilistic systems. In particular we prove soundness and completeness results with respect to trace inclusion: the proof is by coinduction using the generic theory of traces developed by the authors. By suitably instantiating our generic framework, one obtains the appropriate definition of forward/backward simulations for various kinds of systems, for which soundness and completeness come *for free*.

1 Introduction

The theory of forward/backward simulations for non-deterministic automata has been extensively studied, notably by Lynch and Vaandrager [13]. It has been applied successfully in many distributed and concurrent applications, described as transition systems. For example, in [10] trace-based anonymity properties for network protocols are proved by building backward simulations. The notions of forward/backward simulations are also extended to different kinds of state-based systems such as probabilistic ones [19].

In this paper we claim that this theory of forward/backward simulations can actually have more genericity and mathematical clarity. We do so by revealing a simple mathematical structure hidden behind various notions of simulations defined for different kinds of systems. The slogan is:

Forward/backward simulations are lax/oplax morphisms
of coalgebras in Kleisli categories.

Based on this observation, we aim at presenting a generic version of the systematic study [13]. The outcome is satisfactory. We employ the generic theory of traces in [5] and show:

- *Soundness.* Existence of a forward or backward simulation implies trace inclusion.
- *Completeness.* Trace inclusion implies existence of a certain kind of hybrid simulation, namely a backward-forward simulation.

The important point is that all these definitions and proofs are stated in abstract coalgebraic terms, hence come with ample genericity. In fact they are parametrized by:

*This paper is an extended version of the previous ones [5, 6]. Section 7 contains additional materials that have not been presented before.

†本研究の一部は，産業技術総合研究所 システム検証研究センターでの滞在中に行われた．ホストの渡邊 宏氏に謝意を表する．

‡Also part-time at Technical University Eindhoven, the Netherlands.

- The type of branching. It can be either non-determinism (with a set of possible transitions) or probabilism (with a distribution over possible transitions).
- The type of transitions. For example, a context-free grammar can be considered as a state-based system—non-terminals as states—with non-deterministic branching. It has a different transition type from, say, LTS's: a CFG transits to a word over symbols and states, while an LTS transits to a pair of a symbol and a (next) state. Our result covers a wide variety of transition types.

Hence for each application from such a wide variety, one can obtain a definition of forward/backward simulations by instantiating our general framework with suitable parameters. Moreover one is assured that this definition is the *right* one: good properties such as soundness and completeness come for free. Therefore we expect abundant practical implication of this work.

Now let us take a completely different standpoint, namely that of a coalgebra-theorist. This work cultivates a new field of coalgebraic methods in computer science: coalgebras in a Kleisli category. The standard theory of coalgebras (e.g. [18]) is based in **Sets**, establishing the (successful) second row of the table. This paper, following the previous work [5], extends this table downwards.

base category	morphisms of coalgebras	coinduction gives
Sets	functional bisimulation	bisimilarity
Kleisli	lax ··· forward simulation oplax ··· backward simulation [this paper]	trace semantics [5]

What is new in the current version of this paper is an account on internal actions, which is left as future work in the previous version [6]. We describe elimination (or abstraction) of internal actions using our scheme of generic trace theory [5] suitably adapted. Accompanying is a technical result showing the equivalence of the following two different kinds of "trace semantics" for systems with internal actions.

$$\begin{array}{ccc} \boxed{\text{system with internal actions } \tau} & \xmapsto{\text{elimination of } \tau} & \boxed{\text{“closure” system without } \tau} \\ {\scriptstyle\text{trace semantics}}\downarrow & & \downarrow{\scriptstyle\text{trace semantics}} \\ \boxed{\text{trace containing explicit } \tau} & \xmapsto[\text{elimination of } \tau]{} & \boxed{\text{trace without } \tau} \end{array} \tag{1}$$

This result is formulated and proved with the coalgebraic abstraction and genericity.

The paper is organized as follows. In Section 2 our basic (coalgebraic) setting is presented. State-based systems are formulated as coalgebras with explicit start states in Section 3. The key notion of generic forward/backward simulations is presented in Section 4. In Section 5 we recall the generic theory of coalgebraic traces from [5]. The materials of the previous two sections are combined in Section 6 to prove soundness and completeness. The additional account on internal actions is included in Section 7. We conclude in Section 8.

Notation and terminology. In diagrams, triangles and squares with no $\sqsubseteq$ or $\sqsupseteq$ inside are designated to commute. The word *coinduction* refers to an argument using the finality of a final coalgebra.

2 Preliminaries

This section presents preliminaries from category theory and theory of coalgebras. They are put in an elementary and descriptive manner. For more details the reader is referred to [5].

In this paper we identify forward/backward simulations as lax/oplax morphisms of coalgebras in a Kleisli category $\mathcal{K}\ell(T)$ for a monad T on **Sets**. This observation is inspired by a series of work [17, 8, 4, 5] on trace semantics for/via coalgebras: a Kleisli category is a suitable base category there. Our basic story is as follows.

We model a state-based system as a coalgebra $X \to TFX$ in **Sets**, with T a monad, F a functor and a distributive law $FT \Rightarrow TF$ implicit. The intuition is:

- a monad T describes the type of *branching* (non-determinism, probabilism, etc.) of the system;
- a functor F describes the *transition* type of the system, which determines the type of linear-time behavior (e.g. words over action symbols);
- a distributive law $FT \Rightarrow TF$ describes the way how T's effect of branching is distributed over the transition type represented by F.

It turns out that having $X \to TFX$ in **Sets** is equivalent to having a coalgebra $X \to \overline{F}X$ in the Kleisli category $\mathcal{K}\ell(T)$, where $\overline{F} : \mathcal{K}\ell(T) \to \mathcal{K}\ell(T)$ is a canonical lifting of $F : \mathbf{Sets} \to \mathbf{Sets}$ with $\overline{F}X = FX$. This lifting $\overline{F}$ is induced by the distributive law. To summarize:

- In modelling a system as a coalgebra $X \to TFX$, we separate the type of branching T from the transition type F.
- By moving from **Sets** to $\mathcal{K}\ell(T)$, this coalgebra becomes a coalgebra $X \to \overline{F}X$ for a functor $\overline{F}$—instead of a combination TF. Then we can start the usual coalgebraic business such as morphisms, final coalgebras and coinduction.

2.1 Monads for types of branching

A *monad* T on **Sets** is an endofunctor on **Sets** equipped with two kinds of functions: for each set X, the *unit* $X \stackrel{\eta_X}{\to} TX$ and the *multiplication* $TTX \stackrel{\mu_X}{\to} TX$. These functions must satisfy certain coherence conditions.

The use of monads in computer science is most notably announced in the seminal paper [15]. There monads are used for modelling computations with effects. This leads to monadic types in functional programming languages such as Haskell.

In coalgebraic settings, it is shown in [5] that monads with a certain order structure are suitable for modelling state-based systems with branching, especially for analyzing their trace semantics. We are interested in such monads in this paper. We have two examples:

- The *powerset monad* $\mathcal{P}$, modelling the *non-deterministic* branching.
- The *subdistribution monad* $\mathcal{D}$, modelling the *probabilistic* branching. For a set X, $\mathcal{D}X$ is given by: $\mathcal{D}X = \{\, \xi : X \to [0,1] \mid \sum_{x \in X} \xi(x) \le 1 \,\}$. Here ξ is called a *probability subdistribution* over X. It is "sub" because the sum of all probabilities is not necessarily equal to 1.

The reason that we take the subdistribution monad $\mathcal{D}$, instead of the distribution monad $\mathcal{D}_{=1}X = \{\xi \mid \sum_x \xi(x) = 1\}$, is that the latter lacks a suitable order structure. This point is elaborated in Section 2.3.

2.2 Kleisli categories for monads

For each monad T on **Sets**, we construct the *Kleisli category* for T, denoted by $\mathcal{K}\ell(T)$, in the following way. The crucial part is that an arrow $X \to Y$ in $\mathcal{K}\ell(T)$ is actually a function $X \to TY$ in **Sets**.

- Objects in $\mathcal{K}\ell(T)$ are the same as in **Sets**: they are just sets.

- An arrow $X \to Y$ in $\mathcal{K}\ell(T)$ is a function $X \to TY$ in **Sets**.
- Composition of arrows is defined using multiplication $\mu_X : TTX \to TX$.
- The identity arrow $X \stackrel{\mathrm{id}}{\to} X$ in $\mathcal{K}\ell(T)$ is the unit $X \stackrel{\eta_X}{\to} TX$ in **Sets**.

This $\mathcal{K}\ell(T)$ will be our base category. Notice that when we write $X \to Y$ in $\mathcal{K}\ell(T)$, a branching nature of this arrow is implicit because it is a function $X \to TY$.

For the monads $\mathcal{P}$ and $\mathcal{D}$ of our interest, we shall describe more details of their Kleisli categories.

The category $\mathcal{K}\ell(\mathcal{P})$ is in fact isomorphic to the category **Rel** of sets and relations. That is, an arrow $X \to Y$ in $\mathcal{K}\ell(\mathcal{P})$ is a relation between X and Y via the standard "relation-into-fuction" trick: given a function $f : X \to \mathcal{P}Y$ in **Sets** we obtain a relation $R_f = \{(x,y) \mid y \in f(x)\}$. In particular, composition of arrows in $\mathcal{K}\ell(\mathcal{P})$ is given by the relational composition $S \circ R = \{(x,z) \mid \exists y.\, xRy \wedge ySz\}$ of the corresponding relations. The identity arrow id_X is the diagonal relation $\{(x,x) \mid x \in X\}$.

In $\mathcal{K}\ell(\mathcal{D})$ an arrow $X \to Y$ assigns to each $x \in X$ a probability subdistribution over Y. The identity arrow $X \stackrel{\mathrm{id}}{\to} X$ maps $x \in X$ to the so-called *Dirac distribution* for x. The composition of arrows $X \stackrel{f}{\to} Y \stackrel{g}{\to} Z$ in $\mathcal{K}\ell(\mathcal{D})$ is such that: for $x \in X$ and $z \in Z$, $(g \circ f)(x)(z) = \sum_{y \in Y} f(x)(y) \cdot g(y)(z)$.

2.3 Order-enriched structure of Kleisli categories

The notion of branching—such as non-determinism and probabilism—come with natural notions of order. For non-determinism we have the inclusion order between sets of possible transitions. For probabilism a subdistribution ξ is bigger than ψ if, to each possible transition, ξ assigns bigger probability than ψ does.

These natural orders accompanying the notion of branching appear in our setting as a $\mathbf{DCpo}_\perp$-enriched structure of Kleisli categories. This order structure is fully exploited in the definition of forward/backward simulations: a system simulates another one if it has *more* behavior.

For $T = \mathcal{P}$ or $\mathcal{D}$, the Kleisli category $\mathcal{K}\ell(T)$ is $\mathbf{DCpo}_\perp$*-enriched*. This means:

- For any pair of sets X and Y, the set $\mathrm{Hom}_{\mathcal{K}\ell(T)}(X,Y)$ of the arrows from X to Y has a dcpo structure $\sqsubseteq$ with bottom. In particular we can take the supremum $\bigsqcup_{n<\omega} f_n$ of an increasing chain $f_0 \sqsubseteq f_1 \sqsubseteq \cdots$ of arrows, and there is the minimum arrow $\perp_{X,Y} : X \to Y$.
- Composition of arrows is continuous: $g \circ (\bigsqcup_n f_n) = \bigsqcup_n (g \circ f_n)$ and $(\bigsqcup_n f_n) \circ h = \bigsqcup_n (f_n \circ h)$. In particular composition is monotone.

Indeed, for $T = \mathcal{P}$ or $\mathcal{D}$, a set TY has a $\mathbf{DCpo}_\perp$ structure $\sqsubseteq_{TY}$. This extends to the order between arrows in $\mathcal{K}\ell(T)$ in a pointwise manner: for $f, g : X \rightrightarrows Y$, $f \sqsubseteq g$ if for each $x \in X$, $f(x) \sqsubseteq_{TY} g(x)$.

We need the minimum arrow $\perp_{X,Y} : X \to Y$ in $\mathcal{K}\ell(T)$ for the trace semantics results in Section 5. It is not available for the distribution monad $\mathcal{D}_{=1}$: that is why we use the subdistribution monad $\mathcal{D}$ instead.

