COALGEBRAIC TRACE SEMANTICS VIA FORGETFUL LOGICS

BARTEK KLIN [a] AND JURRIAAN ROT [b]

[a] University of Warsaw
  e-mail address: klin@mimuw.edu.pl

[b] Radboud University, Nijmegen
  e-mail address: jrot@cs.ru.nl

ABSTRACT. We use modal logic as a framework for coalgebraic trace semantics, and show the flexibility of the approach with concrete examples such as the language semantics of weighted, alternating and tree automata, and the trace semantics of generative probabilistic systems. We provide a sufficient condition under which a logical semantics coincides with the trace semantics obtained via a given determinization construction. Finally, we consider a condition that guarantees the existence of a canonical determinization procedure that is correct with respect to a given logical semantics. That procedure is closely related to Brzozowski's minimization algorithm.

1. INTRODUCTION

The theory of coalgebras [31, 14] is a framework of choice to model and study state-based systems at a high level of generality. Coalgebraic methods have been rather successful in modeling branching time behaviour of various kinds of transition systems, with a general notion of bisimulation and final semantics as the main contributions. But in a wide range of settings, from automata theory to verification and model checking, one is interested in the linear time behaviour of systems, such as trace semantics of transition systems or language semantics of automata. Indeed, coalgebraic modeling of linear time behaviour has also attracted significant attention.

However, the emerging picture is considerably more complex: several approaches have been developed whose scopes and connections are not yet fully understood. Here we suggest a new approach which, contrary to previously considered coalgebraic ideas, crucially and explicitly defines trace semantics not by coinduction but by induction, based on techniques from modal logic. This provides a transparent framework that covers new examples such as weighted tree automata, allows us to naturally relate trace semantics to determinization

constructions, and, in certain well-behaved cases, provides canonical determinization and minimization procedures.

To study trace semantics coalgebraically, one usually considers systems whose behaviour type is a composite functor of the form TB or BT, where T represents a branching aspect of behaviour that trace semantics is supposed to "resolve", and B represents the transition aspect that should be recorded in system traces. Typically it is assumed that T is a monad, and its multiplication structure is used to resolve branching. For example, in [30, 12], a distributive law of B over T is used to lift B to the Kleisli category of T, and trace semantics is obtained as final semantics for the lifted functor. Additional assumptions on T are needed for this, so this approach does not work for coalgebras such as weighted automata. On the other hand, in [16, 33] a distributive law of T over B is used to lift B to the Eilenberg-Moore category of T, with trace semantics again obtained as final semantics for the lifted functor. This can be seen as a coalgebraic generalization of the powerset determinization procedure for non-deterministic automata. While it applies to many examples, that approach does not work for systems that do not determinize, such as tree automata. A detailed comparison of these two approaches is in [16].

In this paper, we study trace semantics in terms of modal logic. The basic idea is very simple: we view traces as formulas in suitable modal logics, and trace semantics of a state arises from all formulas that hold for it. A coalgebraic approach to modal logic based on dual adjunctions is by now well developed [29, 20, 17, 22], and we apply it to speak of traces generally. Obviously not every logic counts as a trace logic: assuming a behaviour type of the form BT or TB, we construct logics from arbitrary (but usually expressive) logics for B and special logics for T whose purpose is to resolve branching. We call such logics *forgetful*.

Our approach differs from previous studies in a few ways:

- Trace semantics is obtained not as final semantics of coalgebras, but by initial semantics of algebras. Fundamentally, we view trace semantics as an inductive concept and not a coinductive one akin to bisimulation, although in some well-behaved cases the inductive and coinductive views coincide.
- We do not assume that T is a monad, unless we want to relate our logical approach to ones that do, in particular to determinization constructions.
- Instead of using monad multiplication $\mu\colon TT \Rightarrow T$ to resolve branching, we use a natural transformation $\alpha\colon TG \Rightarrow G$, where G is a contravariant functor that provides the basic infrastructure of logics. In case of nondeterministic systems, T is the covariant powerset functor and G the contravariant powerset, so TT and TG act the same on objects, but they carry significantly different intuitions.
- Thanks to the flexibility of modal logics, we are able to cover examples such as the language semantics of weighted tree automata, that does not quite fit into previously studied approaches, or alternating automata.

Example 1.1. Consider weighted automata over a semiring $\mathbb{S}$. Every state of such an automaton, given an input letter, has a linear combination of successor states, with coefficients in $\mathbb{S}$. The semantics of a weighted automaton with state space X is a function $s^\flat\colon X \to \mathbb{S}^{A^*}$, mapping every state to a weighted language. In our framework, the actual map $s^\flat$ is computed as the transpose of a certain map $s\colon A^* \to \mathbb{S}^X$, which arises by induction on words. This computation is determined by a certain (rather simplistic) *forgetful logic*; the formulas of this logic are just the words in A^*. In this example, it is crucial to distinguish between the branching structure of automata, modeled by linear combinations of states, and the

functor $\mathbb{S}^-$ that allows arbitrary assignments of weights to words and states. This distinction is an instance of a general distinction between a covariant functor (or a monad) T and a contravariant functor G, which will be a recurring theme in this paper.

Applying the term *logic* to our framework is admittedly a bit of an exaggeration. We do not consider syntactic aspects usually connected with logical systems, such as deduction relations. We merely study ways to interpret certain terms (formulas) on models (coalgebras) in an inductive fashion. One might call it a testing framework, as in [29], but similar systems are often called logics in coalgebraic literature, and we reluctantly stick to this custom.

The idea of using modal logics for coalgebraic trace semantics is not new; it is visible already in [29]. In [12] it is related to behavioural equivalence, and applied to non-deterministic systems. A generalized notion of relation lifting is used in [6] to obtain infinite trace semantics, and applied in [7] to get canonical linear time logics. In [19], coalgebraic modal logic is combined with the idea of lifting behaviours to Eilenberg-Moore categories, with trace semantics in mind. In [16], a connection to modal logics is sketched from the perspective of coalgebraic determinization procedures. In a sense, this paper describes the same connection from the perspective of logic. In [8], the framework of [6, 7] is rephrased in terms of coalgebraic modal logic. The result is rather similar to the one we considered, with forgetful modalities to resolve branching. Unlike our approach, [8] relies on T being a monad, and under some more assumptions it studies canonical forgetful modalities that give rise to particularly well-behaved logics. In [23, 28], monads feature even more prominently, with the entire behaviour functor embedded in a so-called graded monad. In [11], it is embedded in a more complex functor with a so-called observer.

Our main new contribution is the notion of forgetful logic and its ramifications. Basic definitions are provided in Section 3 and some illustrative examples in Section 4. We introduce a systematic way of relating trace semantics to determinization, by giving sufficient conditions for a given determinization procedure, understood in a slightly more general way than in [16], to be correct with respect to a given forgetful logic (Section 6). For instance, this allows showing in a coalgebraic setting that the determinization of alternating automata into non-deterministic automata preserves language semantics.

A correct determinization procedure may not exist in general. In Section 7 we study a situation where a canonical correct determinization procedure exists. It turns out that even in the simple case of non-deterministic automata that procedure is not the classical powerset construction; instead, it relies on a double application of contravariant powerset construction. Interestingly, this is what also happens in Brzozowski's algorithm for automata minimization [4], so as a by-product, we get a new perspective on that algorithm which has recently attracted much attention in the coalgebraic community [1, 2, 3].

Although we do not assume the branching functor T to be a monad, a forgetful logic for T is equivalent to a transformation from T to a certain monad which, in the case of sets, is the double contravariant powerset monad (a special case of the continuation monad). One might say that the continuation monad is rich enough to handle all types of branching that can be "forgotten" within our framework.

This paper is an extended version of [21], adding full proofs and a treatment of the trace semantics of probabilistic systems as a non-trivial instance of the framework.

pointed out Lemma 5.2 and whose many insightful comments let us significantly improve
the paper.

2. Preliminaries

We assume familiarity with basic notions of category theory (see, e.g., [24]). A coalgebra
for a functor $B\colon \mathcal{C} \to \mathcal{C}$ consists of an object X and a map $f\colon X \to BX$. A homomorphism
from $f\colon X \to BX$ to $g\colon Y \to BY$ is a map $h\colon X \to Y$ such that $g \circ h = Bh \circ f$. The category
of B-coalgebras is denoted by $\mathsf{Coalg}(B)$. Algebras for a functor L are defined dually; the
category of L-algebras and homomorphisms is denoted by $\mathsf{Alg}(L)$.

We list a few examples where $\mathcal{C} = \mathsf{Set}$, the category of sets and functions. Consider the
functor $\mathcal{P}_\omega(A \times -)$, where $\mathcal{P}_\omega$ is the finite powerset functor and A is a fixed set. A coalgebra
$f\colon X \to \mathcal{P}_\omega(A \times X)$ is a finitely branching labelled transition system: it maps every state
to a finite set of next states. Coalgebras for the functor $(\mathcal{P}_\omega-)^A$ are image-finite labelled
transition systems, i.e., the set of next states for every label is finite. When A is finite the
two notions coincide. A coalgebra $f\colon X \to \mathcal{P}_\omega(A \times X + 1)$, where $1 = \{*\}$ is a singleton, is
a non-deterministic automaton; a state x is accepting whenever $* \in f(x)$.

Consider the functor $BX = 2 \times X^A$, where 2 is a two-element set of truth values. A coal-
gebra $\langle o, f \rangle \colon X \to BX$ is a deterministic automaton; a state x is accepting if $o(x) = \mathsf{tt}$, and
$f(x)$ is the transition function. The composition $B\mathcal{P}_\omega$ yields non-deterministic automata,
presented in a different way than above. We shall also consider $B\mathcal{P}_\omega\mathcal{P}_\omega$-coalgebras, which
represent a general version of alternating automata.

Let $\mathbb{S}$ be a semiring. Define $\mathcal{M}X = \{\varphi \in \mathbb{S}^X \mid \mathrm{supp}(\varphi) \text{ is finite}\}$ where $\mathrm{supp}(\varphi) = \{x \mid \varphi(x) \neq 0\}$, and $\mathcal{M}(f\colon X \to Y)(\varphi)(y) = \sum_{x \in f^{-1}(y)} \varphi(x)$. A *weighted automaton* is a
coalgebra for the functor $\mathcal{M}(A \times - + 1)$. Let Σ be a polynomial functor corresponding to
an algebraic signature. A *top-down weighted tree automaton* is a coalgebra for the functor
$\mathcal{M}\Sigma$. For $\mathbb{S}$ the Boolean semiring these are *non-deterministic tree automata*. Similar to
non-deterministic automata above, one can present weighted automata as coalgebras for
$\mathbb{S} \times (\mathcal{M}-)^A$.

We note that $\mathcal{P}_\omega$ is a monad, by taking the unit $\eta_X(x) = \{x\}$ and the multiplication μ
to be set union. More generally, the functor $\mathcal{M}$ extends to a monad, by taking $\mu_X(\varphi)(x) = \sum_{\psi \in \mathbb{S}^X} \varphi(\psi) \cdot \psi(x)$. The case of $\mathcal{P}_\omega$ is obtained by taking the Boolean semiring. Notice that
the finite support condition is required for μ to be well-defined.

2.1. **Contravariant adjunctions.**
The basic framework of coalgebraic logic is formed of
two categories $\mathcal{C}$, $\mathcal{D}$ connected by functors $F\colon \mathcal{C}^{op} \to \mathcal{D}$ and $G\colon \mathcal{D}^{op} \to \mathcal{C}$ that form an
adjunction $F^{op} \dashv G$. For example, one may take $\mathcal{C} = \mathcal{D} = \mathsf{Set}$ and $F = G = 2^-$, for
2 a two-element set of logical values. The intuition is that objects of $\mathcal{C}$ are collections of
processes, or states, and objects of $\mathcal{D}$ are logical theories.

To avoid cluttering the presentation with too much of the $(-)^{op}$ notation, we opt to
treat F and G as *contravariant functors*, i.e., ones that reverse the direction of all arrows
(maps), between $\mathcal{C}$ and $\mathcal{D}$. The adjunction then becomes a contravariant adjunction "on
the right", meaning that there is a natural bijection

$$\mathcal{C}(X, G\Phi) \cong \mathcal{D}(\Phi, FX) \qquad \text{for } X \in \mathcal{C}, \Phi \in \mathcal{D}.$$

Slightly abusing the notation, we shall denote both sides of this bijection by $(-)^\flat$. Applying
the bijection to a map is referred to as transposing the map.

In such an adjunction, GF is a monad on $\mathcal{C}$, whose unit we denote by $\iota\colon \mathsf{Id} \Rightarrow GF$, and FG is a monad on $\mathcal{D}$, with unit the denoted by $\epsilon\colon \mathsf{Id} \Rightarrow FG$. As usual, multiplication of the monad GF is $G\epsilon F\colon GFGF \Rightarrow GF$, and multiplication of FG is $F\iota G\colon FGFG \Rightarrow FG$. The counit-unit equations for adjunctions amount to:

$$
\begin{array}{ccc}
F \xrightarrow{\;\epsilon F\;} FGF & \qquad & G \xrightarrow{\;\iota G\;} GFG \\
& \searrow \;\Big\Downarrow {\scriptstyle F\iota} & \qquad \searrow \;\Big\Downarrow {\scriptstyle G\epsilon} \\
F & \qquad & G
\end{array}
\tag{2.1}
$$

Both F and G map colimits to limits, by standard preservation results for adjoint functors.

In what follows, the reader need only remember that F and G are contravariant, i.e., they reverse maps and natural transformations. All other functors, except a few that lift F and G to other categories, are standard covariant functors.

2.2. Coalgebraic modal logic.

We recall an approach to coalgebraic modal logic based on contravariant adjunctions, see, e.g., [20, 17]. Consider categories $\mathcal{C}$, $\mathcal{D}$ and functors F, G as in Section 2.1. Given an endofunctor $B\colon \mathcal{C} \to \mathcal{C}$, a *coalgebraic logic* to be interpreted on B-coalgebras is built of *syntax*, i.e., an endofunctor $L\colon \mathcal{D} \to \mathcal{D}$, and *semantics*, a natural transformation $\rho\colon LF \Rightarrow FB$. We will usually refer to ρ simply as a logic. If an initial L-algebra $a\colon L\Phi \to \Phi$ exists then, for any B-coalgebra $h\colon X \to BX$, the *logical semantics* of ρ on h is a map $s^\flat\colon X \to G\Phi$ obtained by transposing the map defined by initiality of a.

$$
\begin{array}{ccc}
L\Phi & \xrightarrow{\;Ls\;} & LFX \\
{\scriptstyle a}\Big\downarrow & & \Big\downarrow {\scriptstyle \rho_X} \\
& & FBX \\
& & \Big\downarrow {\scriptstyle Fh} \\
\Phi & \xrightarrow[\;s\;]{} & FX
\end{array}
\tag{2.2}
$$

The mapping of a B-coalgebra $h\colon X \to BX$ to an L-algebra $Fh \circ \rho_X\colon LFX \to FX$ determines a contravariant functor $\hat{F}$ that lifts F, i.e., acts as F on carriers.

$$
\begin{array}{ccc}
\mathsf{Coalg}(B) & \xrightarrow{\;\hat{F}\;} & \mathsf{Alg}(L) \\
\Big\downarrow & & \Big\downarrow \\
\mathcal{C} & \xrightarrow[\;F\;]{} & \mathcal{D}
\end{array}
\tag{2.3}
$$

The functor $\hat{F}$ has no (contravariant) adjoint in general; later in Section 7 we shall study well-behaved situations when it does. Notice that $\hat{F}$ maps coalgebra homomorphisms to algebra homomorphisms, and indeed the logical semantics factors through coalgebra homomorphisms, i.e., behavioural equivalence implies logical equivalence. The converse holds if ρ is *expressive*, meaning that the logical semantics decomposes as a coalgebra homomorphism followed by a mono.

