Lifting Adjunctions to Coalgebras to (Re)Discover Automata Constructions

Henning Kerstan, Barbara König, Bram Westerbaan

Lifting Adjunctions to Coalgebras to (Re)Discover Automata Constructions

Henning Kerstan[1], Barbara König[1] and Bram Westerbaan[2]

[1] Universität Duisburg-Essen
{henning.kerstan, barbara_koenig}@uni-due.de
[2] Radboud Universiteit Nijmegen
bram.westerbaan@cs.ru.nl

Abstract. It is a well-known fact that a nondeterministic automaton can be transformed into an equivalent deterministic automaton via the powerset construction. From a categorical perspective this construction is the right adjoint to the inclusion functor from the category of deterministic automata to the category of nondeterministic automata. This is in fact an adjunction between two categories of coalgebras: deterministic automata are coalgebras over **Set** and nondeterministic automata are coalgebras over **Rel**. We will argue that this adjunction between coalgebras originates from a canonical adjunction between **Set** and **Rel**.
In this paper we describe how, in a quite generic setting, an adjunction can be lifted to coalgebras, and we compare some sufficient conditions. Then we illustrate this technique in length: we recover several constructions on automata as liftings of basic adjunctions including determinization of nondeterministic and join automata, codeterminization, and the dualization of linear weighted automata. Finally, we show how to use the lifted adjunction to check behavioral equivalence.

1 Introduction

Coalgebra offers a general framework for specifying transition systems with various branching types. Given a functor $F\colon \mathbf{Set} \to \mathbf{Set}$, describing the branching type of the transition system, an F-coalgebra is a function $c\colon X \to FX$, where X represents the state set and c the transition function. Depending on the choice of F, one can describe labeled, nondeterministic, probabilistic or various other types of branching and it is possible to combine several of them. Coalgebras come with a natural notion of behavioral equivalence, and coalgebra homomorphisms can be seen as functional bisimulations, mapping states to equivalent states.

In recent years, it has also become customary to study coalgebras in categories different from **Set**. There are several reasons, for instance, one can impose an algebraic structures on the states, or one can work in presheaf categories in order to model name passing [7]. Particularly relevant to this paper is the study of coalgebras in Kleisli categories, where a monad offers a way to model side-effects that can also be understood as implicit branching, different from the explicit branching that can be modeled directly in **Set**. Such coalgebras have for instance

been studied in [8], where nondeterministic automata are specified as coalgebras in **Rel**, the category of sets and relations, which is isomorphic to the Kleisli category of the powerset monad on **Set**. The behavioral equivalence induced by such coalgebras in **Rel** is indeed trace (or language) equivalence, as desired, and not bisimilarity.

When studying coalgebras in various categories a natural question to ask is how to transform such coalgebras from one representation into another. Our motivating examples come from the world of deterministic and nondeterministic automata where various forms of determinization can be seen as functors which map coalgebras living in one category, into coalgebras living in another category. For instance, nondeterministic automata living in **Rel** can be transformed into deterministic automata in **Set** via the powerset construction. In the other direction, a deterministic automaton in **Set** can be trivially regarded as a nondeterministic automaton in **Rel**. It turns out that the transformations together form an adjunction between categories of coalgebras where the powerset construction is the right adjoint. In the same vein various other determinization-like constructions arise as adjunctions.

In the following we will first show under which circumstances adjunctions on categories can be lifted to adjunctions on coalgebras. Part of the answer was already given by Hermida and Jacobs [9] and we extend their characterization by giving another, equivalent, condition. Then we study several examples in detail, especially various forms of automata. Apart from the well-known deterministic and nondeterministic automata, we consider codeterministic automata (also known as átomata, see [6], or backwards-deterministic automata) and deterministic join automata, i.e. automata that have an algebraic structure on the states, allowing to take the join of a given set of states. Such automata live in the category of join semilattices **JSL**, which is the Eilenberg-Moore category of the powerset monad on **Set** (whereas **Rel** is the Kleisli category of the powerset monad) and have already been considered in [17]. In total we consider four different adjunctions between such automata.

In addition we consider an adjunction in the realm of linear weighted automata, where we take up an example from [3], transforming input into output linear weighted automata (and vice versa).

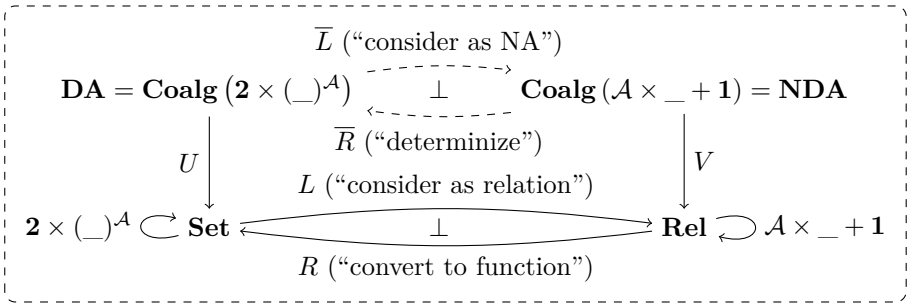
In order to explain what these adjunctions really mean in terms of behavioral equivalence, we study a general notion of behavioral equivalence for arbitrary categories. We first observe that the final coalgebra, if it exists, is preserved by right adjoints and hence can be "inherited" from coalgebras living in a different category. Furthermore we show how queries on behavioral equivalence can be translated to equivalent queries on coalgebras in another category. This reflects the well-known construction of determinizing a nondeterministic automaton before answering questions about language equivalence.

2 Theoretical Background – Lifting Adjunctions

Within this section we are first going to present a short, motivating example which introduces our approach. Then we will recall some basic definitions from the theory of adjunctions (mainly to introduce our notation) and start to develop our theory which is then summarized in our main theoretical result. The result itself is not very surprising and, in fact, was discovered already earlier by C. Hermida and B. Jacobs [9] in a different setting (we will compare our approach with their result) and can be obtained using standard (2-)categorical methods [12]. However, the focus of our work is *not just the theory itself* but we are more interested in *how this theory helps to understand* certain (algorithmic) constructions on automata by *applying* it to various types of automata, modeled as coalgebras.

2.1 Motivating Example

Consider the following (non-commutative) diagram of functors where the bottom part is a (canonical) adjunction (see below for a definition) between **Set** and **Rel**.



Let $\mathcal{A}$ be an alphabet, i.e. a finite set of labels. It is known (and we will also recall this in Section 3.1) that the coalgebras for the functor $\mathbf{2} \times (\_)^{\mathcal{A}}$ on **Set** are the deterministic automata (**DA**) and the coalgebras for the functor $\mathcal{A} \times \_ + \mathbf{1}$ on **Rel** are the nondeterministic automata (**NDA**). (For final coalgebra semantics of **NDA** see [8].) We aim at finding the functors $\overline{L}$, $\overline{R}$ (dashed arrows on top) that form an adjunction which is a *lifting* of the original adjunction and we will see that for this particular example everything works out as planned and the lifted right adjoint $\overline{R}$ "performs" the well-known powerset construction to determinize an **NDA**.

2.2 Adjunctions

We recall some basics from category theory [14,2] to fix our notation.

Definition 1 (Adjunction, Adjoint Functors). *Let* $\mathbf{C}$ *and* $\mathbf{D}$ *be categories. An* adjunction *between* $\mathbf{C}$ *and* $\mathbf{D}$ *consists of a functor* $L\colon \mathbf{C} \to \mathbf{D}$*, called* left adjoint*, a functor* $R\colon \mathbf{D} \to \mathbf{C}$*, called* right adjoint *and two natural transformations*

$\eta\colon 1_{\mathbf{C}} \Rightarrow RL$, *called* unit, *and* $\varepsilon\colon LR \Rightarrow 1_{\mathbf{D}}$, *called* counit, *satisfying*

$$\varepsilon L \circ L\eta = 1_L\,, \qquad \text{and} \qquad R\varepsilon \circ \eta R = 1_R\,. \tag{1}$$

We denote such an adjunction by $\langle L \dashv R, \eta, \varepsilon\rangle : \mathbf{C} \to \mathbf{D}$.

A functor $L\colon \mathbf{C} \to \mathbf{D}$ [$R\colon \mathbf{D} \to \mathbf{C}$] is a *left adjoint [right adjoint]* if it is the left adjoint [right adjoint] of some adjunction $\langle L \dashv R, \eta, \varepsilon\rangle : \mathbf{C} \to \mathbf{D}$. One can prove that L [R] determines the other parts of the adjunction unique up to isomorphism. Since this is not trivial, we will always give the full adjunction.

There are special cases of adjunctions which have their own names.

Definition 2 (Equivalence and Duality of Categories). *We call an adjunction* $\langle L \dashv R, \eta, \varepsilon\rangle : \mathbf{C} \to \mathbf{D}$ *an* equivalence *if both* η *and* ε *are natural isomorphisms. Whenever such an equivalence exists, the categories* $\mathbf{C}$ *and* $\mathbf{D}$ *are called* equivalent*. Similarly, we say that the categories* $\mathbf{C}$ *and* $\mathbf{D}$ *are* dually equivalent *if there is an equivalence* $\langle L \dashv R, \varepsilon, \eta\rangle : \mathbf{C} \to \mathbf{D}^{\mathrm{op}}$.

Let us now consider our first example of an adjunction (which is well-known in the literature and also the adjunction from our motivating example).