2.4 Shapely functors for transition types

We restrict a functor F—which models the transition type of a system—to be *shapely*. The reason to do so is: we know the results on coalgebraic trace semantics in Section 5 hold for shapely functors,[1] and also in most of the interesting examples we can take as F a shapely functor. The family of shapely functors is almost as broad as that of polynomial functors: it is defined inductively by the following BNF notation.

$$F, G, F_i ::= \mathrm{id} \mid \Sigma \mid F \times G \mid \coprod_{i \in I} F_i \quad ,$$

[1] This does not say that those results hold exclusively for shapely functors.

where Σ denotes the constant functor into an arbitrary set Σ, and I is an arbitrary index set. Here are some virtue of shapely functors which we will exploit.

- An initial F-algebra exists, obtained via the initial sequence of length ω.
- For $T = \mathcal{P}$ or $\mathcal{D}$, there is a canonical distributive law $FT \Rightarrow TF$. Equivalently, F has a canonical lifting $\overline{F}$ on $\mathcal{K}\ell(T)$. On objects $\overline{F}X = FX$, and on arrows $\overline{F}$'s action is what one might think of at first sight.

3 Coalgebraic modelling of systems

In this section we model a wide variety of branching state-based systems as what we call (T, F)-*systems*. A (T, F)-system is a $\overline{F}$-coalgebra in the Kleisli category $\mathcal{K}\ell(T)$ plus explicit start states. This definition of (T, F)-systems will be motivated by several illustrating examples.

Two parameters in the notion of (T, F)-systems are: T is a monad, being either $\mathcal{P}$ or $\mathcal{D}$, representing the branching type; F is a shapely functor describing the transition type. In the sequel we assume that T and F are such.

Definition 3.1 ((T, F)-systems) A (T, F)-*system* is a pair of arrows

$$1 \xrightarrow{s} X \xrightarrow{c} \overline{F}X \qquad \text{in the Kleisli category } \mathcal{K}\ell(T).$$

That is, a pair of functions $(s : 1 \to TX,\ c : X \to TFX)$ in **Sets**, recalling that $\overline{F}X = FX$. The arrow s is called the *start states map*, and the $\overline{F}$-coalgebra c is called the *dynamics*. The set X is called the *state space*. The only element of the singleton 1 appearing here[2] is denoted by $*$.

In most literature on coalgebras the start state (or the set of start states) is usually left implicit. However in this paper start states are explicit as one ingredient of the notion of systems. The reason is explained in Appendix A.2.

Example 3.2 (Non-deterministic automata) Let us take the powerset monad $\mathcal{P}$ for T, hence non-deterministic branching. For an endofunctor F we take $1 + \Sigma \times \_$, where $1 = \{\checkmark\}$ is a singleton and Σ is a non-empty set of symbols. A (T, F)-system then is a pair of functions in **Sets**,

$$\left(1 \xrightarrow{s} \mathcal{P}X, \quad X \xrightarrow{c} \mathcal{P}(1 + \Sigma \times X) \right) ,$$

which should be interpreted as follows. The subset $s(*)$ of X is the set of possible start states. For a state $x \in X$, the set $c(x)$ contains $\checkmark$ if x is an accepting state; it contains a tuple (a, x') if there is a (possible) transition $x \xrightarrow{a} x'$. In this way a (T, F)-system for these T and F is thought of as a non-deterministic automaton.

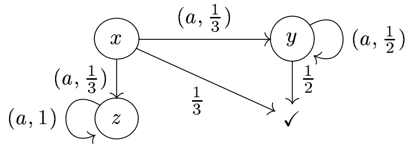
Example 3.3 (Probabilistic automata) Let us take $T = \mathcal{D}$ instead of $\mathcal{P}$ in the previous example. A (T, F)-system is a pair of functions in **Sets**:

$$\left(1 \xrightarrow{s} \mathcal{D}X, \quad X \xrightarrow{c} \mathcal{D}(1 + \Sigma \times X) \right) .$$

This is understood as follows. The subdistribution $s(*)$ over X represents the probability with which each state $x \in X$ is chosen as a starting state. An execution successfully terminates at x with the probability

[2] In this paper we will have singletons with different computational meanings. Accordingly, their only elements will be denoted by different symbols.

$c(x)(\checkmark)$; a transition $x \stackrel{a}{\to} x'$ is made with the probability $c(x)(a, x')$. Such a system is called a *generative probabilistic transition system* [22, 21]: in this paper we shall call it simply a *probabilistic automaton*. Here is an example of a probabilistic automaton.



with start states: $\begin{bmatrix} x & \mapsto & 1/3 \\ y & \mapsto & 2/3 \end{bmatrix}$.

This is modelled as the following $(\mathcal{D}, 1 + \Sigma \times \_)$-system.

- The start state map $1 \stackrel{s}{\to} \mathcal{D}X$ is such that $s(*) = \begin{bmatrix} x & \mapsto & 1/3 \\ y & \mapsto & 2/3 \end{bmatrix}$, and
- the dynamics coalgebra $X \stackrel{c}{\to} \mathcal{D}(1 + \Sigma \times X)$ is such that

  $c(x) = [\,(a, y) \mapsto 1/3,\ (a, z) \mapsto 1/3,\ \checkmark \mapsto 1/3\,]$, etc.

Example 3.4 (Systems with distinct input/output actions) In some cases we would like to distinguish two different kinds of transitions: those with an input action and with an output action. This is the case for (non-deterministic) I/O automata [14] and probabilistic I/O automata [24]. This is done by replacing the functor F in the previous examples: we take $F = 1 + I \times \_ + O \times \_$ instead, where I and O are disjoint sets of input and output actions.

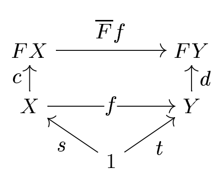
Remark 3.5 (Example 3.4 vs. probabilistic I/O automata [24]) $(\mathcal{D}, F)$-systems in the previous example have two significant differences from the well-studied notion of probabilistic I/O automata. One is that successful termination is explicit by the presence of $1 = \{\checkmark\}$ in F. This is due to our choice of finite traces—which have good characterization via coinduction—as the semantics: without explicit termination the set of finite traces is always empty.

The other is that probabilistic I/O automata have both non-deterministic and probabilistic branching at the same time, while $(\mathcal{D}, F)$-systems in the previous example lack non-deterministic branching. Modelling the combination of non-determinism and probabilism using a suitable monad is left as future work.[3]

Example 3.6 (Context-free grammar, [4]) When $T = \mathcal{P}$ and $F = (\Sigma + \_)^*$, a (T, F)-system is thought of as a context-free grammar (without finiteness assumptions), together with a set of possible starting non-terminals.

The notion of morphisms of coalgebras extends to (T, F)-systems.

Definition 3.7 (Morphisms of systems) Let $1 \stackrel{s}{\to} X \stackrel{c}{\to} FX$ and $1 \stackrel{t}{\to} Y \stackrel{d}{\to} FY$ be (T, F)-systems, presented in $\mathcal{K}\ell(T)$. A *morphism of* (T, F)*-systems* from (s, c) to (t, d) is an arrow $f : X \to Y$ in $\mathcal{K}\ell(T)$ that makes the following diagram commute.



[3]In [23] a monad for the combination of non-determinism and probabilism is proposed. However we have not yet found a suitable dcpo structure for this monad.

4 Forward/backward simulations, coalgebraically

This section presents the key notions of this paper: generic forward, backward and backward-forward simulations. The intuition about order accompanying the notion of "branching"—now substantiated as the $\mathbf{DCpo}_\perp$-enriched structure of a Kleisli category—is fully exploited here.

In this section again $T = \mathcal{P}$ or $\mathcal{D}$, and F is a shapely functor.

Definition 4.1 (Forward simulation) Let $1 \overset{s}{\to} X \overset{c}{\to} FX$ and $1 \overset{t}{\to} Y \overset{d}{\to} FY$ be (T, F)-systems, presented in $\mathcal{K}\ell(T)$. A *forward simulation* from (t, d) to (s, c) is an arrow $f : X \to Y$ in $\mathcal{K}\ell(T)$ such that:

$$t \sqsubseteq f \circ s \quad \text{and} \quad d \circ f \sqsubseteq \overline{F} f \circ c \ ,$$

where $\sqsubseteq$ refers to the order available due to the $\mathbf{DCpo}_\perp$-enriched structure of the Kleisli category. Diagramatically presented,

$$\begin{array}{ccc} FX & \xrightarrow{\ \overline{F}f\ } & FY \\ {\scriptstyle c}\uparrow & \sqsupseteq & \uparrow{\scriptstyle d} \\ X & \xrightarrow{\ f\ } & Y \\ & {\scriptstyle s}\nwarrow \ \sqsupseteq \ \nearrow{\scriptstyle t} & \\ & 1 & \end{array} \ . \qquad (2)$$

In other words, a forward simulation is a *lax morphism* from (s, c) to (t, d).

We write $(t, d) \sqsubseteq_{\mathbf{F}} (s, c)$ if there is a forward simulation from (t, d) to (s, c).

The use of lax morphisms in categorical accounts of simulation/refinement is found in [11]. In a coalgebraic setting, [2] uses lax morphisms of coalgebras to investigate order-enriched version of bisimulation. However, to the best of our knowledge, we are the first to notice the significance of lax morphisms in Kleisli categories.

The dual notion, with the order of arrows opposed, has also a significant computational meaning.

Definition 4.2 (Backward simulation) Let $1 \overset{s}{\to} X \overset{c}{\to} FX$ and $1 \overset{t}{\to} Y \overset{d}{\to} FY$ be (T, F)-systems, presented in $\mathcal{K}\ell(T)$. A *backward simulation* from (s, c) to (t, d) is an arrow $f : X \to Y$ in $\mathcal{K}\ell(T)$ such that: $f \circ s \sqsubseteq t$ and $\overline{F} f \circ c \sqsubseteq d \circ f$. Diagramatically presented,

$$\begin{array}{ccc} FX & \xrightarrow{\ \overline{F}f\ } & FY \\ {\scriptstyle c}\uparrow & \sqsubseteq & \uparrow{\scriptstyle d} \\ X & \xrightarrow{\ f\ } & Y \\ & {\scriptstyle s}\nwarrow \ \sqsubseteq \ \nearrow{\scriptstyle t} & \\ & 1 & \end{array} \ . \qquad (3)$$

Hence a backward simulation is an *oplax morphism* of systems.

We write $(s, c) \sqsubseteq_{\mathbf{B}} (t, d)$ if there is a backward simulation from (s, c) to (t, d).

Remark 4.3 Note the direction of forward/backward simulations and lax/oplax morphisms. In general, the system which appears on the smaller sides of inequalities is simulated by the other one. For example, a lax morphism from (s, c) to (t, d) in Diagram (2) is a forward simulation *from* (t, d) *to* (s, c), through which (s, c) *forward-simulates* (t, d); hence $(t, d) \sqsubseteq_{\mathbf{F}} (s, c)$.

Let us be convinced of these abstract definitions by looking at examples.

