Enhanced Coinduction

Rot, J.C.

Chapter 3

Preliminaries

In the previous chapter, we studied coinduction for languages and deterministic automata. Deterministic automata are a special case of the theory of coalgebras, which encompasses coinduction principles for a wide variety of systems. In the remaining chapters we develop theory at this more abstract coalgebraic level, so that the results in Chapter 2 are just one instance, among others. In the current chapter we recall standard notions and results on coalgebras, coinduction and algebras. We assume familiarity with basic concepts from category theory such as functors and natural transformations (see, e.g., [Awo10, Lan98]).

Below, we first fix some basic notation regarding sets, relations, functions and categories. Then we introduce coalgebras, homomorphisms and bisimulations, and discuss examples of coinductive techniques (Section 3.1). We proceed to discuss a more classical interpretation of coinduction, and relate this to the coalgebraic perspective in Section 3.2. This discussion of coinduction is continued in Section 3.3, where we recall an approach to coinduction based on the categorical notion of fibrations. We recall algebras for functors and monads in Section 3.4, and conclude this chapter with a discussion of distributive laws and bialgebras (Section 3.5).

Section 3.3, on coinduction in a fibration, can be challenging to understand without prior knowledge of fibrations. However, in this thesis it is only required for Chapter 5. Moreover, most of Chapter 4 requires only basic concepts on coalgebras (Section 3.1) and algebras (the beginning of Section 3.4).

Most of the material in this chapter is taken from the literature; for more information, see, e.g., [Rut00, JR12, Jac12, Len98] (coalgebras and coinduction), [HJ98, HCKJ13] (coinduction in a fibration), [BW05, Awo10, Tur96] (algebras and monads) and [TP97, Kli11, Bar04] (distributive laws and bialgebras).

Sets. By Set we denote the category of sets and functions. We write 1 for the singleton $\{*\}$, 2 for the two elements set $\{0, 1\}$, $\mathbb{N}$ for the set of natural numbers and $\mathbb{R}$ for the set of real numbers. Given sets X and Y, $X \times Y$ is the Cartesian product of X and Y (with the usual projection maps π_1 and π_2) and $X + Y$ is the coproduct, i.e., disjoint union (with coproduct injections κ_1, κ_2).

Relations. Given a relation $R \subseteq X \times Y$, we write $\pi_1 \colon R \to X$ and $\pi_2 \colon R \to Y$ for its left and right projection, respectively. Given another relation $S \subseteq Y \times Z$ we denote the composition of R and S by $R \circ S$. We let $R^{op} = \{(y, x) \mid (x, y) \in R\}$. The diagonal relation on a set X is $\Delta_X = \{(x, x) \mid x \in X\}$.

Functions. Let $f \colon X \to Y$ be a function. The direct image of a set $S \subseteq X$ under f is denoted simply by $f(S) = \{f(x) \mid x \in S\}$, and the inverse image of $V \subseteq Y$ by $f^{-1}(V) = \{x \mid f(x) \in V\}$. The kernel of f is given by $\mathsf{ker}(f) = \{(x, y) \mid f(x) = f(y)\}$. The pairing of two functions f, g with a common domain is denoted $\langle f, g \rangle$ and the copairing (for functions f, g with a common codomain) is denoted by $[f, g]$. The set of functions from X to Y is denoted by Y^X; if we fix X, this yields a (covariant) functor on Set. The i-fold application of a function $f \colon X \to X$ is denoted by f^i, i.e., $f^0 = \mathsf{id}$ and $f^{i+1} = f \circ f^i$.

Categories. On any category, we write Id for the identity functor, and id_X or simply id for the identity morphism of an object X. The product of categories $\mathcal{C}$ and $\mathcal{D}$ is denoted by $\mathcal{C} \times \mathcal{D}$; an object of $\mathcal{C} \times \mathcal{D}$ is a pair consisting of an object from $\mathcal{C}$ and one from $\mathcal{D}$, and an arrow is a pair of arrows from $\mathcal{C}$ and $\mathcal{D}$ of the matching types. Any two functors $F \colon \mathcal{C} \to \mathcal{D}$ and $G \colon \mathcal{C}' \to \mathcal{D}'$ yield a functor $F \times G \colon \mathcal{C} \times \mathcal{C}' \to \mathcal{D} \times \mathcal{D}'$. We use the same notation for the product of functors (in a category of functors and natural transformations), i.e., given F, G as above so that $\mathcal{C} = \mathcal{C}'$, $\mathcal{D} = \mathcal{D}'$ and $\mathcal{D}$ has products, we let $(F \times G)(X) = FX \times GX$. It should always be clear from the context which meaning of $\times$ is referred to.

Given a set X, $\mathcal{P}(X)$ is the set of subsets of X, and $\mathcal{P}_\omega(X)$ is the set of finite subsets of X. Both $\mathcal{P}$ and $\mathcal{P}_\omega$ extend to functors on Set, defined on functions by direct image: $\mathcal{P}(f)(V) = f(V)$ and $\mathcal{P}_\omega(f)(V) = f(V)$. Given a semiring $\mathbb{S}$, we denote by $\mathcal{M}X$ the set of linear combinations of X with coefficients in $\mathbb{S}$. Formally, it is defined by $\mathcal{M}X = \{\varphi \in \mathbb{S}^X \mid \mathsf{supp}(\varphi) \text{ is finite}\}$, where $\mathsf{supp}(\varphi) = \{x \mid \varphi(x) \neq 0\}$. This extends to a functor $\mathcal{M} \colon \mathsf{Set} \to \mathsf{Set}$, sending $f \colon X \to Y$ to $\mathcal{M}(f)(\varphi) = \lambda y. \sum_{x \in f^{-1}(y)} \varphi(x)$. We often denote a linear combination $\varphi \in \mathcal{M}X$ by a formal sum of the form $\sum s_i x_i$, where $s_i \in \mathbb{S}$ and $x_i \in X$ for all i.

3.1 Coalgebras

A *coalgebra* for a functor $B \colon \mathcal{C} \to \mathcal{C}$, or *$B$-coalgebra*, is a pair (X, δ) where X is an object in $\mathcal{C}$ and $\delta \colon X \to BX$ a morphism. We often refer to X as the carrier or state space, δ as the transition map or transition structure, and B as the behaviour functor. A *(coalgebra) homomorphism* from (X, δ) to (Y, ϑ) is a map $h \colon X \to Y$ such that $\vartheta \circ h = Bh \circ \delta$:

$$\begin{array}{ccc} X & \xrightarrow{h} & Y \\ {\scriptstyle\delta}\downarrow & & \downarrow{\scriptstyle\vartheta} \\ BX & \xrightarrow[Bh]{} & BY \end{array}$$

The category of B-coalgebras is denoted by B-coalg.

A B-coalgebra (Z, ζ) is *final* if there exists, for any B-coalgebra (X, δ), a unique homomorphism from (X, δ) to (Z, ζ). Final coalgebras are unique up to isomorphism, therefore we often speak about *the* final coalgebra. In general, a final B-coalgebra does not necessarily exist, but there are mild conditions on B under which it does: for instance, when B is a bounded functor on Set (see, e.g., [Rut00]). The *coinductive extension* of a coalgebra (X, δ) is the unique homomorphism into the final coalgebra. Following [JR12], we make a conceptual identification of (coalgebraic) *coinduction* with the use of finality in categories of coalgebras. As we will see below, the unique existence of morphisms gives rise both to *definition* principles and to *proof* principles.

In the remainder of this section we assume that B is a functor on Set. Given a B-coalgebra (X, δ) and states $x, y \in X$, we say x and y are *behaviourally equivalent* or *observationally equivalent* if there exists a coalgebra homomorphism h from (X, δ) to some B-coalgebra so that $h(x) = h(y)$. In particular, $x, y \in X$ are behaviourally equivalent precisely if they are identified by the coinductive extension of δ. The largest relation on X containing only behaviourally equivalent pairs is called *behavioural equivalence*. We denote this relation by $\approx_\delta$, or simply $\approx$.

Example 3.1.1. We list several examples of coalgebras; see, e.g., [Rut00] for more.

1. Let $BX = A \times X$, for a fixed set A. A B-coalgebra $\langle o, t \rangle \colon X \to A \times X$ is a *stream system* (over A). For each state $x \in X$, we observe an output $o(x) \in A$, and a next state $t(x) \in X$.