Example 1. Let $L\colon \mathbf{Set} \to \mathbf{Rel}$ be the inclusion functor from $\mathbf{Set}$ to $\mathbf{Rel}$ mapping each set X to itself and each function $f\colon X \to Y$ to the corresponding relation $f\colon X \leftrightarrow Y$. Moreover, let $R\colon \mathbf{Rel} \to \mathbf{Set}$ be the functor which maps each set X to its powerset $\mathbf{2}^X$ and each relation $f\colon X \leftrightarrow Y$ to the the function $Rf\colon \mathbf{2}^X \to \mathbf{2}^Y$ given by $(Rf)(S) = \{y \in Y \mid \exists x \in S : \langle x, y\rangle \in f\}$ for every $S \in \mathbf{2}^X$. We obtain an adjunction $\langle L \dashv R, \eta, \varepsilon\rangle : \mathbf{Set} \to \mathbf{Rel}$ where the unit $\eta\colon 1_{\mathbf{Set}} \Rightarrow \mathbf{2}^-$ is given by all the functions $\eta_X\colon X \to \mathbf{2}^X, \eta_X(x) = \{x\}$ for every set X and the counit $\varepsilon\colon \mathbf{2}^- \Rightarrow 1_{\mathbf{Rel}}$ consists of all the relations $\varepsilon_X\colon \mathbf{2}^X \leftrightarrow X$ where $\langle S, x\rangle \in \varepsilon_X \iff x \in S$ for every set X.

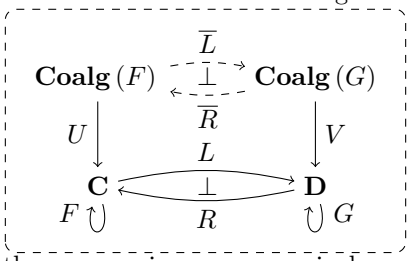
2.3 Lifting an Adjunction to Coalgebras

The theory of coalgebras [10,16] has proven to be an appropriate tool for modeling and analyzing various types of transition systems. We will examine several of the standard examples (deterministic, nondeterministic, codeterministic and linear weighted automata) in detail in the following sections and hence will not provide any examples here. Besides these examples also probabilistic automata [18] and even arbitrary labeled Markov processes [15] or probabilistic transition systems [13] can be seen as coalgebras in suitable categories.

Recently the coalgebraic treatment of automata has provided new views on algorithms for minimization and (co)determinization [1,4,3]. In these works the authors make use of certain adjunctions of categories to obtain the minimization of (various kinds of) automata.

We shall hereafter try to find and analyze a common and generic pattern on how we can make use of an adjunction $\langle L \dashv R, \eta, \varepsilon\rangle : \mathbf{C} \to \mathbf{D}$ to reason

about and to find constructions (algorithms) on automata modeled as coalgebras. For that purpose let us fix two endofunctors $F\colon \mathbf{C} \to \mathbf{C}$ and $G\colon \mathbf{D} \to \mathbf{D}$ and look at the (non-commutative) diagram of functors on the right where $U\colon \mathbf{Coalg}\,(F) \to \mathbf{C}$ and $V\colon \mathbf{Coalg}\,(G) \to \mathbf{D}$ are the forgetful functors mapping a coalgebra to its carrier and a coalgebra homomorphism to the underlying arrow.



The question we are interested in is whether we can in some canonical way obtain the functors $\overline{L}$ and $\overline{R}$ as indicated by the dashed lines such that they form an adjunction which "arises" from the initial adjunction. A precise definition for this is given below. In several cases such adjoint functors transform coalgebras in a way that we (re)discover algorithmic constructions on the modeled automata and we will back this hypothesis by the examples given in the following sections.

Definition 3 (Lifting). *Let* $\mathbf{C}$ *and* $\mathbf{D}$ *be categories,* $F\colon \mathbf{C} \to \mathbf{C}$, $G\colon \mathbf{D} \to \mathbf{D}$ *be endofunctors and* $U\colon \mathbf{Coalg}\,(F) \to \mathbf{C}$ *and* $V\colon \mathbf{Coalg}\,(G) \to \mathbf{D}$ *be the forgetful functors mapping a coalgebra to its carrier and a coalgebra morphism to the underlying arrow. Let* $\langle L \dashv R, \eta, \varepsilon\rangle : \mathbf{C} \to \mathbf{D}$ *be an adjunction.*

1. *We call a functor* $\overline{L}\colon \mathbf{Coalg}\,(F) \to \mathbf{Coalg}\,(G)$ $\left[\ \overline{R}\colon \mathbf{Coalg}\,(G) \to \mathbf{Coalg}\,(F)\ \right]$ *a* lifting *of* L $[R]$ *if it satisfies the equality* $V\overline{L} = LU$ $\left[\ U\overline{R} = RV\ \right]$.
2. *We call an adjunction* $\left\langle \overline{L} \dashv \overline{R}, \overline{\eta}, \overline{\varepsilon}\right\rangle : \mathbf{Coalg}\,(F) \to \mathbf{Coalg}\,(G)$ *a* lifting *of* $\langle L \dashv R, \eta, \varepsilon\rangle : \mathbf{C} \to \mathbf{D}$ *if* $\overline{L}$ *is a lifting of* L, $\overline{R}$ *is a lifting of* R *and we have* $U\overline{\eta} = \eta$ *and* $V\overline{\varepsilon} = \varepsilon$.

Although this definition is straightforward it has one setback: it does not tell us how to construct a lifted adjunction. Let us therefore introduce a method for handling this. If we had a natural transformation $\alpha\colon LF \Rightarrow GL$ it is not hard to see that we obtain a functor $\overline{L}\colon \mathbf{Coalg}\,(F) \to \mathbf{Coalg}\,(G)$ by defining

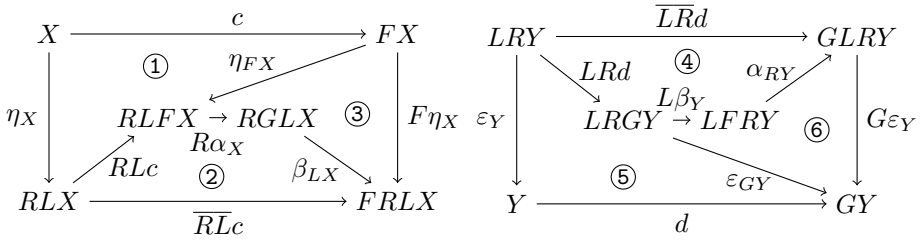
$$\overline{L}\left(X \xrightarrow{c} FX\right) = \left(LX \xrightarrow{Lc} LFX \xrightarrow{\alpha_X} GLX\right), \quad \overline{L}f = Lf \tag{2}$$

for all F-coalgebras $c\colon X \to FX$ and all F-coalgebra homomorphisms f and analogously, given a natural transformation $\beta\colon RG \Rightarrow FR$ we can define a functor $\overline{R}\colon \mathbf{Coalg}\,(G) \to \mathbf{Coalg}\,(F)$ by

$$\overline{R}\left(Y \xrightarrow{d} GY\right) = \left(RY \xrightarrow{Rd} RGY \xrightarrow{\beta_Y} FRY\right), \quad \overline{R}g = Rg \tag{3}$$

for all G-coalgebras $d\colon Y \to GY$ and all G-coalgebra homomorphisms g. By definition these functors are liftings and thus the only remaining question is whether we obtain a lifting of the adjunction. The equation $U\overline{\eta} = \eta$ can be spelled out as the requirement that for all F-coalgebras $c\colon X \to FX$ the arrow $\eta_X\colon X \to RLX$ is an F-coalgebra homomorphism $c \to \overline{RL}c$ and likewise the equation $V\overline{\varepsilon} = \varepsilon$ translates to the requirement that for every G-coalgebra $d\colon Y \to GY$ the arrow $\varepsilon_Y\colon LRY \to Y$ is a G-coalgebra homomorphism $\overline{LR}d \to d$.

This is the case iff the outer rectangles of the following two diagrams commute.



These diagrams certainly commute if their inner parts commute: ① commutes because η is a natural transformation, ② by definition of $\overline{RL}c$ and commutativity of ③ is equivalent to $F\eta_X = \beta_{LX} \circ R\alpha_X \circ \eta_{FX}$. Moreover, ④ commutes by definition of $\overline{LR}d$ and ⑤ because ε is a natural transformation. Finally, the commutativity of ⑥ is equivalent to $\varepsilon_{GY} = G\varepsilon_Y \circ \alpha_{RY} \circ L\beta_Y$.

With these observations at hand it is easy to spell out a sufficient condition for the existence of a lifting which we will do in the following theorem.

Theorem 1 (Lifting an Adjunction to Coalgebras). *Let $F\colon \mathbf{C} \to \mathbf{C}$ and $G\colon \mathbf{D} \to \mathbf{D}$ be endofunctors and $\langle L \dashv R, \eta, \varepsilon\rangle : \mathbf{C} \to \mathbf{D}$ be an adjunction. There is a lifting $\langle \overline{L} \dashv \overline{R}, \overline{\eta}, \overline{\varepsilon}\rangle : \mathbf{Coalg}\,(F) \to \mathbf{Coalg}\,(G)$ of the adjunction if one of the following equivalent conditions is fulfilled.*

(i) There are two natural transformations $\alpha\colon LF \Rightarrow GL$ and $\beta\colon RG \Rightarrow FR$ satisfying the following equalities.

$$F\eta = \beta L \circ R\alpha \circ \eta F \tag{4}$$
$$\varepsilon G = G\varepsilon \circ \alpha R \circ L\beta \tag{5}$$

(ii) There is a natural isomorphism $\beta\colon RG \Rightarrow FR$. [9, 2.15 Corollary]

If (i) *holds, the* adjoint mate $\alpha^\bullet$ *of α, which is defined as*

$$\alpha^\bullet := RG\varepsilon \circ R\alpha R \circ \eta FR \tag{6}$$

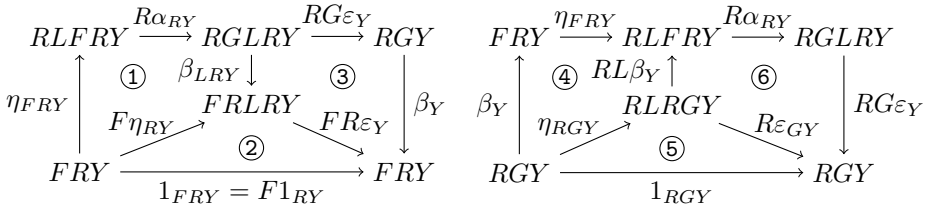
is the inverse of β. Conversely, if (ii) *holds we can define α as the adjoint mate $(\beta^{-1})^\bullet$ of β^{-1} which is defined as*

$$(\beta^{-1})^\bullet = \varepsilon GL \circ L\beta^{-1}L \circ LF\eta\,. \tag{7}$$

In both cases $\overline{L}$ and $\overline{R}$ are defined by (2) *and* (3).