Example 4.4 (Non-deterministic automata) In the setting of Example 3.2, an arrow $X \to Y$ in $\mathcal{K}\ell(T)$ is a relation R from X to Y since $\mathcal{K}\ell(\mathcal{P}) \cong \mathbf{Rel}$. The previous definitions boil down as follows: R

is a forward simulation from (t,d) to (s,c) if and only if it satisfies the following conditions.

$$\begin{array}{rcl} y \in \mathsf{start}_{(t,d)} & \Longrightarrow & \exists x \in \mathsf{start}_{(s,c)}.\quad xRy \ , \\ xRy \ \wedge \ y \to_d \checkmark & \Longrightarrow & x \to_c \checkmark \ , \\ xRy \ \wedge \ y \xrightarrow{a}_d y' & \Longrightarrow & \exists x' \in X. \quad \big(\, x \xrightarrow{a}_c x' \ \wedge \ x'Ry' \, \big) \ , \end{array}$$

where $\mathsf{start}_{(s,c)}$ denotes the set $s(*)$. These conditions are much like those in the standard literature [13]. Notice in particular that the third condition is of the following familiar form, working "forwards".

$$\begin{array}{ccc} x & & \\ R\,\vdots & & \\ y \xrightarrow{a} y' & & \end{array} \quad \Longrightarrow \quad \begin{array}{ccc} x & \xrightarrow{a} & \exists x' \\ R\,\vdots & & \vdots\,R \\ y & \xrightarrow{a} & y' \end{array}$$

Similarly, a relation R from X to Y is a backward simulation from (s,c) to (t,d) if and only if:

$$\begin{array}{rcl} x \in \mathsf{start}_{(s,c)} \ \wedge \ xRy & \Longrightarrow & y \in \mathsf{start}_{(t,d)} \ , \\ x \to_c \checkmark & \Longrightarrow & \exists y \in Y. \quad \big(\, xRy \ \wedge \ y \to_d \checkmark \, \big) \ , \\ x \xrightarrow{a}_c x' \ \wedge \ x'Ry' & \Longrightarrow & \exists y \in Y. \quad \big(\, xRy \ \wedge \ y \xrightarrow{a}_d y' \, \big) \ . \end{array}$$

The third condition here works "backwards" in the following way.

$$\begin{array}{ccc} x & \xrightarrow{a} & x' \\ & & \vdots\,R \\ & & y' \end{array} \quad \Longrightarrow \quad \begin{array}{ccc} x & \xrightarrow{a} & x' \\ R\,\vdots & & \vdots\,R \\ \exists y & \xrightarrow{a} & y' \end{array}$$

Example 4.5 (Probabilistic automata) In the setting of Example 3.3, the abstract Definition 4.1 is instantiated as follows: a function $f : X \to \mathcal{D}Y$ in **Sets** is a forward simulation from (t,d) to (s,c) if and only if:

$$\begin{array}{rcl} t(*)(y) & \leq & \sum_{x \in X} s(*)(x) \cdot f(x)(y) \ , \\ \sum_{y \in Y} f(x)(y) \cdot d(y)(\checkmark) & \leq & c(x)(\checkmark) \ , \\ \sum_{y \in Y} f(x)(y) \cdot d(y)(a, y') & \leq & \sum_{x' \in X} c(x)(a, x') \cdot f(x')(y') \ . \end{array} \tag{4}$$

It is also straightforward to instantiate Definition 4.2 of backward simulations.

One may wonder why we can call such f a forward simulation, although one can notice that a "forward" argument similar to the previous example is going on. The point is that, however, by the abstract theorems in the following sections we know that this definition (4) of forward simulations—derived from the coalgebraic definition—satisfies desirable properties such as soundness/completeness with respect to trace inclusion.

We define a simulation from one probabilistic system to another to be a function $X \to \mathcal{D}Y$. This is different from the approach in [7]: there a simulation is always a relation between state spaces X and Y.

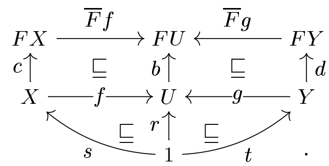
It is also straightforward to instantiate notions of generic forward/backward simulations with T and F in Example 3.6. Then we get appropriate notions of simulations for context-free grammars.

Forward and backward simulations will be shown to be sound with respect to trace inclusion. But they in general fail to be complete. Instead, a completeness result is proved for a certain combination of forward and backward simulations (*hybrid* simulations), as is done in [13].

Definition 4.6 (Backward-forward simulations) Let (s,c) and (t,d) be (T,F)-systems. A *backward-forward simulation* from (s,c) to (t,d) is a pair of

- a backward simulation f from (s,c) to some intermediate (T,F)-system (r,b), and
- a forward simulation g from the intermediate system (r,b) to (t,d).

Diagramatically presented in $\mathcal{K}\ell(T)$ (note the direction of arrows),



(5)

We write $(s,c) \sqsubseteq_{\mathbf{BF}} (t,d)$ if there is a backward-forward simulation from (s,c) to (t,d). Obviously,

$$(s,c) \sqsubseteq_{\mathbf{BF}} (t,d) \qquad \Longleftrightarrow \qquad \exists (r,b).\quad \big(\, (s,c) \sqsubseteq_{\mathbf{B}} (r,b) \;\wedge\; (r,b) \sqsubseteq_{\mathbf{F}} (t,d) \,\big) \enspace .$$

Remark 4.7 (Forward-backward simulations) It is straightforward to define the notion of *forward-backward simulations* and the relation $\sqsubseteq_{\mathbf{FB}}$, as a suitable dual of Definition 4.6. This is done in [13] for a restricted class of non-deterministic systems. In the same paper $\sqsubseteq_{\mathbf{BF}}$ and $\sqsubseteq_{\mathbf{FB}}$ are shown to coincide.

However we have not yet found the coincidence of $\sqsubseteq_{\mathbf{BF}}$ and $\sqsubseteq_{\mathbf{FB}}$ in general: in the light of Theorem 6.2, it seems that $\sqsubseteq_{\mathbf{BF}}$ is the more fundamental notion. The coincidence for non-deterministic systems in [13] may be because $\mathcal{K}\ell(\mathcal{P})$ is self-dual, i.e. $\mathcal{K}\ell(\mathcal{P}) \cong \mathcal{K}\ell(\mathcal{P})^{\mathrm{op}}$. Details are yet to be elaborated.

5 Finite trace semantics via coinduction

In this paper we take (finite) traces as our semantics for systems. It is with respect to trace semantics that soundness and completeness of forward/backward simulations are shown. This section establishes the basics of trace semantics for systems by revisiting our previous work [5]. The main points are:

- a final coalgebra in the Kleisli category $\mathcal{K}\ell(T)$ is (interestingly) induced by an initial algebra in **Sets**;
- the principle of coinduction, when employed in $\mathcal{K}\ell(T)$, yields finite trace semantics for branching systems.

We also cite a fact from [2] about an order-theoretic property of a final coalgebra. Again in this section a monad T is $\mathcal{P}$ or $\mathcal{D}$ and F is a shapely functor.

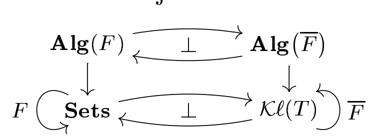
The following result identifies a final coalgebra in the Kleisli category.

Theorem 5.1 (Main theorem of [5]) *Let $\alpha : FA \xrightarrow{\cong} A$ be an initial F-algebra in* **Sets**.

1. *An initial $\overline{F}$-algebra in $\mathcal{K}\ell(T)$ is induced by α as $\eta_A \circ \alpha : FA \xrightarrow{\cong} A$ in $\mathcal{K}\ell(T)$.*
2. *In $\mathcal{K}\ell(T)$, an initial $\overline{F}$-algebra and a final $\overline{F}$-coalgebra coincide. The latter is given as follows. We shall denote this coalgebraic structure map by ζ.*

$$\zeta = (\eta_A \circ \alpha)^{-1} = \eta_{FA} \circ \alpha^{-1} :\; A \xrightarrow{\cong} FA \qquad \text{in } \mathcal{K}\ell(T) \enspace .$$

Proof. The first point is standard [16]. Due to the distributive law the Kleisli adjunction on the bottom is lifted to the top one, which preserves initial objects.



The second point of initial algebra/final coalgebra coincidence essentially follows from the classic work [20] of limit-colimit coincidence. □

As a corollary we obtain the final coalgebra semantics for an $\overline{F}$-coalgebra. Recall that such a coalgebra is a dynamics of a (T, F)-system.

Corollary 5.2 (Finite trace semantics for coalgebras, [5]) *Given an $\overline{F}$-coalgebra $X \stackrel{c}{\to} FX$ in $\mathcal{K}\ell(T)$, there exists a unique morphism tr_c which makes the following diagram commute. Here $\alpha : FA \stackrel{\cong}{\Rightarrow} A$ is an initial F-algebra in* **Sets**.

$$\begin{array}{ccc} FX & \xrightarrow{\overline{F}(\mathsf{tr}_c)} & FA \\ c\uparrow & & \cong\uparrow\zeta \text{ (final)} \\ X & \xrightarrow[\mathsf{tr}_c]{} & A \end{array} \tag{6}$$

□

The induced map

$$\mathsf{tr}_c : X \to A \quad \text{in } \mathcal{K}\ell(T), \qquad \text{that is,} \qquad \mathsf{tr}_c : X \to TA \quad \text{in } \mathbf{Sets},$$

in fact becomes what is usually called the finite trace map: it assigns to each state its "trace" in a suitable sense. The following examples show that the commutation of Diagram (6) actually amounts to standard and natural recursive definition of finite trace maps.

Example 5.3 (Non-deterministic automata) In the setting of Example 3.2, an initial F-algebra in **Sets** is carried by finite lists, or words, over Σ.

$$1 + \Sigma \times \Sigma^* \xrightarrow[\cong]{[\mathsf{nil}, \mathsf{cons}]} \Sigma^*$$

Now Diagram (6) commutes if and only if the function $\mathsf{tr}_c : X \to \mathcal{P}(\Sigma^*)$ satisfies the following conditions. For each $a \in \Sigma$ and $\sigma \in \Sigma^*$.

$$\begin{array}{ccc} \langle\rangle \in \mathsf{tr}_c(x) & \iff & x \to \checkmark\ , \\ a \cdot \sigma \in \mathsf{tr}_c(x) & \iff & \exists x' \in X.\ \ \big(\, x \stackrel{a}{\to} x' \ \wedge\ \sigma \in \mathsf{tr}_c(x')\, \big)\ . \end{array}$$

This is the standard recursive (or corecursive, if you like) definition of the *accepted languages* of non-deterministic automata. The language $\mathsf{tr}_c(x) \subseteq \Sigma^*$ is the set of all the linear-time behavior of x which eventually terminates within a finite number of steps (hence the name *finite* trace).

Example 5.4 (Probabilistic automata) Let us look at the example of a probabilistic automaton in Example 3.3. What is the "trace" of the state x of this system? A natural answer, as suggested in [9], is the probability subdistribution over lists on Σ:

$$\langle\rangle \mapsto \frac{1}{3}, \quad a \mapsto \frac{1}{3} \cdot \frac{1}{2}, \quad aa \mapsto \frac{1}{3} \cdot \frac{1}{2} \cdot \frac{1}{2}, \quad \cdots, \quad a^n \mapsto \frac{1}{3} \cdot \left(\frac{1}{2}\right)^{n-1} \cdot \frac{1}{2}, \quad \cdots \tag{7}$$

This is explained as follows. For the state x to output the list aa, it has to take the path of transitions: $x \stackrel{a}{\to} y \stackrel{a}{\to} y \to \checkmark$. This path occurs with the probability $\frac{1}{3} \cdot \frac{1}{2} \cdot \frac{1}{2}$.

This notion of "probabilistic trace" is again obtained via coinduction in the Kleisli category. Let us instantiate Diagram (6) with T and F in Example 3.3. The commutativity of the diagram amounts to the following (co)recursive definition of a function $\mathsf{tr}_c : X \to \mathcal{D}(\Sigma^*)$:

$$\mathsf{tr}_c(x) = \left[\begin{array}{ccc} \langle\rangle & \mapsto & c(x)(\checkmark) \\ a \cdot \sigma & \mapsto & \sum_{y \in X} c(x)(a, y) \cdot \mathsf{tr}_c(y)(\sigma) \end{array} \right] .$$

Here the probability $c(x)(a, y) \cdot \mathsf{tr}_c(y)(\sigma)$ is for the event that x makes an a-move to y and then y yields the list σ as its trace. Taking the sum over all the possible successors y of x, we get a natural recursive definition of the probability with which x yields $a \cdot \sigma$ as its trace.

As an additional remark we point out that the subdistribution (7) sums up only to $2/3$. The remaining $1/3$ is for the path $x \stackrel{a}{\to} z \stackrel{a}{\to} z \stackrel{a}{\to} \cdots$: the probability for a^ω, or *livelock*. This entry $a^\omega \mapsto 1/3$ is absent in $\mathsf{tr}_c(x)$ because $\mathsf{tr}_c : X \to \mathcal{D}(\Sigma^*)$ is the *finite* trace. This also demonstrates why we use the subdistribution monad $\mathcal{D}$ instead of the distribution monad $\mathcal{D}_{=1}$: although the system can be described using $\mathcal{D}_{=1}$, we do not get tr_c of the type $X \to \mathcal{D}_{=1}(\Sigma^*)$.