   The *final* B-coalgebra is $\langle (-)_0, (-)' \rangle \colon A^\omega \to A \times A^\omega$, where $A^\omega = \{\sigma \mid \sigma \colon \mathbb{N} \to A^\omega\}$ is the set of *streams* over A, and for any stream $\sigma \in A^\omega$: $\sigma_0 = \sigma(0)$ and $\sigma'(n) = \sigma(n+1)$ for all $n \in \mathbb{N}$. The coinductive extension of a stream system $\langle o, t \rangle \colon X \to A \times X$ maps a state x to the stream $(o(x), o(t(x)), o(t(t(x))), \ldots)$.

   Stream systems do not involve termination, and therefore they generate only infinite streams. The final coalgebra of $(A \times \mathsf{Id}) + 1$ consists of all finite and infinite streams over A.

2. A *labelled transition system* over a set of labels A is a coalgebra for the functor $BX = \mathcal{P}(A \times X)$. Indeed, a B-coalgebra consists of a set X of states and a map $\delta \colon X \to \mathcal{P}(A \times X)$ that sends each state to a set of transitions. We write $x \xrightarrow{a} y$ if $(a, y) \in \delta(x)$. Labelled transition systems can equivalently be presented as coalgebras for the functor $(\mathcal{P}-)^A$. A *finitely branching* transition system is a coalgebra for the functor $\mathcal{P}_\omega(A \times \mathsf{Id})$. An *image finite* transition system is a coalgebra for $(\mathcal{P}_\omega -)^A$.

   The functor $\mathcal{P}(A \times \mathsf{Id})$ does not have a final coalgebra, for cardinality reasons (the transition map of any final coalgebra is an isomorphism). Nevertheless, $\mathcal{P}_\omega(A \times \mathsf{Id})$ has a final coalgebra: it consists of (finitely branching) trees edge-labelled in A, and quotiented by strong bisimilarity in the usual sense (see below). Similarly, $(\mathcal{P}_\omega -)^A$ has a final coalgebra given by equivalence classes of trees in which every node has only a finite number of a-successors, for each $a \in A$.

3. Let $BX = 2 \times X^A$. A coalgebra $\langle o, t \rangle \colon X \to BX$ is a deterministic automaton; a state x is accepting if $o(x) = 1$, and x makes an a-transition to y (denoted $x \xrightarrow{a} y$) if $t(x)(a) = y$.

   The final coalgebra for $2 \times \mathsf{Id}^A$ is the deterministic automaton introduced in Section 2.1: its carrier is given by the set 2^{A^*} of all languages over A, a state accepts if the corresponding language contains the empty word, and the transition map is given by language derivative. Given any deterministic automaton $\langle o, t \rangle \colon X \to 2 \times X^A$, the coinductive extension $l \colon X \to 2^{A^*}$ is the usual language semantics, i.e., for any $x \in X$: $l(x)(\varepsilon) = o(x)$ and $l(x)(aw) = l(t(x)(a))(w)$.

   More generally, we can consider *Moore automata*, which are coalgebras for the functor $BX = S \times X^A$, where S is a set of outputs. The carrier of the final coalgebra is S^{A^*}.

4. A *non-deterministic automaton* is a coalgebra for $BX = 2 \times (\mathcal{P}_\omega X)^A$. Given a coalgebra $\langle o, t \rangle \colon X \to 2 \times (\mathcal{P}_\omega X)^A$, for each state $x \in X$, a state is accepting if $o(x) = 1$, and for each $a \in A$ there is a set of next states $t(x)(a)$. We write $x \xrightarrow{a} y$ for $y \in t(x)(a)$.

   The final coalgebra of B does *not* consist of languages. Rather, it consists of trees edge-labelled in A and node-labelled in 2, quotiented by strong bisimilarity. Thus, the branching behaviour of automata is taken into account, and therefore we obtain a finer notion of behavioural equivalence than that arising from the usual language semantics.

5. Let $\mathbb{S}$ be a semiring, and $\mathcal{M}$ the associated functor mapping sets to linear combinations with coefficients in $\mathbb{S}$. A *weighted transition system* is a coalgebra for the functor $BX = (\mathcal{M}X)^A$. A *weighted automaton* is a weighted transition system where states additionally feature output, i.e., a coalgebra for the functor $\mathbb{S} \times (\mathcal{M}-)^A$. Weighted automata accept weighted languages, but the final coalgebra of B distinguishes more, similar to the case of non-deterministic automata; see [BBB$^+$12] for details.

3.1.1 Coinductive definitions

The final B-coalgebra provides a canonical semantics for B-coalgebras. In particular, we can use finality to define operations on the final coalgebra. As an elementary example, consider the functor $\mathbb{R} \times \mathsf{Id}$ of stream systems over the reals, and recall that its final coalgebra is the set of streams $\mathbb{R}^\omega$. To define a pointwise sum on streams, we construct a coalgebra $\langle o, t \rangle \colon \mathbb{R}^\omega \times \mathbb{R}^\omega \to \mathbb{R} \times (\mathbb{R}^\omega \times \mathbb{R}^\omega)$ as follows: $o(\sigma, \tau) = \sigma_0 + \tau_0$ and $t(\sigma, \tau) = (\sigma', \tau')$ (where we use the operations $(-)_0$ and $(-)'$, which form the transition map of the final coalgebra, see Example 3.1.1 (1)). By

finality this gives rise to a unique homomorphism h:

$$\begin{array}{ccc} \mathbb{R}^\omega \times \mathbb{R}^\omega & \overset{h}{- \!\!-\!\!\!\rightarrow} & \mathbb{R}^\omega \\ \Big\downarrow{\scriptstyle \langle o,t\rangle} & & \Big\downarrow{\scriptstyle \langle (-)_0,(-)'\rangle} \\ \mathbb{R} \times (\mathbb{R}^\omega \times \mathbb{R}^\omega) & \xrightarrow[\mathsf{id}\times h]{} & \mathbb{R} \times \mathbb{R}^\omega \end{array}$$

which maps a pair of streams to their pointwise sum.

The above way of coinductively specifying and defining operations on streams is a special case of *behavioural differential equations* [Rut03] (see also Chapter 2), in which an operation is defined by specifying its initial value $(-)_0$ and its derivative $(-)'$. We illustrate this by defining several operators:

Initial value	Differential equation	Name
$(\sigma + \tau)_0 = \sigma_0 + \tau_0$	$(\sigma + \tau)' = \sigma' + \tau'$	sum
$(\sigma \times \tau)_0 = \sigma_0 \cdot \tau_0$	$(\sigma \times \tau)' = \sigma' \times \tau + [\sigma_0] \times \tau'$	convolution product
$[r]_0 = r$	$[r]' = [0]$	constant (for any $r \in \mathbb{R}$)

In the first column, the operations $+$ and $\cdot$ on the right of the equations are the standard operations on $\mathbb{R}$. We associate a set T of *terms* to the above operators, defined by the grammar

$$t ::= \sigma \mid t_1 + t_2 \mid t_1 \times t_2 \mid [r] \tag{3.1}$$

where σ ranges over $\mathbb{R}^\omega$ and $[r]$ ranges over $\{[r] \mid r \in \mathbb{R}\}$. Now the above differential equations specify how to define a stream system $T \to \mathbb{R} \times T$. The unique coalgebra morphism $T \to \mathbb{R}^\omega$ then provides the semantics of the operators [Rut03, HKR14]. In Section 3.5 we will see how to study such coinductive definition methods in a structured, categorical way.

In Chapter 2 we have seen behavioural differential equations for languages; notice that the characterization of union and concatenation of Lemma 2.1.3 resembles the above definition of the sum and convolution product on streams. One difference to the previous chapter is that there, we *characterize* pre-defined operations using differential equations, whereas here we use the differential equations to *define* the operations.

Two more operations on streams, which we study in the next chapter, are *shuffle* and *shuffle inverse*:

Initial value	Differential equation	Name
$(\sigma \otimes \tau)_0 = \sigma_0 \cdot \tau_0$	$(\sigma \otimes \tau)' = \sigma' \otimes \tau + \sigma \otimes \tau'$	shuffle product
$(\sigma^{-1})_0 = (\sigma_0)^{-1}$	$(\sigma^{-1})' = -\sigma' \otimes (\sigma^{-1} \otimes \sigma^{-1})$	shuffle inverse

The inverse is only defined on streams σ for which $\sigma_0 \neq 0$. We abbreviate $[-1] \otimes \sigma$ by $-\sigma$. The set of terms involving sum, shuffle product and inverse can be defined as before by a grammar. However, since the inverse is only defined when $\sigma_0 \neq 0$, it is not directly clear how to turn the set of terms into a stream system. We call

a term *well-formed* if the inverse is never applied to a subterm with initial value 0; this notion can be straightforwardly defined by induction, and we let T_{wf} be the set of well-formed terms. This set can now be turned into a stream system by induction, using the above specification.

