Trace Semantics via Generic Observations

Sergey Goncharov

Department of Computer Science,
Friedrich-Alexander-Universität Erlangen-Nürnberg

Abstract. Recent progress on defining abstract trace semantics for coalgebras rests upon two observations: (i) coalgebraic bisimulation for deterministic automata coincides with trace equivalence, and (ii) the classical powerset construction for automata determinization instantiates the generic idea of lifting a functor to the Eilenberg-Moore category of an appropriate monad $\mathbb{T}$. We take this approach one step further by rebasing the latter kind of trace semantics on the novel notion of $\mathbb{T}$-*observer*, which is just a certain natural transformation of the form $F \to GT$, and thus allowing for elimination of assumptions about the structure of the coalgebra functor. As a specific application of this idea we demonstrate how it can be used for capturing trace semantics of push-down automata. Furthermore, we show how specific forms of observers can be used for coalgebra-based treatment of internal automata transitions as well as weak bisimilarity of processes.

1 Introduction

Perhaps the most impressive and productive category-theoretic archetypes adapted by theoretical computer science are the notions of coalgebra and computational monad. Whereas computational monads are typically used for sakes of denotational semantics in order to encapsulate, i.e. internalize a computational effect, and thus make it invisible, coalgebras are better known for their extroverted character, exhibited by their tendency to actively interact with the outside.

As is usually the case, except oversimplified, one needs both kinds of features: a way to hide some information, but also a possibility to stay reactive. Generic trace semantics for coalgebras, originated in [11], can be viewed as an attempt to resolve the mismatch between internal and external behaviours of coalgebras by means of the generic notion of a trace, provided one treats nondeterminism as a sort of intrinsic effect to be abstracted away from. More recently in [13], it has been ascertained that generic traces naturally appear in a generalization of the powerset construction for nondeterministic automata by identifying the powerset functor as a monad and abstracting away from it. In view of the latter work, the distinction between coalgebraic behaviours and traces can be explained most easily by the core example of nondeterministic automata as follows. A nondeterministic automaton is presented by a coalgebra of the functor $F\mathcal{P}_\omega$ where $\mathcal{P}_\omega$ stands for finite powersets and F is the deterministic automata functor $2 \times -^A$. The universal arrow from such a coalgebra to the final coalgebra $\nu F\mathcal{P}_\omega$ captures behaviours, while traces are obtained as behaviours of the determinized

R. Heckel and S. Milius (Eds.): CALCO 2013, LNCS 8089, pp. 158–174, 2013.

version of the original automaton. The underlying property involved here is a fact, known from experience: for deterministic automata, trace equivalence and behavioural equivalence coincide. Notably, the theory does not explain why this is the case. Moreover, the obtained trace semantic becomes rigidly bound to the syntactic form of the functor, which limits the application power of the construction drastically.

In this work, which is heavily based on [13] as well as on the more recent [28], we attempt to push the existing development forward in two respects. First, we generalise the framework by introducing the concept of a $\mathbb{T}$-*observer*, which is a natural transformation of the form

$$F \to GT$$

with G subject to a distributive law $\pi : \mathbb{T}G \to G\mathbb{T}$. This will allow us to detach from any assumptions about the shape of the coalgebra functor; Therefore, we shall see how this helps to naturally define trace semantics of push-down automata (and thus refine the earlier attempt from [28]). Second, we establish the notion of $\mathbb{T}$-observer as a generic phenomenon, arising from a natural adjoint construction. The latter, somewhat surprisingly, suggests a distinction between *real-time observers*, viewing only the current transition of the system, and *lookahead observers*, viewing potentially infinite sequence of transitions starting from the current one on.

Paper Organization. In Sections 2 and 3 we give the necessary preliminaries on coalgebras and (computational) monads; Moreover, in the latter we introduce a stack monad and its nondeterministic counterpart and give operational characterizations for them. In Section 4 we introduce the crucial notion of a $\mathbb{T}$-observer and then, in Section 5, reesteblish it by a category-theoretic argument. Sections 6, 7 and 8 contain the main corpus of examples: internal actions of automata, weak bisimilarity, pushdown automata, and reactive programs with side-effects.

2 Systems and (Final) Coalgebras

In general, one can sensibly speak about a system as of something sequentially evolving in time while changing its internal state and possibly interacting with the outer world. The notion of *coalgebra* is known to provide a suitable mathematical abstraction, basically, as general as that.

Given a category $\mathbf{C}$ (typically the category **Set** of sets and functions) and an endofunctor $F : \mathbf{C} \to \mathbf{C}$, an F-coalgebra is any morphism of the form $f : X \to FX$. The object X it often referred to as the *state space* of f. For a fixed F, F-coalgebras form a category, which we denote as $\mathsf{coalg}_F(\mathbf{C})$, with morphisms being the morphisms of the state spaces, subject to coherence with the coalgebra structure as follows: for $f : X \to FX$ and $g : Y \to FY$, $h : X \to Y$ is an F-morphism iff $gh = (Fh)f$. A terminal object in the category of F-coalgebras, called a *final coalgebra*, (whose state space is) often denoted as νF, if it exists, plays a critical role in the theory of systems: given an F-coalgebra $f : X \to FX$, the terminal morphism $\hat{f} : X \to \nu F$ produces *behaviours* of the system,

i.e. the complete characterization of its evolution for every chosen initial state $s : 1 \to X$. States with the same behaviour are called *behaviourally equivalent*. For weak pullback preserving functors on **Set**, behavioural equivalence is known to capture the core notion of *strong bisimilarity*, stemming from the realm of process algebra [26]

We now recap two quite representative examples (see e.g. [13]): deterministic (DA) and nondeterministic (NA) automata.

Example 1 (DA). Let $FX = 2 \times X^A$ be an endofunctor over **Set** with finite A, understood as an alphabet of atomic symbols. A coalgebra of this functor has the form $\langle o, t\rangle : X \to 2 \times X^A$. With X being finite $\langle o, t\rangle$ represents a deterministic automaton: $o : X \to 2$ models the acceptance condition, equivalently a subset of final states; and $t : X \to X^A$, whose uncurried version has the profile $X \times A \to X$, models the transition function.

The final coalgebra has as the state space the set of all formal languages over A, i.e. $\nu F = 2^{A^*}$. This can be equipped with the F-coalgebra structure by the isomorphism:

$$\iota : 2^{A^*} \simeq 2^{1+A\times A^*} \simeq 2 \times (2^{A^*})^A.$$

Now the final map $\hat{f}$ generated by $f = \langle o, t\rangle$ sends every state x of the automaton to the language accepted by f at this state.

The functor $2 \times -^A$ plays a particular role in trace semantics. We denote it by L_A and call *(formal) language functor*, which name is suggested by the form of the final coalgebra νL_A. We shall, when appropriate, treat the elements $0, 1$ of 2 as truth values $\bot$ and $\top$ correspondingly.