Example 5.5 (Context-free grammar, [4]) Let us take T and F as in Example 3.6. Via coinduction in $\mathcal{K}\ell(T)$ we get a trace map tr_c: this assigns to each non-terminal x the set of finite-depth parse trees generated by the context-free grammar c starting from x.

From a different point of view, the previous examples are seen as proofs that standard recursive definitions uniquely determine trace maps, due to the finality result in Corollary 5.2.

The trace map tr_c, being a morphism of coalgebras, automatically becomes a lax morphism of coalgebras. It is in fact characterized as the biggest lax morphism.

Proposition 5.6 (Trace map as the biggest lax morphism) *In the situation of Diagram (6), the trace map* tr_c *is the biggest one among the lax coalgebra morphisms from* c *to the final* ζ*. That is, in* $\mathcal{K}\ell(T)$*:*

$$\begin{array}{ccc} FX & \xrightarrow{\overline{F}f} & FA \\ c\uparrow & \sqsupseteq & \cong\uparrow\zeta \\ X & \xrightarrow[f]{} & A \end{array} \quad\Longrightarrow\quad X \underset{\mathsf{tr}_c}{\overset{f}{\rightrightarrows}} A \quad (f \sqsubseteq \mathsf{tr}_c) \enspace .$$

Dually, the trace map tr_c *is the smallest one among the oplax coalgebra morphisms from* c *to the final* ζ*.*

Proof. Although the proposition follows from a general result [2, Proposition 6.7], in this specific setting of the Kleisli category we can give another proof. It does not depend on the local continuity of $\overline{F}$ but only on the local monotonicity. This alternative proof is in Appendix A.1. □

So far the trace map induced by coinduction gives the semantics for a single state of a coalgebra. This is extended to the semantics of a (T, F)-system—a coalgebra with explicit start states—in the obvious way.

Definition 5.7 (Finite trace semantics of (T, F)-systems) Given a (T, F)-system $1 \xrightarrow{s} X \xrightarrow{c} FX$ in $\mathcal{K}\ell(T)$, its *finite trace* (or just *trace*) $\mathsf{tr}_{(s,c)}$ is the following composite in $\mathcal{K}\ell(T)$.

$$\begin{array}{ccc} FX & \xrightarrow{\overline{F}(\mathsf{tr}_c)} & FA \\ c\uparrow & & \cong\uparrow\zeta \\ X & \xrightarrow{\mathsf{tr}_c} & A \\ s\uparrow & \nearrow_{\mathsf{tr}_{(s,c)}} & \\ 1 & & \end{array}$$

Notice that Diagram (6) of coinduction is appearing in this diagram.

Proposition 5.8 (Morphisms of systems yield trace equivalence) *Assume we have a morphism* f *of* (T, F)*-systems from* $1 \stackrel{s}{\to} X \stackrel{c}{\to} FX$ *to* $1 \stackrel{t}{\to} Y \stackrel{d}{\to} FY$*. Then* $\mathsf{tr}_{(s,c)} = \mathsf{tr}_{(t,d)}$.

Proof. By Definition 3.7 of morphism of systems, f is in particular a morphism of coalgebras. By finality we have $\mathsf{tr}_c = \mathsf{tr}_d \circ f$. Hence

$$\mathsf{tr}_{(s,c)} = \mathsf{tr}_c \circ s = \mathsf{tr}_d \circ f \circ s = \mathsf{tr}_d \circ t = \mathsf{tr}_{(t,d)} \enspace .$$ □

6 Soundness and completeness theorems

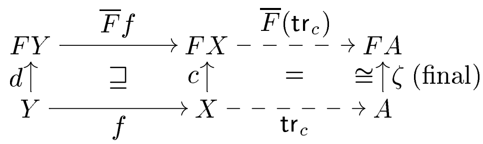
In the last two sections we have built up the notions of (and some results on) forward/backward simulations and trace semantics, with a high level of genericity and abstraction. In this section we relate those materials—with the same genericity and abstraction—by proving soundness of $\sqsubseteq_{\mathbf{F}}, \sqsubseteq_{\mathbf{B}}, \sqsubseteq_{\mathbf{BF}}$ and completeness of $\sqsubseteq_{\mathbf{BF}}$ with respect to trace inclusion. This is the main technical result of this paper.

In the rest of this section we assume $1 \stackrel{s}{\to} X \stackrel{c}{\to} FX$ and $1 \stackrel{t}{\to} Y \stackrel{d}{\to} FY$ to be (T,F)-systems, where $T = \mathcal{P}$ or $\mathcal{D}$ and F is shapely.

Theorem 6.1 (Soundness of $\sqsubseteq_{\mathbf{F}}, \sqsubseteq_{\mathbf{B}}, \sqsubseteq_{\mathbf{BF}}$)

$$\begin{array}{lrcl} \textit{1.} & (s,c) \sqsubseteq_{\mathbf{F}} (t,d) & \Longrightarrow & \mathsf{tr}_{(s,c)} \sqsubseteq \mathsf{tr}_{(t,d)} \ , \\ \textit{2.} & (s,c) \sqsubseteq_{\mathbf{B}} (t,d) & \Longrightarrow & \mathsf{tr}_{(s,c)} \sqsubseteq \mathsf{tr}_{(t,d)} \ , \\ \textit{3.} & (s,c) \sqsubseteq_{\mathbf{BF}} (t,d) & \Longrightarrow & \mathsf{tr}_{(s,c)} \sqsubseteq \mathsf{tr}_{(t,d)} \ . \end{array}$$

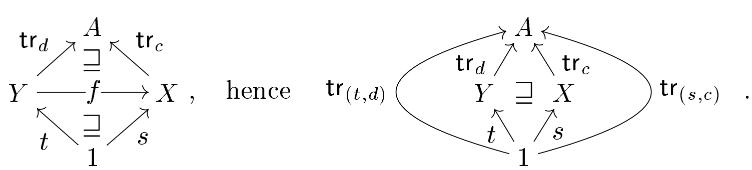
Proof. 1. By definition of $\sqsubseteq_{\mathbf{F}}$ we have a forward simulation $f : Y \to X$. In particular we have in $\mathcal{K}\ell(T)$,



where the coinduction diagram appears on the right. This shows that the arrow $\mathsf{tr}_c \circ f$ is a lax coalgebra morphism from d to the final coalgebra. Indeed,

$$\begin{aligned} \zeta \circ \mathsf{tr}_c \circ f &= \overline{F}(\mathsf{tr}_c) \circ c \circ f && (\mathsf{tr}_c \text{ is a morphism of coalgebras}) \\ &\sqsubseteq \overline{F}(\mathsf{tr}_c) \circ \overline{F}f \circ d && (\text{Composition is continuous}) \\ &= \overline{F}(\mathsf{tr}_c \circ f) \circ d \ . \end{aligned}$$

Since the trace map is the biggest lax coalgebra morphism (Proposition 5.6), we have $\mathsf{tr}_c \circ f \sqsubseteq \mathsf{tr}_d$. This inequality is combined with f's condition on start states.



This proves 1. Similar arguments prove 2.

3. The relation $\sqsubseteq_{\mathbf{BF}}$ is a relational composition $\sqsubseteq_{\mathbf{F}} \circ \sqsubseteq_{\mathbf{B}}$. We use 1. and 2. of the theorem and transitivity of the order $\sqsubseteq$ between arrows $1 \rightrightarrows A$. □

Completeness—the converse of the soundness result above—does not hold for $\sqsubseteq_{\mathbf{F}}, \sqsubseteq_{\mathbf{B}}$ but does hold for the weaker notion of $\sqsubseteq_{\mathbf{BF}}$. For a restricted class of non-deterministic systems the completeness result is shown in [12, 13].

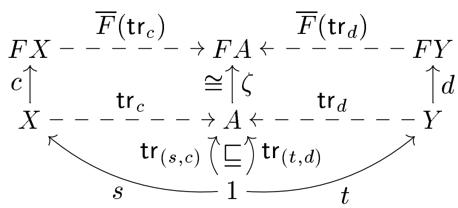
Theorem 6.2 (Completeness of $\sqsubseteq_{\mathbf{BF}}$)

$$\mathsf{tr}_{(s,c)} \sqsubseteq \mathsf{tr}_{(t,d)} \quad \Longrightarrow \quad (s,c) \sqsubseteq_{\mathbf{BF}} (t,d) \ .$$

Proof. From a (T, F)-system (s, c), we construct its "canonical system" as

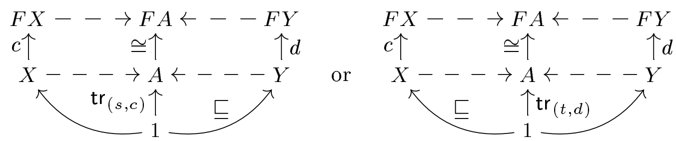
$$1 \xrightarrow{\mathsf{tr}_{(s,c)}} A \xrightarrow[\cong]{\zeta} FA \qquad \text{in } \mathcal{K}\ell(T)\ .$$

That is, the dynamics is the final $\overline{F}$-coalgebra and the start states map is the trace of the system. It is obvious by definition that the map tr_c is a morphism of systems from (s, c) to this canonical system (the left side of Diagram (8) below). We apply the same construction to (t, d) yielding the right side of the diagram. Then the assumption $\mathsf{tr}_{(s,c)} \sqsubseteq \mathsf{tr}_{(t,d)}$ fits in the lower middle of the diagram.



(8)

From this we have two diagrams of backward-forward simulations—like Diagram (5) in Definition 4.6—depending on our choice of the intermediate system.



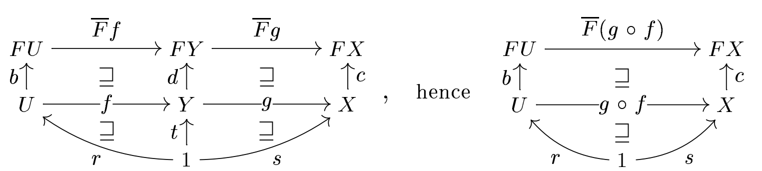
Either diagram shows $(s, c) \sqsubseteq_{\mathbf{BF}} (t, d)$. □

Next we shall prove that three kinds of simulation relations $\sqsubseteq_{\mathbf{F}}, \sqsubseteq_{\mathbf{B}}$ and $\sqsubseteq_{\mathbf{BF}}$ are indeed preorders. The proof has been postponed until now because: for transitivity of $\sqsubseteq_{\mathbf{BF}}$ we need the soundness and completeness results for $\sqsubseteq_{\mathbf{BF}}$.

For $\sqsubseteq_{\mathbf{F}}$ and $\sqsubseteq_{\mathbf{B}}$ the proof is straightforward.

Proposition 6.3 ($\sqsubseteq_{\mathbf{F}}, \sqsubseteq_{\mathbf{B}}$ are preorders) *The forward/backward simulation relations $\sqsubseteq_{\mathbf{F}}$ and $\sqsubseteq_{\mathbf{B}}$ are preorders. That is, they are reflexive and transitive.*

Proof. Reflexitivity is obvious: take the identity arrow in the Kleisli category as a forward (or backward) simulation. Assume $(s, c) \sqsubseteq_{\mathbf{F}} (t, d) \sqsubseteq_{\mathbf{F}} (r, b)$. There exist forward simulations f and g such that



This shows $(s, c) \sqsubseteq_{\mathbf{F}} (r, b)$. Transitivity of $\sqsubseteq_{\mathbf{B}}$ is proved in a similar way. □

Proposition 6.4 ($\sqsubseteq_{\mathbf{BF}}$ is a preorder) *The backward-forward simulation relation $\sqsubseteq_{\mathbf{BF}}$ is a preorder.*

Proof. Reflexitivity is trivial by taking the system itself as an intermediate one. Assume $(s, c) \sqsubseteq_{\mathbf{BF}} (t, d) \sqsubseteq_{\mathbf{BF}} (r, b)$. By soundness of $\sqsubseteq_{\mathbf{BF}}$ in Theorem 6.1 and transitivity of $\sqsubseteq$, we have $\mathsf{tr}_{(s,c)} \sqsubseteq \mathsf{tr}_{(r,b)}$. This in turn yields $(s, c) \sqsubseteq_{\mathbf{BF}} (r, b)$ by completeness of $\sqsubseteq_{\mathbf{BF}}$ in Theorem 6.2. □

7 Eliminating internal actions

In this section we give a coalgebraic account on internal actions. We first motivate our investigation by illustrating how internal actions arise in exercise of formal verification. Then we introduce a generic scheme of eliminating internal actions via coalgebraic trace semantics. Equivalence of two different "trace semantics" for systems with internal actions are proved as the main technical result.