A different use of coalgebras is to study *determinization* constructions at an abstract level, so that language semantics arises by finality [JSS12, SBBR13, Rut00]. Consider a non-deterministic automaton $\langle o, t\rangle \colon X \to 2 \times (\mathcal{P}_\omega X)^A$. As discussed in Example 3.1.1, the coinductive extension of such an automaton does not map a state to the language it accepts. However, we can turn this coalgebra into a deterministic automaton

$$\langle o^\sharp, t^\sharp\rangle \colon \mathcal{P}_\omega X \to 2 \times (\mathcal{P}_\omega X)^A$$

according to the standard powerset construction. This is a deterministic automaton, and the language accepted by a singleton $\{x\}$ is precisely the language accepted by the state x of the original non-deterministic automaton.

For another example of a determinization construction, consider a weighted automaton $\langle o, t\rangle \colon X \to \mathbb{S}\times(\mathcal{M}X)^A$. This induces a Moore automaton $\langle o^\sharp, t^\sharp\rangle \colon \mathcal{M}X \to \mathbb{S} \times (\mathcal{M}X)^A$ where $o^\sharp \colon \mathcal{M}X \to \mathbb{S}$ and $t^\sharp \colon \mathcal{M}X \to (\mathcal{M}X)^A$ are the linear extensions of o and t. By finality, a unique coalgebra homomorphism $l \colon \mathcal{M}X \to \mathbb{S}^{A^*}$ arises, which corresponds to the language semantics of weighted automata. For a detailed explanation see [BBB$^+$12, Section 3] and Example 3.5.2.

3.1.2 Bisimulations and coinductive proofs

The definition of coalgebra homomorphisms provides us with a canonical notion of behavioural equivalence. However, this does not directly give us associated proof techniques, other than the rather abstract property that coinductive extensions are unique. A more concrete proof method is provided by the notion of bisimilarity, which is another fundamental part of the theory of coalgebras. Next, we introduce bisimulations and show a number of concrete examples, and subsequently relate bisimilarity to behavioural equivalence.

A relation $R \subseteq X \times Y$ is a *bisimulation* between coalgebras (X, δ) and (Y, ϑ) if there exists a transition map $\gamma \colon R \to BR$ such that the projections π_1 and π_2 of R are coalgebra homomorphisms, which means that the following diagram commutes [AM89]:

$$\begin{array}{ccccc} X & \xleftarrow{\pi_1} & R & \xrightarrow{\pi_2} & Y \\ {\scriptstyle\delta}\downarrow & & \downarrow{\scriptstyle\gamma} & & \downarrow{\scriptstyle\vartheta} \\ BX & \xleftarrow[B\pi_1]{} & BR & \xrightarrow[B\pi_2]{} & BY \end{array}$$

If $(X, \delta) = (Y, \vartheta)$ then we call R a bisimulation *on* (X, δ). The greatest bisimulation on a given coalgebra (X, δ) is called *bisimilarity* and is denoted by $\sim_\delta$, or simply $\sim$ if δ is clear from the context.

Example 3.1.2.

1. Let $\langle o, t\rangle \colon X \to A \times X$ be a stream system. A relation $R \subseteq X \times X$ is a bisimulation if for all $(x, y) \in R$: $o(x) = o(y)$ and $(t(x), t(y)) \in R$.

   As an example, let T be the set of terms as defined in Equation (3.1), and let $\langle (-)_0, (-)'\rangle \colon T \to \mathbb{R} \times T$ be the stream system defined by the corresponding behavioural differential equations. Let us prove that $s + u \sim u + s$ for any streams s, u. To this end, consider the relation $R = \{(s+u, u+s) \mid s, u \in \mathbb{R}^\omega\}$. For any s, u we have $(s + u)_0 = s_0 + u_0 = u_0 + s_0 = (u + s)_0$. Moreover, $(s + u)' = (s' + u')\ R\ (u' + s') = (u + s)'$. Thus, R is a bisimulation. As we will see below in a more general fashion, this implies that s and u are mapped to the same element in the final coalgebra, meaning that they are assigned the same behaviour. Commutativity of the sum is admittedly a rather trivial property, but it serves here to illustrate the basic methodology of constructing a bisimulation. For many examples of such proofs for streams, see [Rut03, HKR14]; we will also see more advanced proofs in Section 4.2.

2. On labelled transition systems, bisimilarity coincides with the classical notion of strong bisimilarity introduced by Milner and Park [Mil80, Par81]. Given $\delta \colon X \to \mathcal{P}(A \times X)$, a relation $R \subseteq X \times X$ is a bisimulation if for all $(x, y) \in R$: if $x \xrightarrow{a} x'$ then there is y' such that $y \xrightarrow{a} y'$ and $(x', y') \in R$; and if $y \xrightarrow{a} y'$ then there is x' such that $x \xrightarrow{a} x'$ and $(x', y') \in R$.

3. Let $\langle o, t\rangle \colon X \to S \times X^A$ be a Moore automaton. A relation $R \subseteq X \times X$ is a bisimulation if for all $(x, y) \in R$: $o(x) = o(y)$ and for all $a \in A$: $(t(x)(a), t(y)(a)) \in R$. The notion of bisimulation on deterministic automata (Definition 2.1.1) is a special case, and a concrete example of such a bisimulation is in Example 2.1.5.

4. Let $\delta \colon X \to X + 1$ be a coalgebra (for the functor $BX = X + 1$). A relation $R \subseteq X \times X$ is a bisimulation if for any pair $(x, y) \in R$: either $\delta(x) = * = \delta(y)$ or $(\delta(x), \delta(y)) \in R$.

Coalgebra homomorphisms preserve bisimilarity.

Lemma 3.1.3 ([Rut00], Lemma 5.3)**.** *Suppose $f \colon X \to Y$ and $g \colon X \to Z$ are coalgebra homomorphisms. If $R \subseteq X \times X$ is a bisimulation then $(f \times g)(R)$ is a bisimulation.*

If the functor B preserves weak pullbacks, then the inverse image of a bisimulation along a coalgebra homomorphism is again a bisimulation [Rut00, Lemma 5.9]. Thus, in that case, homomorphisms also reflect bisimilarity.

The uniqueness of morphisms into the final coalgebra is, by Lemma 3.1.3 and the fact that the diagonal relation on any coalgebra is a bisimulation [Rut00, Proposition 5.1], equivalent to the following property.

Theorem 3.1.4. *Suppose B has a final coalgebra (Z, ζ). For any $x, y \in Z$:*

$$x \sim y \text{ iff } x = y\,.$$

This is sometimes called strong extensionality, the coinductive proof principle or simply coinduction. Together with Lemma 3.1.3, it entails that, given the bisimilarity relation $\sim$ on any coalgebra:

$$x \sim y \text{ implies } h(x) = h(y) \tag{3.2}$$

where h is the coinductive extension of that coalgebra. Thus, in order to prove that two states have the same behaviour, it suffices to construct a bisimulation.

Example 3.1.5. The foundation of the previous chapter is its coinduction principle Theorem 2.1.2, which states that bisimilarity of languages implies their equality. Indeed, languages form the final coalgebra for the functor $BX = 2 \times X^A$ of deterministic automata, and thus that coinduction principle is an instance of Theorem 3.1.4. Further, Equation (3.2) asserts that bisimilarity on any deterministic automaton implies behavioural equivalence. This means that, to prove that two states of an arbitrary deterministic automaton accept the same language, it suffices to prove that they are bisimilar.

If the functor B preserves weak pullbacks, then homomorphisms reflect bisimilarity, and thus together with Theorem 3.1.4 it implies the converse of (3.2).

Lemma 3.1.6. *If B preserves weak pullbacks then bisimilarity and behavioural equivalence coincide, on any B-coalgebra.*

As an example, the functor $BX = 2 \times X^A$ preserves weak pullbacks. Consequently, two states of a deterministic automaton accept the same language if and only if they related by a bisimulation.

Weak pullback preservation is a mild condition: for instance, it is satisfied by all functors mentioned in Example 3.1.1, except weighted automata and weighted transition systems. For weighted systems, weak pullback preservation only holds under certain conditions on the semiring [GS01, Kli09, BBB$^+$12]. In the cases where it does not hold, behavioural equivalence seems to be of more interest.

3.2 Classical and coalgebraic coinduction

A standard formalization of coinduction is in terms of complete lattices. This is, for instance, the basis of Sangiorgi's introductory text on coinduction [San12a]. This perspective on coinduction, which we call *classical* coinduction (as opposed to coalgebraic coinduction) also plays an important role in this thesis, therefore we recall the basics. In this section we also see how to define coalgebraic bisimulations in the lattice-theoretic setting, and how classical coinduction is generalized by coalgebraic coinduction, i.e., the finality principle in categories of coalgebras.