Example 2 (NA). In order to switch to the nondeterministic case it suffices to take the functor by $FX = 2 \times \mathcal{P}_\omega(X)^A$. The transition function of an NA $\langle c, t\rangle : X \to 2 \times \mathcal{P}_\omega(X)^A$ now becomes equivalent to a finitary relation over $X \times A \times X$. The state space of the final F-coalgebra consists of all finitely-branching trees whose nodes are labelled either by $\bot$ or by $\top$ and whose edges are labelled by action names from A, modulo a suitable bisimilarity relation.

As indicated above, although DA and NA recognize the same languages, coalgebra-driven behavioural equivalences diverge for them.

3 Monads, Theories, and Effects

Monads play a crucial role in denotational semantics, for they capture the very essence of various side-effects, most prominently their composability [19, 18]. Monads are also known to be successfully applied in coalgebra-based formalisations of systems, although in a somewhat restricted way. The underlying philosophy of this paper is to identify monads with computational effects in spirit of [19]. In accordance to this we view e.g. nondeterministic automata as those featuring nondeterminism as a side-effect, etc.

Given a Cartesian category **C**, i.e. a category with finite Cartesian products (including the empty one, which is the terminal object), a monad $\mathbb{T}$ can be given by a so-called *Kleisli triple* $(T, \eta, -^\dagger)$ consisting of an endomap $T : \mathrm{Ob}(\mathbf{C}) \to \mathrm{Ob}(\mathbf{C})$;

a family of morphisms $\eta_A : A \to TA$; and an operator sending any $f : A \to TB$ to $f^\dagger : TA \to TB$, called *Kleisli lifting*. These data are subject to the equations

$$\eta_A^\dagger = \mathrm{id}, \qquad f^\dagger \eta_A = f, \qquad (f^\dagger g)^\dagger = f^\dagger g^\dagger.$$

Intuitively, T is used to form a type of computations TA with outcomes of type A; Therefore η_A injects a value into a trivial computation returning that value and $-^\dagger$ lifts a morphism $f : A \to TB$ over values to a morphism $f^\dagger : TA \to TB$ over computations.

It can be shown that T from the definition of the Kleisli triple can be lifted to an endofunctor and η_A is natural in A, which leads us to an equivalent definition of a monad, customary in the category theory: A monad $\mathbb{T}$ is given by an endofunctor T and two natural transformations $\eta : Id \to T$ and $\mu : T^2 \to T$ called *unit* and *multiplication* respectively, subject to standard commutativity diagrams (see e.g. [16]). A monad $\mathbb{T}$ is *strong* if it has *strength*, a natural transformation $\tau_{A,B} : A \times TB \to T(A \times B)$, which is well-behaved w.r.t. the monad structure [19]. Strength is a natural technical condition ensuring that the monad is suitable for multivariable computations. If strength exists then it is unique up to natural isomorphism. Moreover, for Cartesian closed categories strength is equivalent to enrichment [15]; Therefore, every monad over **Set** is strong. We agree that all the monads in remainder of this paper are strong.

The definition of a monad by a Kleisli triple naturally suggests the idea of the *Kleisli category* $\mathbf{C}_\mathbb{T}$, the category whose objects are the same as those of $\mathbf{C}$ and whose morphisms from $\mathbf{C}_\mathbb{T}(A, B)$ are those from $\mathbf{C}(A, TB)$; The identity morphisms of $\mathbf{C}_\mathbb{T}$ are thus given by $\eta_A : A \to TA$ and the composition of $f : B \to TC$ with $g : A \to TB$ is the so-called *Kleisli composition*: $f \diamond g = f^\dagger g$. Morphisms of $\mathbf{C}_\mathbb{T}$ are sometimes called *Kleisli morphisms*. In terms of computational effects, the Kleisli category is precisely the category of side-effecting morphisms w.r.t. $\mathbf{C}$. In particular, one can "include" $\mathbf{C}$ in $\mathbf{C}_\mathbb{T}$ by postcomposing every morphism $f : A \to B$ with η_B. The obtained functor has a right adjoint and as such gives rise to a monad, which happens to be the original monad $\mathbb{T}$. This provides one of the two extremal solutions to the question, if any monad is generated by an adjunction. The other solution, known as *Eilenberg-Moore construction*, is obtained by involving the so-called *Eilenberg-Moore category* $\mathbf{C}^\mathbb{T}$, i.e. the category of algebras of the monad $\mathbb{T}$. Such algebras are simply morphisms $f : TA \to A$ satisfying certain coherence conditions [16]. The Kleisli category $\mathbf{C}_\mathbb{T}$ can be faithfully embedded into $\mathbf{C}^\mathbb{T}$ as the subcategory of free algebras, which gives rise to a functor $\mathcal{F}^\mathbb{T}_{\mathrm{EM}} : \mathbf{C} \to \mathbf{C}^\mathbb{T}$ that has a right adjoint $\mathcal{G}^\mathbb{T}_{\mathrm{EM}} : \mathbf{C}^\mathbb{T} \to \mathbf{C}$ and therewith again yields the original monad $\mathbb{T}$.

Plain dualization of the monadic universe brings about *comonads* and a whole bunch of associated concepts, which we do not elaborate here (but see [30]). We note, however, that the Eilenberg-Moore category $\mathsf{Coalg}_\mathbb{K}(\mathbf{C})$ of a comonad $\mathbb{K}$ consists of Eilenberg-Moore $\mathbb{K}$-coalgebras. The latter category is not the same as $\mathsf{coalg}_K(\mathbf{C})$ with K being the functorial part of $\mathbb{K}$, but closely related to it. In particular, $\mathsf{Coalg}_\mathbb{K}(\mathbf{C})$ is a full subcategory of $\mathsf{coalg}_K(\mathbf{C})$.

Examples of computationally relevant monads include the following.

Example 3 (Computational monads). We assume that the category $\mathbf{C}$ possesses sufficient structure to makes sense of what follows.

- *Exception monad:* $TX = X + E$ where E is an object of exceptions. One obtains the *partiality monad* by taking $E = 1$.
- *Powerset monad:* $TX = \mathcal{P}X$ where $\mathcal{P}$ is a covariant powerset functor. Some variants of it are $\mathcal{P}^\star$ for non-empty subsets; $\mathcal{P}_\kappa$ for subsets of cardinality strictly less then κ with a regular cardinal κ (e.g. finite powerset $\mathcal{P}_\omega$); etc.
- *Multiset monad:* $TX = \{m : X \to \mathbb{N} \mid |\mathrm{supp}(m)| < \omega\}$ where $\mathbb{N}$ stands for the set of natural numbers including 0.
- *Subdistribution monad:* $TA = \left\{ d : A \to [0,1] \mid \sum_{x \in A} d(x) \leq 1 \right\}$ where d ranges over subprobability distributions, deviating from probability distributions in that they might sum up to less than 1.
- *Store monad:*[1] $TX = (X \times S)^S$ where S is a global store often identified with V^L, a space of maps from locations L to values V.

Often, different effects can be combined, e.g. the nondeterministic store monad $TX = \mathcal{P}(X \times S)^S$, the Java monad [12] $TA = S \rightharpoonup S \times A + E \times A$, etc.

