Traces, Executions and Schedulers, Coalgebraically

Bart Jacobs[1] and Ana Sokolova[2]*

[1] Radboud University Nijmegen, The Netherlands
[2] University of Salzburg, Austria

Abstract. A theory of traces of computations has emerged within the field of coalgebra, via finality in Kleisli categories. In concurrency theory, traces are traditionally obtained from executions, by projecting away states. These traces and executions are sequences and will be called "thin". The coalgebraic approach gives rise to both "thin" and "fat" traces/executions, where in the "fat" case the structure of computations is preserved. This distinction between thin and fat will be introduced first. It is needed for a theory of schedulers in a coalgebraic setting, of which we only present the very basic definitions and results.

1 Introduction

This paper is about traces and executions, in the general setting of coalgebra. It introduces what we call "thin" and "fat" style semantics, both for traces and executions. Roughly speaking, "thin" semantics is what is traditionally considered for traces and executions, especially for labelled transition systems (LTSs) [6]. It involves sequences/lists of observable actions (for traces) or lists of actions and intermediate states (for executions). The "fat" approach emerged from more recent work on traces in a coalgebraic setting [9,10]. It applies to systems as coalgebras $c\colon X \to TF(X)$ of type TF where T is a monad for computational effect or branching type, and F is a functor that determines the transition type, subject to a set of conditions. The semantics is described as a map $X \to I$ in the Kleisli category of the monad T, where I is the initial algebra of F. Elements of this initial algebra incorporate the "fat", tree-like structure of computations of type F.

Here we describe how to understand the thin and fat approaches in a common framework. Figure 1 gives an overview. It will be explained in the course of this article. At this stage we can already see that thin semantics (of both traces and executions) involves lists, via the Kleene star $(-)^\star$—which can of course be described via the μ fixed point (initial algebra) operator. The fat semantics involves initial algebras of the functors F and $F(X \times -)$, where the latter involves the state space X, in order to accommodate states in executions. These initial algebras may have much more (tree) structure than lists. It should be noted however that they need not always exist.

* Research funded by the Austrian Science Fund (FWF) Project Nr. V00125

Initial algebras wrt. $X \to TF(X)$ with $F = F(0) + F_\bullet$	**Thin** (non-determinism only, with $T = \mathcal{P}$)	**Fat** (for general monads T)
Traces	$F_\bullet(1)^\star \times F(0)$ $= \mu Y.\, F(0) + F_\bullet(1) \times Y$	$\mu Y.\, F(Y)$
Executions	$(F_\bullet(X) \times X)^\star \times F(0)$ $= \mu Y.\, F(0) + (F_\bullet(X) \times X) \times Y$	$\mu Y.\, F(X \times Y)$

Fig. 1. Traces and executions

The first part of this paper concentrates on this table. It is needed in order to properly capture schedulers. Schedulers are often used to resolve non-determinism, by making particular choices. They are used for instance in the semantics of programming languages [16], (probabilistic) verification [5,14,15] or security [2,4]. They are not always described in the mathematically most rigorous manner. Hence a precise understanding is valuable. It was the original focus of the paper. But the "preliminary" work on thin and fat traces and executions turned out to be more involved than expected, so that in the end only the last part of the paper (Section 8) is left for schedulers. It does not do much more than setting the scene, by introducing some basic definitions and a "soundness" theorem. It ends with a definition of "completeness" of scheduler semantics, as a cliffhanger. It will be further developed and illustrated in subsequent work.

2 Preliminaries

We assume the reader is reasonably familiar with categorical notation and terminology and with the theory of coalgebras. We shall briefly review our notation. Cartesian projections will be written as $\pi_i \colon X_1 \times X_2 \to X_i$ with $\langle f, g \rangle$ for tupling. By δ we denote the diagonal, $\delta = \langle \mathrm{id}, \mathrm{id} \rangle \colon X \to X \times X$. Dually, coprojections are written as $\kappa_i \colon X_i \to X_1 + X_2$ with cotupling $[f, g]$. In **Sets** coprojections are disjoint, meaning that the pullback of κ_1 and κ_2 is empty. Coproducts are also universal: given $f \colon Y \to X_1 + X_2$ we can split up $Y_1 + Y_2 \xrightarrow{\cong} Y$ via the two pullbacks $Y_i \to Y$ of the coprojections along f, see *e.g.* [3].

We recall that every functor $F \colon \mathbf{Sets} \to \mathbf{Sets}$ is strong, via a "strength" map $\mathsf{st} \colon X \times F(Y) \to F(X \times Y)$ given by $\mathsf{st}(x, v) = F(\lambda y.\, \langle x, y \rangle)(v)$. If F happens to be a monad, then it is strong, meaning that its unit η and multiplication μ commute appropriately with this strength. A map of monads $\sigma \colon S \Rightarrow T$ is called strong if it commutes with strength. Examples of monads that occur in this setting are powerset $\mathcal{P}$ for non-determinism, lift $1 + (-)$ for partiality, or (sub)distribution $\mathcal{D}$ for probabilism. The Kleisli category of a monad T will be written as $\mathcal{K}\ell(T)$.

3 A motivating example: binary trees with output

We start with a simple example of a transition type functor generating binary trees with output, namely $F(X) = A + (B \times X^2)$ for constant sets A and B.

In a state $x \in X$ a transition in $F(X)$ of such a binary tree functor either produces an output in A and terminates, or makes a step in $B \times X^2$, consisting of an observable output element in B together with a pair of children states which will (both!) be active in the next step. In this section we shall concretely describe both "thin" and "fat" executions and traces for this transition type functor F. The general construction of these executions and traces via finality is described later, namely in Sections 5 and 7.