By the observations from above it should be quite clear, that (i) is sufficient for a lifting to exist. The fact that the second condition (ii) of this theorem is also sufficient for the existence of a lifting is due to a result by C. Hermida and B. Jacobs [9, 2.15 Corollary]. They derive this as a "by-product" from a quite generic result in 2-categories using the fact that coalgebras are certain inserters in the 2-category **CAT** of categories, functors and natural transformations. Thus in order to prove the theorem we just have to show that (i) and (ii) are equivalent using the provided definitions of β^{-1} (6) and α (7).

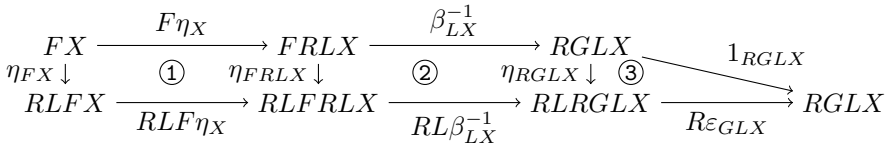
Proof (of Theorem 1).
(i) ⇒ (ii): The equations $\beta_Y \circ \alpha^\bullet_Y = 1_{FRY}$ and $\alpha^\bullet_Y \circ \beta_Y = 1_{RGY}$ are equivalent to commutativity of the outer rectangles of the following diagrams.



The diagrams commute because their inner parts commute: For the left diagram ① is (4) applied to $X = RY$, ② is F applied to the second unit-counit equation (1) and ③ is the natural transformation diagram for β. For the right diagram we observe that ④ is the natural transformation diagram for η, ⑤ is the second unit-counit equation (1) applied to GY and ⑥ is R applied to (5). Thus β is indeed a natural isomorphism with inverse $\alpha^\bullet$.

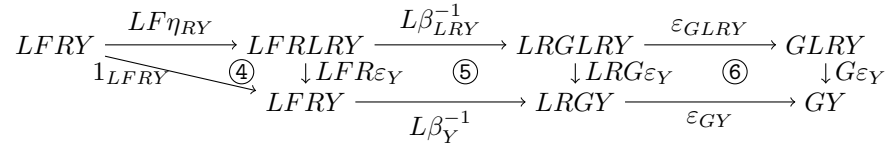
(ii) ⇒ (i): We have to show that α defined by (7) satisfies (4) and (5).

(4)*:* Let X be an arbitrary $\mathbf{C}$-object. Then $F\eta_X = \beta_{LX} \circ R\alpha_X \circ \eta_{FX}$ holds if and only if $\beta^{-1}_{LX} \circ F\eta_X = R\varepsilon_{GLX} \circ RL\beta^{-1}_{LX} \circ RLF\eta_X \circ \eta_{FX}$ holds which in turn is equivalent to commutativity of the outer part of the following diagram.



① and ② commute because η is a natural transformation from $1_{\mathbf{C}}$ to RL, functors preserve inverses and ③ is the second unit-counit equation (1) applied to GLX.

(5)*:* Let Y be an arbitrary $\mathbf{D}$-object. Then: $\varepsilon_{GY} = G\varepsilon_Y \circ \alpha_{RY} \circ L\beta_Y$ holds if and only if $\varepsilon_{GY} \circ L\beta_Y^{-1} = G\varepsilon_Y \circ \left(\varepsilon_{GLRY} \circ L\beta^{-1}_{LRY} \circ LF\eta_{RY}\right)$ holds which in turn is equivalent to commutativity of the outer part of the following diagram.



④ commutes by applying LF to the second unit-counit equation (1) , ⑤ due to the fact that β^{-1} is a natural transformation and ⑥ because ε is a natural transformation. □

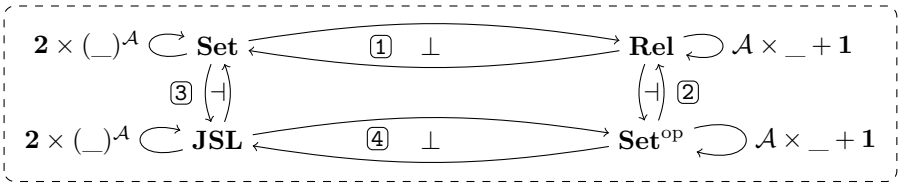
Remark 1. We immediately make the following observations about Theorem 1.

(a) Due to the fact that $\mathbf{Coalg}\,(F) \cong \mathbf{Alg}\,(F^{\mathrm{op}})$, where F^{op} is the opposite functor to F, we can apply the theorem to obtain liftings to algebras.
(b) If $\langle L \dashv R, \eta, \varepsilon\rangle$ is an equivalence [a dual equivalence] of categories then $\left\langle \overline{L} \dashv \overline{R}, \eta, \varepsilon\right\rangle$ is an equivalence [a dual equivalence] of categories.

3 Nondeterministic Automata and Determinization

Within this section we will first shortly recall how deterministic (**DA**), nondeterministic (**NDA**) and codeterministic (**CDA**) automata can be modeled as coalgebras in suitable categories. We will then consider adjunctions between these categories and apply our theorem to obtain a lifting. Via this we will recover the (co)determinization of an automaton via the powerset construction.

The content of this (and the following) section can be summarized in the diagram of categories and functors below. While **DA** live in **Set**, **NDA** can be seen as arrows in **Rel** and **CDA** as arrows in $\mathbf{Set}^{\mathrm{op}}$ (see Section 3.1). Furthermore in Section 4 we will in addition consider deterministic join automata (**DJA**) which live in the category of complete join semilattices (**JSL**), see Sections 4.1 and 4.2. Between these categories of coalgebras there are four adjunctions, which will be treated in the following sections.



For the rest of this and the following section let $\mathcal{A}$ denote an alphabet, i.e. a finite set of labels. In a coalgebraic treatment of labeled transition systems, one usually omits initial states and the state spaces are not required to be finite.

3.1 Automata as Coalgebras

Deterministic Automata. In the category **Set** of sets and functions, deterministic automata can be modeled as coalgebras for the functor $\mathbf{2} \times (\_)^{\mathcal{A}}$. We can represent a deterministic automaton with states X and alphabet $\mathcal{A}$ as a coalgebra $c\colon X \to \mathbf{2} \times X^{\mathcal{A}}$ where each state $x \in X$ is mapped to a tuple $\langle o, s\rangle$ in which the output flag $o \in \{0, 1\}$ determines whether x is final (if and only if $o = 1$) and the successor function $s\colon \mathcal{A} \to X$ determines for each letter $a \in \mathcal{A}$ the unique a-successor $s(a) \in X$ of the state x. We thus define the category of deterministic automata and automata morphisms to be $\mathbf{DA} := \mathbf{Coalg}\left(\mathbf{2} \times (\_)^{\mathcal{A}}\colon \mathbf{Set} \to \mathbf{Set}\right)$.

Nondeterministic Automata. Given a set X of states, we model a nondeterministic automaton by a coalgebra for the functor $\mathcal{A} \times \_ + \mathbf{1}$ in **Rel**. Given a coalgebra $c\colon X \leftrightarrow \mathcal{A} \times X + \mathbf{1}$, each state $x \in X$ is in relation with ✓ if and only if it is a final state. For any letter $a \in \mathcal{A}$ the $y \in X$ such that $\langle x, \langle a, y\rangle\rangle \in c$ are the a-successor(s) (one, multiple or none) of x. We thus define the category of nondeterministic automata and automata morphisms to be $\mathbf{NDA} := \mathbf{Coalg}\left(\mathcal{A} \times \_ + \mathbf{1}\colon \mathbf{Rel} \to \mathbf{Rel}\right)$.

Codeterministic Automata. Given a set X of states, a codeterministic (backwards deterministic) automaton (**CDA**) is given by a function $c\colon \mathcal{A}\times X+\mathbf{1}\to X$ where $c(\checkmark)\in X$ is the unique final state and for each pair $\langle a,x\rangle\in\mathcal{A}\times X$ the unique a-predecessor of x is $c(\langle a,x\rangle)$. Hence we can model them as coalgebras for the functor $\mathcal{A}\times X+\mathbf{1}$ on $\mathbf{Set}^{\mathrm{op}}$ and define the category of codeterministic automata and their morphisms to be $\mathbf{CDA}=\mathbf{Coalg}\left(\mathcal{A}\times \_+\mathbf{1}\colon \mathbf{Set}^{\mathrm{op}}\to\mathbf{Set}^{\mathrm{op}}\right)$.