The starting point is a *complete lattice*: a partial order $(L, \leq)$ in which each subset of L has both a least upper bound and a greatest lower bound. Given a function $f\colon L \to L$, an element $x \in L$ is a fixed point of f if $f(x) = x$, and a post-fixed point if $x \leq f(x)$. If f is monotone (that is, $x \leq y$ implies $f(x) \leq f(y)$) then

by the Knaster-Tarski theorem it has a greatest fixed point $\mathsf{gfp}(f)$, which is also the greatest post-fixed point (see, e.g., [San12a]).

The existence of a greatest fixed point constitutes a coinductive *definition* principle: we call $\mathsf{gfp}(f)$ the *coinductive predicate* defined by f. The fact that it is the greatest post-fixed point constitutes a coinductive *proof* principle: to prove that $x \leq \mathsf{gfp}(f)$, it suffice to show that $x \leq f(x)$. In the sequel we shall sometimes refer to post-fixed points of f as *f-invariants*.

Example 3.2.1. Consider the lattice $L = \mathcal{P}(A^\omega \times A^\omega)$ consisting of relations on streams, ordered by inclusion. Define the monotone function $f \colon L \to L$ by

$$f(R) = \{(\sigma, \tau) \mid \sigma_0 = \tau_0 \text{ and } (\sigma', \tau') \in R\}$$

where $(-)_0$ and $(-)'$ form the transition structure of the final stream system, as in Example 3.1.1 (1). A relation R is an f-invariant (post-fixed point of f) precisely if it is a bisimulation on the final stream system. Since f is monotone, the coinductive predicate (the greatest fixpoint) exists: it is given by bisimilarity on the final coalgebra of stream systems, that is, the diagonal relation on streams. The coinductive proof principle asserts that any bisimulation is contained in bisimilarity.

Notice that the above example can be adapted to define bisimilarity on any stream system with carrier X, by replacing $A^\omega \times A^\omega$ by $X \times X$, and replacing $(-)_0$ and $(-)'$ in the definition of f by the transition map of the stream system under consideration. Classical coinduction easily accommodates other predicates than bisimilarity, as shown by a few basic examples below.

Example 3.2.2.

1. Let $\langle o, t \rangle \colon X \to A \times X$ be a stream system where A is equipped with a partial order $\leq$, and consider the lattice $\mathcal{P}(X \times X)$ of relations on X ordered by inclusion. We define a monotone function $f \colon \mathcal{P}(X \times X) \to \mathcal{P}(X \times X)$ on this lattice:
$$f(R) = \{(x, y) \mid o(x) \leq o(y) \text{ and } (t(x), t(y)) \in R\}\,.$$
A relation R is an f-invariant if for all $(x, y) \in R$, we have $o(x) \leq o(y)$ and $(t(x), t(y)) \in R$. The coinductive predicate defined by f is the greatest such relation. Two states $x, y \in X$ are related by this coinductive predicate if the stream generated by x is pointwise less than the stream generated by y.

2. Let $\langle o, t \rangle \colon X \to A \times X$ be a stream system. Consider the lattice $\mathcal{P}(X)$ of subsets of X, ordered by inclusion, and define the monotone function $f \colon \mathcal{P}(X) \to \mathcal{P}(X)$ on this lattice as follows:
$$f(P) = \{x \mid o(x) \leq o(t(x)) \text{ and } t(x) \in P\}\,.$$
Then an f-invariant is a set $P \subseteq X$ so that for all $x \in P$: $o(x) \leq o(t(x))$, and $t(x) \in P$. The coinductive predicate defined by f, which is the largest f-invariant, thus captures increasing streams.

3. Let $\langle o, t\rangle \colon X \to 2 \times X^A$ be a deterministic automaton, and consider the monotone function f on the lattice of relations on X, defined as follows:

$$f(R) = \{(x, y) \mid o(x) \leq o(y) \text{ and } \forall a \in A.\, (t(x)(a), t(y)(a)) \in R\}$$

A relation R is an f-invariant precisely if it is a *simulation* (Definition 2.4.1). The coinductive predicate defined by f is *similarity*, the greatest simulation.

3.2.1 Coalgebraic bisimulations via relation lifting

In Example 3.2.1 we have seen how to capture bisimulations on stream systems as invariants for a monotone function. Next, we recall a general method of defining a monotone operator on the lattice of relations on the state space of a given coalgebra, so that the coinductive predicate defined by this monotone operator is bisimilarity. This approach was introduced in [HJ98, Rut98b] (see also [Jac12]; and see [Sta11] for a comparison of different notions of bisimulations).

For a functor $B \colon \mathsf{Set} \to \mathsf{Set}$, the (canonical) *relation lifting* $\mathsf{Rel}(B)$ of B maps a relation on X to a relation on BX (for any X). It is defined as follows:

$$\mathsf{Rel}(R \subseteq X \times X) = \{(x, y) \in BX \times BX \mid \exists z.\, B\pi_1(z) = x \text{ and } B\pi_2(z) = y\}$$

where π_1, π_2 are the projections of R. Thus, $\mathsf{Rel}(B)$ is the image of BR under $\langle B\pi_1, B\pi_2\rangle$. For certain classes of functors there are concrete, inductively defined characterizations of relation lifting [HJ98, Jac12].

Now, given a coalgebra $\delta \colon X \to BX$ we define a function

$$\mathsf{b}_\delta = (\delta \times \delta)^{-1} \circ \mathsf{Rel}(B) : \mathcal{P}(X \times X) \to \mathcal{P}(X \times X) \tag{3.3}$$

on the lattice of relations on X ordered by inclusion. Invariants of the function b_δ are bisimulations on δ (defined as in Section 3.1.2), as stated below.

Lemma 3.2.3. *A relation $R \subseteq X \times X$ on the carrier of a coalgebra $\delta \colon X \to BX$ is a bisimulation if and only if $R \subseteq \mathsf{b}_\delta(R)$. Bisimilarity on (X, δ) is the greatest fixed point of b_δ.*

A bisimulation is a relation with a transition *structure*, whereas a b_δ-invariant is a relation with a special *property*. This formulation is taken from [Jac12], to which we refer for a more elaborate comparison. Lemma 3.2.3 asserts that both characterizations are equivalent.

Relation lifting satisfies a number of properties that are be used in subsequent chapters; see [Jac12, Section 4.4] for proofs.

Lemma 3.2.4. *For any functor $B \colon \mathsf{Set} \to \mathsf{Set}$:*

1. $\mathsf{Rel}(B)(\Delta_X) = \Delta_{BX}$.
2. *If $R \subseteq S$ then* $\mathsf{Rel}(B)(R) \subseteq \mathsf{Rel}(B)(S)$.
3. $(\mathsf{Rel}(B)(R))^{op} = \mathsf{Rel}(B)(R^{op})$.

4. $\mathsf{Rel}(B)(R \circ S) \subseteq \mathsf{Rel}(B)(R) \circ \mathsf{Rel}(B)(S)$.

5. $\mathsf{Rel}(B)((f \times f)^{-1}(S)) \subseteq (Bf \times Bf)^{-1}(\mathsf{Rel}(B)(S))$.

If B preserves weak pullbacks, then the inclusions in items 4 and 5 are equalities.

As a consequence of item 2 above, b_δ is monotone.

Theorem 3.2.5. *Let $B \colon \mathsf{Set} \to \mathsf{Set}$ be a functor. The following are equivalent:*

1. *B preserves weak pullbacks.*

2. *$\mathsf{Rel}(B)$ preserves composition, i.e., $\mathsf{Rel}(B)(R \circ S) = \mathsf{Rel}(B)(R) \circ \mathsf{Rel}(B)(S)$.*

This is originally due to Trnková [Trn80]; for an accessible proof, see [Jac12, Theorem 4.4.6] or [KKV12, Fact 3.6].

3.2.2 Classical coinduction in a category

Classical coinduction can be phrased in terms of categories, via the basic observation that any preorder $(X, \leq)$ (and thus in particular any complete lattice) forms a category, whose set of objects is X, and which has an arrow from x to y if and only if $x \leq y$. A *functor* F on such a category is a monotone function on the preorder, and F-coalgebras are post-fixed points of F (seen as a monotone function). The *final* F-coalgebra then corresponds to the coinductive predicate defined by (the monotone map) F (see, e.g., [NR09, HCKJ13]).