For certain monads, especially those involved in coalgebraic trace semantics, it is customary to consider their presentation, based on algebraic theories. For example, the finite powerset monad $\mathcal{P}_\omega$ can be considered as generated by the algebraic theory featuring two operations $\emptyset$ and $+$, subject to the axioms of bounded semi-latices (i.e. semi-lattices with a bottom element). The object $\mathcal{P}_\omega X$ is then identified with the free bounded semi-lattice over X. More recently, it has been shown how the store monad can be presented by an algebraic theory, which example we consider in more detail.

Consider the store S of the form V^n with natural n. This corresponds to a computational model of n locations that can be filled with the elements of V. Let the underlying category be **Set** and assume, for simplicity, V to be finite. We consider the family of operations:

$$lookup_i : X^V \to X \qquad\qquad update_{i,v} : X \to X$$

parametrized by $i \in n$ and $v \in V$. The intuitive meaning of them is as follows: $lookup_i(x_1, \ldots, x_V)$ reads the location i and depending on the value discovered, returns the corresponding argument; $update_{i,v}(x)$ updates the location i with v and returns x. These two operations can be viewed as elementary commands, while the terms composed from them can be viewed as programs over these commands. A complete set of axioms for *lookup* and *update* is provided in [24]; Moreover the following result is proven.

Proposition 4. [24] *The axioms for lookup and update from [24] identify an equational theory generating the store monad over* V^n.

The approach to computational effects sketched above is developed mainly in a series of work by Plotkin and Power [21, 22, 23] and scales rather well as far as to any strong monad over a complete and cocomplete Cartesian closed **C** [5].

[1] We avoid the term "state monad" to prevent confusion with coalgebraic states.

Various forms of memory organization naturally call for various algebraic theories. We conclude this section by giving two novel characterisation results in spirit of Proposition 4 for stack-like stores. Analogously to *lookup* and *update* we introduce $pop : X^{S+1} \to X$ and $push_i : X \to X$, with $S = \{s_1, \ldots, s_n\}$ denoting a finite alphabet of stack elements. The idea is as follows:

- $pop(x_1, \ldots, x_n, y)$ results in y if the stack is empty; Otherwise it removes the top element of it and results in x_i where $s_i \in S$ is the removed stack element.
- $push_i(x)$ pushes $s_i \in S$ onto the stack and returns x.

We postulate the following axioms:

$$push_i(pop(x_1, \ldots, x_n, y)) = x_i \tag{1}$$
$$pop(push_1(x), \ldots, push_n(x), x) = x \tag{2}$$
$$pop(x_1, \ldots, x_n, pop(y_1, \ldots, y_n, z)) = pop(x_1, \ldots, x_n, z) \tag{3}$$

The *stack monad* is the submonad of the store monad $(- \times S^*)^{S^*}$ with every TX consisting of those $\langle r, t\rangle : S^* \to X \times S^*$ for which there is a natural $n \geq 0$ such that whenever $w, u \in S^*$ with $|u| > n$,

$$t(u \cdot w) = t(u) \cdot w \qquad\qquad r(u \cdot w) = t(u)$$

In other words, TX captures exactly those operations $\langle r, t\rangle$, which may only access the stack up to a certain depth n where n is associated with $\langle r, t\rangle$ and independent from the stack content.

Proposition 5. *The algebraic theory of $push_i$ and pop with the axioms* (1)–(3) *is equivalent to the stack monad over* **Set**.

A more advanced theory accommodating stacks together with finite nondeterminism will be needed in Section 7 in order to give a coalgebraic account of push-down automata. It is obtained by adding binary $+$ and nullary $\varnothing$, to the signature of operations $\{push_i \mid i \leq n\} \cup \{pop\}$ used above and by completing the axioms (1)–(3) with the following new identities:

$$(x + y) + z = x + (y + z) \qquad x + y = y + x \qquad x + \varnothing = x + x = x \tag{4}$$

$$pop(\varnothing) = \varnothing \tag{5}$$
$$push(\varnothing, \ldots, \varnothing, \varnothing) = \varnothing \tag{6}$$
$$pop(x + x') = pop(x) + pop(x') \tag{7}$$
$$push(x_1 + x'_1, \ldots, x_n + x'_n, y + y') = push(x_1, \ldots, x_n, y) + push(x'_1, \ldots, x'_n, y') \tag{8}$$

Here (4) are the obvious axioms of bounded semi-lattices whereas the laws (5)–(8) express commutativity of stack operations and nondeterminism over each other as computational effects. In other words, the theory for (1)–(8) is the tensor product of the theory for stacks (1)–(3) and the theory for finite nondeterminism (4) (see [10] for more details). The corresponding *nondeterministic*

stack monad is the submonad of the nondeterministic store monad $\mathcal{P}_\omega(- \times S^*)^{S^*}$ so that every TX consists of those $f : S^* \to \mathcal{P}_\omega(X \times S^*)$ for which there is a natural $n \geq 0$ such that for all $w, u \in S^*$ whenever $|u| > n$,

$$f(u \cdot w) = \{\langle x, v \cdot w\rangle \mid \langle x, v\rangle \in f(u)\}.$$

Proposition 6. *The algebraic theory of push, pop, $\varnothing$ and $+$ with the axioms* (1)–(8) *is equivalent to the nondeterministic stack monad over* **Set**.

The latter proposition is essentially a consequence of the analysis of the structure of *powermonads* given in [8], which are tensor products of monads with variants of the powerset monad.

4 Trace Semantics via Observation

We consider here a monad $\mathbb{T}$ and a pair of endofunctors F, G over a category $\mathbf{C}$ having sufficient structure to interpret the constructions being discussed.

As indicated previously, process-like bisimilarity is rather well captured by coalgebraic behavioural equivalence. Proper treatment of *trace equivalence*, however, turns out to be a rather more delicate issue and a seemingly prevailing approach to tackle it, originated in [11], is to assume F to be a composite functor and capture trace equivalence using the same finality argument in another category with a functor obtained as a syntactic component of F. One concrete implementation of this idea is to assume F to be of the form TG and obtain the trace semantics in the Kleisli category $\mathbf{C}_\mathbb{T}$ for a lifting $G_\mathbb{T}$ of G, equivalently given by a distributivity law $\pi : G\mathbb{T} \to \mathbb{T}G$ [9]. A more recent approach, pursued in [13], follows similar lines with TG replaced by GT, the Kleisli category $\mathbf{C}_\mathbb{T}$ by the Eilenberg-Moore category $\mathbf{C}^\mathbb{T}$ and the distributive law $G\mathbb{T} \to \mathbb{T}G$ by a distributive law $\mathbb{T}G \to G\mathbb{T}$. We stick to this latter style of semantics further on.

Definition 7 ($\mathbb{T}$-distributivity). We call G $\mathbb{T}$-*distributive* if there is a distributive law $\pi : \mathbb{T}G \to G\mathbb{T}$, in which case we call π a $\mathbb{T}$-*distributivity* of G.