Example 2.1. Let $\mathcal{C} = \mathcal{D} = \mathsf{Set}$, $F = G = 2^-$, $B = 2 \times -^A$ and $L = A \times - + 1$. The initial algebra of L is the set A^* of words over A. We define a logic $\rho\colon LF \Rightarrow FB$ as follows: $\rho_X(*)(o, t) = o$ and $\rho_X(a, \varphi)(o, t) = \varphi(t(a))$. For a coalgebra $\langle o, f \rangle\colon X \to 2 \times X^A$

the logical semantics is a map $s^\flat\colon X \to 2^{A^*}$, yielding the usual language semantics of the automaton: $s^\flat(x)(\varepsilon) = o(x)$ for the empty word ε, and $s^\flat(x)(aw) = s^\flat(f(x)(a))(w)$ for any $a \in A, w \in A^*$.

Note that logical equivalences, understood as kernel relations of logical semantics, are conceptually different from behavioural equivalences typically considered in coalgebra theory, in that they do not arise from finality of coalgebras, but rather from initiality of algebras (albeit in a different category). Fundamentally, logical semantics for coalgebras is defined by induction rather than coinduction. In some particularly well-behaved cases the inductive and coinductive views coincide; we shall study such situations in Section 7.

A logic $\rho\colon LF \Rightarrow FB$ gives rise to its *mate* $\rho^\flat\colon BG \Rightarrow GL$, defined by

$$BG \xRightarrow{\iota BG} GFBG \xRightarrow{G\rho G} GLFG \xRightarrow{GL\epsilon} GL, \tag{2.4}$$

where ι and ϵ are as in Section 2.1. A routine calculation shows that ρ in turn is the mate of $\rho^\flat$ (with the roles of F, G, ι and ϵ swapped), giving a bijective correspondence between logics and their mates. Some important properties of logics are conveniently stated in terms of their mates; e.g., under mild additional assumptions (see [20]), if the mate is pointwise monic then the logic is expressive. Two simple but useful diagrams show how logics relate to their mates along the basic adjunction:

Lemma 2.2. *For any logic* $\rho\colon LF \Rightarrow FB$, *the following diagrams commute:*

$$
\begin{array}{ccc}
B & \xRightarrow{\iota B} & GFB \\
{\scriptstyle B\iota}\Big\Downarrow & & \Big\Downarrow{\scriptstyle G\rho} \\
BGF & \xRightarrow[\rho^\flat F]{} & GLF
\end{array}
\qquad\qquad
\begin{array}{ccc}
L & \xRightarrow{\epsilon L} & FGL \\
{\scriptstyle L\epsilon}\Big\Downarrow & & \Big\Downarrow{\scriptstyle F\rho^\flat} \\
LFG & \xRightarrow[\rho G]{} & FBG.
\end{array}
$$

Proof. For the first diagram, chase:

where everything commutes, clockwise starting from top-left: by naturality of ι, by naturality of ρ, by (2.1) above, and by definition of $\rho^\flat$.

The other diagram is similar. $\qquad\square$

There is a direct characterization of logical semantic maps in terms of mates, first formulated in [29]. Indeed, by transposing (2.2) it is easy to check that the logical semantics $s^\flat\colon X \to G\Phi$ on a coalgebra $h\colon X \to BX$ is a unique map that makes the following "twisted coalgebra

morphism" diagram commute.

$$
\begin{array}{ccc}
BX & \xrightarrow{\;Bs^{\flat}\;} & BG\Phi \\
\big\uparrow{\scriptstyle h} & & \big\downarrow{\scriptstyle \rho^{\flat}_{\Phi}} \\
& & GL\Phi \\
& & \big\uparrow{\scriptstyle Ga} \\
X & \xrightarrow[\;s^{\flat}\;]{} & G\Phi.
\end{array}
\tag{2.5}
$$

2.3. **Morphisms of logics.** Given logics $\rho\colon LF \Rightarrow FB$ and $\theta\colon MF \Rightarrow FK$, a *morphism* from θ to ρ consists of a pair of natural transformations $\tau\colon M \Rightarrow L$ and $\kappa\colon B \Rightarrow K$ such that the following diagram commutes:

$$
\begin{array}{ccc}
MF & \xRightarrow{\theta} & FK \\
{\scriptstyle \tau F}\Big\Downarrow & & \Big\Downarrow{\scriptstyle F\kappa} \\
LF & \xRightarrow[\rho]{} & FB
\end{array}
\qquad \text{or, equivalently,} \qquad
\begin{array}{ccc}
BG & \xRightarrow{\rho^{\flat}} & GL \\
{\scriptstyle \kappa G}\Big\Downarrow & & \Big\Downarrow{\scriptstyle G\tau} \\
KG & \xRightarrow[\theta^{\flat}]{} & GM.
\end{array}
\tag{2.6}
$$

Natural transformations τ and κ as above induce functors

$$
\mathsf{Alg}(L) \xrightarrow{\;-\circ\tau\;} \mathsf{Alg}(M) \qquad \text{and} \qquad \mathsf{Coalg}(B) \xrightarrow{\;\kappa\circ-\;} \mathsf{Coalg}(K)
$$

defined by composition.

Lemma 2.3. *Suppose (τ, κ) is morphism from $\theta\colon MF \Rightarrow FK$ to $\rho\colon LF \Rightarrow FB$. Then the following diagram commutes:*

$$
\begin{array}{ccc}
\mathsf{Coalg}(B) & \xrightarrow{\;\kappa\circ-\;} & \mathsf{Coalg}(K) \\
{\scriptstyle \bar{F}_{\rho}}\Big\downarrow & & \Big\downarrow{\scriptstyle \bar{F}_{\theta}} \\
\mathsf{Alg}(L) & \xrightarrow[\;-\circ\tau\;]{} & \mathsf{Alg}(M)
\end{array}
$$

where $\bar{F}_{\rho}$ and $\bar{F}_{\theta}$ are the liftings of F induced by ρ and θ respectively, as in (2.3).

Proof. Let $f\colon X \to BX$ be a coalgebra, and consider:

$$
\begin{array}{ccccccc}
MFX & \xrightarrow{\theta_X} & FKX & \xrightarrow{F\kappa_X} & FBX & \xrightarrow{Ff} & FX \\
\big\| & & & & \big\| & & \big\| \\
MFX & \xrightarrow[\tau_{FX}]{} & LFX & \xrightarrow[\rho_X]{} & FBX & \xrightarrow[Ff]{} & FX
\end{array}
$$

The upper path is $\bar{F}_{\theta}(\kappa_X \circ f)$, and the lower path is $(\bar{F}_{\rho}(f)) \circ \tau_X$. The diagram commutes since (τ, κ) is a morphism of logics. $\qquad\square$

Morphisms of logics apppear in [5], where the category of logics and morphisms between them is studied. The examples in [5] involve a translation of syntax determined by τ. Our main interest in morphisms of logics is cases where $\tau = \mathsf{id}$. Then, it is a direct consequence of Lemma 2.3 that the logical semantics of ρ on a coalgebra $f\colon X \to BX$ coincides with the logical semantics of θ on $\kappa_X \circ f$. In the sequel, in a situation where $\tau = \mathsf{id}$ we simply say that κ is a morphism of logics.

3. Forgetful logics

In most abstract approaches to coalgebraic trace semantics, the behaviour functor under consideration is a composition TB or BT, where T is the branching aspect and B is the type of observations of interest. Our approach is to capture trace semantics as the logical semantics of a suitable logic for TB or BT. The logics that we consider are defined as the composition of a logic for B and a special kind of logic for T which has trivial syntax. This special logic for T specifies how the branching behaviour should be "forgotten" in the resulting logical theory.

Logics for composite functors can often be obtained from logics of their components. Consider functors $B, T\colon \mathcal{C} \to \mathcal{C}$ and logics for them $\rho\colon LF \Rightarrow FB$ and $\alpha\colon NF \Rightarrow FT$, for some functors $L, N\colon \mathcal{D} \to \mathcal{D}$. One can then define logics for the functors TB and BT:

$$\alpha \circledcirc \rho = \alpha B \circ N\rho \colon NLF \Rightarrow FTB, \qquad \rho \circledcirc \alpha = \rho T \circ L\alpha \colon LNF \Rightarrow FBT.$$

It is easy to see that taking the mate of a logic respects this composition operator, i.e., that $(\alpha \circledcirc \rho)^\flat = \alpha^\flat \circledcirc \rho^\flat$. Such compositions of logics appear in [14] and were studied in a slightly more concrete setting in [9, 32]. In [5, Lemma 3.12], it is shown how to (horizontally) compose morphisms of logics, turning $\circledcirc$ into a bifunctor.

We shall be interested in the case where the logic for T has a trivial syntax; in other words, where $N = \mathsf{Id}$. Intuitively speaking, we require a logic for T that consists of a single unary operator, which could therefore be elided in a syntactic presentation of logical formulas. The semantics of such an operator is defined by a natural transformation $\alpha\colon F \Rightarrow FT$ or equivalently by its mate $\alpha^\flat\colon TG \Rightarrow G$. Intuitively, the composite logics $\alpha \circledcirc \rho$ and $\rho \circledcirc \alpha$, when interpreted on TB- and BT-coalgebras respectively disregard, or forget, the aspect of their behaviour related to the functor T, in a manner prescribed by α. We call logics obtained in this fashion *forgetful logics*.

Transformations $\alpha\colon F \Rightarrow FT$ (and their mates $\alpha^\flat$) are also in bijective correspondence with natural transformations $\alpha^\dagger\colon T \Rightarrow GF$. Indeed, define

$$\alpha^\dagger = G\alpha \circ \iota T \qquad \text{or equivalently} \qquad \alpha^\dagger = \alpha^\flat F \circ T\iota,$$

and recover

$$\alpha = F\alpha^\dagger \circ \epsilon F \qquad \text{and} \qquad \alpha^\flat = G\epsilon \circ \alpha^\dagger G$$

in what is easily seen to be mutually inverse operations. This means that, intuitively, a forgetful logic for T is equivalent to an *encoding* of T into the monad GF. In all examples considered in the next section $\alpha^\dagger$ is (pointwise) monic, which justifies the name "encoding", but we do not use the monicity for anything and we do not have an understanding of its significance.

As long as we do not assume T to be a monad, it makes no sense to ask e.g. whether $\alpha^\dagger$ is a monad morphism. However, composing forgetful logics for multiple behaviour functors does agree with the multiplication structure of GF. Specifically, for $\alpha\colon F \Rightarrow FT$ and

$\beta\colon F \Rightarrow FS$, one may consider the composite $\alpha \odot \beta\colon F \Rightarrow FTS$. The corresponding encoding $(\alpha \odot \beta)^\dagger\colon TS \Rightarrow GF$ is then equal to:

$$TS \xRightarrow{\alpha^\dagger \beta^\dagger} GFGF \xRightarrow{G\epsilon F} GF.$$

Encodings $\alpha^\dagger$ will be technically useful in Section 7.2.

4. Examples

We instantiate the setting of Section 3 and use forgetful logics to obtain trace semantics for several concrete types of coalgebras: non-deterministic automata, transition systems, alternating automata, weighted tree automata and probabilistic systems.

In the first few examples we let $\mathcal{C} = \mathcal{D} = \mathsf{Set}$ and $F = G = 2^-$, and consider TB or BT-coalgebras, where $T = \mathcal{P}_\omega$ is the finite powerset functor. Our examples involve the logic $\alpha\colon 2^- \Rightarrow 2^{\mathcal{P}_\omega}$ defined by:

$$\alpha_X(\varphi)(S) = \mathsf{tt} \iff \exists x \in S.\varphi(x) = \mathsf{tt}. \tag{4.1}$$

This choice of F and G has been studied thoroughly in the field of coalgebraic logic, and our α is an example of the standard notion of predicate lifting [14, 22] corresponding to the so-called diamond modality. Its mate $\alpha^\flat\colon \mathcal{P}_\omega 2^- \Rightarrow 2^-$ and the corresponding encoding $\alpha^\dagger\colon \mathcal{P}_\omega \Rightarrow 2^{2^-}$ are as follows:

$$\alpha_\Phi^\flat(S)(w) = \mathsf{tt} \iff \exists \varphi \in S.\varphi(w) = \mathsf{tt}$$

$$\alpha_X^\dagger(U)(\varphi) = \mathsf{tt} \iff \exists x \in U.\varphi(x) = \mathsf{tt}$$

Here and in all examples below, $\mathcal{P}_\omega$ could be replaced by the full powerset $\mathcal{P}$ without any problems.

Example 4.1. We define a forgetful logic $\alpha \odot \rho$ for the functor $\mathcal{P}_\omega(A \times - + 1)$ whose coalgebras are non-deterministic automata, so that the logical semantics is the usual language semantics. To this end, we let:

- $\mathcal{C} = \mathcal{D} = \mathsf{Set}$, $F = G = 2^-$;
- $T = \mathcal{P}_\omega$, $B = L = (A \times - + 1)$;
- α be as in (4.1), and
- ρ be defined by its mate $\rho^\flat\colon A \times 2^- + 1 \to 2^{A \times - + 1}$ as follows:

$$\rho_\Phi^\flat(*)(t) = \mathsf{tt} \iff t = * \qquad \rho_\Phi^\flat(a, \varphi)(t) = \mathsf{tt} \iff t = (a, w) \text{ and } \varphi(w) = \mathsf{tt}.$$

for any set Φ.

The choice of L is motivated by the fact that the initial algebra of $A \times - + 1$ is A^*, hence the logical semantics will be a map from states to languages (elements of 2^{A^*}). Now the logical semantics of the logic $\alpha \odot \rho$ on an automaton $f\colon X \to \mathcal{P}_\omega BX$ is the map $s^\flat$ from (2.5), i.e., the unique map that makes the following diagram commute:

It is easy to calculate (see Appendix A) that for any $x \in X$:

$$s^\flat(x)(\varepsilon) = \mathsf{tt} \iff * \in f(x),$$

$$s^\flat(x)(aw) = \mathsf{tt} \iff \exists y \in X.(a,y) \in f(x) \text{ and } s^\flat(y)(w) = \mathsf{tt},$$

for ε the empty word, and for all $a \in A$ and $w \in A^*$.

Note that the logic ρ in the above example is expressive. One may expect that given a different expressive logic θ involving the same functors, the forgetful logics $\alpha \odot \rho$ and $\alpha \odot \theta$ yield the same logical equivalences, but this is not the case. For instance, define $\theta^\flat \colon BG \Rightarrow GL$ as $\theta^\flat_\Phi(*)(t) = \mathsf{tt}$ for all t, and $\theta^\flat_\Phi(a,\varphi) = \rho^\flat_\Phi(a,\varphi)$. This logic is expressive as well (since $\theta^\flat$ is componentwise monic) but in the semantics of the forgetful logic $\alpha \odot \theta$, information on final states is discarded.

Example 4.2 (Length of words). The initial algebra of the functor L defined by $LX = X+1$ is $\Phi = \mathbb{N}$, the set of natural numbers. Define a logic for $BX = A \times X + 1$ by its mate $\rho^\flat \colon A \times 2^- + 1 \Rightarrow 2^{-+1}$ as follows:

$$\rho^\flat(*)(t) = \mathsf{tt} \iff t = * \qquad \rho^\flat(a,\varphi)(t) = \mathsf{tt} \iff t = x \text{ and } \varphi(x) = \mathsf{tt}.$$

Note that this logic is not expressive. With the above α (Equation 4.1), we have a logic $\alpha \odot \rho$, and given any $f \colon X \to \mathcal{P}_\omega(A \times X + 1)$, this yields the following map $s^\flat \colon X \to 2^\Phi$:

$$s^\flat(x)(0) = \mathsf{tt} \iff * \in f(x),$$

$$s^\flat(x)(n+1) = \mathsf{tt} \iff \exists a \in A, y \in X.(a,y) \in f(x) \text{ and } s^\flat(y)(n) = \mathsf{tt}.$$

Thus, $s^\flat(x)$ is the binary sequence which is tt at position n iff the automaton f accepts a word of length n, starting in state x.