Note that $\mathbf{Set}^{\mathrm{op}}$ is equivalent to **CABA**, the category of all complete atomic boolean algebras, with boolean algebra homomorphisms. So instead of thinking of these automata as codeterministic, one could think of them as deterministic automata with a rich algebraic structure on the states, even richer than the deterministic join automata introduced in Section 4.

Codeterministic automata are studied in [6] under the name àtomata. An example automaton is shown in Figure 1 (right).

3.2 Determinization of Nondeterministic Automata

Let us reconsider the adjunction ① between **Set** and **Rel** which we presented in Example 1. We aim at applying Theorem 1 to this adjunction to get a lifting and claim that this will yield the well known powerset construction to determinize nondeterministic automata.

Recall that, by Theorem 1 (ii), for such a lifting to exist it is sufficient to define a natural isomorphism $\beta\colon \mathbf{2}^{\mathcal{A}\times \_+\mathbf{1}}\Rightarrow \mathbf{2}\times(\mathbf{2}^{-})^{\mathcal{A}}$. Thus we define for every set X the function $\beta_X\colon \mathbf{2}^{\mathcal{A}\times X+\mathbf{1}}\to \mathbf{2}\times\left(\mathbf{2}^X\right)^{\mathcal{A}}$ via

$$\beta_X(S)=\left\langle \chi_S(\checkmark),\left(s\colon\mathcal{A}\to\mathbf{2}^X, s(a)=\{x\in X\mid\langle a,x\rangle\in S\}\right)\right\rangle \tag{8}$$

for every $S\in\mathbf{2}^{\mathcal{A}\times X+\mathbf{1}}$. Here χ_S is the characteristic function[3] of S. The inverse function $\beta_X^{-1}\colon \mathbf{2}\times\left(\mathbf{2}^X\right)^{\mathcal{A}}\to\mathbf{2}^{\mathcal{A}\times X+\mathbf{1}}$ is given by

$$\beta_X^{-1}(\langle o,s\rangle):=\{\checkmark\mid o=1\}\cup\bigcup_{a\in\mathcal{A}}\{a\}\times s(a) \tag{9}$$

for every $\langle o,s\rangle\in\mathbf{2}\times(\mathbf{2}^X)^{\mathcal{A}}$. By Theorem 1 (ii) we obtain a lifting. We calculate the natural transformation $\alpha\colon \mathbf{2}\times(\_)^{\mathcal{A}}\Rightarrow\mathcal{A}\times\_+\mathbf{1}$ by using (7) to obtain for every set X the relation $\alpha_X\colon \mathbf{2}\times X^{\mathcal{A}}\leftrightarrow\mathcal{A}\times X+\mathbf{1}$ given by

$$\alpha_X=\left\{\left\langle\langle 1,s\rangle,\checkmark\right\rangle,\left\langle\langle o,s\rangle,\langle a,s(a)\rangle\right\rangle\;\middle|\; o\in\mathbf{2},s\in X^{\mathcal{A}},a\in\mathcal{A}\right\}. \tag{10}$$

With these preparations at hand we can now construct the lifted functors. The new left adjoint $\overline{L}\colon \mathbf{DA}\to\mathbf{NDA}$ maps a **DJA** $c\colon X\to\mathbf{2}\times X^{\mathcal{A}}$ to the **NDA** $\overline{L}(c)\colon X\leftrightarrow\mathcal{A}\times X+\mathbf{1}$ which is given by[4]

$$\overline{L}(c)=\left\{\langle x,\checkmark\rangle\;\middle|\;\begin{array}{c}x\in X\\ \pi_1(c(x))=1\end{array}\right\}\cup\left\{\left\langle x,\left\langle a,\pi_2\left(c(x)\right)(a)\right\rangle\right\rangle\;\middle|\;\begin{array}{c}x\in X\\ a\in\mathcal{A}\end{array}\right\} \tag{11}$$

[3] Given a set X, the characteristic function $\chi_S\colon X\to\{0,1\}$ is defined for any subset $S\subseteq X$ by $\chi_S(x)=1$ iff $x\in S$ and $\chi_S(x)=0$ otherwise.

[4] $\pi_1\colon \mathbf{2}\times X^{\mathcal{A}}\to\mathbf{2}$ and $\pi_2\colon \mathbf{2}\times X^{\mathcal{A}}\to X^{\mathcal{A}}$ are the projections of the product.

which is simply the same automaton, but interpreted as a nondeterministic one.

The lifted right adjoint $\overline{R}\colon \mathbf{NDA} \to \mathbf{DA}$ maps a nondeterministic automaton $d\colon Y \leftrightarrow \mathcal{A} \times Y + \mathbf{1}$ to the deterministic automaton $\overline{R}(d)\colon \mathbf{2}^Y \to \mathbf{2} \times (\mathbf{2}^Y)^{\mathcal{A}}$. A state of this new automaton is just a set of states $Q \in \mathbf{2}^Y$ of the original automaton. For each such Q the tuple $\overline{R}(d)(Q) = \langle o, s\colon \mathcal{A} \to \mathbf{2}^Y \rangle$ is given as follows: We have $o = 1$ if and only if there is a $q \in Q$ such that $\langle q, \checkmark \rangle \in d$, i.e. Q is final if and only if one of the original states in Q is final. Moreover, the a-successor of Q is determined by $s(a) = \{y \in Y \mid \exists q \in Q : \langle q, \langle a, y \rangle\rangle \in d\}$ which we can easily identify to be exactly the definition of the transition function of the usual powerset automaton construction.

3.3 Codeterminization of Nondeterministic Automata

Let us now consider adjunction ② between **Rel** and $\mathbf{Set}^{\mathrm{op}}$ which we automatically obtain by dualizing the adjunction between **Set** and **Rel** from Example 1 and the fact that **Rel** is a self-dual category. This adjunction has already been considered in [1,8].

The left adjoint $L\colon \mathbf{Rel} \to \mathbf{Set}^{\mathrm{op}}$ maps a set X to its powerset $\mathbf{2}^X$ and a relation $f\colon X \leftrightarrow Y$ to the function $Lf\colon \mathbf{2}^X \leftarrow \mathbf{2}^Y$ given by, for every $S \in \mathbf{2}^Y$, $(Lf)(S) = \{x \in X \mid \exists y \in S : \langle x, y \rangle \in f\}$. The right adjoint $R\colon \mathbf{Set}^{\mathrm{op}} \to \mathbf{Rel}$ is the inclusion, i.e. it maps a set X to itself and a function $f\colon X \leftarrow Y$ to the corresponding relation $f\colon X \leftrightarrow Y$. The unit η consists of all the relations $\eta_X\colon X \leftrightarrow \mathbf{2}^X$ defined via $\langle x, S \rangle \in \varepsilon_X$ iff $x \in S$ and the counit ε is given by all the functions $\varepsilon_X\colon \mathbf{2}^X \leftarrow X$ mapping each $x \in X$ to the singleton set $\{x\}$.

We proceed to define a lifting as before by specifying a natural isomorphism $\beta\colon \mathcal{A} \times \_ + \mathbf{1} \Rightarrow \mathcal{A} \times \_ + \mathbf{1}$. The obvious choice is to let β_X be the identity relation on $\mathcal{A} \times X + \mathbf{1}$ which indeed yields a natural isomorphism. Moreover, using (7) we obtain the natural transformation $\alpha\colon \mathbf{2}^{\mathcal{A} \times \_ + \mathbf{1}} \Rightarrow \mathcal{A} \times \mathbf{2}^- + \mathbf{1}$ where for each set X the function $\alpha_X\colon \mathbf{2}^{\mathcal{A} \times X + \mathbf{1}} \leftarrow \mathcal{A} \times \mathbf{2}^X + \mathbf{1}$ is given by $\alpha(\checkmark) = \mathbf{1}$ and $\alpha(\langle a, S \rangle) = \{a\} \times S$ for every $\langle a, S \rangle \in \mathcal{A} \times \mathbf{2}^X$.

The lifted left adjoint $\overline{L}\colon \mathbf{NDA} \to \mathbf{CDA}$ performs codeterminization: Given an **NDA** $c\colon X \leftrightarrow \mathcal{A} \times X + \mathbf{1}$ we obtain a **CDA** $\overline{L}(c)\colon \mathbf{2}^X \leftarrow \mathcal{A} \times \mathbf{2}^X + \mathbf{1}$ where $\checkmark$ is mapped to the set $\{x \in X \mid \langle x, \mathbf{1} \rangle \in c\}$, i.e. the unique final state of the new automaton is the set of all final states of the original automaton. Given a set of states $S \in \mathbf{2}^X$ and a letter $a \in A$ the a-predecessor of S is the set

$$\overline{L}(c)(\langle a, S \rangle) = \{x \in X \mid \exists y \in S : \langle x, \langle a, y \rangle\rangle \in c\} \tag{12}$$

containing all the a-predecessors of the states in S. We conclude that the new automaton is indeed a codeterministic automaton which is (language) equivalent to the original one.

While in the previous example the lifted left adjoint was trivial (as was the original left adjoint), in this case we obtain a trivial lifted right adjoint $\overline{R}\colon \mathbf{CDA} \to \mathbf{NDA}$: it "interprets" a **CDA** $d\colon Y \leftarrow \mathcal{A} \times Y + 1$ as **NDA**, i.e. as the corresponding relation $\overline{R}(d)\colon Y \leftrightarrow \mathcal{A} \times Y + 1$.

4 Deterministic Join Automata

We will now try to take a different perspective to look at powerset automata instead of just considering them to be determinized nondeterministic automata. In order to do that we briefly recall the notion of complete join semilattices and the corresponding category.