The *definition* principle of classical coinduction here is reformulated to the definition of a final F-coalgebra, whereas the *proof* principle is the existence of a morphism from any F-coalgebra into the final coalgebra. In this setting, we will often refer to F-coalgebras as *F-invariants*. Instantiated to a lattice, the finality principle entails that any F-invariant is below the coinductive predicate defined by F. In this sense, the identification of coinduction with finality in categories of coalgebras still applies in this setting. However, the definition and proof principles carry a significantly different intuition than those discussed in Section 3.1; there, we were defining maps into the final coalgebra and reasoning about them, whereas here we take the final coalgebra itself as the defined object of interest, and the existence of arrows as a proof principle.

Example 3.2.6. Let Pred_X be the category of predicates on a fixed set X, as given by the complete lattice of subsets of X. Let $\delta \colon X \to \mathcal{P}_\omega(A \times X)$ be a labelled transition system, where the set of labels A contains a distinguished element $\tau \in A$. We define a functor $F \colon \mathsf{Pred}_X \to \mathsf{Pred}_X$ by

$$F(P \subseteq X) = \{x \in X \mid \exists y.(\tau, y) \in \delta(x)\}\,. \tag{3.4}$$

An F-invariant (F-coalgebra) is a predicate $P \subseteq X$ so that for any $x \in P$, there exists a τ-transition into a state that is again in P, that is, there is $y \in P$ such that $(\tau, y) \in \delta(x)$. The coinductive predicate defined by F is simply the greatest

fixed point of F seen as a monotone function; thus, it is the largest subset of states $x \in X$ that *may diverge*, that is, states that have an infinite path of τ steps. In terms of modal logic, these are the states that satisfy $\nu u.\langle\tau\rangle u$.

In the above example, the *system* of interest is modelled by a coalgebra for the functor $\mathcal{P}_\omega(A \times \mathsf{Id})\colon \mathsf{Set} \to \mathsf{Set}$. The *invariants* of interest are coalgebras in a category of predicates.

3.3 Liftings and coinduction in a fibration

We have established coinduction as the principle of finality in a category of coalgebras. In Section 3.1 we have seen how to instantiate this to a setting where coalgebras model the systems of interest, yielding a canonical way of assigning behaviour and equivalence to a coalgebra. In this setting, coinduction provides a systematic account of bisimilarity and behavioural equivalence for all systems of the given type. On the other hand, in Section 3.2 we have seen how a different instantiation of coinduction yields the classical lattice-theoretic account, which is very flexible and allows to define many other predicates than bisimilarity, but is mainly suitable to define predicates on a single system. Here, bisimilarity and other predicates can be seen as objects that live in a category of predicates.

A very general and systematic approach for studying coinductive predicates on coalgebras can be achieved if the coalgebras of interest live in the base category of a *fibration*. This provides a means to speak about *properties* or *predicates* on coalgebras of interest. In this setting, invariants and coinductive predicates on a given coalgebra, are themselves coalgebras in a category of predicates, similar to the situation in the previous section. The functor on these predicates is defined in a uniform manner, based on a *lifting* of the behaviour functor.

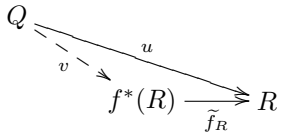
This fibrational approach to coinductive predicates for coalgebras was proposed in [HJ98], and further developed in [HCKJ13] (as well as [AGJJ12, GJF13]). Below, we first list the necessary definitions related to fibrations (Section 3.3.1), and then describe the fibrational approach to coinductive predicates (Section 3.3.2). All of the examples in this thesis are based on two fibrations, described in Example 3.3.1 and Example 3.3.2. Of the remaining chapters in this thesis, the material in the current section is only necessary to understand Chapter 5.

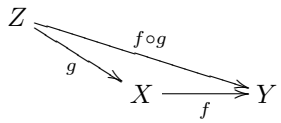
3.3.1 Fibrations

We refer to [Jac99] for more information on fibrations, and recall only a few basic definitions and results.

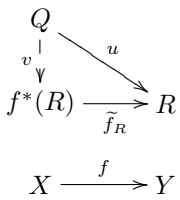
A functor $p\colon \mathcal{E} \to \mathcal{A}$ is called a *fibration* when for every morphism $f\colon X \to Y$ in $\mathcal{A}$ and every R in $\mathcal{E}$ with $p(R) = Y$ there exists an object $f^*(R)$ with $p(f^*(R)) = X$ and a morphism $\widetilde{f}_R\colon f^*(R) \to R$ such that $p(\widetilde{f}_R) = f$ and $\widetilde{f}_R$ is *Cartesian*, which means that the following universal property holds: for all morphisms $g\colon Z \to X$ in $\mathcal{A}$ and $u\colon Q \to R$ in $\mathcal{E}$ sitting above $f \circ g$ (i.e., $p(u) = f \circ g$) there is a unique

morphism $v\colon Q \to f^*(R)$ such that $u = \widetilde{f}_R \circ v$ and $p(v) = g$.



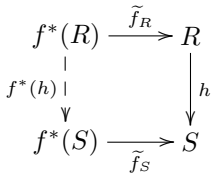


We shall often use a special case of the universal property of $\widetilde{f}_R$ where $p(Q) = X$. Then for any $u\colon Q \to R$ sitting above f there exists a unique $v\colon Q \to f^*(R)$ above id_X such that $\widetilde{f}_R \circ v = u$:



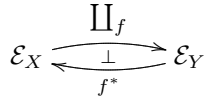
Given a fibration $p\colon \mathcal{E} \to \mathcal{A}$, we call $\mathcal{E}$ the *total category*, and $\mathcal{A}$ the *base category*. The *fibre* above an object X in $\mathcal{A}$, denoted by $\mathcal{E}_X$, is the subcategory of $\mathcal{E}$ with objects mapped by p to X and arrows mapped to the identity on X. We give a few examples of fibrations below; see [Jac99] for many more.

A morphism $\widetilde{f}$ as above is called a *(p)-Cartesian lifting* of f, and is unique up to isomorphism. If we make a choice of Cartesian liftings, the association $R \mapsto f^*(R)$ gives rise to the *reindexing functor* $f^*\colon \mathcal{E}_Y \to \mathcal{E}_X$. On a morphism $h\colon R \to S$ in $\mathcal{E}_Y$, it is defined using the universal property of the Cartesian lifting $\widetilde{f}_S$:



Given morphisms $f\colon X \to Y$ and $g\colon Y \to Z$ in $\mathcal{A}$, there is a natural isomorphism $(g \circ f)^* \cong f^* \circ g^*$ between reindexing functors.

A functor $p\colon \mathcal{E} \to \mathcal{A}$ is called a *bifibration* if both p and $p^{op}\colon \mathcal{E}^{op} \to \mathcal{A}^{op}$ are fibrations. Equivalently [Jac99, Lemma 9.1.2], p is a bifibration if each reindexing functor $f^*\colon \mathcal{E}_Y \to \mathcal{E}_X$ has a left adjoint $\coprod_f$:



We call $\coprod_f$ the *direct image* along f. This choice becomes more clear in the examples below.

For a fibration $p\colon \mathcal{E} \to \mathcal{A}$ we say that p has *fibred finite (co)products* if each fibre has finite (co)products, preserved by reindexing functors. If p is a bifibration with fibred finite products and coproducts, and $\mathcal{A}$ has finite products and coproducts, then the total category $\mathcal{E}$ also has finite products and coproducts, strictly preserved by p [Jac99, Example 9.2.5]. All bifibrations considered in this thesis are assumed to have this structure.

Example 3.3.1 (The predicate bifibration)**.** Let Pred be the category whose objects are pairs of sets (P, X) with $P \subseteq X$ and morphisms $f\colon (P, X) \to (Q, Y)$ are maps $f\colon X \to Y$ so that $f(P) \subseteq Q$. The functor $p\colon \mathsf{Pred} \to \mathsf{Set}$ mapping (P, X) to X is a fibration. The fibre Pred_X above X is the complete lattice of subsets of X ordered by inclusion. For any map $f\colon X \to Y$ in Set the reindexing functor $f^*\colon \mathsf{Pred}_Y \to \mathsf{Pred}_X$ maps (Q, Y) to $(f^{-1}(Q), X)$. Products and coproducts in a fibre Pred_X correspond to intersection and union, respectively. Products and coproducts in the total category $\mathcal{E}$ are simply computed as in Set. The functor f^* has a left adjoint $\coprod_f$ mapping (P, X) to the direct image $(f(P), Y)$.

We note that predicates can alternatively be seen as functions $X \to 2$. Reindexing along a function f then simply becomes precomposition with f.