Thin traces and executions. As illustration we consider a coalgebra $c\colon X \to \mathcal{P}(FX)$, where $FX = A + B \times X^2$ as above. Starting from a state $x_0 \in X$ we can consider the "thin" executions starting in x_0. They are (finite) sequences of the form:

$$b_0, x_1, b_1, x_2, \ldots, b_n, x_n, a \tag{1}$$

where:

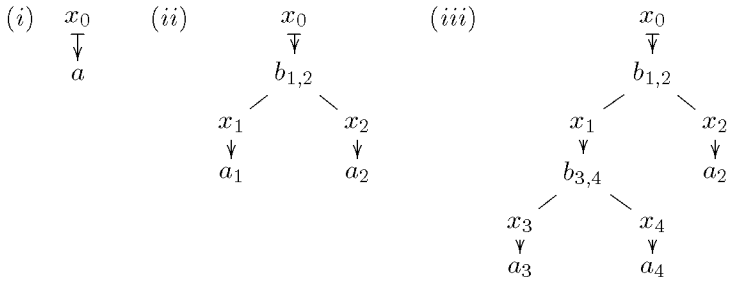
$$\begin{aligned} c(x_0) &\ni \begin{cases} (b_0, x_1, x_1') \text{ for some } x_1' \in X, \text{ or} \\ (b_0, x_1', x_1) \text{ for some } x_1' \in X \end{cases} \\ c(x_1) &\ni \begin{cases} (b_1, x_2, x_2') \text{ for some } x_2' \in X, \text{ or} \\ (b_1, x_2', x_2) \text{ for some } x_2' \in X \end{cases} \\ &\vdots \\ c(x_n) &\ni a. \end{aligned}$$

These executions thus capture possible computation paths, involving a specific choice of left or right successor state.

We shall write $\mathsf{texc}(x_0)$ for the set of all such "thin" executions; hence $\mathsf{texc}(x_0) \in \mathcal{P}((B \times X)^\star \times A)$. It will be described later as a map $\mathsf{texc}\colon X \to (B \times X)^\star \times A$, in the Kleisli category $\mathcal{K}\ell(\mathcal{P})$, obtained by finality using the result of Section 4 below.

Roughly, a trace is an execution with the states removed. So if we remove states x_i from (1) we are left with a "thin" trace, as element of $B^\star \times A$. The trace is a Kleisli map $\mathsf{ttr}\colon X \to \mathcal{P}(B^\star \times A)$. As we shall see, it can also be obtained by finality.

Fat traces and executions. Thin executions and traces describe computation *paths*. What we call fat executions or traces does not involve paths but trees that retain the structure of the transition type. Hence, examples of fat executions from a state $x_0 \in X$ are:



(2)

A fat trace can be understood as what remains when states are removed from such trees. But there is a more direct way of understanding such traces, namely as elements of the initial algebra of the functor F. As usual, this initial algebra I is obtained as a colimit $I = \mathrm{colim}_{i\in\mathbb{N}} F^i(\emptyset)$ of the initial sequence, where:

$$\begin{aligned}
F(\emptyset) &= A \\
F^2(\emptyset) &= A + (B \times A^2) \\
F^3(\emptyset) &= A + B \times \big(A + (B \times A^2)\big)^2 \\
&= A + (B \times A^2) + (B \times A \times B \times A^2) \\
&\qquad + (B \times B \times A^2 \times A) + (B \times B \times A^2 \times B \times A^2) \\
F^4(\emptyset) &= \cdots
\end{aligned}$$

Given a coalgebra $c\colon X \to \mathcal{P}(FX)$, coalgebraic trace theory provides us with a trace map $\mathsf{ftr}_c : X \to I$ in the Kleisli category $\mathcal{K}\ell(\mathcal{P})$, obtained by finality. For a state $x \in X$ the set $\mathsf{ftr}_c(x) \in \mathcal{P}(I)$ contains:

$$\begin{aligned}
a \in \mathsf{ftr}_c(x) &\iff a \in c(x) \\
\langle b, \alpha_1, \alpha_2\rangle \in \mathsf{ftr}_c(x) &\iff \exists x_1, x_2 \in X.\, \langle b, x_1, x_2\rangle \in c(x) \text{ and} \\
&\qquad \alpha_1 \in \mathsf{ftr}_c(x_1) \text{ and } \alpha_2 \in \mathsf{ftr}_c(x_2).
\end{aligned}$$

How to understand "fat" executions? It is not hard to see that the trees in (2) are elements of the initial algebra of the functor $F(X \times -)$. Indeed, this initial algebra is obtained as colimit of the chain $(F(X \times -))^i(\emptyset)$, which starts with:

$$\begin{aligned}
(F(X \times -))^1(\emptyset) &= F(X \times \emptyset) \\
&= A \\
(F(X \times -))^2(\emptyset) &= F(X \times F(X \times \emptyset)) \\
&= F(X \times A) \\
&= A + (B \times (X \times A)^2) \\
(F(X \times -))^3(\emptyset) &= \cdots
\end{aligned}$$

Let us write I_X for this initial algebra. The fat execution map then appears as a map $\mathsf{fexc}_c\colon X \to I_X$ in the Kleisli category $\mathcal{K}\ell(\mathcal{P})$, that is, as function $\mathsf{fexc}_c\colon X \to \mathcal{P}(I_X)$ in **Sets**. It can be obtained by finality. Its key properties are:

$$\begin{aligned}
a \in \mathsf{fexc}_c(x) &\iff a \in c(x) \\
\langle b, x_1, \alpha_1, x_2, \alpha_2\rangle \in \mathsf{fexc}_c(x) &\iff \langle b, x_1, x_2\rangle \in c(x) \text{ and} \\
&\qquad \alpha_1 \in \mathsf{fexc}_c(x_1) \text{ and } \alpha_2 \in \mathsf{fexc}_c(x_2).
\end{aligned}$$

An analogous definition of fat traces and executions, as well as a connection between them, can be made for an arbitrary monad T and an arbitrary functor F satisfying the requirements of the coalgebraic trace theorem. These fat traces and executions will be presented below, in Section 5.

4 Final coalgebras in Kleisli categories

Coalgebraic trace semantics has been developed for coalgebras of the form $X \to TF(X)$ where T is a suitable monad, see [9,10]. It can be formulated for arbitrary categories, but here we shall restrict ourselves to **Sets**. There are some technical requirements.

- There must be a distributive law $\lambda\colon FT \Rightarrow TF$; it induces a lifting of F to $\overline{F}\colon \mathcal{K}\ell(T) \to \mathcal{K}\ell(T)$, which commutes with the canonical functor $J\colon \mathbf{Sets} \to \mathcal{K}\ell(T)$;
- The Kleisli category $\mathcal{K}\ell(T)$ must be suitably order-enriched, with order $\sqsubseteq$ on Kleisli homsets, bottom element $\bot$ and suprema $\bigvee$ of directed subsets;
- The lifting $\overline{F}\colon \mathcal{K}\ell(T) \to \mathcal{K}\ell(T)$ must be locally monotone.