4.1 Complete Join Semilattices

A complete join semilattice is a partially ordered set X such that for every (possibly infinite) set $S \in \mathbf{2}^X$ there is a least upper bound, called *join* and denoted by $\sqcup S$. If Y is another join semilattice we call a function $f\colon X \to Y$ *join-preserving* if, for all S, it satisfies $f(\sqcup S) = \sqcup \{f(s) \mid s \in S\}$. The join semilattices and the join-preserving functions form a category which we will denote by **JSL**. It is isomorphic to the Eilenberg-Moore category for the powerset monad on set.

If we equip the set $\mathbf{2} = \{0, 1\}$ with the partial order $0 \leq 1$, we get a complete join semilattice with $\sqcup\emptyset = 0$, $\sqcup\{0\} = 0$, $\sqcup\{1\} = 1$ and $\sqcup\mathbf{2} = 1$.

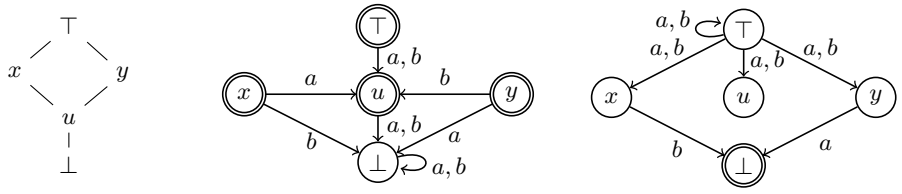
The product of two join semilattices X and Y is the cartesian product of the base sets equipped with a partial order given by $\langle x_1, y_1\rangle \leq \langle x_2, y_2\rangle$ if and only if $x_1 \leq y_1$ and $y_1 \leq y_2$ for all $x_1, x_2 \in X$, $y_1, y_2 \in Y$ and analogously, given a set X we can equip $X^{\mathcal{A}}$ with a partial order based on a given one on X by defining for $f, g \in X^{\mathcal{A}}$ that $f \leq g$ if and only if $f(a) \leq g(a)$ for every $a \in \mathcal{A}$. This is a complete join semilattice if X is one. Given a subset $F \subseteq X^{\mathcal{A}}$ its join is given by $\sqcup F\colon \mathcal{A} \to X, \sqcup F(a) = \sqcup\{f(a) \mid f \in F\}$.

4.2 Deterministic Join Automata

We can interpret a coalgebra $c\colon X \to \mathbf{2} \times X^{\mathcal{A}}$ for the functor $\mathbf{2} \times (\_)^{\mathcal{A}}$ on **JSL** as a deterministic automaton just as we did before on **Set**. Since the arrows of **JSL** are join-preserving functions, such an automaton possesses a certain additional property. Given a set $S \in \mathbf{2}^X$ of states, we know that there is a *join-state* $\sqcup S$. By the join-preserving property of the transition function c we know that $c(\sqcup S) = \sqcup\{c(x) \mid x \in S\}$ and we conclude that $\sqcup S$ is final if and only if one of the states $x \in S$ is final and moreover any transition of an $x \in S$ can be "simulated" (see below) by $\sqcup S$. We define the category $\mathbf{DJA} := \mathbf{Coalg}\left(\mathbf{2} \times (\_)^{\mathcal{A}}\colon \mathbf{JSL} \to \mathbf{JSL}\right)$ and call its objects *deterministic join automata*.

Example 2. Take a look at Figure 1. If we equip the set $X = \{\bot, u, x, y, \top\}$ with the partial order given by the Hasse diagram on the left we obtain a complete join semilattice. The diagram in the middle shows a deterministic join automaton on this join semilattice. Note that the join of two final states is again final and that for every pair of states and alphabet symbol a, the a-successor of the join of the states is the join of the a-successors. This implies a general property of **DJA** that for every subset of states there exists a state accepting the union of the languages of the given states.

Fig. 1. Hasse diagram of a complete join semilattice, a deterministic join automaton and its codeterminization (from left to right)



4.3 From Deterministic Automata to Deterministic Join Automata

We will now consider adjunction ③ in order to transform **DJA** into **DA**s and vice versa. Given a conventional **DA**, a suitable algorithm to obtain a **DJA** is (again) the powerset construction. We will see that this is a reasonable construction in the sense that it arises from an adjunction between **Set** and **JSL**. As said before, the category of complete join semilattices is equivalent to the Eilenberg-Moore category for the powerset monad on **Set**. The theory of adjunctions gives us a generic construction of an adjunction which we will now spell out in details.

The left adjoint $L\colon \mathbf{Set} \to \mathbf{JSL}$ maps any set X to $\mathbf{2}^X$ which is partially ordered by set inclusion. The join operation is set theoretic union and it is easy to see that we indeed obtain a complete join semilattice. Any function $f\colon X \to Y$ is mapped to its image map $f[.]\colon \mathbf{2}^X \to \mathbf{2}^Y$ which we can easily identify as join- (i.e. union-)preserving function. The right adjoint $R\colon \mathbf{JSL} \to \mathbf{Set}$ takes a complete join semilattice to its base set and forgets about the order and the join operation. Analogously, a join-preserving function is just considered as a function. The unit of the adjunction is given, for every set X, by the function $\eta_X\colon X \to \mathbf{2}^X, \eta_X(x) = \{x\}$. The counit consists of the join-preserving functions $\varepsilon_{\langle Y,\sqcup\rangle}\colon \langle \mathbf{2}^Y, \cup\rangle \to \langle Y, \sqcup\rangle$ mapping each set $S \in \mathbf{2}^Y$ to its join $\sqcup S$ in Y.

In order to obtain the lifting we define $\beta_{\langle X,\sqcup\rangle}\colon \mathbf{2} \times X^{\mathcal{A}} \to \mathbf{2} \times X^{\mathcal{A}}$ to be the identity function on $\mathbf{2} \times X^{\mathcal{A}}$ for every join semilattice $\langle X, \sqcup\rangle$ which obviously yields a natural isomorphism β. Using (7) we construct the natural transformation α where for each set X the join preserving function $\alpha_X\colon \mathbf{2}^{\mathbf{2}\times X^{\mathcal{A}}} \to \mathbf{2} \times (\mathbf{2}^X)^{\mathcal{A}}$ is given by, for every $S \in \mathbf{2}^{\mathbf{2}\times X^{\mathcal{A}}}$,

$$\alpha_X(S) = \left\langle \bigsqcup \{o \mid \langle o, s\rangle \in S\}, \bigsqcup \{s \mid \langle o, s\rangle \in S\} \right\rangle . \tag{13}$$

The lifted left adjoint $\overline{L}\colon \mathbf{DA} \to \mathbf{DJA}$ performs the powerset construction on a deterministic automaton: For a deterministic automaton $c\colon X \to \mathbf{2} \times X^{\mathcal{A}}$ the deterministic join automaton $\overline{L}(c)\colon \mathbf{2}^X \to \mathbf{2} \times (\mathbf{2}^X)^{\mathcal{A}}$ is given by, for all $S \in \mathbf{2}^X$,

$$\overline{L}(c)(S) = \left\langle \bigsqcup \{\pi_1(c(x)) \mid x \in S\}, \bigsqcup \{\pi_2(c(x)) \mid x \in S\} \right\rangle . \tag{14}$$

The lifted right adjoint $\overline{R}\colon \mathbf{DJA} \to \mathbf{DA}$ takes a **DJA** and interprets it as **DA** by forgetting about its join property.

4.4 Codeterminization of Deterministic Join Automata

Finally, we will describe an unusual construction translating **DJA** into **CDA**, based on adjunction ④. It is unusual, since the unit of the adjunction (which must be join-preserving) maps every element to the complement (!) of its upward-closure (more details are given below).

The left adjoint $L\colon \mathbf{JSL} \to \mathbf{Set}^{\mathrm{op}}$ maps any join semilattice $\langle X, \sqcup\rangle$ to its base set X and each join-preserving function $f\colon \langle X,\sqcup\rangle \to \langle Y,\sqcup\rangle$ to the $\mathbf{Set}^{\mathrm{op}}$-arrow

$$Lf\colon X \leftarrow Y, \quad Lf(y) = \bigsqcup \{x \in X \mid f(x) \sqsubseteq y\}\ . \tag{15}$$

The right adjoint $R\colon \mathbf{Set}^{\mathrm{op}} \to \mathbf{JSL}$ maps a set X to its powerset $\mathbf{2}^X$ equipped with the subset order, i.e. to the join semilattice $\langle \mathbf{2}^X, \cup\rangle$ and each $\mathbf{Set}^{\mathrm{op}}$-arrow $f\colon X \leftarrow Y$ to the reverse image $f^{-1}[.]\colon \mathbf{2}^X \to \mathbf{2}^Y$. The unit of this adjunction is given by the join-preserving functions

$$\eta_{\langle X,\sqcup\rangle}\colon \langle X,\sqcup\rangle \to \langle \mathbf{2}^X, \cup\rangle, \quad x \mapsto \overline{\uparrow x} = \{x' \in X \mid x' \not\supseteq x\} \tag{16}$$

for every join semilattice $\langle X,\sqcup\rangle$ and the counit is given by, for every set X,

$$\varepsilon_X\colon \mathbf{2}^X \leftarrow X, \quad x \mapsto \overline{\{x\}}\,. \tag{17}$$

We construct a natural isomorphism $\beta\colon \langle \mathbf{2}^{\mathcal{A}\times \_ + \mathbf{1}}, \cup\rangle \Rightarrow \langle \mathbf{2} \times (\mathbf{2}^-)^{\mathcal{A}}, \sqcup\rangle$ in order to get a lifting. For every join semilattice $\langle X,\sqcup\rangle$ we take β_X to be the same function as in (8) of our first example given in Section 3.2 and claim that this is a join-preserving function: for two sets $Q_1, Q_2 \in \mathbf{2}^{\mathcal{A}\times X + \mathbf{1}}$ let $\langle o, s\rangle := \beta_X(Q_1 \cup Q_2)$ and $\langle o_i, s_i\rangle = \beta_X(Q_i)$ for $i \in \{1,2\}$ then we have $o = \chi_{Q_1\cup Q_2}(\checkmark) = \max\{\chi_{Q_1}(\checkmark), \chi_{Q_2}(\checkmark)\} = \max\{o_1, o_2\} = o_1 \sqcup o_2$ and for each $a \in A$ we have

$$\begin{aligned} s(a) &= \{x \in X \mid \langle a, x\rangle \in Q_1 \cup Q_2\} \\ &= \{x \in X \mid \langle a, x\rangle \in Q_1\} \cup \{x \in X \mid \langle a, x\rangle \in Q_2\} = s_1(a) \cup s_2(a) \end{aligned}$$

which can be generalized to arbitrary unions.