Example 3.3.2 (The relation bifibration)**.** Similarly, we can consider the category Rel whose objects are pairs of sets (R, X) with $R \subseteq X \times X$ and morphisms $f\colon (R, X) \to (S, Y)$ are maps $f\colon X \to Y$ such that $(f \times f)(R) \subseteq S$. The functor $p\colon \mathsf{Rel} \to \mathsf{Set}$ mapping (R, X) to X is a fibration. The fibre Rel_X above X is the complete lattice of relations on X ordered by inclusion. For $f\colon X \to Y$ in Set the reindexing functor $f^*\colon \mathsf{Rel}_Y \to \mathsf{Rel}_X$ maps (R, Y) to $((f \times f)^{-1}(R), X)$. Its left adjoint $\coprod_f$ is given by direct image, that is, $\coprod_f(R, X) = ((f \times f)(R), Y)$.

Given fibrations $p\colon \mathcal{E} \to \mathcal{A}$ and $p'\colon \mathcal{E}' \to \mathcal{A}'$ and a functor $B\colon \mathcal{A} \to \mathcal{A}'$, we call $\overline{B}\colon \mathcal{E} \to \mathcal{E}'$ a *lifting* of B if the following diagram commutes:

$$\begin{array}{ccc} \mathcal{E} & \xrightarrow{\overline{B}} & \mathcal{E}' \\ {\scriptstyle p}\downarrow & & \downarrow{\scriptstyle p'} \\ \mathcal{A} & \xrightarrow[B]{} & \mathcal{A}' \end{array} \tag{3.5}$$

Such a lifting $\overline{B}$ restricts to a functor $\overline{B}_X\colon \mathcal{E}_X \to \mathcal{E}'_{BX}$ between fibres, for any X in $\mathcal{A}$. We sometimes omit the subscript X when it is clear from the context. A lifting $(\overline{B}, B)$ is a *fibration map* if it maps Cartesian morphisms to Cartesian morphisms. This means that there is an isomorphism

$$(Bf)^* \circ \overline{B} \cong \overline{B} \circ f^* \tag{3.6}$$

for any $\mathcal{A}$-morphism f. An important example for this thesis is the canonical relation lifting $\mathsf{Rel}(B)$, which is a fibration map whenever B preserves weak pullbacks.

Lemma 3.3.3. *For any $B\colon \mathsf{Set} \to \mathsf{Set}$, the lifting $(\mathsf{Rel}(B), B)$ is a fibration map (from the relation fibration to itself) if B preserves weak pullbacks.*

The isomorphism (3.6) means that $\mathsf{Rel}(B)$ preserves inverse images if B preserves weak pullbacks, that is, in that case the inclusion $\mathsf{Rel}(B)((f \times f)^{-1}(S)) \subseteq (Bf \times Bf)^{-1}(\mathsf{Rel}(B)(S))$ is an equality (see Lemma 3.2.4). The inclusion holds for any functor B; it is a special case of the following lemma.

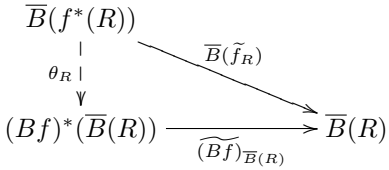
Lemma 3.3.4. *Let $p\colon \mathcal{E} \to \mathcal{A}$ and $p'\colon \mathcal{E}' \to \mathcal{A}'$ be fibrations, and assume $\overline{B}\colon \mathcal{E} \to \mathcal{E}'$ is a lifting of some functor $B\colon \mathcal{A} \to \mathcal{A}'$. For any morphism $f\colon X \to Y$ in $\mathcal{A}$ there is a natural transformation*

$$\theta\colon \overline{B}_X \circ f^* \Rightarrow (Bf)^* \circ \overline{B}_Y : \mathcal{E}_Y \to \mathcal{E}'_{BX}\,.$$

If p and p' are bifibrations then there is another natural transformation

$$\theta'\colon \coprod_{Bf} \circ \overline{B}_X \Rightarrow \overline{B}_Y \circ \coprod_f : \mathcal{E}_X \to \mathcal{E}'_{BY}\,.$$

Proof. To define θ_R on an object R in $\mathcal{E}_Y$, we apply $\overline{B}$ to the p-Cartesian lifting $\widetilde{f}_R\colon f^*(R) \to R$ and use the universal property of the p'-Cartesian lifting $\widetilde{(Bf)}_{\overline{B}R}$:



Naturality follows from the universal property of $\widetilde{(Bf)}_{\overline{B}R}$ and the definition of reindexing functors.

The natural transformation θ' can be defined as follows:

$$\begin{array}{ccc} \coprod_{Bf} \circ \overline{B}_X & \Longrightarrow & \coprod_{Bf} \circ \overline{B}_X \circ f^* \circ \coprod_f \\ & & \Big\Downarrow \coprod_{Bf} \theta \coprod_f \\ & & \coprod_{Bf} \circ (Bf)^* \circ \overline{B}_Y \circ \coprod_f \Longrightarrow \overline{B}_Y \circ \coprod_f \end{array}$$

using the unit of the adjunction $\coprod_f \dashv f^*$ and the counit of the adjunction $\coprod_{Bf} \dashv (Bf)^*$. (The way we obtain θ' from θ is an instance of a more general construction: θ' is called the *(adjoint) mate* of θ.) □

3.3.2 Coinductive predicates in a fibration

Let $p\colon \mathcal{E} \to \mathcal{A}$ be a fibration, and let $B\colon \mathcal{A} \to \mathcal{A}$ be a functor whose coalgebras model the systems of interest. We show how to define functors on the fibre above

the carrier of a B-coalgebra, such that the coalgebras for those functors are the invariants that model coinductive properties of the B-coalgebra in the base category. Following [HJ98], we then say a *coinductive predicate* is a *final coalgebra in a fibre*.

The crucial observation of this approach to coinductive predicates, is that we can uniformly define a functor $\mathcal{E}_X \to \mathcal{E}_X$ for any B-coalgebra $\delta\colon X \to BX$, from a given lifting $\overline{B}\colon \mathcal{E} \to \mathcal{E}$ of B. For a coalgebra $\delta\colon X \to BX$ it is defined as follows:

$$\mathcal{E}_X \xrightarrow{\overline{B}_X} \mathcal{E}_{BX} \xrightarrow{\delta^*} \mathcal{E}_X$$

A coalgebra $R \to \delta^* \circ \overline{B}_X(R)$ is called a $\delta^* \circ \overline{B}_X$*-invariant*; sometimes we shall refer only to the carrier R as an invariant and leave the transition structure implicit. The *final* $\delta^* \circ \overline{B}_X$*-coalgebra* (if it exists) can be seen as the coinductive predicate determined by $\overline{B}$ on the coalgebra δ. Finality of this coinductive predicate amounts to a proof principle: any invariant has a morphism to the coinductive predicate. For instance, if the state space X is a set and $\mathcal{E}_X$ is the lattice of predicates, this principle means that the carrier of any invariant is contained in the coinductive predicate. We refer to [HCKJ13] for more details on the existence of final coalgebras in a fibre.

Example 3.3.5. Recall from Example 3.2.6 the functor F whose final coalgebra is the divergence predicate on some coalgebra for the functor $BX = \mathcal{P}_\omega(A \times X)$. We define a lifting $\overline{B}\colon \mathsf{Pred} \to \mathsf{Pred}$ of B:

$$\overline{B}_X(P \subseteq X) = \{S \subseteq \mathcal{P}_\omega(A \times X) \mid \exists y \in P.(\tau, y) \in S\}$$

Then, given any $\delta\colon X \to BX$, we consider the composition

$$\mathsf{Pred}_X \xrightarrow{\overline{B}_X} \mathsf{Pred}_{BX} \xrightarrow{\delta^*} \mathsf{Pred}_X$$

where δ^* is the reindexing functor, i.e., inverse image along δ. The functor $\delta^* \circ \overline{B}_X$ now coincides with F from Example 3.2.6. Here it is defined uniformly on any B-coalgebra, based on a lifting of B that does not mention any concrete transition system.

Example 3.3.6. The functor $\mathsf{b}_\delta\colon \mathsf{Rel}_X \to \mathsf{Rel}_X$, defined for a given coalgebra $\delta\colon X \to BX$ using relation lifting (see Section 3.2.1), decomposes as

$$\mathsf{Rel}_X \xrightarrow{\mathsf{Rel}(B)_X} \mathsf{Rel}_{BX} \xrightarrow{\delta^*} \mathsf{Rel}_X$$

A $\delta^* \circ \mathsf{Rel}(B)_X$-invariant is simply a b_δ-invariant. Equivalently, it is a bisimulation (Lemma 3.2.3).