7.1 Internal actions in formal verification

A typical scenario of verification via simulations is as follows. We have two systems at hand: one is a simpler *specification* system $\mathcal{S}$ which is already known to satisfy the desirable properties; the other is a more complex *implementation* system $\mathcal{I}$ to be verified. Existence of a simulation from $\mathcal{I}$ to $\mathcal{S}$ implies trace inclusion by soundness Theorem 6.1. From this we conclude that: (linear-time) safety property which holds in $\mathcal{S}$ also holds $\mathcal{I}$.

In this course of verification it often happens that $\mathcal{I}$ can make a bigger variety of actions than $\mathcal{S}$ can. An example: $\mathcal{S}$ is a $(\mathcal{P}, 1 + \Sigma \times \_)$-system and $\mathcal{I}$ is a $(\mathcal{P}, 1 + \Sigma' \times \_)$-system, with $\Sigma \subseteq \Sigma'$, say

$$\Sigma = \{\mathsf{getCoin}, \mathsf{brewCoffee}\} \ ,$$
$$\Sigma' = \{\mathsf{getCoin}, \mathsf{brewCoffee}, \mathsf{boilWater}, \mathsf{grindBeans}\} \ .$$

One can imagine a coffee machine here.

In such cases we employ *internal actions* to apply the verification method via simulations. We first replace the actions which are allowed only in $\mathcal{I}$ (boilWater and grindBeans in the above example) by the symbol τ denoting internal actions. This transforms

$$\mathcal{I} = \big(1 \xrightarrow{s} X \xrightarrow{c} 1 + \Sigma' \times X \big) \quad \text{in } \mathcal{K}\ell(\mathcal{P}) \qquad \text{into} \qquad \mathcal{I}' = \big(1 \xrightarrow{s} X \xrightarrow{c'} 1 + \Sigma \times X + \{\tau\} \times X \big) \ ,$$

$$\text{where} \qquad (\tau, x') \in c'(x) \quad \text{if and only if} \quad \begin{cases} (\mathsf{boilWater}, x') \in c(x), & \text{or} \\ (\mathsf{grindBeans}, x') \in c(x) \end{cases}$$

The resulting $\mathcal{I}'$ is a $(\mathcal{P}, 1 + \Sigma \times \_ + \_)$-system: such a system shall be called a $(\mathcal{P}, 1 + \Sigma \times \_)$-system *with internal actions*. Now we would like to compare a system $\mathcal{I}'$ with internal actions τ to another system $\mathcal{S}$. One way to do so is to eliminate τ's of $\mathcal{I}'$ and obtain a system $\overline{\mathcal{I}'}$ (called the *closure*) without τ's. The closure $\mathcal{I}'$ shall be such that: the closure can make a certain transition if and only if the original system $\mathcal{I}'$ can make it after finite number of τ's. Formally,

$$\overline{\mathcal{I}'} = \big(1 \xrightarrow{s} X \xrightarrow{\overline{c'}} 1 + \Sigma \times X \big) \quad \text{in } \mathcal{K}\ell(\mathcal{P})$$
$$\text{where} \quad \begin{cases} x \xrightarrow{a} x' \text{ in } \overline{c'} & \Longleftrightarrow \quad \exists x_0, x_1, \ldots, x_n \in X. \quad \big(x = x_0 \xrightarrow{\tau} x_1 \xrightarrow{\tau} \cdots \xrightarrow{\tau} x_n \xrightarrow{a} x' \text{ in } c' \big) \\ x \to \checkmark \text{ in } \overline{c'} & \Longleftrightarrow \quad \exists x_0, x_1, \ldots, x_n \in X. \quad \big(x = x_0 \xrightarrow{\tau} x_1 \xrightarrow{\tau} \cdots \xrightarrow{\tau} x_n \to \checkmark \text{ in } c' \big) \end{cases}$$

Now let us consider the "trace semantics" of the system $\mathcal{I}'$ with internal actions. There are two natural ways to do so, as illustrated in (1).

- We first eliminate τ's from $\mathcal{I}'$, obtaining its closure $\overline{\mathcal{I}'}$ as a $(\mathcal{P}, 1+\Sigma\times\_)$-system. Then we apply the trace semantics (Corollary 5.2) and obtain the trace map $\mathsf{tr}_{\overline{\mathcal{I}'}} : X \to \mathcal{P}(\Sigma^*)$ in **Sets**. For example, $\mathsf{tr}_{\overline{\mathcal{I}'}}(x) = \{\langle \mathsf{getCoin}, \mathsf{brewCoffee} \rangle\}$.
- We first apply the trace semantics to $\mathcal{I}'$ itself. The trace map is now of the type $\mathsf{tr}_{\mathcal{I}'} : X \to \big(\mathcal{P}(\Sigma + \{\tau\})^*\big)$ in **Sets**: the codomain is an initial $(1 + \Sigma \times \_ + \_)$-algebra. For example, $\mathsf{tr}_{\mathcal{I}'}(x) = \{\langle \mathsf{getCoin}, \tau, \tau, \mathsf{brewCoffee} \rangle\}$.

There is an inductively defined map $(\Sigma + \{\tau\})^* \stackrel{j}{\to} \Sigma^*$ in **Sets** removing τ's: for example $\langle \mathsf{getCoin}, \tau, \tau, \mathsf{brewCoffee} \rangle \mapsto \langle \mathsf{getCoin}, \mathsf{brewCoffee} \rangle$. The trace semantics is given as a composite

$$X \xrightarrow{\mathsf{tr}_{\mathcal{I}'}} (\Sigma + \{\tau\})^* \xrightarrow{Jj} \Sigma^* \qquad \text{in } \mathcal{K}\ell(\mathcal{P})$$

where $J : \mathbf{Sets} \to \mathcal{K}\ell(\mathcal{P})$ is the canonical left-adjoint.

7.2 Internal actions, coalgebraically

Definition 7.1 (Systems with internal actions) A (T, F)-*system with internal actions* is a pair of arrows

$$1 \xrightarrow{s} X \xrightarrow{c} \overline{F}X + X \qquad \text{in the Kleisli category } \mathcal{K}\ell(T).$$

That is, a pair $\big(1 \stackrel{s}{\to} TX,\ X \stackrel{c}{\to} T(FX + X)\big)$ in **Sets**.

The additional summand "$+X$" in the codomain $\overline{F}X + X$ of the dynamics models internal actions: this "$+X$" is understood as "$+\{\tau\} \times X$".

Elimination of internal actions is done via trace semantics. Given a (T, F)-system with internal actions $1 \stackrel{s}{\to} X \stackrel{c}{\to} \overline{F}X + X$, its dynamics can be thought of as a coalgebra for a (shapely) functor $\overline{F}X + \_$.[4] This functor has the following initial algebra in **Sets**, much like $\mathbb{N}$ is an initial $(1 + \_)$-algebra.

$$\begin{array}{rccc} & \overline{F}X + \mathbb{N} \times \overline{F}X & \kappa_1(t) & \kappa_2(n,t) \\ {\scriptstyle [\,\langle 0, \mathrm{id}\rangle,\ \mathsf{s} \times \mathrm{id}\,]} & \big\downarrow & \big\downarrow & \big\downarrow \\ & \mathbb{N} \times \overline{F}X & (0,t) & (n+1,t) \end{array}$$

By Corollary 5.2, we obtain a map $\widetilde{c} : X \to \mathbb{N} \times \overline{F}X$ in $\mathcal{K}\ell(T)$ via coinduction.

$$\begin{array}{ccc} \overline{F}X + X & \dashrightarrow & \overline{F}X + (\mathbb{N} \times \overline{F}X) \\ {\scriptstyle c}\big\uparrow & & \big\uparrow{\scriptstyle \cong} \\ X & \underset{\widetilde{c}}{\dashrightarrow} & \mathbb{N} \times \overline{F}X \end{array} \tag{9}$$

We define the *closure*

$$X \xrightarrow{\overline{c}} \overline{F}X \qquad \text{in } \mathcal{K}\ell(T)$$

of the dynamics $c : X \to \overline{F}X + X$ as the following composite.

$$\begin{array}{ccc} X & \overset{\widetilde{c}}{\dashrightarrow} & \mathbb{N} \times \overline{F}X \\ & {\scriptstyle \overline{c}}\searrow & \big\downarrow{\scriptstyle J\pi_2} \\ & & \overline{F}X \end{array}$$

Here J is the standard left-adjoint $\mathbf{Sets} \to \mathcal{K}\ell(T)$. To summarize:

Definition 7.2 (Closure, with internal actions eliminated) Given a (T, F)-system with internal actions $1 \stackrel{s}{\to} X \stackrel{c}{\to} \overline{F}X + X$, its *closure* is the (T, F)-system (without internal actions)

$$1 \xrightarrow{s} X \xrightarrow{\overline{c}} \overline{F}X$$

with $\overline{c}$ obtained via coalgebraic trace semantics (i.e. coinduction in $\mathcal{K}\ell(T)$).

---

[4] Note the difference between two functors $\overline{F}X + \_$ and $\overline{F}\_ + \_$.

Example 7.3 (LTS's with explicit termination) For $T = \mathcal{P}$ and $F = 1 + \Sigma \times \_$, let us illustrate the intermediate map $\widetilde{c} : X \to \mathcal{P}\big(\mathbb{N} \times (1 + \Sigma \times X)\big)$ in **Sets** in the above construction. Its diagramatic definition (9) of $\widetilde{c}$ is equivalent to the following equations. For $t \in FX$,

$$\begin{aligned} (0,t) \in \widetilde{c}(x) \quad &\iff \quad \kappa_1(t) \in c(x) \enspace , \\ (n+1,t) \in \widetilde{c}(x) \quad &\iff \quad \exists x' \in X. \, \big(\, \kappa_2(x') \in c(x) \;\wedge\; (n,t) \in \widetilde{c}(x') \,\big) \enspace . \end{aligned}$$

This implies:

$$\begin{aligned} (n, \checkmark) \in \widetilde{c}(x) \quad &\iff \quad \overbrace{x \stackrel{\tau}{\to} \cdots \stackrel{\tau}{\to} x_n}^{n \text{ times}} \to \checkmark \enspace , \\ (n, (a, x')) \in \widetilde{c}(x) \quad &\iff \quad \overbrace{x \stackrel{\tau}{\to} \cdots \stackrel{\tau}{\to} x_n}^{n \text{ times}} \stackrel{a}{\to} x' \enspace . \end{aligned}$$

Hence $\widetilde{c}$ carries information about after how many internal actions the non-internal action is made. By removing this additional information we obtain the closure $\overline{c}$.

In the rest of this section we shall prove coincidence of two "trace semantics" for systems with internal actions.

To get the first "trace semantics" for a given system $1 \stackrel{s}{\to} X \stackrel{c}{\to} \overline{F}X + X$ with internal actions, we first take its closure $1 \stackrel{s}{\to} X \stackrel{\overline{c}}{\to} \overline{F}X$ as defined in Definition 7.2. The resulting (T, F)-system (without internal actions) yields the trace map $\mathsf{tr}_{(s,\overline{c})} : 1 \to A$ in $\mathcal{K}\ell(T)$ via coinduction, as in Corollary 5.2. Here $\alpha : FA \stackrel{\cong}{\to} A$ is an initial F-algebra in **Sets**.