It is well-known that $\mathbb{T}$-distributivities of G bijectively correspond to liftings $G^\mathbb{T}$ of G to $\mathbf{C}^\mathbb{T}$, which can be expressed by the following commutative diagram

$$\begin{array}{ccc} \mathbf{C}^\mathbb{T} & \xrightarrow{G^\mathbb{T}} & \mathbf{C}^\mathbb{T} \\ {\scriptstyle \mathcal{G}^\mathbb{T}_{\mathrm{EM}}}\downarrow & & \downarrow{\scriptstyle \mathcal{G}^\mathbb{T}_{\mathrm{EM}}} \\ \mathbf{C} & \xrightarrow{G} & \mathbf{C} \end{array} \tag{9}$$

Moreover, as shown in [13], a GT-coalgebra in $\mathbf{C}$ gives rise to a $G^\mathbb{T}$-coalgebra in $\mathbf{C}^\mathbb{T}$ and thus traces of the original coalgebra can be identified as behaviours of the lifted $G^\mathbb{T}$-coalgebra in $\mathbf{C}^\mathbb{T}$. Finally, application of the forgetful functor $\mathcal{G}^\mathbb{T}_{\mathrm{EM}} : \mathbf{C}^\mathbb{T} \to \mathbf{C}$ yields a trace semantics in the original category for, as turns out, $\mathcal{G}^\mathbb{T}_{\mathrm{EM}}$ it sends the final $G^\mathbb{T}$-coalgebra exactly to a final G-coalgebra.

Example 8. [13] A standard example of the presented scenario is given by NA. The language functor L_A is $\mathcal{P}_\omega$-distributive with $\pi : \mathcal{P}_\omega L_A \to L_A\mathcal{P}_\omega$, for every $p \in \mathcal{P}_\omega L_A$ given by

$$\mathsf{pr}_1(\pi(p)) = \exists m.\, \langle \top, m\rangle \in p, \qquad \mathsf{pr}_2(\pi(p)) = \lambda a.\, \{m(a) \mid \langle b, m\rangle \in p\}.$$

It can now be seen by coinduction that the induced transformation of $L_A\mathcal{P}_\omega$-coalgebras to L_A-coalgebras implements automata determinization.

As shown in [13], involving more sophisticated forms of nondeterminism, such as captured by multisets and subdistributions, allows for a treatment of somewhat less standard kinds of machines, such as generative probabilistic systems and weighted automata.

We extend the outlined framework just one step further by introducing the core concept of this paper.

Definition 9 (Real-time $\mathbb{T}$-observer). Given a $\mathbb{T}$-distributive functor G and an endofunctor F, a *real-time $\mathbb{T}$-observer for F*, $F \to G\mathbb{T}$ is given by a natural transformation $\delta : F \to GT$. Disregarding the reference to the original functor F, we call any natural transformations in this format a *generic $\mathbb{T}$-observer*.

The term "real-time" here refers to the fact that the observer depends only on the coalgebra functor, and hence the observation is always performed stepwise with no way to delay until the next step. We elaborate on this further in Section 5.

Essentially, given an observer $F \to G\mathbb{T}$ we can transform any F-coalgebra to a $G\mathbb{T}$-coalgebra and then simply apply the generalised powerset construction to the result. However, we prefer to spell the details as they will be relevant for the remaining presentation.

Note that an observer $\delta : F \to G\mathbb{T}$ gives rise to a natural transformation $\delta_* : TF \to GT$ by the following composition:

$$\delta_* : TF \xrightarrow{T\delta} TGT \xrightarrow{\pi T} GT^2 \xrightarrow{G\mu} GT \tag{10}$$

where π is the $\mathbb{T}$-distributivity of G. The intended use of $\mathbb{T}$-observers is to provide trace semantics for a coalgebra $f : X \to FX$ according to diagram

$$\begin{array}{ccccc} X & \xrightarrow{f} & FX & & \\ {\scriptstyle\eta}\downarrow & & \downarrow{\scriptstyle\eta} & & \\ TX & \xrightarrow{Tf} & TFX & \xrightarrow{\delta_*} & GTX \\ {\scriptstyle\hat{s}}\downarrow & \overset{s}{\cdots\cdots\nearrow} & & & \downarrow{\scriptstyle G\hat{s}} \\ \nu G & & \xrightarrow{\iota} & & G\nu G \end{array} \tag{11}$$

Here, ι is the final coalgebra structure, $s = \delta_*(Tf)$ and $\hat{s}$ is the universal arrow induced by s. Given some $a : 1 \to X$ we call $\hat{s}\eta a$ the *δ-trace* of f at a. This naturally generalizes the construction from [13] by taking $F := GT$ and $\delta := (G\mu)\pi$. The DA and NA examples can now be treated on the same footing.

Example 10. Let $FX = 2 \times (\mathcal{P}_\omega X)^A$ be the NA-functor as in Example 8. The generalised powerset construction from [13] amounts to the identity $\mathcal{P}_\omega$-observer $F \simeq L_A \mathcal{P}_\omega$ and therefore yields the expected trace semantics w.r.t. L_A.

In case of the DA-functor $FX = 2 \times X^A$ we take as the $\mathcal{P}_\omega$-observer $L_A(\eta_X) : 2 \times X^A \to L_A \mathcal{P}_\omega X$ where η_X is the unit of $\mathcal{P}_\omega$ and obtain the conventional trace semantics, which does, of course, coincide with the canonical behavioural equivalence.

Proposition 11. *To give an observer $\delta : F \to G\mathbb{T}$ is the same as to give a natural transformation for the following pasting diagram:*

$$\begin{array}{ccc} \mathbf{C} & \xrightarrow{F} & \mathbf{C} \\ {\scriptstyle \mathcal{F}^{\mathbb{T}}_{\mathrm{EM}}}\downarrow & \swarrow & \downarrow{\scriptstyle \mathcal{F}^{\mathbb{T}}_{\mathrm{EM}}} \\ \mathbf{C}^{\mathbb{T}} & \xrightarrow{G^{\mathbb{T}}} & \mathbf{C}^{\mathbb{T}} \end{array} \tag{12}$$

5 Lookahead Observers

We introduced the notion of a $\mathbb{T}$-observer as a fairly modest generalization of an existing device. Here, we would like to argue that this notion is in fact derivable from some rather general category-theoretic considerations.

Let us consider a pair of categories $\mathbf{C}$, $\mathbf{D}$, a pair of endofunctors F, G over them and the corresponding categories of coalgebras $\mathsf{coalg}_F(\mathbf{C})$, $\mathsf{coalg}_H(\mathbf{D})$. The idea is: objects of $\mathsf{coalg}_F(\mathbf{C})$ represent original systems of interest, while objects of $\mathsf{coalg}_H(\mathbf{D})$ represent systems of observable behaviours of the latter. It appears reasonable to capture such kind of an observation scenario by a pair of functors $V : \mathbf{C} \to \mathbf{D}$ and $\widehat{V} : \mathsf{coalg}_F(\mathbf{C}) \to \mathsf{coalg}_H(\mathbf{D})$ such that $\widehat{V}$ is a lifting of V, in other words the diagram

$$\begin{array}{ccc} \mathsf{coalg}_F(\mathbf{C}) & \xrightarrow{\widehat{V}} & \mathsf{coalg}_H(\mathbf{D}) \\ {\scriptstyle U_F}\downarrow & & \downarrow{\scriptstyle U_G} \\ \mathbf{C} & \xrightarrow{V} & \mathbf{D} \end{array} \tag{13}$$

with U_F, U_G being the evident forgetful functors, commutes.