Given a lifting $\overline{B}$ of B, we thus have a way of defining a functor on the fibre $\mathcal{E}_X$ above the carrier of any B-coalgebra. We now emphasize that this *uniformly* defines a predicate on B-coalgebras, by showing that coalgebra homomorphisms preserve invariants (and also reflect them, under a certain condition). The second item appears as Proposition 3.11 in [HCKJ13], with a proof in the appendix.

Proposition 3.3.7. *Let $p\colon \mathcal{E} \to \mathcal{A}$ be a bifibration, $\overline{B}\colon \mathcal{E} \to \mathcal{E}$ a lifting of a functor $B\colon \mathcal{A} \to \mathcal{A}$ and let $h\colon X \to Y$ be a coalgebra morphism from $\delta\colon X \to BX$ to $\vartheta\colon Y \to BY$.*

- *If R is a $\delta^* \circ \overline{B}_X$-invariant, then $\coprod_h(R)$ is a $\vartheta^* \circ \overline{B}_Y$-invariant.*
- *If S is a $\vartheta^* \circ \overline{B}_Y$-invariant and $(\overline{B}, B)$ is a fibration map, then $h^*(S)$ is a $\delta^* \circ \overline{B}_X$-invariant.*

Proof. Since h is a coalgebra homomorphism, we have $Bh \circ \delta = \vartheta \circ h$. Thus

$$\delta^* \circ (Bh)^* \cong (Bh \circ \delta)^* = (\vartheta \circ h)^* \cong h^* \circ \vartheta^* \,. \tag{3.7}$$

Using the unit of the adjunction $\coprod_{Bh} \dashv (Bh)^*$ and the counit of $\coprod_h \dashv h^*$ we construct the mate of the above natural transformation (read from left to right):

$$\coprod_h \circ \delta^* \Longrightarrow \coprod_h \circ \delta^* \circ (Bh)^* \circ \coprod_{Bh} \Longrightarrow \coprod_h \circ h^* \circ \vartheta^* \circ \coprod_{Bh} \Longrightarrow \vartheta^* \circ \coprod_{Bh} .$$

We can use this to construct a natural transformation

$$\gamma\colon \coprod_h \circ \delta^* \circ \overline{B}_X \Longrightarrow \vartheta^* \circ \coprod_{Bh} \circ \overline{B}_X \Longrightarrow \vartheta^* \circ \overline{B}_Y \circ \coprod_h$$

where the second part is given by Lemma 3.3.4. Then any $\delta^* \circ \overline{B}$-invariant, that is, a coalgebra $R \to \delta^* \circ \overline{B}_X(R)$ in $\mathcal{E}_X$, yields a coalgebra (invariant) $\coprod_h(R) \to \vartheta^* \circ \overline{B}_Y \circ \coprod_h(R)$ in $\mathcal{E}_Y$, simply by applying $\coprod_h$ and the natural transformation γ.

For the second item, we construct a natural isomorphism

$$h^* \circ \vartheta^* \circ \overline{B}_Y \cong \delta^* \circ (Bh)^* \circ \overline{B}_Y \cong \delta^* \circ \overline{B}_X \circ h^*$$

using (3.7) and the fact that $(\overline{B}, B)$ is a fibration map. Then, given an invariant $S \to \vartheta^* \circ \overline{B}_Y(S)$, we apply the isomorphism to get the invariant $h^*(S) \to h^* \circ \vartheta^* \circ \overline{B}_Y(S) \cong \delta^* \circ \overline{B}_X \circ h^*(S)$. □

As stated in Lemma 3.2.3, a relation R is a bisimulation on a coalgebra δ precisely if it is a $\delta^* \circ \mathsf{Rel}(B)$-invariant. Thus, the first item of the above Proposition 3.3.7 is a generalization of the fact that coalgebra homomorphisms preserve bisimilarity (Lemma 3.1.3), for $\overline{B}$ instantiated to the canonical relation lifting $\mathsf{Rel}(B)$ (Lemma 3.1.3 mentions two homomorphisms rather than one; this can also be accommodated in the current setting by choosing a slightly different fibration). Moreover, if B preserves weak pullbacks, then $(\mathsf{Rel}(B), B)$ is a fibration map (Lemma 3.3.3). Hence, a special case of the second item is that bisimulations are preserved by inverse image along coalgebra homomorphisms, whenever the functor B preserves weak pullbacks.

3.4 Algebras

In this thesis, algebras play an important role to model coalgebras whose carrier has algebraic structure; for example, the set of closed terms over some signature.

Other examples include automata over sets or linear combinations of states, which arise in determinization constructions.

An *algebra* for a functor $T\colon \mathcal{C} \to \mathcal{C}$, or *$T$-algebra*, is a pair (X, α) where X is an object in $\mathcal{C}$ and $\alpha\colon TX \to X$ is a morphism. We call X the carrier and α the algebra structure. An *(algebra) homomorphism* from $\alpha\colon TX \to X$ to $\beta\colon TY \to Y$ is a function $h\colon X \to Y$ such that $h \circ \alpha = \beta \circ Th$. The category of algebras and their homomorphisms is denoted by T-alg.

An *initial T-algebra* is an initial object in the category T-alg. Thus, given an initial T-algebra (A, κ) there exists, for each T-algebra (X, α) a unique algebra homomorphism from (A, κ) to (X, α). We call such a morphism the *inductive extension* of (X, α). Similar to the case of final coalgebras, initial algebras exist under mild conditions on the functor.

A *signature* Σ is a (possibly infinite) set of operator names $\sigma \in \Sigma$ with (finite) arities $|\sigma| \in \mathbb{N}$. Equivalently, it is a polynomial functor on Set:

$$\begin{aligned} \Sigma X &= \coprod_{\sigma\in\Sigma} \{\sigma\} \times X^{|\sigma|} \cong \{\sigma(x_1, \ldots, x_{|\sigma|}) \mid \sigma \in \Sigma \text{ and } \forall i.\, x_i \in X\} \\ (\Sigma(f\colon X \to Y))(\sigma(x_1, \ldots, x_n)) &= \sigma(f(x_1), \ldots, f(x_n)) \end{aligned} \tag{3.8}$$

A Σ-algebra coincides with the standard notion of an interpretation of the signature Σ: a set X together with a function of type $X^{|\sigma|} \to X$ for every operator σ (see also Section 2.3). The carrier of the initial Σ-algebra is given by the set of all *closed terms* over the signature.

3.4.1 Monads

A *monad* is a triple $\mathcal{T} = (T, \eta, \mu)$ where $T\colon \mathcal{C} \to \mathcal{C}$ is a functor, and $\eta\colon \mathsf{Id} \Rightarrow T$ and $\mu\colon TT \Rightarrow T$ are natural transformations called *unit* and *multiplication* respectively, such that the following diagrams commute:

$$\begin{array}{ccccc} T & \xRightarrow{\eta T} & TT & \xLeftarrow{T\eta} & T \\ & \searrow & \Big\Downarrow \mu & \swarrow & \\ & & T & & \end{array} \qquad \begin{array}{ccc} TTT & \xRightarrow{T\mu} & TT \\ {\scriptstyle \mu T}\Big\Downarrow & & \Big\Downarrow{\scriptstyle \mu} \\ TT & \xRightarrow[\mu]{} & T \end{array} \tag{3.9}$$

An Eilenberg-Moore algebra for $\mathcal{T}$ (or *$\mathcal{T}$-algebra*, or *algebra for the monad $\mathcal{T}$*) is a T-algebra $\alpha\colon TX \to X$ such that the following diagram commutes:

$$\begin{array}{ccccc} X & \xrightarrow{\eta X} & TX & \xleftarrow{T\alpha} & TTX \\ & \searrow & \Big\downarrow{\scriptstyle \alpha} & & \Big\downarrow{\scriptstyle \mu X} \\ & & X & \xleftarrow[\alpha]{} & TX \end{array}$$

A *$\mathcal{T}$-algebra homomorphism* is simply a T-algebra homomorphism. We denote the category of $\mathcal{T}$-algebras and their homomorphisms by $\mathcal{T}$-Alg, and the associated forgetful functor by $U\colon \mathcal{T}\text{-}\mathsf{Alg} \to \mathcal{C}$.

Throughout this thesis we often use $\mathcal{T}$ to denote a monad and T to denote a functor. Accordingly, a $\mathcal{T}$-algebra is an (Eilenberg-Moore) algebra for a *monad*, whereas a T-algebra is an algebra for a *functor*.