We calculate $\alpha\colon \mathbf{2} \times \_^{\mathcal{A}} \Leftarrow \mathcal{A} \times \_ + \mathbf{1}$ where for every join semilattice $\langle X,\sqcup\rangle$ the function $\alpha_{\langle X,\sqcup\rangle}$ is given by $\alpha_{\langle X,\sqcup\rangle}(\checkmark) = \langle 0, (s_\checkmark\colon \mathcal{A} \to X, s_\checkmark(a) = \top)\rangle$ and

$$\alpha_{\langle X,\sqcup\rangle}(\langle a, x\rangle) = \left\langle 1, \left(s_{\langle a,x\rangle}\colon \mathcal{A} \to X, s_{\langle a,x\rangle}(a') = \begin{cases} x, & a' = a \\ \top, & a' \neq a \end{cases} \right) \right\rangle . \tag{18}$$

The lifted left adjoint $\overline{L}\colon \mathbf{DJA} \to \mathbf{CDA}$ maps a $\mathbf{DJA}$ $c\colon \langle X,\sqcup\rangle \to \langle \mathbf{2} \times X^{\mathcal{A}}, \sqcup\rangle$ to the $\mathbf{CDA}$ $\overline{L}c\colon X \leftarrow \mathcal{A} \times X + \mathbf{1}$ whose unique final state is the join of the non-final states of the original automaton, i.e. $\overline{L}c(\checkmark) = \bigsqcup\{x' \in X \mid \pi_1(c(x')) = 0\}$. For every action $a \in \mathcal{A}$ and every state $x \in X$ the unique a-predecessor of x is the join of all the states of the original automaton whose a-successor is less or equal to x, i.e. $\overline{L}c(\langle a, x\rangle) = \bigsqcup\{x' \in X \mid \pi_2(c(x'))(a) \sqsubseteq x\}$.

The lifted right adjoint $\overline{R}\colon \mathbf{CDA} \to \mathbf{DJA}$ maps a $\mathbf{CDA}$ $d\colon Y \leftarrow \mathcal{A} \times Y + \mathbf{1}$ (which can also be regarded as nondeterministic automaton) to its determinization

$\overline{R}(d)\colon \langle \mathbf{2}^Y, \cup \rangle \to \langle \mathbf{2} \times (\mathbf{2}^Y)^{\mathcal{A}}, \sqcup \rangle$ via the usual powerset construction, i.e. for every $S \in \mathbf{2}^Y$ we have $\overline{R}(d)(S) = \beta_Y \circ d^{-1}[S]$. Since the reverse image of any function preserves arbitrary unions, this is indeed a **DJA**.

Example 3. Take another look at Figure 1. The **DJA** of Example 2 (in the middle) is transformed into a **CDA** with the same state set $X = \{\bot, x, y, u, \top\}$ (the diagram on the right). Its unique final state is $\bot$ (the join of the non-final states) and the unique a-predecessor of a state is the join of the previous a-predecessors, for instance the new a-predecessor of $\bot$ is y (the join of $\bot, u, y$). If we transfer this automaton into a **DJA** with state set $\mathbf{2}^X$ via the right adjoint the unit $\eta_{(X,\sqcup)}$ maps every state to the complement of its upward-closure. For instance state x is mapped to $\{\bot, u, y\}$.

5 Linear Weighted Automata

In the previous sections we looked at automata over **Set**, **Rel** and **JSL**. Now we will look at automata over the category **Vect** of linear maps between vector spaces called *linear weighted automata*. There are two flavors: *input linear weighted automata* (category: $\mathbf{WAut}_i$) and *output linear weighted automata* ($\mathbf{WAut}_o$). We will use Theorem 1 to obtain an adjunction between $\mathbf{WAut}_i$ and $\mathbf{WAut}_o$. We have taken this example from [3, Section 4] and extended it from finite to arbitrary vector spaces. For this section, let $\mathcal{A}$ be an arbitrary set (not necessarily finite).

5.1 Vector Spaces

Let us recall some facts about the category **Vect** of vector spaces. To begin **Vect** has all products and coproducts. Let V and W be vector spaces. The product of V and W is simply the cartesian product $V \times W$ with coordinatewise addition and scalar multiplication. The coproduct of V and W is $V \times W$ as well; the coprojections $V \xrightarrow{\kappa_0} V \times W \xleftarrow{\kappa_1} W$ are given by $\kappa_0(v) = (v, 0)$ and $\kappa_1(w) = (0, w)$ for $v \in V$ and $w \in W$.

Similarly, given a set B the vector space V^B of functions from B to V is the B-fold product of V. However, the coproduct of infinitely many vector spaces is a bit more interesting. Let B be a set and V a vector space. The B-fold coproduct of V is the following subspace of V^B.

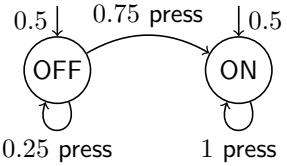
$$B \cdot V \;:=\; \{\; f \in V^B \mid \operatorname{supp} f \text{ is finite } \}$$

Here $\operatorname{supp} f := \{\, b \in B \mid f(b) \neq 0\}$ denotes the *support* of $f \in V^B$. Let $b \in B$ be given. The b-th coprojection $\kappa_b\colon V \to B \cdot V$ is given by, for every $v \in V$: $\kappa_b(v)(b) = v$ and $\kappa_b(v)(b') = 0$ for all $b' \in B$ with $b' \neq b$.

Note that the B-fold product and the B-fold coproduct coincide, i.e. $B \cdot V = V^B$, if and only if $V = \{0\}$ or B is finite. In fact, we even have $B \cdot \mathbb{R} \cong \mathbb{R}^B$ if and only if B is finite.

5.2 Input Linear Weighted Automata

An input linear weighted automaton is a linear map of the form $c\colon \mathcal{A} \cdot V + \mathbb{R} \to V$. That is, it is an algebra on **Vect** of type $\mathcal{A} \cdot (\_) + \mathbb{R}$. Accordingly we define the category $\mathbf{WAut}_i$ of input linear weighted automata to be $\mathbf{Alg}\,(\mathcal{A} \cdot (\_) + \mathbb{R})$.



For an example of an input linear weighted automaton we draw on our experiences with turning on electrical devices such as printers and projectors. Let us say that the state of such device is either "ON" or "OFF" or a superposition of both. We can represent the state space as $\mathbb{R}^2$ where $\mathsf{ON} := \langle 1, 0\rangle$ and $\mathsf{OFF} := \langle 0, 1\rangle$. When we approach the device it is fair to say it is as likely that we find it running as it is that we find it turned off. So the initial state is

$$I \;:=\; \langle 0.5, 0.5\rangle \;=\; 0.5 \cdot \mathsf{ON} + 0.5 \cdot \mathsf{OFF}\,.$$

When the device is turned on and we press the "power on" button, surely nothing will happen, but when the device is turned off and the button is pressed the device will only turn on with probability, say, 0.75 (see diagram above).

Formally, pressing the button is represented by a linear map $P\colon \mathbb{R}^2 \to \mathbb{R}^2$ with $P(\mathsf{ON}) = \langle 1, 0\rangle$ and $P(\mathsf{OFF}) = \langle 0.75, 0.25\rangle$. There is precisely one such P:

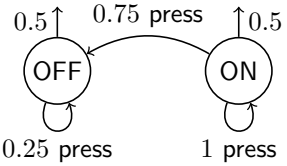
$$P(\langle x, y\rangle) \;=\; \begin{pmatrix} 1 & 0.75 \\ 0 & 0.25 \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} \qquad \text{for } x, y \in \mathbb{R}\,.$$

Together, the initial state I and P form an input linear weighted automaton $c\colon \mathcal{A} \cdot \mathbb{R}^2 + \mathbb{R} \to \mathbb{R}^2$ with $\mathcal{A} := \{\mathsf{press}\}$. Indeed, c is given by, for $x, y, \mu \in \mathbb{R}$,

$$c(\,\langle(\mathsf{press}, \langle x, y\rangle),\, \mu\rangle\,) \;=\; \begin{pmatrix} 1 & 0.75 \\ 0 & 0.25 \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} + \mu \begin{pmatrix} 0.5 \\ 0.5 \end{pmatrix}.$$

5.3 Output Linear Weighted Automata

An output linear weighted automaton is a linear map of the form $d\colon V \to \mathbb{R} \times V^{\mathcal{A}}$. That is, it is a coalgebra on **Vect** of type $\mathbb{R} \times (\_)^{\mathcal{A}}$. The category of output linear weighted automata $\mathbf{WAut}_o$ is the category $\mathbf{Coalg}\left(\mathbb{R} \times (\_)^{\mathcal{A}}\right)$ of coalgebras for $\mathbb{R} \times (\_)^{\mathcal{A}}$.