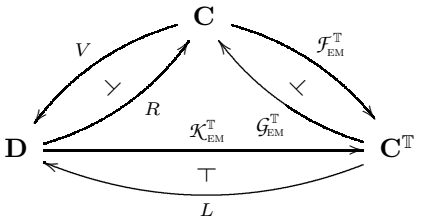
Suppose U_F has a right adjoint R_F. This gives rise to a *cofree comonad F^∞ on* F, explicitly, $F^\infty X = \nu\gamma.(F\gamma \times X)$ (see e.g. [30]). Since $\mathsf{coalg}_F(\mathbf{C})$ is isomorphic to the Eilenberg-Moore category $\mathsf{Coalg}_{F^\infty}(\mathbf{C})$, an endofunctor $\widehat{V}$ on $\mathsf{coalg}_F(\mathbf{C})$ is equivalently an endofunctor on $\mathsf{Coalg}_{F^\infty}(\mathbf{C})$. Furthermore, we have

Lemma 12. *Provided F^∞ exists, to give a lifting $\widehat{V}$ for V is the same as to give a natural transformation*

$$VF^\infty \to HV. \tag{14}$$

Let us assume that the functor $V : \mathbf{C} \to \mathbf{D}$ in (13) has a right adjoint $R : \mathbf{D} \to \mathbf{C}$ and let us therefore obtain a monad $T = RV$ on $\mathbf{C}$. Let $\mathcal{K}^{\mathbb{T}}_{\mathrm{EM}} : \mathbf{D} \to \mathbf{C}^{\mathbb{T}}$ be the

induced comparison functor. Suppose moreover that $\mathcal{K}^{\mathbb{T}}_{\mathrm{EM}}$ has a left adjoint L so that, in summary, we have the following picture:



where $\mathcal{F}^{\mathbb{T}}_{\mathrm{EM}} = \mathcal{K}^{\mathbb{T}}_{\mathrm{EM}} V$ and $R = \mathcal{G}^{\mathbb{T}}_{\mathrm{EM}} \mathcal{K}^{\mathbb{T}}_{\mathrm{EM}}$. Since a composition of left (right) adjoints is again a left (right) adjoint and a left (right) adjoint to a functor is uniquely defined, also $V = L\mathcal{F}^{\mathbb{T}}_{\mathrm{EM}}$ and $\mathcal{G}^{\mathbb{T}}_{\mathrm{EM}} = RL$.

Theorem 13. *In the situation* (13)*, suppose (i) U_F has a right adjoint R_F, (ii) V has a right adjoint R and (iii) the comparison functor $\mathcal{K}^{\mathbb{T}}_{\mathrm{EM}}$ a has a left adjoint L. Let H be an endofunctor $\mathbf{D} \to \mathbf{D}$ such that $GR \simeq RH$ with some $G : \mathbf{C} \to \mathbf{C}$. Then*

1. *$\mathcal{K}^{\mathbb{T}}_{\mathrm{EM}} HL : \mathbf{C}^{\mathbb{T}} \to \mathbf{C}^{\mathbb{T}}$ is a lifting of G to $\mathbf{C}^{\mathbb{T}}$ and thus G is $\mathbb{T}$-distributive;*
2. *the rule sending any $\alpha : F^\infty \to GT$ to*

$$VF^\infty \xrightarrow{V\alpha} VGRV \xrightarrow{V\rho V} VRHV \xrightarrow{\epsilon HV} HV$$

where ρ is the isomorphism $HV \simeq VG$ and ϵ is the counit of the adjunction, determines a one-to-one correspondence between $\mathbb{T}$-observers $F^\infty \to G\mathbb{T}$ and natural transformations (14)*.*

Remark 14. While the conditions (i) and (iii) of Theorem 13 seem to be relatively mild (e.g. (i) is fulfilled when F is accessible, (iii) is fulfilled when $\mathbf{D}$ has coequalizers [1]), condition (ii) is essential. A somewhat unexpected source of situations (13) satisfying (ii) are *logical connections* [20] used in coalgebraic modal logic. These can be related to our view by instantiating $\mathbf{D}$ with $\mathbf{E}^{op}$ for some $\mathbf{E}$; Coalgebras over $\mathbf{D}$ would then be algebras over $\mathbf{E}$. A detailed analysis of the relation between observers and logical connections is subject to further work.

Theorem 13 inspires the following definition.

Definition 15 (Lookahead $\mathbb{T}$-observer). A *lookahead $\mathbb{T}$-observer for F* is a real-time $\mathbb{T}$-observer for F^∞.

For every F-coalgebra $f : X \to FX$ we can form a F^∞-coalgebra $f' : X \to F^\infty X$ as the final arrow from $\langle f, \mathrm{id}\rangle : X \to FX \times X$ to $F^\infty X$. In order to obtain a δ-trace for f by means of a lookahead observer $\delta : F^\infty \to GT$ we just apply the construction (11) to f' instead of f.

We clarify the distinction between real-time and lookahead observers by a small example inspired by [14].

Example 16 (Infinite streams). Consider the following, perhaps slightly artificial, problem: Given an infinite stream over a set L, calculate the stream obtained by missing out all those maximal finite segments of it, which consist of a specified element $a \in L$, e.g. if the whole stream is an infinite repetition of a then the original stream is returned. The functor $FX = L \times X$ is known to generate infinite streams exactly as we need. A real-time observer would then have the format $L \times \mathrm{Id} \to L \times T$, which is not suitable for it would not give a way to access the next element of the stream as required unless the front one is a.

By comparison, consider a lookahead observer $skip_a : F^\infty \to F$ w.r.t. the identity monad. Note that $F^\infty X \simeq \nu\gamma.\,(F\gamma \times X) \simeq (L \times X)^\infty$, i.e. $F^\infty X$ consists of infinite streams over $L \times X$. Hence we define

$$skip_a[\langle a, x_1\rangle, \langle a, x_2\rangle, \ldots, \langle a, x_n\rangle, \langle b, x_{n+1}\rangle, \ldots] = \langle b, x_{n+1}\rangle,$$
$$skip_a[\langle a, x_1\rangle, \langle a, x_2\rangle, \ldots, \langle a, x_n\rangle, \langle a, x_{n+1}\rangle, \ldots] = \langle a, x_2\rangle.$$

A coalgebra $f : X \to FX$ gives rise to a coalgebra $f' : X \to F^\infty X$ so that a behaviour $[a_1, a_2, \ldots]$ of the original system carried out by a state x_1 maps to a behaviour $[\langle a_1, x_1\rangle, \langle a_2, x_2\rangle, \ldots]$ of the transformed system where the second components of the tuples protocol the intermediate states visited. It is now easy to see that a $skip_a$-trace of f at $x : 1 \to X$ is obtained from the corresponding stream of behaviours as expected.

As noted in [14], examples like $skip_a$, considered as such, carry a little of operational meaning, which fact indicates that restricting to real-time observers can well be a reasonable idea. However, as we shall see, allowing for a lookahead pays off as it immediately enables useful coalgebraic meta-constructions for a small added price.