For the other "trace semantics" we first take the trace map in which internal actions (denoted by τ) appear explicit. Specifically, a (T, F)-system $1 \stackrel{s}{\to} X \stackrel{c}{\to} \overline{F}X + X$ with internal actions is thought of as a $(T, F\_ + \_)$-system without internal actions. Hence it yields the trace map $\mathsf{tr}_{(s,c)} : 1 \to B$ in $\mathcal{K}\ell(T)$, with the codomain B carrying an initial $(F\_ + \_)$-algebra in **Sets**:

$$\begin{array}{c} FB + B \\ \cong \big\downarrow \beta \\ B \end{array}$$

We shall remove internal actions from this trace $\mathsf{tr}_{(s,c)}$. It is done categorically, via induction. Recalling $\alpha : FA \stackrel{\cong}{\to} A$ is an initial F-algebra, there are canonical maps $i : A \to B$ and $j : B \to A$ in **Sets** defined via initiality.

$$\begin{array}{ccc} FA & \dashrightarrow & FB \\ & & \big\downarrow \kappa_1 \\ \cong \big\downarrow \alpha & & FB + B \\ & & \beta \big\downarrow \cong \\ A & \underset{i}{\dashrightarrow} & B \end{array} \qquad\qquad \begin{array}{ccc} FB + B & \dashrightarrow & FA + A \\ \cong \big\downarrow \beta & & \big\downarrow [\alpha, \mathrm{id}] \\ B & \underset{j}{\dashrightarrow} & A \end{array} \tag{10}$$

The map j is what we want: it removes τ's from elements of B. Finally, the second "trace semantics" is now defined as the composition

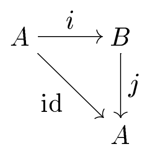
$$1 \xrightarrow{\mathsf{tr}_{(s,c)}} TB \xrightarrow{Tj} TA \quad \text{in } \mathbf{Sets}, \text{ i.e.} \qquad 1 \xrightarrow{\mathsf{tr}_{(s,c)}} B \xrightarrow{Jj} A \quad \text{in } \mathcal{K}\ell(T).$$

Example 7.4 Let us take $T = \mathcal{P}$ and $F = 1 + \Sigma \times \_$ for illustration. Then $A = \Sigma^*$ and $B = (\Sigma + \{\tau\})^*$. The map i is an embedding, while $j : B \to A$ removes appearances of τ from words over $\Sigma + \{\tau\}$. In fact, the diagramatic definition of j is equivalent to the following equational definition, where $\sigma \in (\Sigma + \{\tau\})^*$.

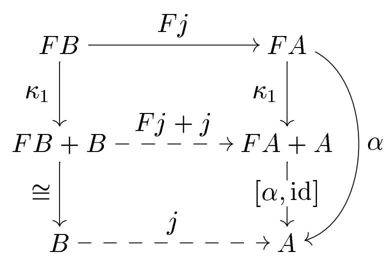
$$
\begin{aligned}
j(\langle\rangle) &= \langle\rangle \\
j(a \cdot \sigma) &= a \cdot j(\sigma) \qquad \text{for } a \in \Sigma \\
j(\tau \cdot \sigma) &= j(\sigma)
\end{aligned}
$$

For example j carries the list $a_1 \tau a_2 \tau$ to $a_1 a_2$.

Lemma 7.5 *The map $i : A \to B$ is a split epi with j a left inverse.*



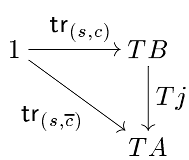
Proof. First we observe that the following diagram in **Sets** commutes.



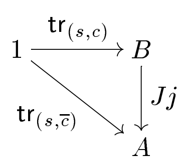
Hence j is a morphism of F-coalgebras as well. The claim follows from the initiality of $\alpha : FA \xrightarrow{\cong} A$. □

Now we can formally state the main technical result of this section.

Theorem 7.6 *The following diagrams commute, where $\mathsf{tr}_{(s,c)}, \mathsf{tr}_{(s,\overline{c})}$ and j are defined as above.*



in **Sets**, *i.e.*



in $\mathcal{K}\ell(T)$.

The trace $\mathsf{tr}_{(s,\overline{c})}$ is what we call the first "trace semantics", while the composite $Jj \circ \mathsf{tr}_{(s,c)}$ is the second kind.

Proof. The proof involves non-trivial, lengthy diagram chasing. It is presented in Appendix A.3. □

8 Conclusions and future work

We have developed a generic theory of branching state-based systems in terms of coalgebras in Kleisli categories. Notions such as forward/backward simulations and traces are defined and related via soundness and completeness results. Several illustrating examples suggest practical implications of this theory.

There are a number of issues on branching systems that remain to be elaborated in our generic framework. To name a few: composition of systems, compositionality of semantics, modal logic, preservation of logical formulas and infinite traces.

As mentioned in Remark 3.4, systems with both non-deterministic and probabilistic branching do not fit in our general framework. There are many semantical questions (see e.g. [1]) around this combination of different branching: hopefully categorical approaches will contribute to clarify the picture.

More examples of types of systems to which our framework applies are to be found. For example, the author is interested in a probabilistic version of anonymous simulations [10] .

IOA Toolset [3] is a formal verification tool in which systems are described as I/O automata and analyzed using simulations. Now that its base theory is made generic, one might as well work on a generic version of the toolset itself.

Acknowledgements

Thanks are due to Chris Heunen, Yoshinobu Kawabe, Koki Nishizawa, Frits Vaandrager and Hiroshi Watanabe for helpful discussions.

References

[1] L. Cheung. *Reconciling Nondeterministic and Probabilistic Choices.* PhD thesis, Radboud Univ. Nijmegen, 2006.

[2] M. Fiore. A coinduction principle for recursive data types based on bisimulation. *Inf. & Comp.*, 127(2):186–198, 1996.

[3] S. Garland, N. Lynch, and M. Vaziri. *IOA: a language for specifying, programming, and validating distributed systems.* MIT Laboratory for Computer Science, 1997.

[4] I. Hasuo and B. Jacobs. Context-free languages via coalgebraic trace semantics. In *Algebra and Coalgebra in Computer Science (CALCO'05)*, volume 3629 of *Lect. Notes Comp. Sci.*, pages 213–231. Springer, Berlin, 2005.

[5] I. Hasuo, B. Jacobs, and A. Sokolova. Generic trace theory. In *Coalgebraic Methods in Computer Science (CMCS 2006)*, Elect. Notes in Theor. Comp. Sci. Elsevier, Amsterdam, 2006.

[6] I. Hasuo. Generic forward and backward simulations. In *International Conference on Concurrency Theory (CONCUR 2006)*, Lect. Notes Comp. Sci., 2006. To appear.

[7] J. Hughes and B. Jacobs. Simulations in coalgebra. *Theor. Comp. Sci.*, 327(1-2):71–108, 2004.

[8] B. Jacobs. Trace semantics for coalgebras. In J. Adámek and S. Milius, editors, *Coalgebraic Methods in Computer Science*, number 106 in Elect. Notes in Theor. Comp. Sci. Elsevier, Amsterdam, 2004.

[9] C. Jou and S. Smolka. Equivalences, congruences and complete axiomatizations for probabilistic processes. In *CONCUR'90*, volume 458 of *Lect. Notes Comp. Sci.*, pages 367–383. Springer-Verlag, 1990.

[10] Y. Kawabe, K. Mano, H. Sakurada, and Y. Tsukada. Backward simulations for anonymity. In *International Workshop on Issues in the Theory of Security (WITS '06)*, 2006.

[11] Y. Kinoshita and J. Power. Data refinement and algebraic structure. *Acta Informatica*, 36:693–719, 2000.

[12] N. Klarlund and F. Schneider. Verifying safety properties using infinite-state automata. Technical Report 89-1039, Department of Computer Science, Cornell University, Ithaca, New York, 1989.

[13] N. Lynch and F. Vaandrager. Forward and backward simulations. I. Untimed systems. *Inf. & Comp.*, 121(2):214–233, 1995.

[14] N. Lynch and M. Tuttle. An introduction to input/output automata. *CWI Quarterly*, 2(3):219–246, 1989.

[15] E. Moggi. Notions of computation and monads. *Inf. & Comp.*, 93(1):55–92, 1991.

[16] A. Pardo. Fusion of recursive programs with computational effects. *Theor. Comp. Sci.*, 260(1–2):165–207, 2001.

[17] J. Power and D. Turi. A coalgebraic foundation for linear time semantics. In *Category Theory and Computer Science*, number 29 in Elect. Notes in Theor. Comp. Sci. Elsevier, Amsterdam, 1999.

[18] J. Rutten. Universal coalgebra: a theory of systems. *Theor. Comp. Sci.*, 249:3–80, 2000.

[19] R. Segala and N. Lynch. Probabilistic simulations for probabilistic processes. *Nordic Journ. Comput.*, 2(2):250–273, 1995.

[20] M. Smyth and G. Plotkin. The category theoretic solution of recursive domain equations. *SIAM Journ. Comput.*, 11:761–783, 1982.

[21] A. Sokolova. *Coalgebraic Analysis of Probabilistic Systems.* PhD thesis, TU Eindhoven, 2005.

[22] R. van Glabbeek, S. Smolka, and B. Steffen. Reactive, generative, and stratified models of probabilistic processes. *Inf. & Comp.*, 121:59–80, 1995.

[23] D. Varacca and G. Winskel. Distributing probabililty over nondeterminism. *Math. Struct. in Comp. Sci.*, 2005. To appear.

[24] S.H. Wu, S.A. Smolka, and E.W. Stark. Composition and behaviors of probabilistic I/O automata. *Theor. Comp. Sci.*, 176(1–2):1–38, 1997.

A Appendix

A.1 Proof of Proposition 5.6

Our proof here heavily relies on the constructions in our previous paper [5], to which the reader is referred for details.

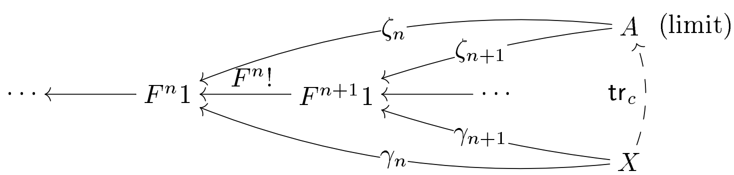
Let us consider the final sequence $1 \stackrel{!}{\leftarrow} F1 \stackrel{\overline{F}!}{\leftarrow} \cdots$ for $\overline{F}$ in $\mathcal{K}\ell(T)$. The following facts are standard.

- A final $\overline{F}$-coalgebra (which coincides with an F-initial algebra in **Sets**, Theorem 5.1) is an ω-limit of this final sequence. We denote this limit by $(\zeta_n : A \to F^n 1)_{n<\omega}$.

- An $\overline{F}$-coalgebra $c : X \to FX$ yields a cone $(\gamma_n : X \to F^n 1)_{n<\omega}$ over the final sequence, in an inductive manner.

- The unique coalgebra morphism tr_c from c to the final coalgebra is the unique mediating arrow $X \to A$ from the cone (γ_n) to the limit (ζ_n).

Hence we have the following situation in $\mathcal{K}\ell(T)$.



In the proof of the initial algebra/final coalgebra coincidence (Theorem 5.1.2), it is crucial that the limit (ζ_n) is also characterized by the order-theoretical notion of **O**-limit. In particular we can take the corresponding embedding $\zeta_n^E : F^n 1 \to A$ of each $\zeta_n : A \to F^n 1$, and moreover, we have $\mathrm{id}_A = \bigsqcup_{n<\omega} \zeta_n^E \circ \zeta_n$.

Now we can prove the first statement of the proposition.

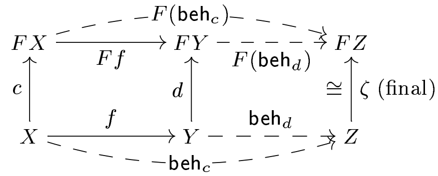
$$
\begin{aligned}
\mathsf{tr}_c &= \left(\textstyle\bigsqcup_{n<\omega} \zeta_n^E \circ \zeta_n\right) \circ \mathsf{tr}_c && (\mathrm{id}_A = \textstyle\bigsqcup_{n<\omega} \zeta_n^E \circ \zeta_n) \\
&= \left(\textstyle\bigsqcup_{n<\omega} \zeta_n^E \circ \zeta_n \circ \mathsf{tr}_c\right) && \text{(Composition is continuous)} \\
&= \left(\textstyle\bigsqcup_{n<\omega} \zeta_n^E \circ \gamma_n\right) && (\mathsf{tr}_c \text{ is a mediating arrow}) \\
&\sqsupseteq \left(\textstyle\bigsqcup_{n<\omega} \zeta_n^E \circ \zeta_n \circ f\right) && (\zeta_n \circ f \sqsubseteq \gamma_n \text{ for each } n, \dagger) \\
&= \left(\textstyle\bigsqcup_{n<\omega} \zeta_n^E \circ \zeta_n\right) \circ f \quad = f \ , && \text{(Composition is continuous)}
\end{aligned}
$$

where the inequality (†) is proved by induction, using that f is a lax morphism. In this proof the local monotonicity of $\overline{F}$ has been used in showing that the limit (ζ_n) is also an **O**-limit.