Example 4.3 (Labelled transition systems). In this example we consider the finite traces of labelled transition systems of the form $f \colon X \to (\mathcal{P}_\omega X)^A$, i.e., BT-coalgebras where $BX = X^A$ and $TX = \mathcal{P}_\omega X$. To this end, let $LX = A \times X + 1$. Define $\rho^\flat \colon (2^-)^A \Rightarrow 2^{A \times - +1}$ as follows:

$$\rho^\flat_\Phi(\varphi)(*) = \mathsf{tt} \qquad \rho^\flat_\Phi(\varphi)(a,w) = \varphi(a)(w). \tag{4.2}$$

Then the logical semantics $s^\flat \colon X \to 2^{A^*}$ of $\rho \odot \alpha$ on a transition system $f \colon X \to (\mathcal{P}_\omega X)^A$ is given by $s^\flat(x)(\varepsilon) = \mathsf{tt}$ and $s^\flat(x)(aw) = \mathsf{tt}$ iff $s^\flat(y)(w) = \mathsf{tt}$ for some $y \in f(x)(a)$.

Example 4.4 (Non-deterministic automata as BT-coalgebras). Consider the functor $BX = 2 \times X^A$. Let $LX = A \times X + 1$, let $\rho^\flat \colon 2 \times (2^-)^A \Rightarrow 2^{A \times - +1}$ be the mate of the logic ρ given in Example 2.1; explicitly, it is the obvious isomorphism given by manipulating exponents:

$$\rho^\flat_\Phi(o,\varphi)(*) = o \qquad \rho^\flat_\Phi(o,\varphi)(a,w) = \varphi(a)(w) \tag{4.3}$$

The logical semantics $s^\flat \colon X \to 2^{A^*}$ of $\rho \odot \alpha$ on a coalgebra $\langle o, f \rangle \colon X \to 2 \times (\mathcal{P}_\omega X)^A$ is the usual language semantics, i.e., for any $x \in X$ we have:

$$s^\flat(x)(\varepsilon) = o(x) \qquad s^\flat(x)(aw) = \mathsf{tt} \iff s^\flat(y)(w) = \mathsf{tt} \text{ for some } y \in f(x)(a).$$

Non-determinism can be resolved differently: in contrast to (4.1), consider $\beta\colon \mathcal{P}_\omega \Rightarrow \mathcal{P}_\omega 2^-$, and the corresponding $\beta^\flat\colon \mathcal{P}_\omega 2^- \Rightarrow \mathcal{P}_\omega$ and $\beta^\dagger\colon \mathcal{P}_\omega \Rightarrow 2^{2^-}$, given by

$$\beta_X(\varphi)(S) = \mathsf{tt} \iff \forall x \in S.\varphi(x) = \mathsf{tt}$$
$$\beta^\flat_\Phi(S)(w) = \mathsf{tt} \iff \forall \varphi \in S.\varphi(w) = \mathsf{tt}$$
$$\beta^\dagger_X(U)(\varphi) = \mathsf{tt} \iff \forall x \in U.\varphi(x) = \mathsf{tt}$$

Similarly to (4.1), β is a predicate lifting that corresponds to the so-called box modality. The semantics $s^\flat$ induced by the forgetful logic $\rho \odot \beta$ accepts a word if *all* paths end in an accepting state: $s^\flat(x)(\varepsilon) = o(x)$, and $s^\flat(x)(aw) = \mathsf{tt}$ iff $s^\flat(y)(w) = \mathsf{tt}$ for all $y \in f(x)(a)$. We call this the conjunctive semantics. In automata-theoretic terms, this is the language semantics for ($B\mathcal{P}_\omega$-coalgebras understood as) co-nondeterministic automata, i.e., alternating automata with only universal states.

Some non-examples. It is not clear how to use forgetful logics to give a conjunctive semantics to coalgebras for $\mathcal{P}_\omega(A \times X + -)$; simply using β together with ρ from Example 4.1 does not yield the expected logical semantics. Also, transition systems as $\mathcal{P}_\omega(A \times -)$-coalgebras do not work well; with α as in (4.1) the logical semantics of a state with no successors is always empty, while it should contain the empty trace.

Example 4.5 (Alternating automata)**.** Consider $B\mathcal{P}_\omega\mathcal{P}_\omega$-coalgebras with $B = 2 \times -^A$. We give a forgetful logic by combining ρ, α, and β from the previous example (more precisely, the logic is $(\rho \odot \alpha) \odot \beta$); recall that α and β resolve the non-determinism by disjunction and conjunction respectively. The logical semantics on a coalgebra $\langle o, f \rangle\colon X \to B\mathcal{P}_\omega\mathcal{P}_\omega X$ then is the map $s^\flat$ in the following diagram (see (2.5)):

$$
\begin{array}{ccc}
X & \xrightarrow{\quad s^\flat \quad} & 2^{A^*} \\
{\scriptstyle \langle o,f \rangle}\big\downarrow & & \big\downarrow \\
B\mathcal{P}_\omega\mathcal{P}_\omega X \xrightarrow[B\mathcal{P}_\omega\mathcal{P}_\omega s^\flat]{} B\mathcal{P}_\omega\mathcal{P}_\omega 2^{A^*} \xrightarrow[B\mathcal{P}_\omega\beta^\flat_{A^*}]{} B\mathcal{P}_\omega 2^{A^*} \xrightarrow[B\alpha^\flat_{A^*}]{} B2^{A^*} \xrightarrow[\rho^\flat_{A^*}]{} & & 2^{LA^*}
\end{array}
\qquad (4.4)
$$

Spelling out the details for a coalgebra $\langle o, f \rangle\colon X \to 2 \times (\mathcal{P}_\omega\mathcal{P}_\omega X)^A$ yields, for any $x \in X$: $s^\flat(x)(\varepsilon) = o(x)$ and for any $a \in A$ and $w \in A^*$: $s^\flat(x)(aw) = \mathsf{tt}$ iff there is $S \in f(x)(a)$ such that $s^\flat(y)(w) = \mathsf{tt}$ for all $y \in S$ (see Appendix A).

Example 4.6 (Weighted tree automata)**.** In this example we let $\mathcal{C} = \mathcal{D} = \mathsf{Set}$ and $F = G = \mathbb{S}^-$ for a semiring $\mathbb{S}$. We consider coalgebras for $\mathcal{M}\Sigma$ (Section 2), where Σ is a polynomial functor corresponding to a signature. The initial algebra of Σ is carried by the set of finite Σ-trees, denoted by $\Sigma^*\emptyset$. Define $\rho\colon \Sigma F \Rightarrow F\Sigma$ by cases on the operators σ in the signature:

$$\rho_X(\sigma(\varphi_1, \ldots, \varphi_n))(\tau(x_1, \ldots, x_m)) = \begin{cases} \prod_{i=1..n} \varphi_i(x_i) & \text{if } \sigma = \tau \\ 0 & \text{otherwise} \end{cases}$$

where n is the arity of σ. Define $\alpha\colon \mathbb{S}^- \Rightarrow \mathbb{S}^\mathcal{M}$ by its mate: $\alpha^\flat_\Phi(\varphi)(w) = \sum_{\psi \in \mathbb{S}^\Phi} \varphi(\psi) \cdot \psi(w)$. Notice that α and ρ generalize the logics of Example 4.1.

The logical semantics of $\alpha \odot \rho$ on a weighted tree automaton $f \colon X \to \mathcal{M}\Sigma X$ is the unique map $s^\flat \colon X \to \mathbb{S}^{\Sigma^*\emptyset}$ making the following diagram commute:

$$\begin{array}{ccc}
X & \xrightarrow{\quad\quad\quad\quad s^\flat \quad\quad\quad\quad} & \mathbb{S}^{\Sigma^*\emptyset} \\
\downarrow{\scriptstyle f} & & \downarrow \\
\mathcal{M}\Sigma X \xrightarrow{\mathcal{M}\Sigma s^\flat} \mathcal{M}\Sigma\mathbb{S}^{\Sigma^*\emptyset} \xrightarrow{\mathcal{M}\rho^\flat_{\Sigma^*\emptyset}} \mathcal{M}\mathbb{S}^{\Sigma\Sigma^*\emptyset} \xrightarrow{\alpha^\flat_{\Sigma\Sigma^*\emptyset}} \mathbb{S}^{\Sigma\Sigma^*\emptyset}
\end{array} \qquad (4.5)$$

This means that for any tree $\sigma(t_1, \ldots, t_n)$ and any $x \in X$ we have (see Appendix A):

$$s^\flat(x)(\sigma(t_1, \ldots, t_n)) = \sum_{x_1,\ldots,x_n \in X} f(x)(\sigma(x_1, \ldots, x_n)) \cdot \prod_{i=1..n} s^\flat(x_i)(t_i)$$

As a special case, we obtain for any *weighted automaton* $f \colon X \to \mathcal{M}(A \times X + 1)$ a unique map $s^\flat \colon X \to \mathbb{S}^{A^*}$ so that for any $x \in X$, $a \in A$ and $w \in A^*$: $s^\flat(x)(\varepsilon) = f(x)(*)$ and $s^\flat(x)(aw) = \sum_{y \in X} f(x)(a,y) \cdot s^\flat(y)(w)$. For $\mathbb{S}$ the Boolean semiring we get the usual semantics of tree automata: $s^\flat(x)(\sigma(t_1, \ldots, t_n)) = \mathsf{tt}$ iff there are $x_1, \ldots, x_n$ such that $\sigma(x_1, \ldots, x_n) \in f(x)$ and for all $i \leq n \colon s^\flat(x_i)(t_i) = \mathsf{tt}$.

The Σ-algebra $\hat{F}(X, f)$ (see (2.3)) is a *deterministic bottom-up tree automaton*. It corresponds to the top-down automaton f, in the sense that the semantics $s^\flat$ of f is the transpose of the unique homomorphism $s \colon \Sigma^*\emptyset \to \mathbb{S}^X$ arising by initiality; the latter is the usual semantics of bottom-up tree automata.

Example 4.7 (Probabilistic systems). Consider generative probabilistic transition systems [34] with explicit termination, modeled as coalgebras $f \colon X \to \Delta(A \times X + 1)$, where Δ is the finitely supported probability distribution functor on Set. One would like to interpret sequences of labels from A as *completed* traces for such coalgebras, i.e., ones ending with a transition to the unique element of 1, and assign probabilities to them.

Although probability distributions on a set X can be seen as functions from X to the interval $[0,1]$, techniques of Example 4.6 are not directly applicable, since $[0,1]$ is not a semiring in the expected sense: addition is not a total operation. One could replace $[0,1]$ with the semiring of nonnegative real numbers and proceed as in Example 4.6, but the resulting trace semantics would obscure an important property of probabilistic traces: every trace has probability at most 1. Actually, in the example considered here, an even stronger property holds: for every process, probabilities of all complete traces generated from it form a subprobability distribution. We wish to design a forgetful logic framework that would make this property apparent.

To this end, put $\mathcal{C} = \mathsf{Set}$ and $\mathcal{D} = \mathsf{PCM}$, the category of partial commutative monoids (see [15] for details). A partial commutative monoid is a set X together with a unit $0 \in X$ and a partial binary addition function $\oslash$ on X that is commutative and associative whenever defined. A morphism of such monoids is a function that preserves units and addition, whenever defined. A paradigmatic example of a partial commutative monoid is the interval $[0,1]$ with 0 as unit and addition defined whenever the result is at most 1.

The obvious forgetful functor $U \colon \mathsf{PCM} \to \mathsf{Set}$ has a left adjoint $(-)_\perp \colon \mathsf{Set} \to \mathsf{PCM}$ that assigns to every set X the partial monoid $X + \perp$ with $\perp$ as the unit, and addition defined by $\perp \oslash x = x \oslash \perp = x$ and undefined otherwise.

The category PCM is symmetric monoidal closed [15] with the internal hom-functor $\Phi \multimap \Psi = \mathrm{hom}_{\mathsf{PCM}}(\Phi, \Psi)$ with the constant function $0(\phi) = 0_\Psi$ as the unit, and addition

defined by
$$(f \oslash g)(\phi) = f(\phi) \oslash g(\phi)$$
if the addition on the right is defined for every $\phi \in \Phi$, and $f \oslash g$ undefined otherwise. The symmetric monoidal closed structure implies that the functor $- \multimap \Psi$ is contravariant self-adjoint for any partial commutative monoid Ψ.

Define contravariant functors
$$FX = X_{\perp} \multimap [0,1] \qquad \text{and} \qquad G\Phi = \text{hom}_{\mathsf{PCM}}(\Phi, [0,1]).$$
They form a contravariant adjunction: a composition of the adjunction $(-)_{\perp} \dashv U$ with the contravariant self-adjunction of $- \multimap [0,1]$. In other words, there is a bijection
$$X \to \text{hom}_{\mathsf{PCM}}(\Phi, [0,1]) \cong \text{hom}_{\mathsf{PCM}}(\Phi, X_{\perp} \multimap [0,1])$$
natural in $X \in \mathsf{Set}$ and $\Phi \in \mathsf{PCM}$.

Define $\alpha \colon F \Rightarrow F\Delta$ by its mate $\alpha^{\flat} \colon \Delta G \Rightarrow G$:
$$\alpha_{\Phi}^{\flat}(\delta)(\phi) = \bigoslash_{f \in G\Phi} \delta(f) \cdot f(\phi).$$
This takes values in $[0,1]$, since
$$0 \le \bigoslash_{f \in G\Phi} \delta(f) \cdot f(\phi) \le \bigoslash_{f \in G\Phi} \delta(f) = 1.$$

PCM has products, defined as expected as cartesian products on carriers. Define a functor L on PCM by
$$L\Phi = \Phi_{\omega}^{A} \times \{\top\}_{\perp}$$
where Φ_{ω}^{A}, for the set A of transition labels, is the A-fold product of Φ restricted to those tuples that have the unit 0_{Φ} on all but finitely many components. If A is finite then this is simply the A-fold product of Φ.

Elements of $L\Phi$ are A-indexed families of elements of Φ that are 0_{Φ} almost everywhere, with an additional component that is either $\top$ or $\perp$, with addition defined componentwise (note that $\top \oslash \top$ is undefined).

Consider the set $\Lambda = \mathcal{P}_{\omega}(A^{*})$ of all finite sets of A-traces, considered as a partial commutative monoid with $\emptyset$ as the unit, and set union as addition, but defined only for disjoint sets. This partial monoid carries an L-algebra structure $h \colon L\Lambda \to \Lambda$ defined by:
$$h(T_a, T_b, \ldots, \perp) = \{aw \mid a \in A, \ w \in T_a\}$$
$$h(T_a, T_b, \ldots, \top) = \{aw \mid a \in A, \ w \in T_a\} \cup \{\epsilon\},$$
for any $T_a, T_b, \ldots \subseteq A^{*}$. It is easy check that h is a bijection and a morphism in PCM. In particular, h takes values in finite sets thanks to the restriction of Φ_{ω}^{A} to tuples that are $\emptyset$ almost everywhere.

Moreover, h is an initial L-algebra. To see this, consider any algebra $k \colon L\Phi \to \Phi$. The unique algebra map $f \colon \Lambda \to \Phi$ from h to k is defined by induction on the length of the longest trace in elements of Λ:
$$f(\emptyset) = 0_{\Phi} = k(0_{\Phi}, 0_{\Phi}, \ldots, \perp)$$
$$f(\{\epsilon\}) = k(0_{\Phi}, 0_{\Phi}, \ldots, \top)$$
$$f(T) = k(f(T_a), f(T_b), \ldots, \perp) \qquad\qquad \text{if } \epsilon \notin T$$
$$f(T) = k(f(T_a), f(T_b), \ldots, \top) \qquad\qquad \text{if } \epsilon \in T$$

where $T_a = \{w \in A^* \mid aw \in T\}$, for $T \subseteq A^*$ and $a \in A^*$. This is a well-formed inductive definition since each T_a only contains traces strictly shorter than the longest trace in T. It is also a partial monoid morphism, since if T and T' are disjoint sets of traces then T_a and T'_a are disjoint for each $a \in A$, and at most one of T and T' contains the empty word ϵ. The fact that f is an algebra morphism follows directly from its definition, and its uniqueness follows by routine induction.