6 Internal Actions

Consider the language functor $L_{A+E}X = 2 \times X^{A+E}$ where the actions are partitioned into visible A and internal ones E. Applying the standard finality argument produces traces over the whole set $A+E$, which can be undesirable because of the presence of internal actions. A real-time observer can not be helpful to tackle this issue precisely because if we run into an internal action we must drop it and hold up the output until a non-internal one occurs. As one would expect, a suitable lookahead observer carries the necessary effect.

Let us note first that $L^\infty_{A+E}X \simeq \nu\gamma.\,(2 \times X \times \gamma^{A+E}) \simeq (2 \times X)^{(A+E)^*}$. Then we define a lookahead observer $\delta_X : L^\infty_{A+E}X \to \mathcal{P}_{\omega 1}L_A X$ by the expression

$$\delta\langle o, t\rangle = \{\langle o(w), \lambda a.\, t(w \cdot a)\rangle \mid w \in E^*\}$$

Intuitively, we form partial runs of the original automaton over $A + E$ corresponding to the trace prefixes of the form $\tau_1, \tau_2, \ldots, \tau_n, a$ with $\tau_i \in E$, $a \in A$ and for every such run construct a one-step a-transition of the target automaton. As it must, the target automaton becomes highly nondeterministic, hence the

use of the countable powerset functor $\mathcal{P}_{\omega 1}$. Determinization is ensured on the spot by defining a $\mathcal{P}_{\omega 1}$-distributivity for L_A as in Example 8.

It is not difficult to instantiate the presented idea to capture weak bisimilarity of processes by reduction to strong bisimilarity [17]. Unfortunately, unlike the case of linear traces, considered above, it appears to be impossible to eliminate the internal action. This seems to be a consequence of the standard fact that weak bisimilarity fails to be a congruence. However, we can make use of the fact that two processes are weakly bisimilar iff they become strongly bisimilar after saturating their state transitions by adjoining pre- and postfixes of chained internal actions [6, 17]. Informally, this amounts to creating one a-transition in the target system corresponding to a chain of transitions of the original system labelled by τ^n, a, τ^m if $a \neq \tau$, and one internal transition corresponding to a chain of transitions labelled by τ^n. Here, τ^n and τ^m denote finite, possibly empty, chains of the internal action τ.

Let $FX = \mathcal{P}_\omega(X)^{A_\tau}$ be the functor of finitely-branching labelled transition systems over finite $A_\tau = A \cup \{\tau\}$ [26]. Note that $F^\infty X \simeq \nu\gamma.\,(\mathcal{P}_\omega(\gamma)^{A_\tau} \times X)$. We define a lookahead observer $\pi_X : \nu\gamma.\,(\mathcal{P}_\omega(\gamma)^{A_\tau} \times X) \to \mathcal{P}_{\omega 1}(X)^{A_\tau}$ for F w.r.t. the identity monad by expression:

$$\pi(t_0)(a) = \bigcup\{\mathsf{pr}_2(t_n)(a_n) \mid t_1 \in \mathsf{pr}_1(t_0)(a_0), \ldots, t_n \in \mathsf{pr}_1(t_{n-1})(a_{n-1})\}$$

where the union is taken over all sequences $a_1, \ldots, a_n$ of the form $\tau, \ldots, \tau, a, \tau, \ldots, \tau$ if $a \neq \tau$ and $\tau, \ldots, \tau$ is $a = \tau$.

The presented construction is the most straightforward one. As a result, the obtained transition system receives an enormous number of junk τ-transitions. A further optimisation for practical purposes should not be difficult, but would demand for involving more sophisticated versions of the observer.

7 Push-Down Automata

We treat push-down automata analogously to NA by using the functor $FX = \mathcal{P}_\omega(S^*) \times \mathcal{P}_\omega(X \times S^*)^{A \times S}$ where A stands for the input alphabet and S stands for the set of stack symbols. Given a finite set X, a coalgebra $\langle o, t\rangle : X \to FX$ captures the automaton carrying the following data:

Finite Set of Accepting Configurations $Acc \subseteq X \times S^*$ consisting of all such pairs $\langle x, w\rangle \in X \times S^*$ that $o(x)(w) = 1$; Common possible choices for Acc include [25]: $\{\langle x, w\rangle \mid x \in Fin\}$, $\{\langle x, w\rangle \mid w = \epsilon\}$, $\{\langle x, w\rangle \mid x \in Fin, w = \epsilon\}$, and $\{\langle x, w\rangle \mid w = sw', s \in S'\}$ where $Fin \subseteq X$ is a distinguished set of final states, $S' \subseteq S$ is a distinguished set of stack symbols, and ϵ denotes the empty stack.

Transition Function $X \times A \times S \to \mathcal{P}_\omega(X \times S^*)$ obtained by uncurrying t; Here, the elements of an input triple $\langle x, a, s\rangle$ refer to a current state x, an input alphabet symbol a and the current top stack symbol s, subject to removal. The outcome of the transition function is a finite set of pairs $\langle x', w\rangle$ where x' is the new state and w is a string of elements to be pushed onto the stack.

As already noticed in [28], transitions of a push-down automaton can be considered as a nondeterministic side-effecting function w.r.t. the set of stacks S^*

as underlying store, which suggests using the monad $\mathcal{P}_\omega(X \times S^*)^{S^*}$. Here, we propose a nondeterministic version of the stack monad from Section 3 as a better choice, for—as we have shown in Proposition 5—it is directly generated by the stack operations. Let $\mathbb{T}$ be that monad henceforth.

Note that T does not appear as a subexpression of the expression defining F. More importantly, operations encoded by F are only commands to be applied to a stack and not ready stack transformers.

Due to an isomorphism $FX \simeq \mathcal{P}_\omega(S^*) \times \mathcal{P}_\omega(X \times (S \times A \times S^*))$, behaviours of F can be understood as finitely-branching trees, modulo bisimilarity, whose nodes are annotated by finite sets of stacks and whose branches are labelled with commands of either of two forms: $(a; s/s_1 \dots s_n)$, $(s/s_1 \dots s_n)$.

Let $\alpha_X : \mathcal{P}_\omega(X \times S^*)^S \to TX$ be the natural transformation defined by the clauses

$$\alpha(f)(\epsilon) = \emptyset, \qquad \alpha(f)(s \cdot w) = \{\langle x, u \cdot w\rangle \mid \langle x, u\rangle \in f(s)\}.$$

This gives rise, in an obvious way, to a real-time $\mathbb{T}$-observer $\delta_X : FX \to \mathcal{P}(S^*) \times (TX)^A$ for F. Finally, we endow $\mathcal{P}(S^*) \times -^A$ with a $\mathbb{T}$-distributivity $\pi_X : T(\mathcal{P}(S^*) \times X^A) \to \mathcal{P}(S^*) \times (TX)^A$ by the equations:

$$\mathsf{pr}_1(\pi(p)) = \{w \in S^* \mid \langle w'', m, w'\rangle \in p(w), w'' \cap w' \neq \emptyset\},$$
$$\mathsf{pr}_2(\pi(p)) = \lambda a.\, \lambda w.\, \{m(a) \mid \langle b, m\rangle \in p(w)\},$$

which is in a perfect correspondence with [28]. The resulting traces are the elements of the final coalgebra $\nu\gamma.\, (\mathcal{P}(S^*) \times X^A) \simeq \mathcal{P}(A^*)^{S^*}$ and are indeed maps from the set of initial stack values to languages over A.