We get an example of an output linear weighted automaton (see diagram) if we reverse all arrows of the input linear weighted automaton of Section 5.2. Formally, let $d\colon \mathbb{R}^2 \to \mathbb{R} \times (\mathbb{R}^2)^{\mathcal{A}}$ with $\mathcal{A} = \{\mathsf{press}\}$ be given by, for $x, y \in \mathbb{R}$,

$$d(\langle x, y\rangle) \;=\; \left(\begin{pmatrix} 0.5 & 0.5 \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix},\; \lambda a^{\mathcal{A}}.\; \begin{pmatrix} 1 & 0 \\ 0.75 & 0.25 \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} \right).$$

We could say that the output linear weighted automaton d is the *dual* (or *transpose*) of the input linear weighted automaton c. We will generalize this construction to arbitrary input linear weighted automata using the dual of a vector space (see below) and Theorem 1.

5.4 Dual of a Vector Space

Let V be a vector space. The dual of V is the following subspace of $\mathbb{R}^V$.

$$V^* \;:=\; \left\{\; \varphi \in \mathbb{R}^V \mid \varphi \text{ is linear} \;\right\}$$

Let us determine the dual of $\mathbb{R}^n$ for some natural number n. Recall that for every linear map $\varphi\colon \mathbb{R}^n \to \mathbb{R}$ there is a unique $u \in \mathbb{R}^n$ with, for all $x \in \mathbb{R}^n$,

$$\varphi(x) \;=\; u_1 x_1 \,+\, \cdots \,+\, u_n x_n \;\equiv\; u \cdot x\,.$$

So we get a bijection $\Phi\colon \mathbb{R}^n \to (\mathbb{R}^n)^*$ given by, for $x, u \in \mathbb{R}^n$

$$\Phi(u)(x) \;=\; u \cdot x\,. \tag{19}$$

It is not hard to see that Φ is linear, so $(\mathbb{R}^n)^*$ is isomorphic to $\mathbb{R}^n$. Consequently, $V^* \cong V$ for any finite dimensional vector space V.

For an infinite dimensional vector space V the situation is different. Let B be a basis for V (so $B \cdot \mathbb{R} \cong V$ and B is infinite). Define $\Psi\colon (B \cdot \mathbb{R})^* \to \mathbb{R}^B$ by, for $\varphi \in (B \cdot \mathbb{R})^*$ and $b \in B$, $\Psi(\varphi)(b) \;=\; \varphi(b)$. Then it is not hard to see that Ψ is an isomorphism. Since B is infinite we know that $B \cdot \mathbb{R} \ncong \mathbb{R}^B$, so $B \cdot \mathbb{R} \ncong (B \cdot \mathbb{R})^*$. Hence $V^* \ncong V$ for any infinite dimensional vector space V.

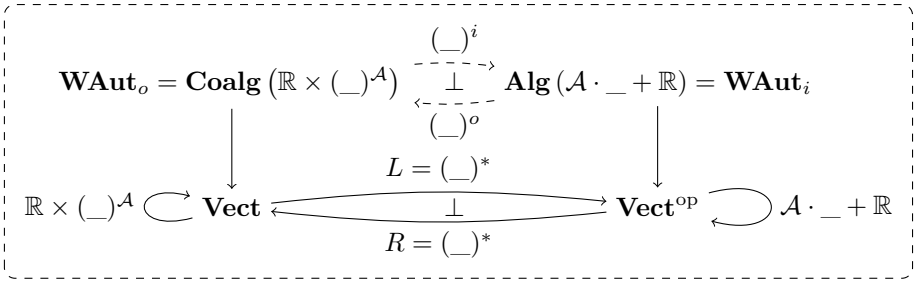
To apply Theorem 1 to the dual vector space construction we need to recognize the assignment $V \mapsto V^*$ as part of an adjunction. To begin, note that $V \mapsto V^*$ extends to a functor $(\_)^*\colon \mathbf{Vect} \to \mathbf{Vect}^{\mathrm{op}}$ as follows. Given a linear map $f\colon V \to W$ define $f^*\colon W^* \to V^*$ by, for $\varphi \in W^*$, $f^*(\varphi) \;=\; \varphi \circ f$.

This also gives us a functor $(\_)^*\colon \mathbf{Vect}^{\mathrm{op}} \to \mathbf{Vect}$. Now, given a vector space V define $\iota_V\colon V \to V^{**}$ by, for $v \in V$ and $\varphi \in V^*$, $\iota_V(v)(\varphi) \;=\; \varphi(v)$. Then we have an adjunction $\langle (\_)^* \dashv (\_)^*, \iota, \iota\rangle : \mathbf{Vect} \to \mathbf{Vect}^{\mathrm{op}}$.

If $V \equiv \mathbb{R}^n$ for some $n \in \mathbb{N}$, then ι_V is an isomorphism. So if we restrict the adjunction to the category $\mathbf{FVect}$ of linear maps between finite dimensional vector spaces we get a duality $\langle (\_)^* \dashv (\_)^*, \iota, \iota\rangle : \mathbf{FVect} \to \mathbf{FVect}^{\mathrm{op}}$.

5.5 Dual of a Linear Weighted Automaton

We will now lift the adjunction between $\mathbf{Vect}$ and $\mathbf{Vect}^{\mathrm{op}}$. The result will be:



For a vector space V define $\beta_V \colon (\mathcal{A}\cdot V+\mathbb{R})^* \to \mathbb{R}\times(V^*)^{\mathcal{A}}$ by, for $f \in (\mathcal{A}\cdot V+\mathbb{R})^*$,

$$\beta_V(f) \;=\; \Big\langle\, f\big(\kappa_1(1)\big),\, \lambda a^{\mathcal{A}}\lambda v^{V}.\, f\Big(\kappa_0\big(\kappa_a(v)\big)\Big)\Big\rangle\ .$$

Then β_V is invertible and we get a natural iso $\beta \colon (\mathcal{A}\cdot(\_)+\mathbb{R})^* \to \mathbb{R}\times((\_)^*)^{\mathcal{A}}$. Hence we get an adjunction lifting by Theorem 1 as depicted above.

Let $c \colon \mathcal{A}\cdot V+\mathbb{R} \to V$ be an input linear weighted automaton. Then

$$c^o \colon V^* \to \mathbb{R}\times(V^*)^{\mathcal{A}} \qquad \text{and} \qquad c^o \;=\; \beta_V \circ c^*.$$

If we take c to be as depicted in the diagram in Section 5.2 then c^o will be as depicted in the diagram in Section 5.3 (if we identify $(\mathbb{R}^2)^*$ with $\mathbb{R}^2$ via the isomorphism in Equation 19).

Let $d \colon V \to \mathbb{R}\times V^{\mathcal{A}}$ be an output linear weighted automaton. Then

$$d^i \colon \mathcal{A}\cdot V^*+\mathbb{R} \to V^* \qquad \text{and} \qquad d^i \;=\; d^* \circ \alpha_V,$$

where $\alpha_V \colon \mathcal{A}\cdot V^*+\mathbb{R} \to (\mathbb{R}\times V^{\mathcal{A}})^*$ and for $h \in \mathcal{A}\cdot V^*$, $\mu,\nu \in \mathbb{R}$, $f \in V^{\mathcal{A}}$,

$$\alpha_V(\langle h,\nu\rangle)(\langle \mu, f\rangle) \;=\; \mu\cdot\nu \;+\; \sum_{a\in\mathcal{A}} h(a)\big(f(a)\big). \tag{20}$$

If we take d as in the diagram of Section 5.3, then d^i will be as depicted in the diagram of Section 5.2. In particular, we see that $(d^i)^o \cong c$. More generally, since we have a duality $\langle (\_)^* \dashv (\_)^*, \iota, \iota\rangle : \mathbf{FVect} \to \mathbf{FVect}^{\mathrm{op}}$ we get a duality

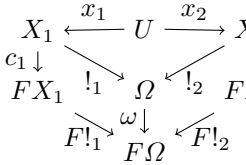
$$\big\langle (\_)^i \dashv (\_)^o, \iota, \iota\big\rangle : \mathbf{FWAut}_o \to \mathbf{FWAut}_i$$

where $\mathbf{FWAut}_o$ and $\mathbf{FWAut}_i$ are variants of $\mathbf{WAut}_o$ and $\mathbf{WAut}_i$, respectively, for finite dimensional vector spaces.

6 Checking Behavioral Equivalences

Finally, we will show how the results on adjunctions can be used to check behavioral equivalences. Since our coalgebras do not necessarily live in **Set**, where we could address elements of a carrier set, we use the following alternative definition, where we specify whether two arrows are behaviorally equivalent. This is reminiscent of equipping a coalgebra with start states, similar to the initial states of an automaton.

Definition 4 (Behavioral Equivalence). *Let* $\mathbf{C}$ *be a category,* $F \colon \mathbf{C} \to \mathbf{C}$ *be an endofunctor such that a final* F*-coalgebra* $\omega \colon \Omega \to F\Omega$ *exists. Furthermore let* $c_1 \colon X_1 \to FX_1$ *and* $c_2 \colon X_2 \to FX_2$ *be two* F*-coalgebras and* U *be a* $\mathbf{C}$*-object.*



We say that two $\mathbf{C}$*-arrows* $x_1 \colon U \to X_1$, $x_2 \colon U \to X_2$ *are* behaviorally equivalent *(in symbols* $x_1 \sim_F^{c_1,c_2} x_2$*), whenever the diagram on the left commutes where* $!_1 \colon c_1 \to \omega$ *and* $!_2 \colon c_2 \to \omega$ *are the unique coalgebra homomorphisms into the final coalgebra.*

In **Set** the choice for U will typically be a singleton set and then the problem reduces to asking whether two given states are behaviorally equivalent.