A logic $\rho\colon LF \Rightarrow FB$, where $BX = A \times X + 1$, can be defined by its mate $\rho^\flat\colon BG \Rightarrow GL$:

$$\rho^\flat_\Phi : A \times \hom_{\mathsf{PCM}}(\Phi, [0,1]) + 1 \to \hom_{\mathsf{PCM}}(\Phi^A \times \{\top\}_\perp, [0,1])$$

$$\rho^\flat_\Phi(a, h)(\phi_a, \phi_b, \ldots, x) = h(\phi_a) \qquad \text{for } x \in \{\top, \perp\}$$

$$\rho^\flat_\Phi(*)(\phi_a, \phi_b, \ldots, x) = \begin{cases} 0 & \text{if } x = \perp \\ 1 & \text{if } x = \top \end{cases}$$

where $*$ is the unique element of 1. Both α and ρ are easily seen to be natural, so this completes an instance of the framework of forgetful logics.

As a result, for any probabilistic transition system $f\colon X \to \Delta(A \times X + 1)$ we obtain a map $s^\flat\colon X \to \hom_{\mathsf{PCM}}(\Lambda, [0,1])$ that assigns, to every state $x \in X$, a map from finite sets of A-traces to the interval $[0,1]$ that is additive as far as disjoint sets of traces are concerned. Such a map gives (indeed, is equivalent to) a subprobability distribution δ on the set of all A-traces. Indeed, put $\delta(w) = s^\flat(\{w\})$; then

$$\sum_{w \in A^*} \delta(w) = \sup_{W \subseteq_{\mathrm{fin}} A^*} \left(\sum_{w \in W} \delta(w) \right) = \sup_{W \subseteq_{\mathrm{fin}} A^*} s^\flat(W) \leq 1.$$

Note that the partial monoid homomorphism $s\colon \Lambda \to (X_\perp \multimap [0,1])$ does *not* map traces to any distributions of states.

In [16, Sec. 7.2], where another coalgebraic approach to generative probabilistic systems was developed, the subprobability distribution monad had to be used instead of Δ. The reason for that was that even though a state in a generative probabilistic system determines a full probability distribution on successors (including immediate termination, i.e. the empty trace, as a degenerate successor), the finite complete traces of a state do not form a probability distribution, but only a subprobability distribution, due to possibility of infinite traces. In the forgetful logic approach the branching type functor Δ is distinguished from the functor G used to collect a structure of traces, so the distribution monad can be used. Note that although the subdistribution monad is naturally isomorphic to the functor $\Delta(-+1)$ that is present in our behaviour functor, the $+1$ component here is not a part of the branching structure: it models trace termination and it is very much a part of the trace behaviour structure. In [16], the subdistribution monad is used on top of that, with an additional $+1$ component necessary to handle global *non*termination.

The technique developed in this example does not work for *reactive* probabilistic transition systems modeled as coalgebras $f\colon X \to [0,1] \times (\Delta X)^A$. Technically, it is not clear how to find a functor L on PCM with a natural transformation $\rho\colon LF \Rightarrow FB$ for $BX = [0,1] \times X^A$, that would model trace semantics as expected. This is not surprising, as under a standard probabilistic semantics of reactive systems, traces accepted from a fixed state do not form a (sub)probability distribution.

5. FORGETFUL LOGICS FOR MONADS

In most coalgebraic attempts to trace semantics [6, 11, 16, 19, 23, 30], the functor T, which models the branching aspect of system behaviour, is assumed to be a monad. The basic definition of a forgetful logic is more relaxed in that it allows an arbitrary functor T but one may notice that in all examples in Section 4, T is a monad.

In coalgebraic approaches cited above, the structure of T is resolved using monad multiplication $\mu\colon TT \Rightarrow T$. Forgetful logics use transformations $\alpha\colon F \Rightarrow FT$ with their mates $\alpha^\flat\colon TG \Rightarrow G$ for the same purpose. If T is a monad, it will be useful to assume a few basic axioms analogous to those of monad multiplication:

Definition 5.1. Let (T, η, μ) be a monad. A natural transformation $\alpha^\flat\colon TG \Rightarrow G$ is a *(T)-action* (on G) if the following diagram commutes:

$$
\begin{array}{ccc}
TTG & \overset{T\alpha^\flat}{\Longrightarrow} TG \overset{\eta G}{\Longleftarrow} & G \\
\mu G \big\Downarrow & \alpha^\flat \big\Downarrow & \Big\Vert \\
TG & \underset{\alpha^\flat}{\Longrightarrow} & G
\end{array}
$$

i.e., if each component of $\alpha^\flat$ is an Eilenberg-Moore algebra for T.

Monad actions on functors are to monads as monoid actions on sets are to monoids.

Logics α whose mates are monad actions have a characterization in terms of their corresponding encodings $\alpha^\dagger : T \Rightarrow GF$ defined as in Section 3:

Lemma 5.2. *For any $\alpha\colon F \Rightarrow FT$, the mate $\alpha^\flat\colon TG \Rightarrow G$ is a monad action if and only if $\alpha^\dagger\colon T \Rightarrow GF$ is a monad morphism.*

Proof. This is a special case of [10, Prop. II.1.4], but see Appendix B for a self-contained proof. □

This means that a forgetful logic for T whose mate is a monad action, is equivalent to an encoding of the monad structure of T (which is used for resolving branching in [30, 12, 16, 33]) into the monad GF. We shall use this connection in Section 6 to relate forgetful logics to the determinization constructions of [16].

It is easy to check by hand that in all examples in Section 4, $\alpha^\flat$ is an action, but it also follows from the following considerations.

In some well-structured cases, one can search for a suitable α by looking at T-algebras in $\mathcal{C}$. We mention it only briefly and not explain the details, as it will not be directly used in the following.

If $\mathcal{C}$ has products, then for any object $V \in \mathcal{C}$ there is a contravariant adjunction as in Section 2.1, where: $\mathcal{D} = \mathsf{Set}$, $F = \mathcal{C}(-, V)$ and $G = V^-$, where V^X denotes the X-fold product of V in $\mathcal{C}$. (This adjunction was studied in [25] for the purpose of combining distributive laws.) By the Yoneda Lemma, natural transformations $\alpha\colon F \Rightarrow FT$ are in bijective correspondence with algebras $g\colon TV \to V$. Routine calculation shows that the mate $\alpha^\flat$ is a T-action if and only if the corresponding g is an Eilenberg-Moore algebra for T.

Alternatively, one may assume that $\mathcal{C} = \mathcal{D}$ is a symmetric monoidal closed category and $F = G = V^-$ is the internal hom-functor based on an object $V \in \mathcal{C}$. (This adjunction was studied in [20] in the context of coalgebraic modal logic.) If, additionally, the functor

T is strong, then every algebra $g\colon TV \to V$ gives rise to $\alpha\colon F \Rightarrow FT$, whose components $\alpha_X\colon V^X \to V^{TX}$ are given by transposing:

$$TX \otimes V^X \xrightarrow{\quad\text{strength}\quad} T(X \otimes V^X) \xrightarrow{T(\text{application})} TV \xrightarrow{\quad g \quad} V$$

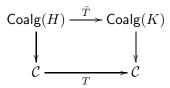If T is a strong monad and g is an Eilenberg-Moore algebra for T then $\alpha^\flat$ is a T-action.

In an enriched setting, if $\mathcal{D}$ is enriched over $\mathcal{C}$, both these constructions are instances of a more general one based on the existence of suitable powers.

If $\mathcal{C} = \mathcal{D} = \mathsf{Set}$ then both constructions apply (and coincide). In this situation more can be said [16, 14]: the resulting contravariant adjunction can be factored through the category of Eilenberg-Moore algebras for T.

6. Determinization

The classical powerset construction turns a non-deterministic automaton into a deterministic one, with states of the former interpreted as singleton states in the latter. More generally, a determinization procedure of coalgebras involves a change of state space. We define it as follows.

Definition 6.1. Let $T, H\colon \mathcal{C} \to \mathcal{C}$ be endofunctors. A (T)-*determinization procedure* of H-coalgebras consists of a natural transformation $\eta\colon \mathsf{Id} \Rightarrow T$, a functor $K\colon \mathcal{C} \to \mathcal{C}$ and a lifting of T:

$$
\begin{array}{ccc}
\mathsf{Coalg}(H) & \xrightarrow{\ \bar{T}\ } & \mathsf{Coalg}(K) \\
\big\downarrow & & \big\downarrow \\
\mathcal{C} & \xrightarrow[\ \ T\ \ ]{} & \mathcal{C}
\end{array}
$$

We will mostly focus on cases where $H = TB$ or $H = BT$, but in Section 7 we will consider situations where T is not directly related to H.

The classical powerset construction is *correct*, in the sense that the language semantics of a state x in a non-deterministic automaton coincides with the final semantics (the accepted language) of the singleton of x in the determinized automaton. At the coalgebraic level, we capture trace semantics by a forgetful logic. Then, a determinization procedure is correct if logical equivalence on the original system coincides with behavioural equivalence on the determinized system along η, formally captured as follows.

Definition 6.2. A T-determinization procedure $(\bar{T}, K, \eta)$ of H-coalgebras is *correct* wrt. a logic for H if, for any H-coalgebra (X, f) with logical semantics $s^\flat$:

(1) $s^\flat$ factors through $h \circ \eta_X$, for any K-coalgebra homomorphism h from $\bar{T}(X, f)$.
(2) there exists a K-coalgebra homomorphism h from $\bar{T}(X, f)$ and a mono m so that $s^\flat = m \circ h \circ \eta_X$.

The first condition states that behavioural equivalence on the determinized system implies logical equivalence on the original system; the second condition states the converse.

It is standard to define a lifting $\bar{T}\colon \mathsf{Coalg}(H) \to \mathsf{Coalg}(K)$ of T from a natural transformation $\lambda\colon TH \Rightarrow KT$, as follows:

$$\bar{T}(X, f) = (TX, \lambda_X \circ Tf) \qquad \bar{T}(h) = Th \tag{6.1}$$

for any H-coalgebra (X, f) and coalgebra morphism h. All of the (liftings in) determinization constructions considered in this paper arise from such natural transformations.

In [16] a more specific kind of determinization for TB-coalgebras was studied, arising from a natural transformation $\kappa\colon TB \Rightarrow KT$ and a monad (T, η, μ). This construction is an instance of (6.1), by setting $H = TB$ and $\lambda = \kappa \circ \mu B : TTB \Rightarrow KT$. We denote the lifting of T arising in this way by T^κ. Spelling out the details, for any TB-coalgebra (X, f) we have:

$$T^\kappa(X, f) = (TX \xrightarrow{Tf} TTBX \xrightarrow{\mu_{BX}} TBX \xrightarrow{\kappa_X} KTX) . \tag{6.2}$$

For examples see, e.g., [16] and the end of this section.

The same type of natural transformation can be used to determinize BT-coalgebras. This is again an instance of (6.1), where $H = BT$ and $\lambda = K\mu \circ \kappa T : TBT \Rightarrow KT$. We denote the lifting of T arising in this way by T_κ. Spelling out the details, for any BT-coalgebra (X, f) we have:

$$T_\kappa(X, f) = (TX \xrightarrow{Tf} TBTX \xrightarrow{\kappa_{TX}} KTTX \xrightarrow{K\mu_X} KTX) . \tag{6.3}$$

This is considered in [33, 16] for the case where $B = K$ and κ is a distributive law of monad over functor.

In Theorem 6.4 below, we give a sufficient condition for the logical semantics on TB or BT-coalgebras to coincide with a logical semantics on determinized K-coalgebras, for the determinization constructions T^κ and T_κ. The same condition was recently studied in [8, Lemma 5.11], where it (more precisely, its mate) was proved to hold, under some additional assumptions, if α arises from the algebra $\mu_1 : TT1 \to T1$ as described in Section 5. First, we prove a general result, in the setting of (6.1), which relates the logical semantics of an H-coalgebra to a logical semantics for the K-coalgebra obtained by applying the lifting $\bar{T}$.

Lemma 6.3. *Assume natural transformations* $\lambda\colon TH \Rightarrow KT$, $\delta\colon LF \Rightarrow FH$, $\theta\colon LF \Rightarrow FK$ *and* $\alpha\colon F \Rightarrow FT$ *such that* λ *is a morphism of logics from* $\theta \odot \alpha$ *to* $\alpha \odot \delta$.

Let $s^\flat$ *be the logical semantics of* δ *on a coalgebra* $f\colon X \to HX$. *Then* $\alpha^\flat_\Phi \circ Ts^\flat$ *is the logical semantics of* θ *on the coalgebra* $(TX, \lambda_X \circ Tf)$.

Proof. Consider the following diagram:

$$
\begin{array}{ccccccc}
TX & \xrightarrow{\quad Ts^\flat \quad} & TG\Phi & \xrightarrow{\ \alpha^\flat_\Phi\ } & G\Phi \\
{\scriptstyle Tf}\downarrow & & \downarrow{\scriptstyle TGa} & & \downarrow{\scriptstyle Ga} \\
THX & \xrightarrow{THs^\flat} THG\Phi & \xrightarrow{T\delta^\flat_\Phi} & TGL\Phi & \xrightarrow{\alpha^\flat_{L\Phi}} & GL\Phi \\
{\scriptstyle \lambda_X}\downarrow & & \downarrow{\scriptstyle \lambda_{G\Phi}} & & \| \\
KTX & \xrightarrow{KTs^\flat} & KTG\Phi & \xrightarrow{K\alpha^\flat_\Phi} & KG\Phi & \xrightarrow{\theta^\flat_\Phi} & GL\Phi
\end{array}
$$

which commutes, clockwise starting from the top left: by definition of the logical semantics $s^\flat$, naturality of $\alpha^\flat$, the assumption that λ is a morphism of logics, and naturality of λ. Commutativity of the outside of the diagram means that $\alpha^\flat_\Phi \circ Ts^\flat$ is indeed the logical semantics of θ on $(TX, \lambda_X \circ Tf)$. $\square$

Theorem 6.4. *Assume:*

- *a monad (T, η, μ);*
- *a forgetful logic $\alpha \colon F \Rightarrow FT$, $\rho \colon LF \Rightarrow FB$, such that $\alpha^\flat$ is a T-action on G (Definition 5.1);*
- *a natural transformation $\kappa \colon TB \Rightarrow KT$, and*
- *another logic $\theta \colon LF \Rightarrow FK$,*

such that κ is a morphism of logics from $\theta \circledcirc \alpha$ to $\alpha \circledcirc \rho$, i.e., the following diagram commutes:

$$
\begin{array}{ccccc}
TBG & \overset{T\rho^\flat}{\Longrightarrow} & TGL & \overset{\alpha^\flat L}{\Longrightarrow} & GL \\
\kappa G \Big\Downarrow & & & & \Big\| \\
KTG & \underset{K\alpha^\flat}{\Longrightarrow} & KG & \underset{\theta^\flat}{\Longrightarrow} & GL.
\end{array}
$$

Let $s^\flat$ be the semantics of $\alpha \circledcirc \rho$ on a coalgebra $f \colon X \to TBX$, and let $s_\theta^\flat$ be the semantics of θ on $T^\kappa(X, f)$ (see (6.2)). Then $s^\flat = s_\theta^\flat \circ \eta_X$.

The same holds for the determinization procedure T_κ (see (6.3)) for BT-coalgebras and the logic $\rho \circledcirc \alpha$.