According to the standard definition (e.g. [25]), A has form $A' + 1$ where the adjoined element can be viewed as a special symbol for internal transitions (otherwise the automaton is called *real-time* and captures precisely non-empty context-free languages over A [25]), which results in the undesirable effect of having the internal symbol in the traces. A solution to that would be to involve a lookahead observer as in Section 6, which we do not spell out here.

8 Reactive Programs with Generic Side-Effects

Reactive coalgebra-based systems over generic side-effects encapsulated by a monad appear in the literature in the form of coalgebraic component-based frameworks [3] or as generic calculi for side-effecting processes [7]. Here, we sketch the perspectives of defining observation-based semantics for systems of this kind.

Let (for simplicity) $\mathbf{C} = \mathbf{Set}$ and consider coalgebras of the form

$$f : X \to T(O \times X)^I \tag{15}$$

where T is a functorial part of a monad $\mathbb{T}$, capturing some generic side-effect, I is an input type, O is an output type. One way to look at a system of this kind is as a generalized Mealy machine [28]—the latter would be obtained by instantiating $\mathbb{T}$ with the identity monad.

Note that $\nu\gamma.\,(O \times \gamma)^I \simeq \nu\gamma.\,(O^I \times \gamma^I) \simeq (O^I)^{I^*} = O^{I^+}$ where I^+ stands for nonempty lists over I. It is easy to see by induction that the function space $I^+ \to O$ can be considered as a subspace of stream transformers $I^\omega \to O^\omega$ formed by so-called *causal maps* [27]: a map $t : I^\omega \to O^\omega$ is *causal* if $t(s)(n) = t(s')(n)$ whenever $s(i) = s'(i)$ for all $i \leq n$; In other words, the n-th element of the output stream $t(s)$ depends only on the elements of s in the positions from the first one up to the n-th. The latter definition of causality is the standard one, however, the former one is more suitable for our generalization, which is as follows.

Definition 17 ($\mathbb{T}$-causality). Given a strong monad $\mathbb{T}$ over a category $\mathbf{C}$ with finite products and coproducts such that any initial algebra $X^* = \mu\gamma.\,(1 + X \times \gamma)$ exists, we call a morphism $t : I^* \to TO^*$ $\mathbb{T}$-*causal* if the diagram

$$\begin{array}{ccc} I^* & \xrightarrow{\quad t \quad} & TO^* \\ {\scriptstyle \iota_I}\downarrow & & \downarrow{\scriptstyle T\iota_O} \\ 1 + I \times I^* & & T(1 + O \times O^*) \\ {\scriptstyle \mathrm{id} + \mathsf{pr}_2}\downarrow & & \downarrow{\scriptstyle T(\mathrm{id} + \mathsf{pr}_2)} \\ 1 + I^* & \xrightarrow{[\eta\,\mathsf{inl}, (T\,\mathsf{inr})t]} & T(1 + O^*) \end{array}$$

commutes where $\iota_X : X^* \to 1 + X \times X^*$ is the initial algebra structure on X^*.

It can be readily verified by induction that for $\mathbf{C} = \mathbf{Set}$ with $\mathbb{T}$ being the identity monad $\mathbb{T}$-causality agrees with the standard definition.

We now proceed with defining a trace semantics for coalgebras of type (15). Let $\mathbb{R}$ be the monad with the functorial part $RX = T(O^* \times X)$ and monadic operations induced by the obvious monoidal structure on O^*. Then we define a real-time $\mathbb{R}$-observer $\delta_X : T(O \times X)^I \to TO^* \times RX^I$ and an $\mathbb{R}$-distributivity $\pi_X : R(TO^* \times X^I) \to TO^* \times RX^I$ by the expressions

$$\delta(t) = \langle \eta\alpha, \lambda x.\, T(\beta \times \mathrm{id})(t(x)) \rangle, \tag{16}$$

$$\pi(t) = \Big(\mathsf{let}\, r = T(\gamma \times \mathrm{id})\mu(T\tau)t \;\; \mathsf{in}\, \langle (T\,\mathsf{pr}_1)r, \lambda x.\, T(\mathrm{id} \times at(x))r \rangle\Big) \tag{17}$$

where $\alpha : 1 \to O^*$, $\beta : O \to O^*$, $\gamma : O^* \times O^* \to O^*$ are the obvious operations for creating the empty list, forming a single-element list from a given element and for list concatenation correspondingly; τ is the strength of $\mathbb{T}$; $at(x) = \lambda f.\, f(x)$; the intermediate value r has $T(O^* \times X^I)$ as the output type. This induces traces for (15) as elements of $\nu\gamma.\,(TO^* \times \gamma^I) \simeq (TO^*)^{I^*}$.

The idea behind the monad $\mathbb{R}$ is to collect the outputs from O while composing the computational effects between iterations. The strength τ of the monad $\mathbb{T}$ plays a crucial role in this process, as it allows for propagating the output values downwards through the layers of $\mathbb{T}$-computations. We can now relate to $\mathbb{T}$-causality as follows.

Proposition 18. *Traces induced by* (16)–(17) *are* $\mathbb{T}$*-causal.*

9 Conclusions

We have laid down the foundations of a generic monad-based notion of observation for systems represented by coalgebras. This notion establishes a bridge

between an implementation of a system and its observable behaviour, and therefore provides a basic abstraction idiom for studying systems from the most general perspective. Whereas, from the technical point of view, our development is only a mild modification of the existing machinery, as we have shown, our notion of observation has a universal character—under reasonable assumptions, it arises from a fairly basic observation scenario. Every observer in our framework comes with an associated generalized notion of trace, which can vary over a large spectrum of equivalences refining the canonical notion of bisimilarity of the original system. We have shown how the introduced notion of observation allows for a smooth and concise treatment of such traditionally delicate examples as internal actions, weak bisimilarity and pushdown automata. As a further improvement of the latter case we have presented two versions of a stack monad and operational characterizations for them in the spirit of Plotkin and Power.

Related Work. This work descends from [13] as well as from the more recent [28] where the generalized powerset construction was introduced. Natural transformations of the form $TF \to GT$, which one can treat as a form of a $\mathbb{T}$-observer occasionally appear in [13] for sakes of relating Kleisli and Eilenberg-Moore styles of semantics. Insufficient expressivity of the standard distributive law argument has been acknowledged in [29] in the specific case of the finitary subdistribution monad D_ω combined with the functor $1 + A \times -$: $D_\omega(1 + A \times X)$ only embeds into $[0, 1] \times (D_\omega X)^A$ (and thus can be naturally regarded as an observer), unlike the analogous case with $\mathcal{P}_\omega$ instead of D_ω when an isomorphism would take place.

Future Work. Format constraints applied to this note do not give a chance to develop the presented theory to a sufficient extent. We would like to develop this further along the following lines.