Now assume that we have an adjunction $\langle L \dashv R, \eta, \varepsilon \rangle : \mathbf{C} \to \mathbf{D}$ that is lifted to coalgebras as specified in Definition 3. Now, since $\overline{R}$ is a right adjoint, it preserves limits, specifically it preserves the final coalgebra. Let us take another look at the first diagram in Section 3: It can easily be determined that $\mathcal{A}^*$, the set of all finite words, is the carrier of the final coalgebra in $\mathbf{Set}^{\mathrm{op}}$. Via the right adjoint, this translates to the carrier set $\mathcal{A}^*$ in **Rel**, where the arrow $!_1$ into the final coalgebra is a relation, relating each state with the words that are accepted by it (in [8] this final coalgebra is discussed in detail). The final coalgebra can also be transferred into **JSL** and **Set**, where it has carrier set $\mathbf{2}^{\mathcal{A}^*}$.

Hence, the adjunctions allow to construct final coalgebras and to transfer results about final semantics from other categories. Furthermore, it is possible to check behavioral equivalence in a different category, by translating queries via the adjunction.

Proposition 1. *Let* $\mathbf{C}$ *be a category,* $F \colon \mathbf{C} \to \mathbf{C}$ *and* $G \colon \mathbf{D} \to \mathbf{D}$ *be endofunctors,* $\langle L \dashv R, \eta, \varepsilon \rangle : \mathbf{C} \to \mathbf{D}$ *be an adjunction together with a lifting in the sense of Definition 3, i.e. an adjunction* $\left\langle \overline{L} \dashv \overline{R}, \eta, \varepsilon \right\rangle : \mathbf{Coalg}\,(F) \to \mathbf{Coalg}\,(G)$*. Furthermore assume that a final* G*-coalgebra exists and that* R *is faithful.*

Let $d_1 \colon Y_1 \to GY_1$*,* $d_2 \colon Y_2 \to GY_2$ *be two* G*-coalgebras and let* $y_1 \colon U \to Y_1$*,* $y_2 \colon U \to Y_2$ *be two arrows in* $\mathbf{D}$*. Then the following equivalence holds.*

$$y_1 \sim_G^{d_1, d_2} y_2 \qquad \Longleftrightarrow \qquad Ry_1 \sim_F^{\overline{R}d_1, \overline{R}d_2} Ry_2$$

In all our examples the right adjoint R is faithful. If this is the case and the final coalgebra exists, this allows us to check behavioral equivalence in a different category, where this might be easier or more straightforward. The classical example is of course the lifted adjunction between **Set** and **Rel**. In order to check language equivalence for nondeterministic automata, the standard technique is to determinize them via the right adjoint into **Set**. Then, language equivalence can be checked on the powerset automaton.

7 Conclusion, Related and Future Work

We have shown how to lift adjunctions between categories to adjunctions on coalgebras. Furthermore we gave several examples for such adjunction liftings and showed how they can be used to transfer behavioral equivalence checks from one category to another.

Several open questions remain, both concerning the examples and the general technique. Our main example involving deterministic and nondeterministic automata is strongly related to the powerset monad, since one adjunction is based on the Kleisli and another on the Eilenberg-Moore construction for this monad. In this specific case we obtain two more adjunctions by considering $\mathbf{Set}^{\mathrm{op}}$, however this will not work for any monad. Still, it would be interesting to investigate which monads allow such a rich structure of adjunctions, in what way these

adjunctions can be lifted to adjunctions to coalgebras and whether this results in well-known constructions. There exists also the comparison functor between the Kleisli and the Eilenberg-Moore category, which however is not necessarily part of an adjunction. It would be interesting to find out which behavioral information can be transported over the comparison functor.d

In [1] the authors used the adjunction between **Rel** and $\mathbf{Set}^{\mathrm{op}}$ in order to characterize a factorization structure that is employed for a minimization algorithm. Hence, an obvious question is whether other adjunctions can be used for such algorithmic purposes, for instance for minimizing a coalgebra in one category, but using the structure of another category. It also seems plausible that up-to techniques can be explained in this way, for instance by checking language equivalence for nondeterministic automata in **Rel**, using the algebraic structure of **JSL** via the comparison functor (similar to [5]).

We also showed how to transfer equivalence checking queries through a right adjoint. Can they also be transferred in the other direction, via a left adjoint?

Finally, the conditions in Theorem 1 (Lifting an Adjunction to Coalgebras) are sufficient for the lifting to exist. However, it is unclear whether they are also necessary.

Related Work. The adjunction between **Rel** and $\mathbf{Set}^{\mathrm{op}}$, transforming nondeterministic automata into codeterministic automata has already been considered in [1] and similarly in [8]. The paper [17] is concerned with the adjunction between **Set** and **JSL** and uses it to determinize automata, but different from the approach in this paper. More concretely, in [17] a *nondeterministic automaton* specified by a function $X \to 2 \times (\mathbf{2}^X)^A$ in **Set** is translated into a join-preserving function $\mathbf{2}^X \to \mathbf{2} \times (\mathbf{2}^X)^A$, which does not give an adjunction on coalgebras. Another closely related paper is [11] which is also concerned with the Kleisli and Eilenberg-Moore constructions for the powerset monad and uses the comparison functor in order to determinize automata. Furthermore our example in Section 5 is based on a duality of categories [3].

Hence, many ideas that are summarized in this paper are not completely new, but have been stated in various forms. However, we think that it is insightful to present this theory strictly from the point of view of adjunction lifting and to clearly spell out what it means to preserve and reflect behavioral equivalences by adjoints. Furthermore, to our knowledge, the adjunction between join semilattices and $\mathbf{Set}^{\mathrm{op}}$, that gives rise to a quite surprising and unusual construction, has never been studied in this setting.

Acknowledgements. We would like to thank Marcello Bonsangue, Alexandra Silva and Filippo Bonchi for raising and discussing the problem with us. Furthermore we would like to acknowledge Ana Sokolova for interesting discussions on this topic.

References

1. Adámek, J., Bonchi, F., Hülsbusch, M., König, B., Milius, S., Silva, A.: A coalgebraic perspective on minimization and determinization. In: Foundations of Software Science and Computational Structures, Lecture Notes in Computer Science, vol. 7213, pp. 58–73. Birkedal, Lars (2012)
2. Awodey, S.: Category Theory. Clarendon Press (2006)
3. Bezhanishvili, N., Kupke, C., Panangaden, P.: Minimization via duality. In: Proc. of WoLLIC '12. pp. 191–205. Springer (2012), LNCS 7456
4. Bonchi, F., Bonsangue, M., Rutten, J., Silva, A.: Brzozowski's algorithm (co)algebraically. In: Logic and Program Semantics – Essays Dedicated to Dexter Kozen on the Occasion of His 60th Birthday. pp. 12–23. Springer (2012), LNCS 7230
5. Bonchi, F., Pous, D.: Checking NFA equivalence with bisimulations up to congruence. In: Proc. of POPL '13. pp. 457–468. ACM (2013)
6. Brzozowski, J., Tamm, H.: Theory of átomata. In: Proc. of DLT '11. pp. 105–116. Springer (2011), LNCS 6795
7. Fiore, M., Turi, D.: Semantics of name and value passing. In: Proc. of LICS '01. pp. 93–104. IEEE (2001)
8. Hasuo, I., Jacobs, B., Sokolova, A.: Generic trace semantics via coinduction. Logical Methods in Computer Science 3 (4:11), 1–36 (November 2007)
9. Hermida, C., Jacobs, B.: Structural induction and coinduction in a fibrational setting. Information and Computation 145(IC982725), 107–152 (1998)
10. Jacobs, B., Rutten, J.: A tutorial on (co)algebras and (co)induction. Bulletin of the European Association for Theoretical Computer Science 62, 222–259 (1997)
11. Jacobs, B., Silva, A., Sokolova, A.: Trace semantics via determinization. In: Proc. of CMCS '12. pp. 109–129. Springer (2012), LNCS 7399
12. Kelly, G., Street, R.: Review of the elements of 2-categories. In: Kelly, G.M. (ed.) Category Seminar, Lecture Notes in Mathematics, vol. 420, pp. 75–103. Springer Berlin Heidelberg (1974), http://dx.doi.org/10.1007/BFb0063101
13. Kerstan, H., König, B.: Coalgebraic trace semantics for continuous probabilistic transition systems. Logical Methods in Computer Science 9 [4:16](834) (dec 2013), http://arxiv.org/abs/1310.7417v3
14. Mac Lane, S.: Categories for the Working Mathematician. Springer, 2nd edn. (1998)
15. Panangaden, P.: Labelled Markov Processes. Imperial College Press (2009)
16. Rutten, J.: Universal coalgebra: a theory of systems. Theoretical Computer Science 249, 3–80 (2000)
17. Silva, A., Bonchi, F., Bonsangue, M.M., Rutten, J.J.M.M.: Generalizing determinization from automata to coalgebras. Logical Methods in Computer Science 9(1:09) (2013)
18. Sokolova, A.: Probabilistic systems coalgebraically: A survey. Theoretical Computer Science 412(38), 5095–5110 (2011), CMCS Tenth Anniversary Meeting