Proof. We first consider TB-coalgebras. Our aim is to use Lemma 6.3, instantiated to $H = TB$, $\delta = \alpha \circledcirc \rho$ and $\lambda = \kappa \circ \mu B$. To this end, we must show that λ is a morphism of logics from $\theta \circledcirc \alpha$ to $\alpha \circledcirc \alpha \circledcirc \rho$, i.e., the outside of the following diagram should commute:

$$
\begin{array}{ccccc}
TTBG & \overset{TT\rho^\flat}{\Longrightarrow} & TTGL & \overset{T\alpha^\flat L}{\Longrightarrow} & TGL \\
\mu BG \Big\Downarrow & & \mu GL \Big\Downarrow & & \Big\Downarrow \alpha^\flat L \\
TBG & \overset{T\rho^\flat}{\Longrightarrow} & TGL & \overset{\alpha^\flat L}{\Longrightarrow} & GL \\
\kappa G \Big\Downarrow & & & & \Big\| \\
KTG & \underset{K\alpha^\flat}{\Longrightarrow} & KG & \underset{\theta^\flat}{\Longrightarrow} & GL
\end{array}
$$

Indeed, the diagram commutes, clockwise starting from the top left: by naturality of μ, by the assumption that $\alpha^\flat$ is a T-action, and by the assumption that κ is a morphism of logics from $\theta \circledcirc \alpha$ to $\alpha \circledcirc \rho$ (in fact, since $\alpha^\flat$ is a T-action, μ is a morphism of logics from α to $\alpha \circledcirc \alpha$, hence μB is a morphism from $\alpha \circledcirc \rho$ to $\alpha \circledcirc \alpha \circledcirc \rho$, and the above diagram is obtained by composing this morphism with κ).

By Lemma 6.3, for any TB-coalgebra (X, f) we obtain

$$
s_\theta^\flat = \alpha_\Phi^\flat \circ T s^\flat
$$

where $s^\flat$ is the logical semantics of $\alpha \circledcirc \rho$ on (X, f) and $s_\theta^\flat$ is the logical semantics of θ on $T^\kappa(X, f) = (TX, \kappa_X \circ \mu_{BX} \circ Tf)$. Hence $s_\theta^\flat \circ \eta_X = \alpha_\Phi^\flat \circ T s^\flat \circ \eta_X = \alpha_\Phi^\flat \circ \eta_{G\Phi} \circ s^\flat = s^\flat$ where the second equality holds by naturality of η, and the third since $\alpha^\flat$ is a T-action.

For BT-coalgebras, we instantiate Lemma 6.3 to $H = BT$, $\delta = \rho \odot \alpha$ and $\lambda = K\mu \circ \kappa T$. We prove that λ is a morphism of logics:

$$
\begin{array}{ccccc}
TBTG & \overset{TB\alpha^\flat}{\Longrightarrow} & TBG & \overset{T\rho^\flat}{\Longrightarrow} & TGL \\
\kappa TG \big\Downarrow & & \big\Downarrow \kappa G & & \\
& KTTG & \overset{KT\alpha^\flat}{\Longrightarrow} & KTG & \\
K\mu G \big\Downarrow & & & \big\Downarrow K\alpha^\flat & \big\Downarrow \alpha^\flat L \\
& KTG & \underset{K\alpha^\flat}{\Longrightarrow} & KG & \underset{\theta^\flat}{\Longrightarrow} GL
\end{array}
$$

The diagram commutes, clockwise starting from the top left: by naturality of κ, the assumption that by κ is a morphism of logics and by the assumption that $\alpha^\flat$ is a T-action (again, the above diagram amounts to a composition of the logic morphisms μ and κ). By Lemma 6.3, for any BT-coalgebra (X, f) we obtain $s_\theta^\flat = \alpha_\Phi^\flat \circ Ts^\flat$ where $s^\flat$ is the logical semantics of $\rho \odot \alpha$ on (X, f) and $s_\theta^\flat$ is the logical semantics of θ on $T_\kappa(X, f) = (TX, K\mu_X \circ \kappa_{TX} \circ Tf)$. The conclusion of the proof is analogous to the above case of TB-coalgebras. □

We apply Theorem 6.4 to obtain a sufficient condition for a determinization construction to be correct with respect to a trace semantics given by a forgetful logic. The main idea is to choose θ to be an expressive logic for K, so that logical equivalence coincides with behavioural equivalence.

Corollary 6.5. *Let (T, η, μ), α, ρ, θ and κ be as in Theorem 6.4, and suppose that θ is an expressive logic. Then the determinization procedure T^κ of TB-coalgebras (6.2) is correct with respect to $\alpha \odot \rho$, and the determinization procedure T_κ of BT-coalgebras (6.3) is correct with respect to $\rho \odot \alpha$.*

Proof. Let $\bar{T}$ be either T^κ or T_κ, let (X, f) be a TB-coalgebra or a BT-coalgebra respectively, and $s^\flat$ the semantics of the forgetful logic on f. Under the above assumptions, by Theorem 6.4 we have $s^\flat = s_\theta^\flat \circ \eta_X$, where $s_\theta^\flat$ is the logical semantics of θ on $\bar{T}(X, f)$. Since $s_\theta^\flat$ is a logical semantics it factors through any coalgebra homomorphism, yielding condition (1) of correctness, and since it is expressive it decomposes as a coalgebra homomorphism followed by a mono, yielding condition (2). □

To illustrate all this, we show that the determinization of weighted automata as given in [16] is correct with respect to weighted language equivalence. (There is no such result for tree automata, as they do not determinize.)

Example 6.6. Fix a semiring $\mathbb{S}$, let $B = A \times - + 1$ and $K = \mathbb{S} \times -^A$. Consider $\kappa \colon \mathcal{M}B \Rightarrow K\mathcal{M}$ defined as follows [16]: $\kappa_X(\varphi) = (\varphi(*), \lambda a.\lambda x.\varphi(a, x))$. This induces a determinization procedure $\mathcal{M}^\kappa$ as in (6.2), for weighted automata. Let $\alpha \odot \rho$ be the forgetful logic for weighted automata introduced in Example 4.6, and recall that the logical semantics on a weighted automaton is the usual notion of acceptance of weighted languages. We use Corollary 6.5 to prove that the determinization procedure $\mathcal{M}^\kappa$ is correct with respect to $\alpha \odot \rho$. To this end, consider the logic $\theta^\flat \colon \mathbb{S} \times (\mathbb{S}^-)^A \Rightarrow \mathbb{S}^{A \times - + 1}$ given by the isomorphism, similar to the logic in Example 4.4. Since $\theta^\flat$ is componentwise injective, θ is expressive. Moreover, $\alpha^\flat$ is an action (see Section 5). The only remaining condition is commutativity of the diagram in Theorem 6.4, which is a straightforward calculation. This proves correctness of the determinization $\mathcal{M}^\kappa$ with respect to the semantics of $\alpha \odot \rho$.

Example 6.7. In [33] it is shown how to determinize non-deterministic automata of the form $B\mathcal{P}_\omega$, where $BX = 2 \times X^A$, based on $\kappa = \langle \kappa^o, \kappa^t \rangle \colon \mathcal{P}_\omega(2 \times -^A) \Rightarrow 2 \times (\mathcal{P}_\omega -)^A$ (note that $B = K$ in this example) where $\kappa^o_X(S) = \mathsf{tt}$ iff $\exists t.(\mathsf{tt}, t) \in S$, and $\kappa^t_X(a) = \{x \mid x \in t(a) \text{ for some } (o, t) \in S\}$. In Example 4.4 we have seen an expressive logic ρ and an α so that the logical semantics of $\rho \odot \alpha$ yields the usual language semantics. It is now straightforward to check that the determinization κ together with the logics ρ, α above satisfies the condition of Theorem 6.4, where $\theta = \rho$. By Corollary 6.5 this shows the expected result that determinization of non-deterministic automata is correct with respect to language semantics.

Moreover, recall that the logic $\rho \odot \beta$, where β is as defined in Example 4.4, yields a conjunctive semantics. Take the natural transformation $\tau = \langle \tau^o, \tau^t \rangle$ of the same type as κ, where $\tau^o(S) = \mathsf{tt}$ iff $o = \mathsf{tt}$ for every $(o, t) \in S$, and $\tau^t = \kappa^t$. Using Corollary 6.5 we can verify that this determinization procedure is correct.

One can also get the finite trace semantics of transition systems (Example 4.4) by turning them into non-deterministic automata (then, B and K are different).

Example 6.8. Alternating automata (see Example 4.5) can be determinized into non-deterministic automata in a process that preserves the language semantics. We shall now see how this arises as an application of Theorem 6.4 without a reference to final semantics of coalgebras.

This problem is more subtle than it may look, and our solution from a previous version of this paper [21] suffered from a serious mistake. We therefore present a corrected solution a little more elaborately.

Let $BX = 2 \times X^A$ as before. We wish to determinize $B\mathcal{P}_\omega\mathcal{P}_\omega$-coalgebras where, according to Example 4.5, the outer $\mathcal{P}_\omega$ is interpreted disjunctively (as in nondeterministic automata), and the inner $\mathcal{P}_\omega$ is interpreted conjunctively (as in co-nondeterministic automata). The result should be a nondeterministic automaton, i.e., a $B\mathcal{P}_\omega$-coalgebra with the $\mathcal{P}_\omega$ interpreted disjunctively. If the original alternating automaton had a carrier X, then the nondeterministic automaton should have a carrier $\mathcal{P}_\omega X$, interpreted conjunctively.

To model all this, we should instantiate Theorem 6.4 so that $T = \mathcal{P}_\omega$, the functors B and K from the theorem are both $B\mathcal{P}_\omega = 2 \times (\mathcal{P}_\omega -)^A$, the logic α from the theorem is instantiated to β from Example 4.5 and the logics ρ and θ from the theorem are both instantiated to $\rho \odot \alpha$ from Example 4.5.

We then need to provide an appropriate natural transformation $\kappa \colon \mathcal{P}_\omega B\mathcal{P}_\omega \Rightarrow B\mathcal{P}_\omega\mathcal{P}_\omega$ that would model the intended determinization procedure. It is natural to define it as a composition of two steps:

$$\kappa = B\chi \circ \tau\mathcal{P}_\omega \colon \mathcal{P}_\omega B\mathcal{P}_\omega \Rightarrow B\mathcal{P}_\omega\mathcal{P}_\omega$$

where $\tau \colon \mathcal{P}_\omega B \Rightarrow B\mathcal{P}_\omega$ is defined as at the end of Example 6.7. As argued there, this distributes B over $\mathcal{P}_\omega$ according to the conjunctive interpretation modeled by the logic β.

The question remains how to define $\chi \colon \mathcal{P}_\omega\mathcal{P}_\omega \Rightarrow \mathcal{P}_\omega\mathcal{P}_\omega$ so that it distributes the "conjunctive" powerset over the "disjunctive" one as intended. One may attempt (as we did in [21]):

$$\chi_X(S) = \{\overrightarrow{g}(S) \mid g \colon S \to X \text{ s.t. } g(U) \in U \text{ for each } U \in S\} \tag{6.4}$$

which, given a family of sets S, returns all possible sets obtained by choosing a single element from each set in S.

Unfortunately this is *not* a natural transformation. The following counterexample is due to Joost Winter: consider

$$X = \{a, b, c\} \qquad Y = \{d, e\}$$
$$f\colon X \to Y \qquad f(a) = f(b) = d \qquad f(c) = e$$

and calculate:

$$(\mathcal{P}_\omega \mathcal{P}_\omega f)(\chi_X(\{\{a,c\}, \{b,c\}\})) = (\mathcal{P}_\omega \mathcal{P}_\omega f)(\{\{a,b\}, \{a,c\}, \{b,c\}, \{c\}\}) = \{\{d\}, \{d,e\}, \{e\}\}$$
$$\chi_Y((\mathcal{P}_\omega \mathcal{P}_\omega f)(\{\{a,c\}, \{b,c\}\})) = \chi_Y(\{\{d,e\}\}) = \{\{d\}, \{e\}\}$$

therefore the naturality square for χ on $f\colon X \to Y$ does not commute for the argument $\{\{a,c\}, \{b,c\}\} \in \mathcal{P}_\omega \mathcal{P}_\omega X$.

This mistake has occured in the literature before. In [27, pp. 183-184], (6.4) was claimed to be a distributive law of the powerset monad over itself, and a monad structure on the double (covariant) powerset was derived from that claim in the standard way. The above counterexample applies there, and the structure defined in [27] is not, in fact, a monad. The mistake can be traced back to [26, pp. 76-79], where the same purported monad structure is defined via a Kleisli triple which, in fact, fails to satisfy one of the axioms of Kleisli triples.

To avoid these problems, we choose another definition of χ: given a finite family S of finite sets, it returns all sets that:

- are contained in the union of S (and are therefore finite, and there are finitely many of them), and
- intersect with every set in S.

Formally:

$$\chi_X(S) = \left\{ V \subseteq \bigcup S \mid V \cap U \neq \emptyset \text{ for each } U \in S \right\}. \tag{6.5}$$

This turns out to be a natural transformation (see Appendix C). As a side remark, note that χ is *not* a distributive law of the monad $\mathcal{P}_\omega$ over itself; it does not even satisfy the unit axioms.

Now the composition $\kappa = B\chi \circ \tau\mathcal{P}_\omega$ yields a determinization procedure. To show that the diagram in Theorem 6.4 commutes, a key ingredient is the fact that χ correctly distributes conjunction over disjunction or, more formally, that the diagram

$$
\begin{array}{ccc}
\mathcal{P}_\omega \mathcal{P}_\omega G & \overset{\mathcal{P}_\omega \alpha^\flat}{\Longrightarrow} & \mathcal{P}_\omega G \\
{\scriptstyle \chi G} \Big\| & & \searrow {\scriptstyle \beta^\flat} \\
\mathcal{P}_\omega \mathcal{P}_\omega G \underset{\mathcal{P}_\omega \beta^\flat}{\Longrightarrow} & \mathcal{P}_\omega G \underset{\alpha^\flat}{\Longrightarrow} & G
\end{array}
\tag{6.6}
$$

commutes, where α and β are as in Example 4.5. (See Appendix C for details.)

By Theorem 6.4 we conclude that for any alternating automaton: $s^\flat = s^\flat_{\rho \odot \alpha} \circ \eta_X$ where X is the set of states, $s^\flat$ is the language semantics of the alternating automaton as in Example 4.5, and $s^\flat_{\rho \odot \alpha}$ is the usual language semantics (as in Example 4.4) of the non-deterministic automaton obtained by determinization.

The determinization procedure formalized by our χ (and therefore κ) is slightly different from the standard procedure for transforming alternating automata into nondeterministic ones. It still returns an automaton where states are sets of states of the original alternating automaton, but the family of successors of each such state is larger than in the standard procedure (in particular, it is closed under taking set-theoretic unions). This does not change

the language semantics (additional successor states, being supersets of some successor states in the standard nondeterministic automaton, do not accept any additional words), but the reachable part of the nondeterministic automaton may be larger than the one obtained by the standard procedure. As a result, although our determinization is correct, it may be less efficient than the standard one. This is the price we paid to salvage the naturality of χ. We do not know how to model precisely the standard transformation of alternating automata into nondeterministic ones in our framework.

7. LOGICS WHOSE MATES ARE ISOMORPHISMS

Corollary 6.5 provides a sufficient condition for a given determinization procedure to be correct with respect to a forgetful logic. However, in general there is no guarantee that a correct determinization procedure for a given logic exists. Indeed it would be quite surprising if it did: the language semantics of (weighted) tree automata (see Example 4.6) is an example of a forgetful logic, and such automata are well known not to determinize in a classical setting.

In this section we provide a sufficient condition for a correct determinization procedure to exist. Specifically, for an endofunctor B, we assume a logic ρ whose mate $\rho^\flat \colon BG \Rightarrow GL$ is a natural isomorphism. This condition holds, for instance, for ρ in Example 4.4 and for θ in Example 6.6. It has been studied before in the context of determinization constructions [16]. Its important consequence is that the logical semantics $s^\flat$ in (2.5) from Section 3 can be seen as a B-coalgebra morphism:

$$
\begin{array}{ccc}
BX & \xrightarrow{\;Bs^\flat\;} & BG\Phi \\
\Big\uparrow & & \Big\uparrow{\scriptstyle(\rho^\flat_\Phi)^{-1}} \\
{\scriptstyle h}\Big\uparrow & & GL\Phi \\
\Big\uparrow & & \Big\uparrow{\scriptstyle Ga} \\
X & \xrightarrow[\;s^\flat\;]{} & G\Phi
\end{array}
\tag{7.1}
$$

Moreover, as shown in [16, Lemma 6] (see also [13]), the construction mapping

$$
LA \xrightarrow{\;g\;} A \qquad \text{to} \qquad GA \xrightarrow{\;Gg\;} GLA \xrightarrow{\;(\rho^\flat_A)^{-1}\;} BGA
$$

defines a functor $\hat{G} \colon \mathsf{Alg}(L) \to \mathsf{Coalg}(B)$, which is a contravariant adjoint to $\hat{F}$ (see (2.3) in Section 3). As a result, $\hat{G}$ maps initial objects to final ones, hence $\hat{G}(\Phi, a) = (G\Phi, (\rho^\flat_\Phi)^{-1} \circ Ga)$ is a final B-coalgebra. Therefore, $s^\flat$ is a final coalgebra morphism from (X, h).