– *Formal languages and machines.* One objective of this paper was to improve the coalgebraic treatment of push-down automata. We believe that an analogous treatment of Turing machines should not be difficult, except that instead of classical Turing machines one should involve *reactive Turing machines* presented in [2] as a more coalgebra-friendly concept.

– *Coalgebraic modal logic.* As indicated in Remark 14, there is a technical relation between observation scenarios (13) and logical connections. In view of the classical relation between testing and observational semantics this may be not a coincidence and should be studied in detail; The notion of a lookahead observer may also suggest a way for a logical characterisation of the weak bisimulation, by involving F^∞-coalgebras derived from F-coalgebras.

– *Rational fixpoints of functors.* The notion of a regular language is known to be nicely captured coalgebraically by a *rational fixpoint* ρL_A of the language functor L_A by Milius and collaborators (see e.g. [4]). As expected, ρL_A sits inside the greatest fixpoint νL_A of all formal languages. The question is, if the theory of rational fixpoints can be extended (somehow) to other kinds of machines under the presented observational treatment, e.g. to push-down automata.

– *Notion of observation.* We intend to study compositions of observation scenarios (13), such as sequential and parallel ones, and their impact on the monads underlying the corresponding observers. As we have seen, the notion of trace

equivalence, induced by an observer can be as fine as the usual bisimilarity. We anticipate the introduction of a class of observers capturing linearity of traces and a construction for generating such observers from a functor and a monad in a universal manner analogously to forming tensors of monads with powersets [8]. Theorem 13 indicates that it might be reasonable to relax the definition of an observer by rebasing it on pairs of adjoint functors and dropping $\mathbb{T}$-distributivity.

Acknowledgements. The author wishes to thank Stefan Milius for a brief but definite update on distributive laws for comonads. Thanks to Alexandra Silva, Daniel Hausmann and anonymous referees for their valuable feedback.

References

[1] Adámek, J., Herrlich, H., Strecker, G.: Abstract and concrete categories. John Wiley & Sons Inc., New York (1990)
[2] Baeten, J.C.M., Luttik, B., van Tilburg, P.: Reactive turing machines. In: Owe, O., Steffen, M., Telle, J.A. (eds.) FCT 2011. LNCS, vol. 6914, pp. 348–359. Springer, Heidelberg (2011)
[3] Barbosa, L.S.: Towards a calculus of state-based software components. Journal of Universal Comp. Sci. 9, 891–909 (2003)
[4] Bonsangue, M.M., Milius, S., Silva, A.: Sound and complete axiomatizations of coalgebraic language equivalence. ACM Trans. Comp. Logic 14(1), 7:1–7:7 (2013)
[5] Dubuc, E.: Kan Extensions in Enriched Category Theory. LNM, vol. 145 (1970)
[6] Fiore, M., Cattani, G.L., Winskel, G.: Weak bisimulation and open maps. In: LICS 1999 (1999)
[7] Goncharov, S., Schröder, L.: A coinductive calculus for asynchronous side-effecting processes. In: Owe, O., Steffen, M., Telle, J.A. (eds.) FCT 2011. LNCS, vol. 6914, pp. 276–287. Springer, Heidelberg (2011)
[8] Goncharov, S., Schröder, L.: Powermonads and tensors of unranked effects. In: LICS 2011, pp. 227–236 (2011)
[9] Hasuo, I., Jacobs, B., Sokolova, A.: Generic trace theory. In: CMCS 2006. Elect. Notes in Theor. Comp. Sci., vol. 164, pp. 47–65. Elsevier (2006)
[10] Hyland, M., Plotkin, G., Power, J.: Combining computational effects: Commutativity & Sum. In: TCS 2002, vol. 223, pp. 474–484. Kluwer (2002)
[11] Jacobs, B.: Trace semantics for coalgebras. Electron. Notes Theor. Comput. Sci. 106, 167–184 (2004)
[12] Jacobs, B., Poll, E.: Coalgebras and Monads in the Semantics of Java. Theoret. Comput. Sci. 291, 329–349 (2003)
[13] Jacobs, B., Silva, A., Sokolova, A.: Trace semantics via determinization. In: Pattinson, D., Schröder, L. (eds.) CMCS 2012. LNCS, vol. 7399, pp. 109–129. Springer, Heidelberg (2012)
[14] Klin, B.: Bialgebras for structural operational semantics: An introduction. Theor. Comput. Sci. 412(38), 5043–5069 (2011)
[15] Kock, A.: Strong functors and monoidal monads. Archiv der Mathematik 23(1), 113–120 (1972)
[16] Mac Lane, S.: Categories for the Working Mathematician. Springer (1971)
[17] Milner, R.: Communication and concurrency. Prentice-Hall, Inc., Upper Saddle River (1989)
[18] Moggi, E.: A modular approach to denotational semantics. In: Curien, P.-L., Pitt, D.H., Pitts, A.M., Poigné, A., Rydeheard, D.E., Abramsky, S. (eds.) CTCS 1991. LNCS, vol. 530, pp. 138–139. Springer, Heidelberg (1991)

[19] Moggi, E.: Notions of computation and monads. Inf. Comput. 93, 55–92 (1991)
[20] Pavlovic, D., Mislove, M., Worrell, J.B.: Testing semantics: Connecting processes and process logics. In: Johnson, M., Vene, V. (eds.) AMAST 2006. LNCS, vol. 4019, pp. 308–322. Springer, Heidelberg (2006)
[21] Plotkin, G., Power, J.: Adequacy for algebraic effects. In: Honsell, F., Miculan, M. (eds.) FOSSACS 2001. LNCS, vol. 2030, pp. 1–24. Springer, Heidelberg (2001)
[22] Plotkin, G., Power, J.: Notions of computation determine monads. In: Nielsen, M., Engberg, U. (eds.) FOSSACS 2002. LNCS, vol. 2303, pp. 342–356. Springer, Heidelberg (2002)
[23] Plotkin, G., Power, J.: Algebraic operations and generic effects. Appl. Cat. Struct. 11, 69–94 (2003)
[24] Power, J., Shkaravska, O.: From comodels to coalgebras: State and arrays. In: CMCS 2004. ENTCS, vol. 106, pp. 297–314 (2004)
[25] Rozenberg, G., Salomaa, A. (eds.): Handbook of formal languages. Word, Language, Grammar, vol. 1. Springer-Verlag New York, Inc. (1997)
[26] Rutten, J.: Universal coalgebra: A theory of systems. Theoret. Comput. Sci. 249, 3–80 (2000)
[27] Rutten, J.J.M.M.: Algebraic specification and coalgebraic synthesis of mealy automata. Electr. Notes Theor. Comput. Sci. 160, 305–319 (2006)
[28] Silva, A., Bonchi, F., Bonsangue, M., Rutten, J.: Generalizing determinization from automata to coalgebras. LMCS 9(1) (2013)
[29] Silva, A., Sokolova, A.: Sound and complete axiomatization of trace semantics for probabilistic systems. Electr. Notes Theor. Comput. Sci. 276, 291–311 (2011)
[30] Uustalu, T., Vene, V.: Comonadic notions of computation. Electron. Notes Theor. Comput. Sci. 203(5), 263–284 (2008)