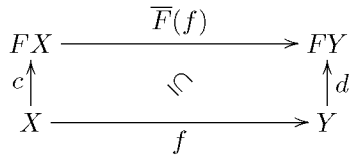
The requirements are discussed in detail in [10]. Examples of monads T and functors F that satisfy these requirements are: the powerset monad $\mathcal{P}$, the sub-distribution monad $\mathcal{D}$ (with probability distributions with sum less than or equal to 1), the lift monad $1+(-)$, and the list monad $(-)^*$, together with all "shapely" functors. We shall not concentrate on these requirements, and assume that they simply hold for the monads/functors that we use in this paper. Of crucial importance is the following result, describing how final coalgebras arise in Kleisli categories from initial algebras in the underlying category. As will be amply illustrated, it will be used throughout to obtain traces and executions, of various forms.

Theorem 1. *Let F and T be a functor and a monad on* **Sets** *satisfying the above requirements. If there is an initial algebra $\alpha\colon F(I) \xrightarrow{\cong} I$ in* **Sets**, *then the associated coalgebra $J(\alpha^{-1})\colon I \xrightarrow{\cong} F(I)$ is final in the category of coalgebras of the lifting $\overline{F}$ to $\mathcal{K}\ell(T)$.* □

Coalgebraic trace semantics shows that linear-time semantics fits into the paradigm of final coalgebra semantics, and can thus benefit from the associated machinery, for instance in showing compositionality / congruence of bisimilarity and trace equivalence for various coalgebras [11]. In the second part of this paper we shall restrict ourselves to the special case $T = \mathcal{P}$ of the powerset monad. Recall that the associated Kleisli category $\mathcal{K}\ell(\mathcal{P})$ is the category of sets and relations between them. The distributive law then follows from preservation of weak pullbacks of F. It means that there is a "relation lifting" operation $\mathrm{Rel}(F)\colon \mathcal{P}(X \times Y) \longrightarrow \mathcal{P}(FX \times FY)$, which induces the "power" distributive law $\lambda\colon F\mathcal{P} \Rightarrow \mathcal{P}F$, namely as $\lambda_X(u) = \{v \in FX \mid \mathrm{Rel}(F)(\in)(u, v)\}$, see [10, Lemma 2.3] (going back to [12]) for details. This λ commutes with the monad operations $\eta = \{-\}$ and $\mu = \bigcup$ of the powerset monad. The order on the Kleisli homsets is the pointwise inclusion order, and will be denoted by $\subseteq$.

For reasoning about schedulers (in Section 8) we borrow some results from Hasuo [7,8] on oplax morphisms in Kleisli categories—for the special case $T = \mathcal{P}$. To start, an *oplax morphism* in $\mathcal{K}\ell(\mathcal{P})$ from a coalgebra $c\colon X \to FX$ to a

coalgebra $d\colon Y \to FY$ is a Kleisli map $f\colon X \to Y$ such that:

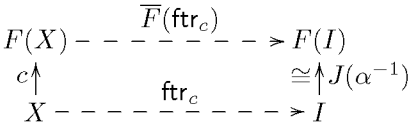


Proposition 1 ([7,8]). *For an arbitrary coalgebra $c\colon X \to TF(X)$ we write $\mathsf{tr}_c\colon X \to T(I)$ for the $\overline{F}$-coalgebra morphism in $\mathcal{K}\ell(T)$ obtained by finality. This map tr_c is the smallest one among the oplax morphisms from $c\colon X \to FX$ to the final coalgebra $I \xrightarrow{\cong} FI$ in $\mathcal{K}\ell(\mathcal{P})$.* □

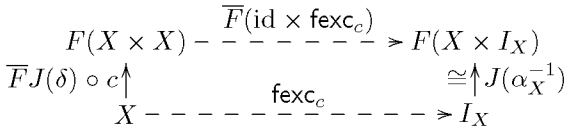
Corollary 1. *If $f\colon X \to Y$ is an oplax morphism from $c\colon X \to FX$ to $d\colon Y \to FY$, then $\mathsf{tr}_d \circ f$ is also an oplax morphism, so that $\mathsf{tr}_c \subseteq \mathsf{tr}_d \circ f$ in $\mathcal{K}\ell(\mathcal{P})$.* □

5 Fat traces and executions

Fat traces. We assume that the functor F has an initial algebra I, in addition to the assumptions from Section 4, with map $\alpha\colon F(I) \xrightarrow{\cong} I$. It yields by the trace Theorem 1 a final coalgebra $J(\alpha^{-1})\colon I \to F(I)$ in the Kleisli category $\mathcal{K}\ell(T)$. Each coalgebra $c\colon X \to TF(X)$ gives rise to a "fat" trace map $\mathsf{ftr}_c\colon X \to I$ in $\mathcal{K}\ell(T)$ by finality, as in:

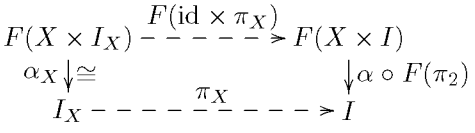


Fat executions. We fix a particular coalgebra $c\colon X \to TF(X)$, and assume that the functor $F(X \times -)$ has an initial algebra I_X, with map $\alpha_X\colon F(X \times I_X) \xrightarrow{\cong} I_X$. We obtain a "fat" execution map, again by finality in $\mathcal{K}\ell(T)$:



The notation is a bit sloppy. The map on top involves the lifting $\overline{F(\mathrm{id} \times -)}$, which exists via strength.

Relating fat executions and traces. We first construct a map $\pi_X\colon I_X \to I$ that projects away states by initiality in **Sets**, as in:

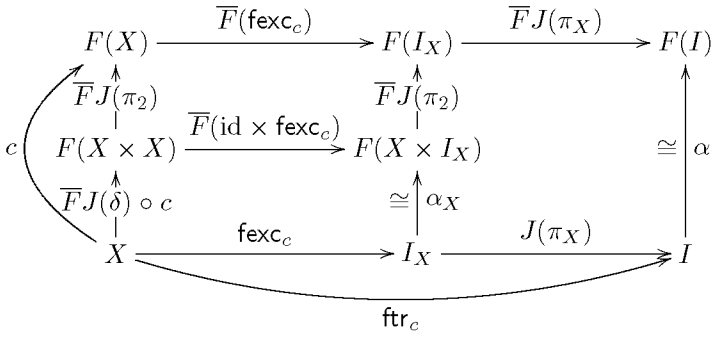


(3)

Then we obtain the basic execution-trace equation in $\mathcal{K}\ell(T)$:

$$\mathsf{ftr}_c = J(\pi_X) \circ \mathsf{fexc}_c. \tag{4}$$

It is proven by a uniqueness argument in $\mathcal{K}\ell(T)$:



The square on the right is essentially $J\colon \mathbf{Sets} \to \mathcal{K}\ell(T)$ applied to (3). The lower-left rectangle commutes by definition of fexc_c and the upper-left one commutes because $\overline{F}J(\pi_2)$ is a natural transformation $\overline{F}(\mathrm{id} \times -) \Rightarrow \overline{F}$ between liftings.