7.1. **Canonical determinization.** The setting of a forgetful logic α, ρ where the mate of ρ is a natural isomorphism gives rise to the following diagrams:

$$
\begin{array}{ccccc}
\mathsf{Coalg}(TB) & \xrightarrow{\tilde{F}} & \mathsf{Alg}(L) & \underset{\hat{F}}{\overset{\hat{G}}{\rightleftarrows}} & \mathsf{Coalg}(B) \\
\downarrow & & \downarrow & & \downarrow \\
\mathcal{C} & \xrightarrow{F} & \mathcal{D} & \underset{F}{\overset{G}{\rightleftarrows}} & \mathcal{C}
\end{array}
\qquad
\begin{array}{ccccc}
\mathsf{Coalg}(BT) & \xrightarrow{\tilde{F}'} & \mathsf{Alg}(L) & \underset{\hat{F}}{\overset{\hat{G}}{\rightleftarrows}} & \mathsf{Coalg}(B) \\
\downarrow & & \downarrow & & \downarrow \\
\mathcal{C} & \xrightarrow{F} & \mathcal{D} & \underset{F}{\overset{G}{\rightleftarrows}} & \mathcal{C}
\end{array}
$$

(7.2)

The functor $\tilde{F}$ arises from the logic $\alpha \odot \rho$, the functor $\tilde{F}'$ arises from $\rho \odot \alpha$, the functor $\hat{F}$ arises from ρ and its contravariant adjoint $\hat{G}$ from the fact that $\rho^\flat$ is iso. Note that we make no assumptions on α; in particular, $\alpha^\flat$ need not be an action.

The composition $\hat{G}\tilde{F}$ is a determinization procedure, turning a coalgebra $f \colon X \to TBX$ into a B-coalgebra with carrier GFX. Explicitly, $\hat{G}\tilde{F}(X, f)$ is

$$
GFX \xrightarrow{GFf} GFTBX \xrightarrow{G\alpha_{BX}} GFBX \xrightarrow{G\rho_X} GLFX \xrightarrow{(\rho^\flat)^{-1}_{FX}} BGFX .
\qquad (7.3)
$$

Similarly, the composition $\hat{G}\tilde{F}'$ is a determinization procedure, mapping a coalgebra $g \colon X \to BTX$ to

$$
GFX \xrightarrow{GFg} GFBTX \xrightarrow{G\rho_{TX}} GLFTX \xrightarrow{GL\alpha_X} GLFX \xrightarrow{(\rho^\flat)^{-1}_{FX}} BGFX .
\qquad (7.4)
$$

These determinization procedures are correct with respect to the logics $\alpha \odot \rho$ and $\rho \odot \alpha$ respectively, in the following sense, much stronger than required by Definition 6.2.

Theorem 7.1. *For any TB-coalgebra (X, f), the logical semantics $s^\flat$ of $\alpha \odot \rho$ on (X, f) coincides with the final semantics of the B-coalgebra $\hat{G}\tilde{F}(X, f)$ precomposed with the unit $\iota \colon \mathsf{Id} \Rightarrow GF$. The same holds for BT-coalgebras and $\rho \odot \alpha$.*

Proof. Let (X, f) be a TB-coalgebra (the case of BT-coalgebras is analogous). The logical semantics $s^\flat$ is defined by $s^\flat = Gs \circ \iota_X$, where $s \colon (L, a) \to \tilde{F}(X, f)$ is the unique algebra morphism arising by initiality. Hence, we only need to show that Gs is the final semantics of $\hat{G}\tilde{F}(X, f)$. To see this, apply $\hat{G}$ to s to get a coalgebra morphism

$$
Gs \colon \hat{G}\tilde{F}(X, f) \to \hat{G}(L, a) .
$$

But $\hat{G}(L, a)$ is a final coalgebra, so we are done. $\qquad\square$

7.2. **Determinization after preprocessing.** Strictly speaking, the above constructions $\hat{G}\tilde{F}$ and $\hat{G}\tilde{F}'$ are not examples of determinization procedures as understood in [16]: the functors $\hat{G}\tilde{F}$ and $\hat{G}\tilde{F}'$ lift GF rather than T, and the liftings do not arise from a natural transformation as described in Section 6. However, they are *almost* examples: after an encoding of TB-coalgebras as GFB-coalgebras and of BT-coalgebras as BGF-coalgebras, they arise from a distributive law $\kappa \colon GFB \Rightarrow BGF$.

Indeed, recall from Section 3 that $\alpha \colon F \Rightarrow FT$ uniquely determines a natural transformation $\alpha^\dagger \colon T \Rightarrow GF$. Furthermore, this induces functors $\mathsf{Coalg}(BT) \xrightarrow{B\alpha^\dagger \circ -} \mathsf{Coalg}(BGF)$ and $\mathsf{Coalg}(TB) \xrightarrow{\alpha^\dagger B \circ -} \mathsf{Coalg}(GFB)$, by simple composition.
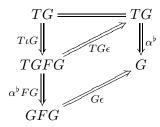
Now, define a canonical logic for GF by

$$\varepsilon = \epsilon F : F \Rightarrow FGF, \qquad \text{or equivalently,} \qquad \varepsilon^\flat = G\epsilon : GFG \Rightarrow G.$$

(Note that $\varepsilon^\flat$ is *always* a GF-action on G.) Given a logic ρ for B this gives rise to forgetful logics $\rho \odot \varepsilon$ and $\varepsilon \odot \rho$, for GFB-coalgebras and BGF-coalgebras respectively. Note that these logics do not depend on α anymore.

Lemma 7.2. *For any $\rho\colon LF \Rightarrow FB$ and $\alpha\colon F \Rightarrow FT$, the natural transformation $\alpha^\dagger B$ is a morphism of logics (with* id *as the translation of syntax) from $\varepsilon \odot \rho$ to $\alpha \odot \rho$, and $B\alpha^\dagger$ is a morphism from $\rho \odot \varepsilon$ to $\rho \odot \alpha$.*

Proof. It suffices to prove that $\alpha^\dagger$ is a morphism of logics from ε to α. This, recalling the definition of $\alpha^\dagger$ from Section 3, means that the outside of the following diagram should commute:

$$
\begin{array}{ccc}
TG & =\!=\!=\!=\!= & TG \\
{\scriptstyle T\iota G}\big\Downarrow & \overset{TG\epsilon}{\nearrow} & \big\Downarrow{\scriptstyle \alpha^\flat} \\
TGFG & & G \\
{\scriptstyle \alpha^\flat FG}\big\Downarrow & \overset{G\epsilon}{\nearrow} & \\
GFG & &
\end{array}
$$

which follows from naturality of $\alpha^\flat$ and from (2.1).   □

As a consequence of Lemma 2.3 and the above, we have commuting diagrams

$$
\begin{array}{ccccccc}
\mathsf{Coalg}(TB) & \xrightarrow{\ \alpha^\dagger B \circ -\ } & \mathsf{Coalg}(GFB) & \qquad & \mathsf{Coalg}(BT) & \xrightarrow{\ B\alpha^\dagger \circ -\ } & \mathsf{Coalg}(BGF) \\
{\scriptstyle \bar{F}}\searrow & & \swarrow{\scriptstyle \bar{F}_\varepsilon} & & {\scriptstyle \bar{F}'}\searrow & & \swarrow{\scriptstyle \bar{F}'_\varepsilon} \\
& \mathsf{Alg}(L) & & & & \mathsf{Alg}(L) &
\end{array}
\qquad (7.5)
$$

where $\bar{F}_\varepsilon$ is the functor defined by $\varepsilon \odot \rho$ as in (2.3), and $\bar{F}'_\varepsilon$ the functor defined by $\rho \odot \varepsilon$. Hence, encoding TB-coalgebras as GFB-coalgebras does not change their logical semantics. More precisely, for any $f\colon X \to TBX$, the map from X to $G\Phi$ defined as the logical semantics of $\varepsilon \odot \rho$ on $\alpha^\dagger_{BX} \circ f$, coincides with semantics of $\alpha \odot \rho$ on f. A similar result holds for BT-coalgebras.

Thanks to the mate $\rho^\flat\colon BG \Rightarrow GL$ being an isomorphism, the functor GF has a distributive law over B, denoted $\kappa\colon GFB \Rightarrow BGF$ and defined by:
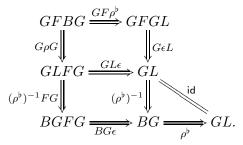
$$GFB \overset{G\rho}{=\!=\!=\!\Longrightarrow} GLF \overset{(\rho^\flat)^{-1}F}{=\!=\!=\!=\!\Longrightarrow} BGF. \qquad (7.6)$$

Remark 7.3. The same construction is used in [18] to prove that a functor lifting to the category of Eilenberg-Moore algebras for a monad induces a distributive law. There, the right adjoint G is the (covariant) forgetful functor; having a lifting of B then means having a functor L with the equality $BG = GL$.

Using κ we can apply the determinization construction from [16] as described in Section 6, putting $K = B$, and taking the monad on GF arising from the adjunction.

Lemma 7.4. *The determinization procedure $(GF)^\kappa$ defined as in (6.2) is correct with respect to $\varepsilon \odot \rho$. The determinization procedure $(GF)_\kappa$ is correct with respect to $\rho \odot \varepsilon$.*

Proof. We use Corollary 6.5, where we put $T = GF$, $\alpha = \varepsilon$ defined above, $K = B$, and $\theta = \rho$. Obviously then θ is expressive, and it is easy to check that $\varepsilon^\flat = G\epsilon$ is an action. The only remaining condition is that κ is a morphism of logics as in Theorem 6.4, which is the outer shape of:

$$
\begin{array}{ccc}
GFBG & \xmapsto{GF\rho^\flat} & GFGL \\
{\scriptstyle G\rho G} \big\Downarrow & & \big\Downarrow {\scriptstyle G\epsilon L} \\
GLFG & \xRightarrow{GL\epsilon} & GL \\
{\scriptstyle (\rho^\flat)^{-1}FG} \big\Downarrow & {\scriptstyle (\rho^\flat)^{-1}} \big\Downarrow & \Big\Downarrow {\scriptstyle \text{id}} \\
BGFG & \xRightarrow[BG\epsilon]{} & BG \xRightarrow[\rho^\flat]{} GL.
\end{array}
$$

Here, the top square commutes by Lemma 2.2 and the bottom square by naturality of $(\rho^\flat)^{-1}$. $\qquad\square$

Altogether, the following two-step determinization procedures arise:

$$
\begin{array}{ccccc}
\mathsf{Coalg}(TB) \xmapsto{\alpha^\dagger B\circ-} \mathsf{Coalg}(GFB) \xmapsto{(GF)^\kappa} \mathsf{Coalg}(B) & & & \mathsf{Coalg}(BT) \xmapsto{B\alpha^\dagger\circ-} \mathsf{Coalg}(BGF) \xmapsto{(GF)_\kappa} \mathsf{Coalg}(B) \\
\Big\downarrow \qquad\qquad \Big\downarrow \qquad\qquad\quad \Big\downarrow & & & \Big\downarrow \qquad\qquad \Big\downarrow \qquad\qquad\quad \Big\downarrow \\
\mathcal{C} \xrightarrow{\ \mathsf{Id}\ } \mathcal{C} \xrightarrow{\ GF\ } \mathcal{C} & & & \mathcal{C} \xrightarrow{\ \mathsf{Id}\ } \mathcal{C} \xrightarrow{\ GF\ } \mathcal{C}
\end{array}
$$

and they are correct with respect to $\alpha \odot \rho$ and $\rho \odot \alpha$ respectively.

Correctness can also be proved without Corollary 6.5, using Theorem 7.1, since the procedures coincide with the constructions from (7.3) and (7.4):

Theorem 7.5. *The following diagrams commute:*

$$
\begin{array}{ccc}
\mathsf{Coalg}(TB) \xmapsto{\alpha^\dagger B\circ-} \mathsf{Coalg}(GFB) & \qquad & \mathsf{Coalg}(BT) \xmapsto{B\alpha^\dagger\circ-} \mathsf{Coalg}(BGF) \\
{\scriptstyle \tilde{F}} \Big\downarrow \qquad\qquad \Big\downarrow {\scriptstyle (GF)^\kappa} & & {\scriptstyle \tilde{F}'} \Big\downarrow \qquad\qquad \Big\downarrow {\scriptstyle (GF)_\kappa} \\
\mathsf{Alg}(L) \xrightarrow[\hat{G}]{} \mathsf{Coalg}(B) & & \mathsf{Alg}(L) \xrightarrow[\hat{G}]{} \mathsf{Coalg}(B).
\end{array}
$$

with $\tilde{F}$ and $\tilde{F}'$ and $\hat{G}$ defined as in (7.2).

Proof. By Lemma 7.2, $\alpha^\dagger B$ and $B\alpha^\dagger$ are morphisms of logics, which means that the diagrams in (7.5) commute. Hence, we only need to check commutativity of:

$$
\begin{array}{ccc}
\qquad \mathsf{Coalg}(GFB) & \qquad\qquad & \qquad \mathsf{Coalg}(BGF) \\
{\scriptstyle \bar{F}_\varepsilon} \swarrow \quad \Big\downarrow {\scriptstyle (GF)^\kappa} & & {\scriptstyle \bar{F}'_\varepsilon} \swarrow \quad \Big\downarrow {\scriptstyle (GF)_\kappa} \\
\mathsf{Alg}(L) \xrightarrow[\hat{G}]{} \mathsf{Coalg}(B) & & \mathsf{Alg}(L) \xrightarrow[\hat{G}]{} \mathsf{Coalg}(B).
\end{array}
$$

For the left triangle, notice that $\bar{F}_\varepsilon$ maps any $f\colon X \to GFBX$ to

$$
LFX \xrightarrow{\ \rho_X\ } FBX \xrightarrow{\ \epsilon_{FBX}\ } FGFBX \xrightarrow{\ Ff\ } FX
$$

and applying $\hat{G}$ yields

$$
GFX \xrightarrow{\ GFf\ } GFGFBX \xrightarrow{\ G\epsilon_{FBX}\ } GFBX \xrightarrow{\ G\rho_X\ } GLFX \xrightarrow{\ (\rho^\flat)^{-1}_{FX}\ } BGFX
$$

which coincides with $(GF)^\kappa(X, f)$. Note that $G\epsilon F$ is the multiplication of the monad GF.

For the right triangle, we compute $\tilde{F}'_\varepsilon(g\colon X \to BGFX)$:

$$LFX \xrightarrow{L\epsilon_{FX}} LFGFX \xrightarrow{\rho_{GFX}} FBGFX \xrightarrow{Fg} FX$$

and apply $\hat{G}$, yielding:

$$GFX \xrightarrow{GFg} GFBGFX \xrightarrow{G\rho_{GFX}} GLFGFX \xrightarrow{GL\epsilon_{FX}} GLFX \xrightarrow{(\rho^\flat)^{-1}_{FX}} BGFX. \qquad (7.7)$$

Further, $(GF)_\kappa(g\colon X \to BGFX)$ is:

$$GFX \xrightarrow{GFg} GFBGFX \xrightarrow{G\rho_{GFX}} GLFGFX \xrightarrow{(\rho^\flat)^{-1}_{FGFX}} BGFGFX \xrightarrow{BG\epsilon_{FX}} BGFX$$

which coincides with (7.7) by naturality of $(\rho^\flat)^{-1}$. $\qquad\square$

7.3. **A connection to Brzozowski's algorithm.** Call a B-coalgebra *observable* if the morphism into a final coalgebra (assuming it exists) is mono [3]. For $\mathcal{D} = \mathsf{Set}$, the above canonical determinization procedure can be adapted to construct, for any TB-coalgebra, an observable B-coalgebra whose final semantics coincides with the logical semantics on the original one.