The dual statement is proved in a similar way.

A.2 Why explicit start states?

We shall explain the reason why we have explicit start states incorporated in the notion of (T, F)-systems. First let us look at the following diagram in **Sets**.



(11)

The maps beh_c and beh_d—since they are induced by coinduction in **Sets**—give the semantics respecting bisimilarity. In particular, by finality we have for each $x \in X$,

$$\mathsf{beh}_d\big(f(x)\big) = \mathsf{beh}_c(x) \enspace .$$

Hence a coalgebra morphism (the map f here) in **Sets** is a behavior-preserving map, respecting bisimilarity.

Now let us try the same trick in the Kleisli category $\mathcal{K}\ell(\mathcal{P})$ of the powerset monad. In Diagram (11), the maps beh_c and beh_d give trace semantics, and the map f is now a relation. By finality we have again $\mathsf{beh}_d \circ f = \mathsf{beh}_c$. However in $\mathcal{K}\ell(\mathcal{P})$ the computational meaning of this equality is unclear. It means: for each $x \in X$,

$$\textstyle\bigcup_{y \in f(x)} \mathsf{beh}_d(y) = \mathsf{beh}_c(x) \enspace .$$

That is, x has more behavior than any $y \in f(x)$, but at the same time any behavior of x is simulated by some $y \in f(x)$. We do not immediately see the significance of this notion.

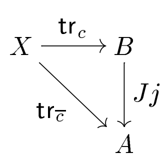
This clumsiness in comparing "one x vs. many y's" is immediately mended by considering explicit start states. In Proposition 5.8 the comparison is made between "many x's vs. many y's" instead. The conclusion $\mathsf{tr}_{(s,c)} = \mathsf{tr}_{(t,d)}$ of the proposition is interpreted as

$$\textstyle\bigcup_{x \in s(*)} \mathsf{beh}_c(x) = \bigcup_{y \in t(*)} \mathsf{beh}_d(y)$$

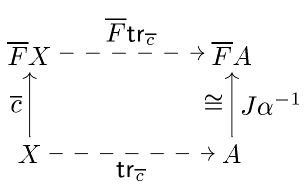
in the context of this remark.

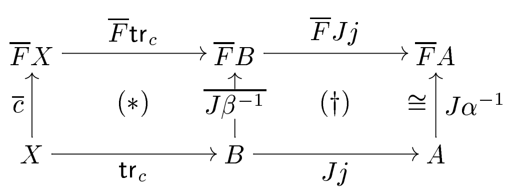
A.3 Proof of Theorem 7.6

By Definition 5.7, it suffices to show the commutativity of the following diagram in $\mathcal{K}\ell(T)$.



This follows from the following two commuting diagrams in $\mathcal{K}\ell(T)$ and the finality of $A \xrightarrow{\cong} \overline{F}A$.





The diagram on the left is the definition of $\mathsf{tr}_{\overline{c}}$. The commutativity of the one on the right is non-trivial and hence shown in later lemmas: $(*)$ in Lemma A.4 and $(\dagger)$ in Lemma A.5.

In the sequel we proceed step-by-step to finally prove the two lemmas.

Lemma A.1 *For a monad $T = \mathcal{P}$ or $\mathcal{D}$, the Kleisli category $\mathcal{K}\ell(T)$ is monoidal.*

Proof. Such a monad T on **Sets** is *commutative*: see [5] for details. A commutative monad is equipped with an additional operation called *double strength*

$$TY \times TW \xrightarrow{\mathsf{dst}_{Y,W}} T(Y \times W)$$

for each $Y, W \in \mathbf{Sets}$. This is exploited to give a monoidal structure of $\mathcal{K}\ell(T)$.

Specifically, a tensor product $\otimes$ is defined as follows. On objects, $X \otimes Y = X \times Y$. Given $f : X \to Y$ and $g : Z \to W$ in $\mathcal{K}\ell(T)$, we obtain $f \otimes g : X \otimes Z \to Y \otimes W$ as follows.

$$\frac{X \otimes Z \xrightarrow{f \otimes g} Y \otimes W \quad \text{in } \mathcal{K}\ell(T)}{X \times Z \xrightarrow[f \times g]{} TY \times TW \xrightarrow[\mathsf{dst}_{Y,W}]{} T(Y \times W) \quad \text{in } \mathbf{Sets}}$$

The unit object I is given by the terminal object 1. It is straightforward to check that the appropriate coherence conditions are satisfied. □

Lemma A.2 *For each set S, the following two functors $\mathcal{K}\ell(T) \rightrightarrows \mathcal{K}\ell(T)$ are identical.*

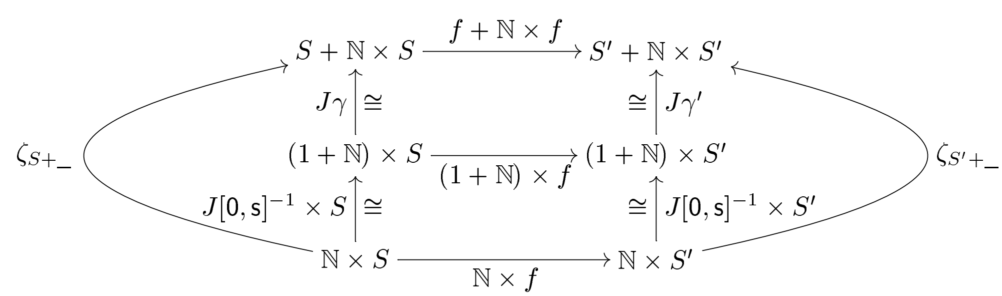
- *The functor $\overline{S \times \_}$. This is the shapely functor $S \times \_$ on* **Sets** *lifted to $\mathcal{K}\ell(T)$, as described in Section 2.4.*
- *The functor $S \otimes \_$. This is the bifunctor $\otimes : \mathcal{K}\ell(T) \times \mathcal{K}\ell(T) \to \mathcal{K}\ell(T)$ with the first argument fixed.*

Proof. Obviously the two functors coincide in their action on objects. For an arrow $f : X \to Y$ in $\mathcal{K}\ell(T)$, we have the following equalities in **Sets**.

$$\begin{aligned}
\overline{S \times _}(f) &= \lambda_Y \circ (S \times f) && \lambda : (S \times _)T \Rightarrow T(S \times _), \text{ distributive law} \\
&= \mathsf{dst}_{S,Y} \circ (\eta_S \times TY) \circ (S \times f) && \text{By inductive construction of } \lambda\text{, see [5]} \\
&= \mathsf{dst}_{S,Y} \circ (\eta_S \times f) && \\
&= (S \otimes _)(f) \ . &&
\end{aligned}$$

□

Lemma A.3 *For any arrow $f : S \to S'$ in $\mathcal{K}\ell(T)$, the following diagram commutes.*



Here the maps γ, γ' in **Sets** *are canonical isomorphisms distributing $\times$ over $+$. Note that $\mathcal{K}\ell(T)$ has small coproducts:* **Sets** *has small coproducts and the left adjoint $J : \mathbf{Sets} \to \mathcal{K}\ell(T)$ preserves them.*

Proof. For the commutativity of the upper square, we have

$$
\begin{array}{ll}
J{\gamma'}^{-1} \circ (f + \mathbb{N} \times f) \circ \kappa_1 & \\
= J{\gamma'}^{-1} \circ J\kappa_1 \circ f & J \text{ preserves coproducts} \\
= J\big(\, S' \xrightarrow{\sigma_{S'}} 1 \times S' \xrightarrow{\kappa_1 \times S'} (1+\mathbb{N}) \times S' \,\big) \circ f & \text{Here } \sigma_{S'} \text{ is the canonical isomorphism} \\
= \overline{(\_ \times S')} J\kappa_1 \circ J\sigma_{S'} \circ f & JF = \overline{F}J \text{ for shapely } F, \text{ see [5]} \\
= (J\kappa_1 \otimes S') \circ J\sigma_{S'} \circ f & \text{Lemma A.2} \\
= (J\kappa_1 \otimes S') \circ (1 \otimes f) \circ J\sigma_S & J\sigma_S : S \to 1 \otimes S \text{ is natural in } S \\
= \big((1+\mathbb{N}) \otimes f\big) \circ (J\kappa_1 \otimes S) \circ J\sigma_S & \otimes \text{ is a bifunctor} \\
= \big((1+\mathbb{N}) \otimes f\big) \circ J(\kappa_1 \times S) \circ J\sigma_S & \\
= \big((1+\mathbb{N}) \otimes f\big) \circ J(\gamma^{-1} \circ \kappa_1) & \\
= \big((1+\mathbb{N}) \otimes f\big) \circ J(\gamma^{-1}) \circ \kappa_1 \ . &
\end{array}
$$

The equality

$$J{\gamma'}^{-1} \circ (f + \mathbb{N} \times f) \circ \kappa_2 = \big((1+\mathbb{N}) \otimes f\big) \circ J(\gamma^{-1}) \circ \kappa_2$$

is shown in the same manner. These prove the commutativity of the upper square.

The commutativity of the lower square follows from the fact that we can substitute all the appearance of $\times$ there with $\otimes$ (Lemma A.2), and bifunctoriality of $\otimes$. □

Lemma A.4 *The diagram on the right commutes. That is, the trace map* tr_c *(as defined in the diagram on the left), is also a morphism of* $\overline{F}$*-coalgebras (as on the right).*

$$
\begin{array}{ccc}
\overline{F}X + X & \dashrightarrow & \overline{F}B + B \\
{\scriptstyle c}\uparrow & & \uparrow{\scriptstyle \cong\, J\beta^{-1}} \\
X & \underset{\mathsf{tr}_c}{\dashrightarrow} & B
\end{array}
\qquad\qquad
\begin{array}{ccc}
\overline{F}X & \xrightarrow{\overline{F}\mathsf{tr}_c} & \overline{F}B \\
{\scriptstyle \overline{c}}\uparrow & & \uparrow{\scriptstyle \overline{J\beta^{-1}}} \\
X & \xrightarrow[\mathsf{tr}_c]{} & B
\end{array}
$$

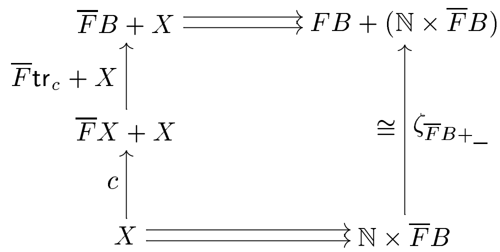
Both diagrams are in $\mathcal{K}\ell(T)$*. The coalgebras* $\overline{c}$ *and* $\overline{J\beta^{-1}}$ *are obtained by taking the closures of the coalgebras* c *and* $J\beta^{-1}$ *with internal actions.*

Proof. It suffices to show that the following diagram commute, where the operation $\widetilde{(\_)}$ is by coinduction as in (9).

$$
\begin{array}{ccc}
\mathbb{N} \times \overline{F}X & \xrightarrow{\mathbb{N} \times \overline{F}\mathsf{tr}_c} & \mathbb{N} \times \overline{F}B \\
{\scriptstyle \widetilde{c}}\uparrow & & \uparrow{\scriptstyle \widetilde{J\beta^{-1}}} \\
X & \xrightarrow[\mathsf{tr}_c]{} & B
\end{array}
$$

This follows from the fact that both maps $\widetilde{J\beta^{-1}} \circ \mathsf{tr}_c$ and $(\mathbb{N} \times F\mathsf{tr}_c) \circ \widetilde{c}$ are morphisms of coalgebras as in the following diagram. Note the finality of the coalgebra on the right: $\mathbb{N} \times \overline{F}B$ carries an initial

$(\overline{F}B + \_)$-algebra in **Sets**.