6 Splitting up functors

The problem that we address in this section can best be illustrated with an example. Consider the functor $F(X) = A + (B \times X)$. It consists of two (sum) components, one containing a successor state, namely $B \times X$, and one without, namely A. If we wish to consider non-terminating executions of an F-coalgebra only the part of the functor containing states is relevant. For traces, on the other hand, one looks at sequences/trees of observables. Then the "state-less" part is also relevant, at the end of a such a sequence. In this section we shall see how to get a handle on these different parts of a functor (with and without states), via a form of linearisation of a functor[3].

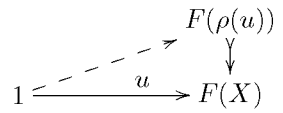
In the remainder of this paper we shall work with what we shall call subpower functors F, as "transition type" functors, in coalgebras $X \to \mathcal{P}(FX)$.

Definition 1. *A functor $F\colon \mathbf{Sets} \to \mathbf{Sets}$ will be called a "subpower" functor if it preserves weak pullbacks and comes with a natural transformation $\rho\colon F \Rightarrow \mathcal{P}$ with the special property that $u \in F(\rho_X(u))$ for each $u \in F(X)$.*

We recall that weak pullback preserving functors preserve monos/injections. Hence the inclusion $\rho_X(u) \hookrightarrow X$, for $u \in F(X)$, yields an injection $F(\rho_X(u)) \rightarrowtail$

[3] This linearisation does not keep track of "positions" or "holes" like in derivatives of functors [13,1].

$F(X)$ so that the requirement $u \in F(\rho_X(u))$ should formally be understood as a (necessarily unique) factorisation:



Like in this diagram, we often omit the subscript X in ρ_X.

It is not hard to see that the identity functor is subpower, with ρ as singleton map and that constant functors are subpower via the empty map. Further, subpower functors are closed under coproducts and products. Additionally, certain special functors are subpower, like the probability distribution functor $\mathcal{D}$, via the support map. Notice that taking the greatest subset $\rho_X(u) = X$ does not yield a natural transformation.

We now use that a powerset $\mathcal{P}(X)$ can be written as coproduct $\mathcal{P}(X) = 1 + \mathcal{P}_\bullet(X)$, where $\mathcal{P}_\bullet(X)$ contains the non-empty subsets of X and 1 corresponds to the empty subset. For a subpower functor with $\rho\colon F \Rightarrow \mathcal{P}$ we can thus form the following two pullbacks.

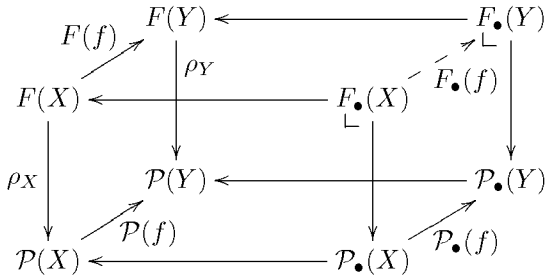
$$\begin{array}{ccccc} F_\emptyset(X) & \rightarrowtail & F(X) & \twoheadleftarrow & F_\bullet(X) \\ \downarrow & \lrcorner & \downarrow{\scriptstyle \rho_X} & & \lrcorner \downarrow \\ 1 & \xrightarrow{\emptyset} & \mathcal{P}(X) & \twoheadleftarrow & \mathcal{P}_\bullet(X) \end{array} \tag{5}$$

Since coproducts are universal [3] in **Sets**, the induced cotuple $F_\emptyset(X) + F_\bullet(X) \to F(X)$ is an isomorphism. In this way we can split up F in two parts, one with output states, and one without them.

Lemma 1. *Consider a subpower functor F.*

1. *Both $F_\emptyset$ and $F_\bullet$ are functors with (coprojection) natural transformations $F_\emptyset \Rightarrow F$ and $F_\bullet \Rightarrow F$.*
2. *$F_\emptyset(X) = F(0)$, for each X, so that $F(X) = F(0) + F_\bullet(X)$—where 0 is the initial object (empty set) in* **Sets**.
3. *The functor $F_\bullet$ is again subpower, via $\rho_\bullet = \big(F_\bullet \Rightarrow F \overset{\rho}{\Rightarrow} \mathcal{P}\big)$; as a consequence, there is a distributive law $\lambda_\bullet\colon F_\bullet\mathcal{P} \Rightarrow \mathcal{P}F_\bullet$, commuting with λ via $F_\bullet \Rightarrow F$ in the obvious way.*

Proof. For the first point we shall do the proof for $F_\bullet$. Consider for a map $f\colon X \to Y$ the following diagram:



The maps $F_\bullet(X) \to F(X)$ are natural by construction of $F_\bullet$. Similarly, there is a natural transformation $F_\emptyset \Rightarrow F$.

For point (2), note first that the empty set is an isomorphism $\emptyset\colon 1 \xrightarrow{\cong} \mathcal{P}(0)$. Hence for $X = 0$ the map $F_\emptyset(0) \to F(0)$ in (5) is also an isomorphism. Thus $F(0) \cong F_\emptyset(0) \hookrightarrow F_\emptyset(X)$. Conversely, if $u \in F_\emptyset(X)$, then $u \in F(\rho(u)) = F(0)$.

For the third point we use that the natural transformation $F_\bullet \Rightarrow F = F(0) + F_\bullet$ is essentially the second coprojection κ_2. Hence if $u \in F_\bullet(X)$, then $\kappa_2(u) \in F(\rho(\kappa_2(u))) = F(0) + F_\bullet(\rho(\kappa_2(u)))$, so that $u \in F_\bullet(\rho_\bullet(u))$.