Indeed, recall that Set has an (epi,mono)-factorization system and L (as every endofunctor on Set) preserves epimorphisms. From this it follows that every L-algebra homomorphism decomposes as a surjective homomorphism followed by an injective one. Given a coalgebra $f\colon X \to TBX$, apply this to decompose the algebra homomorphism $s\colon (\Phi, a) \to \tilde{F}(X, f)$ as $s = m \circ e$, where m and e are injective and surjective respectively; call the L-algebra in the middle (R, r). Recall that Gs is a coalgebra homomorphism into the final coalgebra. In the present situation it decomposes as follows:

$$\hat{G}\tilde{F}(X, f) \xrightarrow[Gm]{} \hat{G}(R, r) \xrightarrow[Ge]{} \hat{G}(\Phi, a)$$

with Gs spanning the outer arc.

and recall that $\hat{G}(\Phi, a)$ is a final coalgebra. Because G is a right adjoint, it maps epis to monos, therefore Ge is injective and $\hat{G}(R, r)$ is observable. Moreover, thanks to Theorem 7.1 we have $s^\flat = Gs \circ \iota_X = Ge \circ Gm \circ \iota_X$, hence the final semantics Ge of $\hat{G}(R, r)$ coincides with the logical semantics on (X, f) along the mapping $Gm \circ \iota_X$.

Note that the construction of $\hat{G}(R, r)$ from (X, f) is not a determinization procedure itself according to Definition 6.1, as it does not lift any functor on $\mathcal{C}$.

The above refers to TB-coalgebras, but as everything else in this section, analogous reasoning works also for BT-coalgebras. For $T = \mathsf{Id}$ and $B = 2 \times -^A$ on $\mathcal{C} = \mathsf{Set}$, that (almost) corresponds to Brzozowski's algorithm for minimization of deterministic automata [4]. Applying $\tilde{F}$ to the given automaton corresponds to reversing transitions and turning final states into initial ones. Epi-mono factorization corresponds to taking the *reachable* part of this automaton. Then, applying $\hat{G}$ reverses transitions again, and turns initial states into final ones. Our abstract approach stops here; the original algorithm concludes by taking the reachable part again, which ensures minimality.

For a more detailed coalgebraic presentation of the full Brzozowski minimization algorithm in several concrete examples, see [3]. Another approach, based on duality theory, is

presented in [2]. The main idea there is similar, in the sense that contravariant adjunctions are lifted to adjunctions between categories of coalgebras and algebras. However, it differs from the above development in that the adjunctions used in [2] are assumed to be dual equivalences, and the lifting of the duality is proved concretely for each example, rather than that a general condition is given. Another coalgebraic approach to minimization, based on partition refinement, is in [1]. It is mentioned in the conclusion that part of Brzozowski's algorithm appears as an instance of the abstract construction introduced there, but the precise connection remains to be understood.

Notice that we only assume the mate of ρ to be iso; there are no requirements on α. The mate of ρ is iso for the logic from Example 4.4. Thus, we can instantiate α to obtain observable deterministic automata from non-deterministic automata or even alternating automata (by taking $T = \mathcal{P}_\omega \mathcal{P}_\omega$ and, for α, the composition of α and β from Example 4.5). The logic θ from Example 6.6 is covered as well, so one can treat Moore automata and weighted automata. However, the abstract construction of an observable automaton does not necessarily yield a concrete algorithm, as discussed for the case of weighted automata in [3].

References

[1] J. Adámek, F. Bonchi, M. Hülsbusch, B. König, S. Milius, and A. Silva. A coalgebraic perspective on minimization and determinization. In *Procs. FoSSaCS 2012*, volume 7213 of *LNCS*, pages 58–73, 2012.

[2] N. Bezhanishvili, C. Kupke, and P. Panangaden. Minimization via duality. In *Procs. WoLLIC 2012*, volume 7456 of *LNCS*, pages 191–205, 2012.

[3] F. Bonchi, M. M. Bonsangue, H. H. Hansen, P. Panangaden, J. J. M. M. Rutten, and A. Silva. Algebra-coalgebra duality in Brzozowski's minimization algorithm. *ACM Trans. Comput. Log.*, 15(1):3, 2014.

[4] J. Brzozowski. Canonical regular expressions and minimal state graphs for definite events. *Mathematical Theory of Automata*, 12:529–561, 1962.

[5] L-T. Chen and A. Jung. On a categorical framework for coalgebraic modal logic. *Electr. Notes Theor. Comput. Sci.*, 308:109–128, 2014.

[6] C. Cîrstea. From branching to linear time, coalgebraically. In *Procs. FICS 2013*, volume 126 of *EPTCS*, pages 11–27, 2013.

[7] C. Cîrstea. A coalgebraic approach to linear-time logics. In *Procs. FoSSaCS 2014*, volume 8412 of *LNCS*, pages 426–440, 2014.

[8] C. Cirstea. Canonical Coalgebraic Linear Time Logics. In *Procs. CALCO 2015*, volume 35 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 66–85, 2015.

[9] C. Cîrstea and D. Pattinson. Modular construction of modal logics. In *Procs. CONCUR 2004*, volume 3170 of *LNCS*, pages 258–275, 2004.

[10] E. J. Dubuc. *Kan extensions in enriched category theory*. Lecture notes in mathematics. Springer, 1970.

[11] S. Goncharov. Trace semantics via generic observations. In *Procs. CALCO 2013*, volume 8089 of *LNCS*, pages 158–174, 2013.

[12] I. Hasuo, B. Jacobs, and A. Sokolova. Generic trace semantics via coinduction. *Log. Meth. Comp. Sci.*, 3(4), 2007.

[13] C. Hermida and B. Jacobs. Structural induction and coinduction in a fibrational setting. *Inf. and Comp.*, 145:107–152, 1997.

[14] B. Jacobs. Introduction to coalgebra. Towards mathematics of states and observations, 2014. Draft.

[15] B. Jacobs and J. Mandemaker. Coreflections in algebraic quantum logic. *Foundations of Physics*, 42(7):932–958, 2012.

[16] B. Jacobs, A. Silva, and A. Sokolova. Trace semantics via determinization. *J. Comput. Syst. Sci.*, 81(5):859–879, 2015.

[17] B. Jacobs and A. Sokolova. Exemplaric expressivity of modal logics. *J. Log. and Comput.*, 20(5):1041–1068, 2010.

[18] P. Johnstone. Adjoint lifting theorems for categories of algebras. *Bulletin of the London Mathematical Society*, 7:294–297, 1975.
[19] C. Kissig and A. Kurz. Generic trace logics. *CoRR*, abs/1103.3239, 2011.
[20] B. Klin. Coalgebraic modal logic beyond sets. *ENTCS*, 173:177–201, 2007.
[21] B. Klin and J. Rot. Coalgebraic trace semantics via forgetful logics. In *FoSSaCS 2015. Proceedings*, pages 151–166, 2015.
[22] C. Kupke and D. Pattinson. Coalgebraic semantics of modal logics: An overview. *Theor. Comput. Sci.*, 412(38):5070–5094, 2011.
[23] A. Kurz, S. Milius, D. Pattinson, and L. Schröder. Simplified coalgebraic trace equivalence. In *Software, Services, and Systems - Essays Dedicated to Martin Wirsing on the Occasion of His Retirement from the Chair of Programming and Software Engineering*, pages 75–90, 2015.
[24] S. Mac Lane. *Categories for the working mathematician*, volume 5. Springer, 1998.
[25] M. Lenisa, J. Power, and H. Watanabe. Category theory for operational semantics. *Theor. Comput. Sci.*, 327(1-2):135–154, 2004.
[26] E. Manes. Monads of sets. *Handbook of algebra*, 3:67–153, 2003.
[27] E. Manes and P. Mulry. Monad compositions I: general constructions and recursive distributive laws. *Theory and Applications of Categories*, 18(7):172–208, 2007.
[28] S. Milius, D. Pattinson, and L. Schröder. Generic Trace Semantics and Graded Monads. In *Procs. CALCO 2015*, volume 35 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 253–269, 2015.
[29] D. Pavlovic, M. Mislove, and J. Worrell. Testing semantics: Connecting processes and process logics. In *AMAST 2006*, volume 4019 of *LNCS*, pages 308–322, 2006.
[30] J. Power and D. Turi. A coalgebraic foundation for linear time semantics. *ENTCS*, 29:259–274, 1999.
[31] J. J. M. M. Rutten. Universal coalgebra: a theory of systems. *Theor. Comput. Sci.*, 249(1):3–80, 2000.
[32] L. Schröder and D. Pattinson. Modular algorithms for heterogeneous modal logics via multi-sorted coalgebra. *Math. Struct. in Comp. Sci.*, 21(2):235–266, 2011.
[33] A. Silva, F. Bonchi, M. M. Bonsangue, and J. J. M. M. Rutten. Generalizing determinization from automata to coalgebras. *Log. Meth. Comp. Sci.*, 9(1), 2013.
[34] R.J. van Glabbeek, S.A. Smolka, and B. Steffen. Reactive, generative, and stratified models of probabilistic processes. *Information and Computation*, 121(1):59 – 80, 1995.

Appendix A. Details of Section 4

In this section we expand on some of the examples considered in Section 4.

Example 4.1. We spell out the details of the semantics of the logic $\alpha \odot \rho$ on an automaton $f \colon X \to \mathcal{P}_\omega BX$:

$$
\begin{aligned}
s^\flat(x)(\varepsilon) = \mathsf{tt} &\iff \alpha^\flat_{LA^*}(\mathcal{P}_\omega \rho^\flat_{A^*}((\mathcal{P}_\omega B s^\flat)(f(x))))(*) = \mathsf{tt} \\
&\iff \exists \varphi \in \mathcal{P}_\omega \rho^\flat_{A^*}((\mathcal{P}_\omega B s^\flat)(f(x))).\ \varphi(*) = \mathsf{tt} \\
&\iff \exists t \in f(x).\ \rho^\flat_{A^*}(Bs^\flat(t))(*) = \mathsf{tt} \\
&\iff * \in f(x)
\end{aligned}
$$

where ε is the empty word, and for all $a \in A$ and $w \in A^*$:

$$
\begin{aligned}
s^\flat(x)(aw) = \mathsf{tt} &\iff \exists t \in f(x).\ \rho^\flat_{A^*}(Bs^\flat(t))(a, w) = \mathsf{tt} \\
&\iff \exists t \in f(x).\ Bs^\flat(t) = (a, \varphi) \wedge \varphi(w) = \mathsf{tt} \\
&\iff \exists y \in X.\ (a, y) \in f(x) \wedge s^\flat(y)(w) = \mathsf{tt}
\end{aligned}
$$

Example 4.5. Spelling out $s^\flat$ yields $s^\flat(x)(\varepsilon) = o(x)$, and

$$s^\flat(x)(aw) = \mathtt{tt} \iff \alpha^\flat_{A^*}(\mathcal{P}_\omega\beta^\flat_{A^*}((\mathcal{P}_\omega\mathcal{P}_\omega s^\flat)(f(x)(a))))(w) = \mathtt{tt}$$
$$\iff \exists\varphi \in \mathcal{P}_\omega\beta^\flat_{A^*}((\mathcal{P}_\omega\mathcal{P}_\omega s^\flat)(f(x)(a))).\ \varphi(w) = \mathtt{tt}$$
$$\iff \exists U \in (\mathcal{P}_\omega\mathcal{P}_\omega s^\flat)(f(x)(a)).\ \forall\varphi \in U.\ \varphi(w) = \mathtt{tt}$$
$$\iff \exists S \in f(x)(a).\ \forall y \in f(x)(a).\ s^\flat(y)(w) = \mathtt{tt}$$

Example 4.6. In order to understand the semantics $s^\flat$ of the forgetful logic on a tree automaton, we first compute the composite logic $\alpha^\flat_\Sigma \circ \mathcal{M}\rho^\flat \colon \mathcal{M}\Sigma\mathbb{S}^- \Rightarrow S^\Sigma$:

$$(\alpha^\flat_{\Sigma\Phi} \circ \mathcal{M}\rho^\flat_\Phi(\varphi))(\sigma(w_1, \ldots, w_n)) = \sum_{\psi \in \mathbb{S}^{\Sigma\Phi}} (\mathcal{M}\rho^\flat_\Phi(\varphi))(\psi) \cdot \psi(\sigma(w_1, \ldots, w_n))$$
$$= \sum_{\psi \in \mathbb{S}^{\Sigma\Phi}} \sum_{\gamma \in \rho^{\flat\,-1}_\Phi(\psi)} \varphi(\gamma) \cdot \psi(\sigma(w_1, \ldots, w_n))$$
$$= \sum_{\varphi_1, \ldots, \varphi_n \in \mathbb{S}^\Phi} \varphi(\sigma(\varphi_1, \ldots, \varphi_n)) \cdot \prod_{i=1..n} \varphi_i(w_i)$$

The next step is to instantiate this to $\Sigma^*\emptyset$ and precompose with $\mathcal{M}\Sigma s^\flat$:

$$(\alpha^\flat_{\Sigma\Sigma^*\emptyset} \circ \mathcal{M}\rho^\flat \circ \mathcal{M}\Sigma s^\flat(\psi))(\sigma(t_1, \ldots, t_n))$$
$$= \sum_{\varphi_1, \ldots, \varphi_n \in \mathbb{S}^{\Sigma^*\emptyset}} (\mathcal{M}\Sigma s^\flat(\psi))(\sigma(\varphi_1, \ldots, \varphi_n)) \cdot \prod_{i=1..n} \varphi_i(t_i)$$
$$= \sum_{\substack{\varphi_1, \ldots, \varphi_n \in \mathbb{S}^{\Sigma^*\emptyset} \\ x_1 \in s^{\flat\,-1}(\varphi_1) \\ \overset{\cdots}{x_n \in s^{\flat\,-1}(\varphi_n)}}} \psi(\sigma(x_1, \ldots, x_n)) \cdot \prod_{i=1..n} \varphi_i(t_i)$$
$$= \sum_{x_1, \ldots, x_n \in X} \psi(\sigma(x_1, \ldots, x_n)) \cdot \prod_{i=1..n} s^\flat(x_i)(t_i)$$

It follows that the diagram (4.5) commutes if and only if for all $\sigma(t_1, \ldots t_n)$ and all $x \in X$:

$$s^\flat(x)(\sigma(t_1, \ldots, t_n)) = \sum_{x_1, \ldots, x_n \in X} f(x)(\sigma(x_1, \ldots, x_n)) \cdot \prod_{i=1..n} s^\flat(x_i)(t_i)$$

which is the semantics presented in Example 4.6.