It is easily checked that $\widetilde{J\beta^{-1}} \circ \mathsf{tr}_c$ is a morphism of coalgebras: in $\mathcal{K}\ell(T)$,

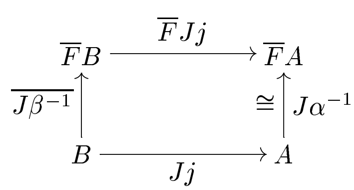
$$\begin{aligned}
\zeta_{\overline{F}B+_} \circ \widetilde{J\beta^{-1}} \circ \mathsf{tr}_c &= (\overline{F}B + \widetilde{J\beta^{-1}}) \circ J\beta^{-1} \circ \mathsf{tr}_c \\
&= (\overline{F}B + \widetilde{J\beta^{-1}}) \circ (\overline{F}\mathsf{tr}_c + \mathsf{tr}_c) \circ c \\
&= (\overline{F}B + \widetilde{J\beta^{-1}}) \circ (\overline{F}B + \mathsf{tr}_c) \circ (\overline{F}\mathsf{tr}_c + X) \circ c \\
&= (\overline{F}B + \widetilde{J\beta^{-1}} \circ \mathsf{tr}_c) \circ (\overline{F}\mathsf{tr}_c + X) \circ c \ .
\end{aligned}$$

It is trickier for $(\mathbb{N} \times F\mathsf{tr}_c) \circ \widetilde{c}$:

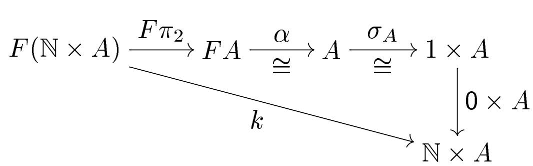
$$\begin{aligned}
&\zeta_{\overline{F}B+_} \circ (\mathbb{N} \times \overline{F}\mathsf{tr}_c) \circ \widetilde{c} \\
&= (\overline{F}\mathsf{tr}_c + \mathbb{N} \times \overline{F}\mathsf{tr}_c) \circ \zeta_{\overline{F}X+_} \circ \widetilde{c} && \text{Lemma A.3} \\
&= (\overline{F}\mathsf{tr}_c + \mathbb{N} \times \overline{F}\mathsf{tr}_c) \circ (\overline{F}X + \widetilde{c}) \circ c && \text{Definition of } \widetilde{c} \\
&= (\overline{F}B + \mathbb{N} \times \overline{F}\mathsf{tr}_c) \circ (\overline{F}\mathsf{tr}_c + \mathbb{N} \times \overline{F}X) \circ (\overline{F}X + \widetilde{c}) \circ c \\
&= (\overline{F}B + \mathbb{N} \times \overline{F}\mathsf{tr}_c) \circ (\overline{F}B + \widetilde{c}) \circ (\overline{F}\mathsf{tr}_c + X) \circ c \\
&= \big(\overline{F}B + \big((\mathbb{N} \times \overline{F}\mathsf{tr}_c) \circ \widetilde{c}\big)\big) \circ (\overline{F}\mathsf{tr}_c + X) \circ c \ .
\end{aligned}$$

This conclude the proof. □

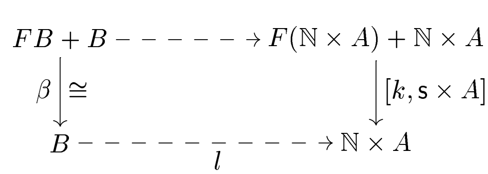
Lemma A.5 *The following diagram in $\mathcal{K}\ell(T)$ commutes. That is, the map Jj with j defined in (10) is a morphism of $\overline{F}$-coalgebra.*



Proof. Let k be the following composite in **Sets**. Here σ_A is the canonical isomorphism.



Let l be the following map in **Sets** induced by initiality.



(12)

Sublemma A.6 *For the map j as defined in (10), the following diagram commutes.*

$$\begin{array}{ccc} B & \xrightarrow{l} & \mathbb{N} \times A \\ & {\scriptstyle j}\searrow & \downarrow{\scriptstyle \pi_2} \\ & & A \end{array}$$

Proof of Sublemma A.6. It suffices to show that the following diagram in **Sets** commutes: then the statement follows from the initiality of β.

$$\begin{array}{ccc} F(\mathbb{N} \times A) + \mathbb{N} \times A & \xrightarrow{F\pi_2 + \pi_2} & FA + A \\ {\scriptstyle [k, \mathsf{s} \times A]}\downarrow & & \downarrow{\scriptstyle [\alpha, A]} \\ \mathbb{N} \times A & \xrightarrow[\pi_2]{} & A \end{array}$$

This is easy. □

By this sublemma, the commutativity of the following diagram in $\mathcal{K}\ell(T)$ proves Lemma A.5.

$$\begin{array}{ccc} \mathbb{N} \times \overline{F}B & \xrightarrow{\mathbb{N} \times \overline{F}Jj} & \mathbb{N} \times \overline{F}A \\ {\scriptstyle \widetilde{J\beta^{-1}}}\uparrow & & \uparrow{\scriptstyle \mathbb{N} \times (J\alpha^{-1})} \\ B & \xrightarrow[Jl]{} & \mathbb{N} \times A \end{array}$$

We shall show that both maps $(\mathbb{N} \times J\alpha^{-1}) \circ Jl$ and $(\mathbb{N} \times \overline{F}Jj) \circ \widetilde{J\beta^{-1}}$ are morphisms of $(FA + \_)$-coalgebras as in the following diagram in $\mathcal{K}\ell(T)$. Then finality of the codomain coalgebra yields the identity of the two maps.

$$\begin{array}{ccc} FA + B & \Longrightarrow & FA + \mathbb{N} \times FA \\ {\scriptstyle \overline{F}Jj + B}\uparrow & & \\ FB + B & & \uparrow{\scriptstyle \zeta_{\overline{F}A + \_}} \\ {\scriptstyle J\beta^{-1}}\uparrow\cong & & \\ B & \Longrightarrow & \mathbb{N} \times FA \end{array} \tag{13}$$

In the following $\alpha_{FA+\_} : FA + \mathbb{N} \times FA \xrightarrow{\cong} \mathbb{N} \times FA$ denotes an initial algebra in **Sets**. Hence $\zeta_{\overline{F}A+\_} = J(\alpha_{FA+\_})^{-1}$.

$$\begin{array}{ll}
\zeta_{\overline{F}A+_} \circ (\mathbb{N} \times J\alpha^{-1}) \circ Jl \circ J\beta \circ \kappa_1 & \\
= J\big(\alpha^{-1}_{FA+_} \circ (\mathbb{N} \times \alpha^{-1}) \circ l \circ \beta \circ \kappa_1\big) & \\
= J\big(\alpha^{-1}_{FA+_} \circ \langle \pi_1 \circ l,\, \alpha^{-1} \circ \pi_2 \circ l \rangle \circ \beta \circ \kappa_1\big) & \\
= J\big(\alpha^{-1}_{FA+_} \circ \langle \pi_1 \circ l \circ \beta \circ \kappa_1,\, \alpha^{-1} \circ j \circ \beta \circ \kappa_1 \rangle\big) & \pi_2 \circ l = j \text{ by Sublemma A.6} \\
= J\big(\alpha^{-1}_{FA+_} \circ \langle\, FB \xrightarrow{!} 1 \xrightarrow{0} \mathbb{N}\,,\, \alpha^{-1} \circ j \circ \beta \circ \kappa_1 \rangle\big) & \pi_1 \circ l \circ \beta \circ \kappa_1 = \; FB \xrightarrow{!} 1 \xrightarrow{0} \mathbb{N} \text{ by the definition of } l \\
= J\big(\kappa_1 \circ \alpha^{-1} \circ j \circ \beta \circ \kappa_1\big) & \text{For any } g,\, (\alpha_{FA+_})^{-1} \circ \langle\, FB \xrightarrow{!} 1 \xrightarrow{0} \mathbb{N}\,,\, g \rangle = \kappa_1 \circ g \\
= J\big(\kappa_1 \circ \alpha^{-1} \circ [\alpha, A] \circ (Fj + j) \circ \kappa_1\big) & \text{Definition of } j \\
= J\big(\kappa_1 \circ Fj\big) & \\
= J\big((FA + \mathbb{N} \times \alpha^{-1}) \circ (Fj + l) \circ \kappa_1\big) & \\
= \big(\overline{F}A + \mathbb{N} \times J(\alpha^{-1})\big) \circ (\overline{F}A + Jl) \circ (\overline{F}Jj + B) \circ \kappa_1 & \\
= \big(\overline{F}A + (\mathbb{N} \times J(\alpha^{-1})) \circ Jl\big) \circ (\overline{F}Jj + B) \circ \kappa_1 \;\; . &
\end{array}$$

$$
\begin{aligned}
&\zeta_{\overline{F}A+\_} \circ (\mathbb{N} \times J\alpha^{-1}) \circ Jl \circ J\beta \circ \kappa_2 \\
&= J\big((\alpha_{FA+\_})^{-1} \circ (\mathbb{N} \times \alpha^{-1}) \circ (\mathsf{s} \times A) \circ l\big) \\
&= J\big((\alpha_{FA+\_})^{-1} \circ (\mathsf{s} \times \alpha^{-1}) \circ l\big) \\
&= J\big(\kappa_2 \circ (\mathbb{N} \times \alpha^{-1}) \circ l\big) \qquad \text{For any } h,\ (\alpha_{FA+\_})^{-1} \circ \big(\mathsf{s} \times (\ X \xrightarrow{h} FA\ )\big) = \kappa_2 \circ (\mathbb{N} \times h) \text{ in } \mathbf{Sets} \\
&= \cdots \\
&= \big(\overline{F}A + (\mathbb{N} \times J(\alpha^{-1})) \circ Jl\big) \circ (\overline{F}Jj + B) \circ \kappa_2 \ .
\end{aligned}
$$

This proves that the map $(\mathbb{N} \times J\alpha^{-1}) \circ Jl$ is a morphism of coalgebras as in (13).

For the other map $(\mathbb{N} \times \overline{F}Jj) \circ \widetilde{J\beta^{-1}}$, we prove the commutativity of (13) as follows.

$$
\begin{aligned}
&\zeta_{\overline{F}A+\_} \circ (\mathbb{N} \times \overline{F}Jj) \circ \widetilde{J\beta^{-1}} \circ J\beta \\
&= \zeta_{\overline{F}A+\_} \circ (\mathbb{N} \times \overline{F}Jj) \circ J(\alpha_{FB+\_}) \circ (\overline{F}B + \widetilde{J\beta^{-1}}) \qquad \text{Definition of } \widetilde{J\beta^{-1}} \\
&= J\big((\alpha_{FA+\_})^{-1} \circ (\mathbb{N} \times FJ) \circ \alpha_{FB+\_}\big) \circ (\overline{F}B + \widetilde{J\beta^{-1}}) \\
&= J\big([(\alpha_{FA+\_})^{-1} \circ (\mathbb{N} \times FJ) \circ \alpha_{FB+\_} \circ \kappa_1,\ (\alpha_{FA+\_})^{-1} \circ (\mathbb{N} \times FJ) \circ \alpha_{FB+\_} \circ \kappa_2]\big) \circ (\overline{F}B + \widetilde{J\beta^{-1}}) \\
&= J\big([\ FB \xrightarrow{Fj} FA \xrightarrow{\kappa_1} FA + \mathbb{N} \times FA\ ,\ \ \mathbb{N} \times FB \xrightarrow{\mathbb{N} \times Fj} \mathbb{N} \times FA \xrightarrow{\kappa_2} FA + \mathbb{N} \times FA\ ]\big) \circ (\overline{F}B + \widetilde{J\beta^{-1}}) \\
&= J(Fj + \mathbb{N} \times Fj) \circ (\overline{F}B + \widetilde{J\beta^{-1}}) \\
&= \overline{F}Jj + (\mathbb{N} \times \overline{F}Jj) \circ \widetilde{J\beta^{-1}} \\
&= \big(\overline{F}A + (\mathbb{N} \times \overline{F}Jj) \circ \widetilde{J\beta^{-1}}\big) \circ (\overline{F}Jj + B) \ .
\end{aligned}
$$

This concludes the proof. □