The distributive law $\lambda_\bullet\colon F_\bullet\mathcal{P} \Rightarrow \mathcal{P}F_\bullet$ exists because $F_\bullet$ preserves weak pullbacks: assume $p_i\colon P \to X_i$ is the weak pullback of $f_i\colon X_i \to Y$, for $i \in \{1, 2\}$. Let $g_i\colon Z \to F_\bullet(X_i)$ satisfy $F_\bullet(f_1) \circ g_1 = F_\bullet(f_2) \circ g_2$. Then by post-composition with $F_\bullet(X_i) \hookrightarrow F(0) + F_\bullet(X_i) = F(X_i)$ we obtain, because F preserves weak pullbacks, a mediating map $h\colon Z \to F(P) = F(0) + F_\bullet(P)$. This h must then factor through $F_\bullet(P)$.

Hence $\rho_\bullet$ makes $F_\bullet$ a subpower functor. The proof of the connection between $\lambda_\bullet$ and λ uses relation lifting. The details are skipped. □

The next map $\mathsf{split}\colon FX \to \mathcal{P}(F(0) + F_\bullet(X) \times X)$ will be important:

$$\begin{array}{rcl} FX = F(0)+F_\bullet(X) & \xrightarrow{\mathrm{id}+\delta} & F(0)+F_\bullet(X) \times F_\bullet(X) \\ & & \downarrow{\scriptstyle \mathrm{id}+\mathrm{id}\times\rho_\bullet} \\ & & F(0)+F_\bullet(X) \times \mathcal{P}(X) \xrightarrow{\mathrm{id}+\mathsf{st}} F(0)+\mathcal{P}(F_\bullet(X) \times X) \\ & & \qquad\qquad \downarrow{\scriptstyle [\eta\circ\kappa_1, \mathcal{P}(\kappa_2)]} \\ & & \qquad\qquad \mathcal{P}(F(0)+F_\bullet(X) \times X) \end{array} \tag{6}$$

It is natural $F \Rightarrow \mathcal{P}(F(0) + F_\bullet \times \mathrm{id})$, in **Sets**.

7 Thin traces and executions for non-determinism

We now restrict ourselves to the powerset monad $\mathcal{P}$ for non-determinism and will assume that the transition type functor is a subpower functor, via $\rho\colon F \Rightarrow \mathcal{P}$. Hence we can write $F(X) = F(0) + F_\bullet(X)$.

Thin traces. We shall write L for the set of lists $L = F_\bullet(1)^\star \times F(0)$, ending with an element in $F(0)$. This L is of course the initial algebra of the functor $Y \mapsto F(0) + F_\bullet(1) \times Y$. The initial algebra structure will be written as $[\mathsf{end}, \mathsf{cons}]\colon F(0) + F_\bullet(1) \times L \xrightarrow{\cong} L$. Therefore, using trace semantics in the Kleisli category $\mathcal{K}\ell(\mathcal{P})$ of the powerset monad, we obtain a "thin" trace map by finality:

$$\begin{array}{ccc} F(0) + F_\bullet(1) \times X & \xrightarrow{\mathrm{id} + \mathrm{id} \times \mathsf{ttr}_c} & F(0) + F_\bullet(1) \times L \\ {\scriptstyle c_{\mathsf{lt}}}\uparrow & & \uparrow{\scriptstyle \cong} \\ X & \xrightarrow{\mathsf{ttr}_c} & L \end{array}$$

where the coalgebra $c_{\mathsf{lt}}\colon X \to \mathcal{P}(F(0) + F_\bullet(1) \times X)$ is defined as composite

$$c_{\mathsf{lt}} = \mathcal{P}(\mathrm{id} + (F_\bullet(!) \times \mathrm{id})) \circ \mu \circ \mathcal{P}(\mathsf{split}) \circ c, \tag{7}$$

with split defined in (6). We note that for "linear" functors such as $F(X) = A + B \times X$ for which $F(X) = F(0) + F_\bullet(1) \times X$ there is no difference between "thin" and "fat" traces (or executions).

Thin executions. We now fix a non-deterministic coalgebra $c\colon X \to \mathcal{P}(FX)$ in advance and write $L_X = (F_\bullet(X) \times X)^\star \times F(0)$ for the set of lists of executions. This L_X is the initial algebra of the functor $F(0) + (F_\bullet(X) \times X) \times (-)$. The associated thin execution map is obtained in:

$$\begin{array}{ccc} F(0) + (F_\bullet(X) \times X) \times X & \xrightarrow{\mathrm{id} + \mathrm{id} \times \mathsf{texc}_c} & F(0) + (F_\bullet(X) \times X) \times L_X \\ {\scriptstyle c_{\mathsf{le}}}\uparrow & & \uparrow{\scriptstyle \cong} \\ X & \xrightarrow{\qquad\qquad \mathsf{texc}_c \qquad\qquad} & L_X \end{array}$$

where the coalgebra $c_{\mathsf{le}}\colon X \to \mathcal{P}(F(0) + (F_\bullet(X) \times X) \times X)$ is defined as:

$$c_{\mathsf{le}} = \mathcal{P}(\mathrm{id} + \langle \mathrm{id}, \pi_2 \rangle) \circ \mu \circ \mathcal{P}(\mathsf{split}) \circ c, \tag{8}$$

where $\mathsf{split}\colon F(X) \to \mathcal{P}(F(0) + F_\bullet(X) \times X)$ is from (6).

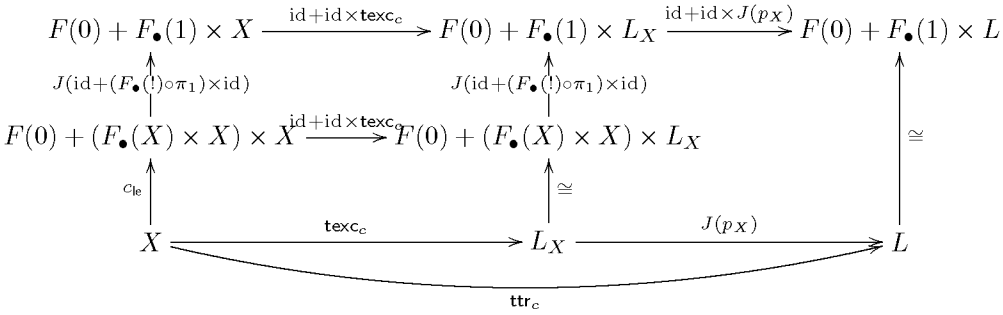
Relating thin executions and traces. The first step in relating thin executions and traces is to get a map $L_X \to L$ between the corresponding sequences. It is of course obtained by initiality (of L_X) in **Sets**, as in:

$$\begin{array}{ccc} F(0) + (F_\bullet(X) \times X) \times L_X & \xrightarrow{\mathrm{id} + \mathrm{id} \times p_X} & F(0) + (F_\bullet(X) \times X) \times L \\ {\scriptstyle [\mathsf{end}, \mathsf{cons}]}\downarrow{\scriptstyle \cong} & & {\scriptstyle [\mathsf{end}, \mathsf{cons}] \circ (\mathrm{id} + (F_\bullet(!) \circ \pi_1) \times \mathrm{id})}\downarrow \\ L_X & \xrightarrow{\qquad\qquad p_X \qquad\qquad} & L \end{array}$$