The map $s^\flat$ is computed by induction; this is done by turning a weighted tree automaton f into the Σ-algebra $\hat{F}(f)$, which can be viewed as a *deterministic weighted bottom-up tree automaton*. We spell out the details. Given a coalgebra $f\colon X \to \mathcal{M}\Sigma X$ the computed Σ-algebra looks as follows:

$$\Sigma(\mathbb{S}^X) \xrightarrow{\rho_X} \mathbb{S}^{\Sigma X} \xrightarrow{\alpha_{\Sigma X}} \mathbb{S}^{\mathcal{M}\Sigma X} \xrightarrow{\mathbb{S}^f} \mathbb{S}^X$$

We have

$$\alpha_X(\varphi)(\psi) = \sum_{x \in X} \varphi(x) \cdot \psi(x)$$

and $\rho = \rho^\flat$:

$$(\mathbb{S}^{\Sigma\eta_X} \circ \mathbb{S}^{\rho^\flat_{\mathbb{S}X}} \circ \eta_{\Sigma\mathbb{S}X}(\sigma(\varphi_1,\ldots,\varphi_n)))(\tau(x_1,\ldots,x_m))$$

$$= (\mathbb{S}^{\rho^\flat_{\mathbb{S}X}\circ\Sigma\eta_X} \circ \lambda\psi.\psi(\sigma(\varphi_1,\ldots,\varphi_n)))(\tau(x_1,\ldots,x_m))$$

$$= (\rho^\flat_{\mathbb{S}X} \circ \Sigma\eta_X(\tau(x_1,\ldots,x_m)))(\sigma(\varphi_1,\ldots,\varphi_n))$$

$$= \rho^\flat_{\mathbb{S}X}(\tau(\lambda\varphi.\varphi(x_1),\ldots,\lambda\varphi.\varphi(x_m)))(\sigma(\varphi_1,\ldots,\varphi_n))$$

$$= \begin{cases} \prod_{i=1..n} \varphi_i(x_i) & \text{if } \sigma = \tau \\ 0 & \text{otherwise} \end{cases}$$
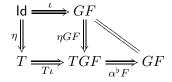
The algebra is as follows:

$$\hat{F}(X,f)(\sigma(\varphi_1,\ldots,\varphi_n))(x) = (\mathbb{S}^f \circ \alpha_{\Sigma X} \circ \rho_X(\sigma(\varphi_1,\ldots,\varphi_n)))(x)$$

$$= (\alpha_{\Sigma X} \circ \rho_X(\sigma(\varphi_1,\ldots,\varphi_n)))(f(x))$$

$$= \sum_{t\in\Sigma X} f(x)(t) \cdot \rho_X(\sigma(\varphi_1,\ldots,\varphi_n))(t)$$

$$= \sum_{x_1,\ldots,x_n\in X} f(x)(\sigma(x_1,\ldots,x_n)) \cdot \prod_{i=1..n} \varphi_i(x_i)$$

Appendix B. Proof of Lemma 5.2

Recall the definitions:

$$\alpha^\dagger = \alpha^\flat F \circ T\iota, \qquad \alpha^\flat = G\epsilon \circ \alpha^\dagger G.$$

First let us assume that $\alpha^\flat : TG \Rightarrow G$ is a monad action, and check the axioms of monad morphisms for $\alpha^\dagger$. The unit axiom is the outer shape of:

where the square commutes by naturality of η, and the triangle by a monad action axiom. The multiplication axiom is the outer shape of:

where everything commutes, from left to right: by naturality of μ, by a monad action axiom, by (2.1) and by naturality of $\alpha^\flat$.

Now assume that $\alpha^\dagger \colon T \Rightarrow GF$ is a monad morphism, and check the axioms of monad actions for $\alpha^\flat$. The unit axiom is the outer shape of:

$$
\begin{array}{ccc}
 & G & \\
 \overset{\eta G}{\swarrow} \quad \downarrow{\iota G} & & \searrow \\
 TG \underset{\alpha^\dagger G}{\Longrightarrow} GFG \underset{G\epsilon}{\Longrightarrow} G
\end{array}
$$

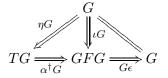where the left triangle is the unit axiom of a monad morphism, and the right triangle is (2.1). The multiplication axiom is the outer shape of:

$$
\begin{array}{ccccc}
TTG & \xRightarrow{T\alpha^\dagger G} & TGFG & \xRightarrow{TG\epsilon} & TG \\
 & & {\scriptstyle \alpha^\dagger GFG}\Big\Downarrow & & \Big\Downarrow{\scriptstyle \alpha^\dagger G} \\
{\scriptstyle \mu G}\Big\Downarrow & & GFGFG & \underset{GFG\epsilon}{\Longrightarrow} & GFG \\
 & & {\scriptstyle G\epsilon FG}\Big\Downarrow & & \Big\Downarrow{\scriptstyle G\epsilon} \\
TG & \underset{\alpha^\dagger G}{\Longrightarrow} & GFG & \underset{G\epsilon}{\Longrightarrow} & G
\end{array}
$$

where the leftmost shape is the multiplication axiom of a monad morphism, and the two smaller squares commute by naturality of $\alpha^\dagger$ and ϵ.

APPENDIX C. DETAILS OF EXAMPLES IN SECTION 6

Example 6.6. The condition from Theorem 6.4 is commutativity of the following diagram:

$$
\begin{array}{ccccc}
\mathcal{M}(A \times \mathbb{S}^- + 1) & \xRightarrow{\mathcal{M}\rho^\flat} & \mathcal{M}(\mathbb{S}^{A \times - +1}) & \xRightarrow{\alpha^\flat_{A \times - +1}} & \mathbb{S}^{A \times - +1} \\
{\scriptstyle \kappa_{\mathbb{S}^-}}\Big\Downarrow & & & & \Big\| \\
\mathbb{S} \times (\mathcal{M}\mathbb{S}^-)^A & \xRightarrow{\mathsf{id} \times (\alpha^\flat)^A} & \mathbb{S} \times (\mathbb{S}^-)^A & \xRightarrow{\theta^\flat} & \mathbb{S}^{A \times - +1}
\end{array}
$$

Indeed, we have

$$K\alpha^\flat_\Phi \circ \kappa_{\mathbb{S}^\Phi}(\varphi) = (\varphi(*), \lambda a.\alpha^\flat(\lambda\psi.\varphi(a,\psi))) = (\varphi(*), \lambda a.\lambda w. \sum_{\psi \in \mathbb{S}^\Phi} \varphi(a,\psi) \cdot \psi(w))$$

and thus

$$(\theta^\flat_\Phi \circ K\alpha^\flat_\Phi \circ \kappa_{\mathbb{S}^\Phi}(\varphi))(*) = \varphi(*)$$

$$(\theta^\flat_\Phi \circ K\alpha^\flat_\Phi \circ \kappa_{\mathbb{S}^\Phi}(\varphi))(a,w) = \sum_{\psi \in \mathbb{S}^\Phi} \varphi(a,\psi) \cdot \psi(w)$$

which coincides with $\alpha^\flat_{A \times \Phi + 1} \circ \mathcal{M}\delta^\flat_\Phi$ as computed (in a more general setting) in Appendix A.

Example 6.7. We treat the determinization $\langle \tau^o, \tau^t \rangle$ described in the example. The relevant condition of Theorem 6.4 instantiates to commutativity of:

$$
\begin{array}{ccc}
\mathcal{P}_\omega(2 \times (2^-)^A) & \xrightarrow{\mathcal{P}_\omega\rho^\flat} \mathcal{P}_\omega(2^{A\times-+1}) \xrightarrow{\beta^\flat L} 2^{A\times-+1} \\
\Big\downarrow{\scriptstyle\langle\tau^o_{2-},\tau^t_{2-}\rangle} & \Big\| \\
2 \times (\mathcal{P}_\omega(2^-))^A & \xrightarrow{\mathsf{id}\times(\beta^\flat)^A} 2 \times (2^-)^A \xrightarrow{\rho^\flat} 2^{A\times-+1}
\end{array}
$$

We have, for any set Φ:

$$
\begin{aligned}
(\beta^\flat_{L\Phi} \circ \mathcal{P}_\omega\rho^\flat_\Phi)(S)(*) = \mathsf{tt} &\iff \forall\varphi \in (\mathcal{P}_\omega\rho^\flat_\Phi)(S).\varphi(*) = \mathsf{tt} \\
&\iff \forall(o,t) \in S.o = \mathsf{tt} \\
&\iff \tau^o_{2\Phi}(S) = \mathsf{tt} \\
&\iff (\rho^\flat_\Phi \circ B\beta^\flat_\Phi \circ \langle\tau^o_{2\Phi}, \tau^t_{2\Phi}\rangle)(S)(*) = \mathsf{tt}
\end{aligned}
$$

and for any $a \in A$, $w \in \Phi$:

$$
\begin{aligned}
(\beta^\flat_{L\Phi} \circ \mathcal{P}_\omega\rho^\flat_\Phi)(S)(a,w) = \mathsf{tt} &\iff \forall\varphi \in (\mathcal{P}_\omega\rho^\flat_\Phi)(S).\varphi(a,w) = \mathsf{tt} \\
&\iff \forall(o,t) \in S.t(a)(w) = \mathsf{tt} \\
&\iff \forall\varphi \in \tau^t_{2\Phi}(S)(a).\varphi(w) = \mathsf{tt} \\
&\iff \beta^\flat_\Phi(\tau^t_{2\Phi}(S)(a))(w) = \mathsf{tt} \\
&\iff ((\beta^\flat_\Phi)^A \circ \tau^t_{2\Phi})(S)(a)(w) = \mathsf{tt} \\
&\iff (\rho^\flat_\Phi \circ B\beta^\flat_\Phi \circ \langle\tau^o_{2\Phi}, \tau^t_{2\Phi}\rangle)(S)(a,w) = \mathsf{tt}
\end{aligned}
$$

which proves commutativity of the diagram.

Example 6.8. First, we check that χ defined by (6.5) is a natural transformation. To this end, for any function $f \colon X \to Y$, and a family $S \in \mathcal{P}_\omega\mathcal{P}_\omega X$, we need to check that

$$
\chi_Y((\mathcal{P}_\omega\mathcal{P}_\omega f)(S)) = (\mathcal{P}_\omega\mathcal{P}_\omega f)(\chi_X(S)).
$$

For any $W \subseteq Y$, calculate:

$$
\begin{aligned}
W \in \chi_Y((\mathcal{P}_\omega\mathcal{P}_\omega f)(S)) &\iff W \subseteq \bigcup(\mathcal{P}_\omega\mathcal{P}_\omega f)(S) \wedge \forall T \in (\mathcal{P}_\omega\mathcal{P}_\omega f)(S). W \cap T \neq \emptyset \\
&\iff W \subseteq \mathcal{P}_\omega f\left(\bigcup S\right) \wedge \forall U \in S. W \cap (\mathcal{P}_\omega f)(U) \neq \emptyset \quad\quad (\text{C.1})
\end{aligned}
$$

$$
\begin{aligned}
W \in (\mathcal{P}_\omega\mathcal{P}_\omega f)(\chi_X(S)) &\iff \exists V \in \chi_X(S). W = (\mathcal{P}_\omega f)(V) \\
&\iff \exists V \subseteq \bigcup S. (W = (\mathcal{P}_\omega f)(V) \wedge \forall U \in S. V \cap U \neq \emptyset) \quad (\text{C.2})
\end{aligned}
$$

For the second equivalence in (C.1), notice that

$$
\bigcup(\mathcal{P}_\omega\mathcal{P}_\omega f)(S) = \mathcal{P}_\omega f\left(\bigcup S\right)
$$

for any $f \colon X \to Y$ and $S \subseteq \mathcal{P}_\omega X$, since direct images preserve unions.

We need to show that (C.1) and (C.2) are equivalent for every W.

For the implication from (C.1) to (C.2), put

$$
V = \overleftarrow{f}(W) \cap \bigcup S.
$$

Then $W = (\mathcal{P}_\omega f)(V)$. Indeed, for the left-to-right containment, pick any $y \in W$. Then, since $W \subseteq \mathcal{P}_\omega f (\bigcup S)$, there is some $x \in \bigcup S$ such that $f(x) = y$. Obviously then $x \in \overleftarrow{f}(W)$, hence $x \in V$ and $y \in \mathcal{P}_\omega f(V)$. The right-to-left containment is equivalent to $\overleftarrow{f}(W) \supseteq V$, which follows directly from the definition of V.

Moreover, for any $U \in S$, by (C.1) there exists some $y \in W \cap (\mathcal{P}_\omega f)(U)$, so there is some $x \in U$ such that $f(x) = y$. Then $x \in \overleftarrow{f}(W)$. Obviously $x \in \bigcup S$ as well, so $x \in V$ and $V \cap U \neq \emptyset$.

For the implication from (C.2) to (C.1), take a $V \subseteq \bigcup S$ that exists by (C.2), and calculate:
$$W = (\mathcal{P}_\omega f)(V) \subseteq (\mathcal{P}_\omega f)\left(\bigcup S\right).$$
Furthermore, for any $U \in S$, by (C.2) we have $V \cap U \neq \emptyset$. Then calculate:
$$W \cap (\mathcal{P}_\omega f)(U) = (\mathcal{P}_\omega f)(V) \cap (\mathcal{P}_\omega f)(U) \supseteq (\mathcal{P}f)(V \cap U) \neq \emptyset.$$
We thus conclude that $\chi \colon \mathcal{P}_\omega \mathcal{P}_\omega \Rightarrow \mathcal{P}_\omega \mathcal{P}_\omega$ is a natural transformation.

We now prove that the diagram (6.6) commutes. Recall that, for any set X and any $S \in \mathcal{P}_\omega \mathcal{P}_\omega 2^X$, $U \in \mathcal{P}_\omega 2^X$ and $x \in X$:
$$\chi_{2^X}(S) \quad = \quad \{V \subseteq \bigcup S \mid \forall U \in S.\ V \cap U \neq \emptyset\}$$
$$\alpha_X^\flat(U)(x) = \mathsf{tt} \quad \Longleftrightarrow \quad \exists \phi \in S.\ \phi(x) = \mathsf{tt}$$
$$\beta_X^\flat(U)(x) = \mathsf{tt} \quad \Longleftrightarrow \quad \forall \phi \in S.\ \phi(x) = \mathsf{tt}$$
Then calculate, for any $S \in \mathcal{P}_\omega \mathcal{P}_\omega 2^X$ and $x \in X$:
$$\beta_X^\flat(\mathcal{P}_\omega \alpha_X^\flat(S))(x) = \mathsf{tt} \iff \forall \phi \in \mathcal{P}_\omega \alpha_X^\flat(S).\ \phi(x) = \mathsf{tt}$$
$$\iff \forall U \in S.\ \alpha_X^\flat(U)(x) = \mathsf{tt}$$
$$\iff \forall U \in S.\ \exists \phi \in U.\phi(x) = \mathsf{tt} \qquad\qquad \text{(C.3)}$$
$$\alpha_X^\flat(\mathcal{P}_\omega \beta_X^\flat(\chi_{2^X}(S)))(x) = \mathsf{tt} \iff \exists \phi \in \mathcal{P}_\omega \beta_X^\flat(\chi_{2^X}(S)).\ \phi(x) = \mathsf{tt}$$
$$\iff \exists V \in \chi_{2^X}(S).\ \beta_X^\flat(V)(x) = \mathsf{tt}$$
$$\iff \exists V \in \chi_{2^X}(S).\ \forall \phi \in V.\phi(x) = \mathsf{tt}$$
$$\iff \exists V \subseteq \bigcup S.(\forall U \in S.\ U \cap V \neq \emptyset) \wedge (\forall \phi \in V.\phi(x) = \mathsf{tt})$$
$$\text{(C.4)}$$
Conditions (C.3) and (C.4) are easily equivalent, therefore (6.6) commutes.

Finally, the condition of Theorem 6.4 in this case is that the following commutes:

$$
\begin{array}{ccccccc}
\mathcal{P}_\omega B \mathcal{P}_\omega G & \xRightarrow{\mathcal{P}_\omega B \alpha^\flat} & \mathcal{P}_\omega B G & \xRightarrow{\mathcal{P}_\omega \rho^\flat} & \mathcal{P}_\omega G L & \xRightarrow{\beta^\flat L} & GL \\
\Big\Downarrow{\scriptstyle \tau \mathcal{P}_\omega G} & & \Big\Downarrow{\scriptstyle \tau G} & & & {\scriptstyle \rho^\flat}\nearrow & \\
B \mathcal{P}_\omega \mathcal{P}_\omega G & \xRightarrow{B \mathcal{P}_\omega \alpha^\flat} & B \mathcal{P}_\omega G & \xRightarrow{B \beta^\flat} & BG & & \\
\Big\Downarrow{\scriptstyle B \chi G} & & & {\scriptstyle B \alpha^\flat}\nearrow & & & \\
B \mathcal{P}_\omega \mathcal{P}_\omega G & \xRightarrow{B \mathcal{P}_\omega \beta^\flat} & B \mathcal{P}_\omega G & & & &
\end{array}
$$

The square commutes by naturality of $\alpha^\flat$, and the upper right shape is the diagram for proving correctness of the determinization procedure τ considered in Example 6.7. The lower shape is (6.6) mapped by B.