As before we obtain the basic execution-trace equation in $\mathcal{K}\ell(\mathcal{P})$, but this time for the thin case:

$$\mathsf{ttr}_c = J(p_X) \circ \mathsf{texc}_c. \tag{9}$$

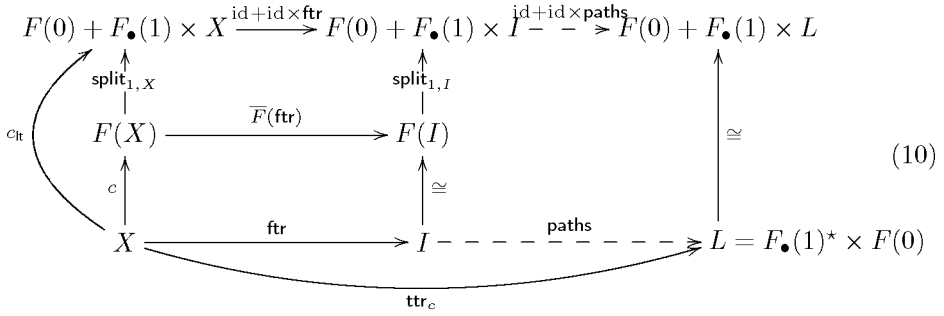
It is proven by uniqueness in:



It requires that we prove that the vertical composite on the left equals c_{lt}. This follows from an easy calculation in **Sets**:

$$\begin{aligned}
&\mathcal{P}(\mathrm{id} + (F_\bullet(!) \circ \pi_1) \times \mathrm{id}) \circ c_{\mathsf{le}} \\
&= \mathcal{P}(\mathrm{id} + (F_\bullet(!) \circ \pi_1) \times \mathrm{id}) \circ \mathcal{P}(\mathrm{id} + \langle \mathrm{id}, \pi_2 \rangle) \circ \mu \circ \mathcal{P}(\mathsf{split}) \circ c \\
&= \mathcal{P}(\mathrm{id} + (F_\bullet(!) \times \mathrm{id})) \circ \mu \circ \mathcal{P}(\mathsf{split}) \circ c \\
&= c_{\mathsf{lt}}.
\end{aligned}$$

From fat to thin traces. As we have seen in the previous sections one can define thin and fat traces separately. Here we show that one can also obtain thin traces from fat ones via a special "paths" map between the corresponding initial algebras, as in the following diagram (in $\mathcal{K}\ell(\mathcal{P})$).



(10)

where $\mathsf{split}_{1,X} = \mathcal{P}(\mathrm{id} + (F_\bullet(!) \times \mathrm{id})) \circ \mathsf{split}_X$. The Kleisli map $\mathsf{paths}\colon I \to \mathcal{P}(L)$ that is (implicitly) defined by finality yields the set of paths/sequences in a tree. The upper left square commutes by naturality of $\mathsf{split}_{1,X}$, from $\overline{F}$ to $\overline{F(0) + F_\bullet(1) \times \mathrm{id}}$ in $\mathcal{K}\ell(\mathcal{P})$. This naturality requires the additional assumption that $\lambda\colon F\mathcal{P} \Rightarrow \mathcal{P}F$ and $\rho\colon F \Rightarrow \mathcal{P}$ are compatible in the sense that $\mu \circ \mathcal{P}(\rho) \circ \lambda = \mu \circ \rho$. Details are skipped.

Thin and fat executions are similarly related via a paths map between initial algebras I_X to L_X.

8 Scheduling

Scheduling is about resolving non-determinism, by choosing some structure like singletons, lists, multisets or distributions instead of plain, unstructured, subsets. How this non-determinism is resolved will be described generically, at first, in terms of another monad S with a (strong) monad map $\sigma\colon S \Rightarrow \mathcal{P}$. Possible examples of S are identity Id, lift $1 + (-)$, list $(-)^\star$, multiset $\mathcal{M}$, or distribution $\mathcal{D}$, each with "obvious" mappings σ to the powerset monad. Very roughly, scheduling "of type S" involves a suitable inverse to this mapping $\sigma\colon S \Rightarrow \mathcal{P}$.

For a set X we shall abbreviate

$$\Xi = X \times (F_\bullet(X) \times X)^*.$$

It contains all the finite (non-terminating) thin "executions", appended to a starting state, as first component. Of course the elements of Ξ are not really executions since there is no coalgebra involved and no one-step connections between the constituents. There are three obvious maps:

$$X \xrightarrow{\mathsf{in}} \Xi = X \times (F_\bullet(X) \times X)^\star \underset{\mathsf{last}}{\overset{\mathsf{first}}{\rightrightarrows}} X$$

where $\mathsf{in}(x) = \langle x, \langle\rangle\rangle$. The map first yields the first state of the execution, simply via the first projection, and last yields the last state of the execution defined as:

$$\mathsf{last}(x, \alpha) = \begin{cases} x & \text{if } \alpha \text{ is the empty sequence } \langle\rangle \\ y & \text{if } \alpha = \beta \cdot \langle u, y\rangle. \end{cases}$$

Clearly, $\mathsf{first} \circ \mathsf{in} = \mathrm{id}$, but also $\mathsf{last} \circ \mathsf{in} = \mathrm{id}$.

We now come to the crucial notion of scheduler. Informally it chooses a computation of type S for a non-deterministic computation, given a scheduling type $\sigma\colon S \Rightarrow \mathcal{P}$.

Definition 2. *A scheduler of type $\sigma\colon S \Rightarrow \mathcal{P}$ for a coalgebra $c\colon X \to \mathcal{P}F(X)$ is a mapping ξ in:*

$$\begin{array}{ccc} \Xi & \xrightarrow{\xi} & SF(X) \\ {\scriptstyle \mathsf{last}}\downarrow & \swarrow & \downarrow{\scriptstyle \sigma} \\ X & \xrightarrow[c]{} & \mathcal{P}F(X) \end{array}$$

Intuitively, a scheduler chooses a next step starting from the last state of an execution in Ξ. The inclusion presented by the diagram, together with the definition of last, ensures that the chosen next step is contained in the next-step options that the coalgebra c provides for the last state of the execution.

Because such a scheduler ξ takes elements from Ξ as input it may be called *history dependent*: the scheduler may take previous execution steps into account when making the current scheduling decision. Here one may object that the set Ξ has too many elements—not just the proper executions. One way to handle this is to let ξ choose in these non-proper execution cases a "bottom" element.

Example 1. We illustrate several variants of schedulers, for the simplest and most well known example of LTS with termination, namely $FX = \{\checkmark\} + A \times X$, in which case $\Xi = X \times (A \times X)^*$.

1. *Deterministic schedulers* have type $\eta\colon Id \Rightarrow \mathcal{P}$. They are maps $\xi\colon \Xi \to (\{\checkmark\} + A \times X)$ that choose a single possibility out of the last state of an execution.
2. *Non-deterministic schedulers* are schedulers of type $\mathrm{id}\colon \mathcal{P} \Rightarrow \mathcal{P}$. They merely reduce non-determinism by pruning out some of the possible options in the last state of an execution. These are maps $\xi\colon \Xi \to \mathcal{P}(\{\checkmark\} + A \times X)$.

3. *Randomized schedulers* replace non-deterministic choice by a probability distribution via the scheduling type $\mathsf{supp}\colon \mathcal{D} \Rightarrow \mathcal{P}$. They are maps $\xi\colon \Xi \to \mathcal{D}(\{\checkmark\} + A \times X)$ such that $\xi(x,\alpha)(z) \neq 0$ implies $z \in c(\mathsf{last}(x,\alpha))$. It means that the support of the distribution $\varphi = \xi(x,\alpha)$ that is produced by the scheduler ξ on a non-terminating execution $\langle x, \alpha\rangle$ is a subset of the set of transitions resulting from the last state $\mathsf{last}(x,\alpha)$ of the execution.

Definition 3. *Let ξ be a scheduler for $c\colon X \to \mathcal{P}F(X)$. The coalgebra of executions of c under the scheduler ξ,*

$$\Xi \xrightarrow{\;c_\xi\;} \mathcal{P}F(\Xi)$$

is the composite of the following pile of maps.

$$\begin{aligned}
\Xi &\xrightarrow{\;\langle id, \xi\rangle\;} \Xi \times SF(X) \\
&\xrightarrow{\;id \times \sigma\;} \Xi \times \mathcal{P}F(X) \\
&\xrightarrow{\;\mathsf{st}\;} \mathcal{P}(\Xi \times (F(0) + F_\bullet(X))) \\
&\xrightarrow{\;\mathcal{P}(\mathit{dist})\;} \mathcal{P}(\Xi \times F(0) + \Xi \times F_\bullet(X)) \\
&\xrightarrow{\;\mathcal{P}(\pi_2 + \langle id, \pi_2\rangle)\;} \mathcal{P}(F(0) + \Xi \times F_\bullet(X) \times F_\bullet(X)) \\
&\xrightarrow{\;\mathcal{P}(id + \mathsf{st})\;} \mathcal{P}(F(0) + F_\bullet(\Xi \times F_\bullet(X) \times X)) \\
&\xrightarrow{\;\mathcal{P}(id + F_\bullet(\mathit{cons}))\;} \mathcal{P}F(\Xi) \qquad .
\end{aligned}$$

where cons extends executions; it satisfies $\mathsf{last} \circ \mathsf{cons} = \pi_3$.

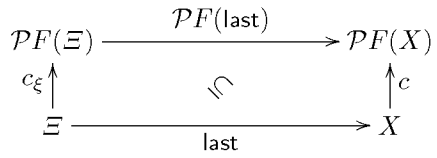
The coalgebra c_ξ yields a fat trace map $\mathsf{ftr}_{c_\xi}\colon \Xi \to I$ in the Kleisli category $\mathcal{K}\ell(\mathcal{P})$, for I being the initial F-algebra. One can also look at the thin trace map of c_ξ but it can be obtained simply via the paths map from (10). The next lemma relates the scheduler-induced coalgebra c_ξ on executions to the original coalgebra c on states.

Lemma 2. *The map $J(\mathsf{last})$ is an oplax morphism from $c_\xi\colon \Xi \to F(\Xi)$ to $c\colon X \to F(X)$ in $\mathcal{K}\ell(\mathcal{P})$, i.e.*

$$\begin{array}{ccc}
F(\Xi) & \xrightarrow{\;\overline{F}J(\mathit{last})\;} & F(X) \\
{\scriptstyle c_\xi}\uparrow & \swarrow & \uparrow{\scriptstyle c} \\
\Xi & \xrightarrow[\;J(\mathit{last})\;]{} & X
\end{array}$$

Proof. We first note that $\overline{F}J = JF$, and that $JF(f) \circ c \subseteq d \circ J(f)$ in $\mathcal{K}\ell(\mathcal{P})$ if and only if $\mathcal{P}F(f) \circ c \subseteq d \circ f$ in **Sets**. Therefore, it is enough to show that the

following oplax diagram commutes in **Sets**.



which we get from the following calculation

$$\begin{aligned}
\mathcal{P}F(\mathsf{last}) \circ c_\xi &= \mathcal{P}F(\mathsf{last}) \circ \mathcal{P}(\mathrm{id} + F_\bullet \mathsf{cons}) \circ \mathcal{P}(\mathrm{id} + \mathsf{st}) \circ \mathcal{P}(\pi_2 + \langle \mathrm{id}, \pi_2 \rangle) \circ \\
&\qquad \mathcal{P}(\mathsf{dist}) \circ \mathsf{st} \circ \langle \mathrm{id}, \sigma \circ \xi \rangle \\
&= \mathcal{P}(\mathrm{id} + F_\bullet \mathsf{last}) \circ \mathcal{P}(\mathrm{id} + F_\bullet \mathsf{cons}) \circ \mathcal{P}(\mathrm{id} + \mathsf{st}) \circ \mathcal{P}(\pi_2 + \langle \mathrm{id}, \pi_2 \rangle) \circ \\
&\qquad \mathcal{P}(\mathsf{dist}) \circ \mathsf{st} \circ \langle \mathrm{id}, \sigma \circ \xi \rangle \\
&= \mathcal{P}(\mathrm{id} + F_\bullet \pi_3) \circ \mathcal{P}(\mathrm{id} + \mathsf{st}) \circ \mathcal{P}(\pi_2 + \langle \mathrm{id}, \pi_2 \rangle) \circ \\
&\qquad \mathcal{P}(\mathsf{dist}) \circ \mathsf{st} \circ \langle \mathrm{id}, \sigma \circ \xi \rangle \\
&= \mathcal{P}(\mathrm{id} + \pi_3) \circ \mathcal{P}(\pi_2 + \langle \mathrm{id}, \pi_2 \rangle) \circ \mathcal{P}(\mathsf{dist}) \circ \mathsf{st} \circ \langle \mathrm{id}, \sigma \circ \xi \rangle \\
&= \mathcal{P}(\pi_2 + \pi_2) \circ \mathcal{P}(\mathsf{dist}) \circ \mathsf{st} \circ \langle \mathrm{id}, \sigma \circ \xi \rangle \\
&= \mathcal{P}(\pi_2) \circ \mathsf{st} \circ \langle \mathrm{id}, \sigma \circ \xi \rangle \\
&= \pi_2 \circ \langle \mathrm{id}, \sigma \circ \xi \rangle \\
&= \sigma \circ \xi \\
&\subseteq c \circ \mathsf{last}
\end{aligned}$$

where the inclusion holds by the definition of a scheduler. □

Definition 4. *For a coalgebra $c\colon X \to \mathcal{P}F(X)$ we define the fat "scheduler" trace map $\mathsf{fstr}_c\colon X \to \mathcal{P}(I)$ as:*

$$\mathsf{fstr}_c(x) = \bigcup \{ \mathsf{ftr}_{c_\xi}(\mathsf{in}(x)) \mid \xi \text{ is scheduler for } c \}.$$

Theorem 2 (Soundness). *The fat scheduler traces are contained in the fat traces, that is, for any coalgebra $c\colon X \to \mathcal{P}F(X)$ it holds that*

$$\mathsf{fstr}_c \subseteq \mathsf{ftr}_c.$$

Proof. By Lemma 2 we have that $J(\mathsf{last})$ is an oplax morphism from $c_\xi\colon \Xi \to F(\Xi)$ to $c\colon X \to F(X)$ in $\mathcal{K}\ell(\mathcal{P})$. Therefore, Corollary 1 yields $\mathsf{ftr}_{c_\xi} \subseteq \mathsf{ftr}_c \circ J(\mathsf{last})$ in $\mathcal{K}\ell(\mathcal{P})$, or equivalently $\mathsf{ftr}_{c_\xi} \subseteq \mathsf{ftr}_c \circ \mathsf{last}$, with composition in **Sets**, which further implies that

$$\mathsf{ftr}_{c_\xi} \circ \mathsf{in} \subseteq \mathsf{ftr}_c \circ \mathsf{last} \circ \mathsf{in} = \mathsf{ftr}_c.$$ □

We end with a definition, which introduces many new questions, such as: which types of scheduling are complete for which functors. These questions will be postponed to future work.

Definition 5. *A scheduler type $\sigma\colon S \Rightarrow \mathcal{P}$ is fat complete if $\mathsf{fstr}_c = \mathsf{ftr}_c$ for any coalgebra $c\colon X \to \mathcal{P}F(X)$.*

9 Conclusions and future work

This paper deepens the study of "traces" in a coalgebraic setting and brings schedulers within scope. Many research issues remain, like completeness of scheduling, for instance for deterministic or probabilistic systems, or scheduling for other monads than powerset. Also, the relevance of derivatives of functors—capturing the idea of a hole where a scheduler should continue—needs further investigation.

References

1. M. Abott, Th. Altenkirch, N. Ghani, and C. McBride. Derivatives of containers. In M. Hofmann, editor, *Typed Lambda Calculi and Applications*, pages 16–30. LNCS 2701, 2003.
2. Mohit Bhargava and Catuscia Palamidessi. Probabilistic anonymity. In Martín Abadi and Luca de Alfaro, editors, *CONCUR'05*, pages 171–185. LNCS 3653, 2005.
3. A. Carboni, S. Lack, and R.F.C. Walters. Introduction to extensive and distributive categories. *JPAA*, 84(2):145–158, 1993.
4. Konstantinos Chatzikokolakis and Catuscia Palamidessi. Making random choices invisible to the scheduler. In *CONCUR'07*, pages 42–58. LNCS 4703, 2007.
5. L. Cheung. *Reconciling Nondeterministic and Probabilistic Choices.* PhD thesis, Radboud University Nijmegen, 2006.
6. R.J. van Glabbeek. The linear time – branching time spectrum (extended abstract). In J.C.M. Baeten and J.W. Klop, editors, *CONCUR'90*, pages 278–297. LNCS 458, 1990.
7. I. Hasuo. Generic forward and backward simulations. In C. Baier and H. Hermanns, editors, *CONCUR'06*, pages 406–420. LNCS 4137, 2006.
8. I. Hasuo. *Tracing Anonymity with Coalgebras.* PhD thesis, Radboud University Nijmegen, 2008.
9. I. Hasuo and B. Jacobs. Context-free languages via coalgebraic trace semantics. In J.L. Fiadeiro, N. Harman, M. Roggenbach, and J. Rutten, editors, *CALCO'05*, pages 213–231. LNCS 3629, 2005.
10. I. Hasuo, B. Jacobs, and A. Sokolova. Generic trace semantics via coinduction. *LMCS*, 3(4:11), 2007.
11. I. Hasuo, B. Jacobs, and A. Sokolova. The microcosm principle and concurrency in coalgebra. In R. Amadio, editor, *FOSSACS'08*, pages 246–260. LNCS 4962, 2008.
12. B. Jacobs. Trace semantics for coalgebras. In J. Adámek and S. Milius, editors, *CMCS'04*. ENTCS 106, 2004.
13. A. Joyal. Foncteurs analytiques et espces de structures. In G. Labelle and P. Leroux, editors, *Combinatoire Enumerative*, pages 126–159. LNM 1234, Springer, Berlin, 1986.
14. R. Segala. A compositional trace-based semantics for probabilistic automata. In I. Lee and S.A. Smolka, editors, *CONCUR'95*, pages 234–248. LNCS 962, 1995.
15. R. Segala. *Modeling and verification of randomized distributed real-time systems.* PhD thesis, MIT, 1995.
16. D. Varacca and G. Winskel. Distributing probability over non-determinism. *MSCS*, 16:87–113, 2006.