Given any $\mathcal{C}$-object X, the algebra (TX, μ_X) satisfies a universal property: for any $\mathcal{T}$-algebra (A, α) and any arrow $f\colon X \to A$, there is a unique algebra homomorphism $f^\sharp\colon TX \to A$ such that $f^\sharp \circ \eta_X = f$, given by $f^\sharp = \alpha \circ Tf$.

Let (T, η, μ) and (K, θ, ν) be monads. A *monad morphism* is a natural transformation $\sigma\colon T \Rightarrow K$ such that the following diagram commutes:

$$\begin{array}{ccccc} \mathsf{Id} & \xRightarrow{\eta} & T & \xLeftarrow{\mu} & TT \\ & \searrow^{\theta} & \Downarrow{\scriptstyle\sigma} & & \Downarrow{\scriptstyle\sigma\sigma} \\ & & K & \xLeftarrow[\nu]{} & KK \end{array} \tag{3.10}$$

where $\sigma\sigma = K\sigma \circ \sigma T = \sigma K \circ T\sigma$.

Example 3.4.1. We list a few examples of monads.

1. The powerset functor $\mathcal{P}$ is a monad, with unit $\eta\colon \mathsf{Id} \Rightarrow \mathcal{P}$ and multiplication $\mu\colon \mathcal{P}\mathcal{P} \Rightarrow \mathcal{P}$ given by:

   $$\eta_X(x) = \{x\} \quad \text{and} \quad \mu_X(S) = \bigcup_{U \in S} U\,.$$

   The finite powerset functor $\mathcal{P}_\omega$ extends to a monad in a similar way.

2. Given a semiring $\mathbb{S}$, the functor $\mathcal{M}$ extends to a monad, by taking

   $$\eta_X(x)(y) = \begin{cases} 1 & \text{if } x = y \\ 0 & \text{otherwise} \end{cases} \qquad \mu_X(\varphi)(x) = \sum_{\psi \in \mathbb{S}^X} \varphi(\psi) \cdot \psi(x)$$

   The case of $\mathcal{P}_\omega$ is obtained by taking the Boolean semiring. Notice that μ is well-defined since its argument φ has finite support, by definition of $\mathcal{M}$.

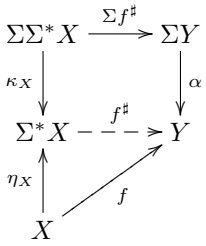
3. Suppose Σ is a polynomial functor representing a signature (3.8). Consider the functor Σ^*, which maps a set X to the set of terms over Σ with variables in X, as given by the grammar $t ::= x \mid \sigma(t_1, \ldots, t_{|\sigma|})$, where x ranges over X and σ ranges over the operator names. Given $f\colon X \to Y$, the function $\Sigma^* f\colon \Sigma^* X \to \Sigma^* Y$ is defined by substitution. The functor Σ^* extends to a monad, where the multiplication μ glues terms over terms, and the unit η interprets a variable as a term. This monad is defined properly below; it is called the *free monad* for Σ.

Let $\Sigma\colon \mathcal{C} \to \mathcal{C}$ be an arbitrary functor. A *free Σ-algebra* for a $\mathcal{C}$-object X is an initial $\Sigma + X$-algebra. The existence of free algebras for every object X amounts to the existence of a left adjoint F to the forgetful functor $U\colon \Sigma\text{-}\mathsf{alg} \to \mathcal{C}$. It is a standard fact in category theory that an adjunction yields a monad; we spell out

some of the details. Suppose a left adjoint F to U exists, let $\Sigma^* = UF \colon \mathcal{C} \to \mathcal{C}$ and let $\eta \colon \mathsf{Id} \Rightarrow \Sigma^*$ be the unit of the adjunction. The functor F induces a natural transformation $\kappa \colon \Sigma\Sigma^* \Rightarrow \Sigma^*$ such that (the copairing of)

$$\Sigma\Sigma^* X \xrightarrow{\kappa_X} \Sigma^* X \xleftarrow{\eta_X} X \tag{3.11}$$

is the free Σ-algebra for X. This means that for any Σ-algebra (Y, α) and any $f \colon X \to Y$ there exists a unique homomorphism $f^\sharp$ as in the following diagram:



Then the *free monad* for Σ is defined as (Σ^*, η, μ) where η is the unit of the adjunction, and μ is defined on a component X as the unique morphism $\mu_X \colon \Sigma^*\Sigma^* X \to \Sigma^* X$ such that $\mu_X \circ \eta_{\Sigma^* X} = \mathsf{id}$.

Example 3.4.2. Suppose Σ^* is the (underlying functor of the) free monad for Σ arising from a signature, as described in Example 3.4.1 (3). The fact that $\Sigma^* X$ is a free Σ-algebra amounts to the following: given any Σ-algebra A, there is a one-to-one correspondence between maps $f \colon X \to A$ and algebra homomorphisms $f^\sharp \colon \Sigma^* X \to A$. Here f can be viewed as a variable assignment, and $f^\sharp$ as its inductive extension to terms.

Suppose (Σ^*, η, μ) is the free monad for a functor Σ. Any Σ-algebra $\alpha \colon \Sigma X \to X$ then yields an Eilenberg-Moore algebra $\widehat{\alpha} \colon \Sigma^* X \to X$, defined by the unique extension of id_X to an algebra morphism from $\Sigma^* X$ to X. In fact, this construction yields an isomorphism between the category Σ-alg of algebras for the functor Σ and the category Σ^*-Alg of algebras for the free monad Σ^*.

3.5 Bialgebras and distributive laws

Bialgebras consist of an algebra and a coalgebra structure over a common carrier. The interaction between algebra and coalgebra can be captured by *distributive laws*. These provide enough structure to study operational semantics, determinization and recursive equations in a systematic manner; see [TP97, Bar04, Kli11, JSS12] for more information.

Let $T, B \colon \mathcal{C} \to \mathcal{C}$ be functors. A *distributive law* of T over B is a natural transformation $\lambda \colon TB \Rightarrow BT$. This is the simplest type of distributive law, and we sometimes refer to it as a distributive law between functors. Given such a λ, a

λ-*bialgebra* is a triple (X, α, δ) so that $\alpha\colon TX \to X$ is a T-algebra, $\delta\colon X \to BX$ is a B-coalgebra and the following diagram commutes:

$$\begin{array}{ccccc} TX & \xrightarrow{\alpha} & X & \xrightarrow{\delta} & BX \\ {\scriptstyle T\delta}\downarrow & & & & \uparrow{\scriptstyle B\alpha} \\ TBX & & \xrightarrow[\lambda_X]{} & & BTX \end{array} \tag{3.12}$$

A λ-bialgebra homomorphism is a map that is both an algebra and a coalgebra homomorphism. Any distributive law defines *liftings* of T and B:

$$\begin{array}{ccc} B\text{-coalg} & \xrightarrow{\overline{T}} & B\text{-coalg} \\ \downarrow & & \downarrow \\ \mathcal{C} & \xrightarrow[T]{} & \mathcal{C} \end{array} \qquad \begin{array}{ccc} T\text{-alg} & \xrightarrow{\overline{B}} & T\text{-alg} \\ \downarrow & & \downarrow \\ \mathcal{C} & \xrightarrow[B]{} & \mathcal{C} \end{array}$$

defined on objects by

$$\overline{T}(X, \delta) = (TX, \lambda_X \circ T\delta) \qquad \overline{B}(X, \alpha) = (BX, B\alpha \circ \lambda_X)$$

Notice that (3.12) commutes iff δ is a $\overline{B}$-coalgebra with carrier (X, α) iff α is a $\overline{T}$-algebra with carrier (X, δ). Indeed, the category of λ-bialgebras is isomorphic to the category of $\overline{B}$-coalgebras and the category of $\overline{T}$-algebras.

If B has a final coalgebra (Z, ζ), we can use $\overline{T}$ and coinduction to construct a bialgebra on Z:

$$\begin{array}{ccc} TZ & \overset{\alpha}{\dashrightarrow} & Z \\ {\scriptstyle T\zeta}\downarrow & & \downarrow{\scriptstyle \zeta} \\ TBZ & & \\ {\scriptstyle \lambda_Z}\downarrow & & \\ BTZ & \xrightarrow[B\alpha]{} & BZ \end{array}$$

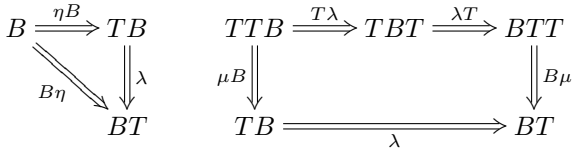
This bialgebra is *final* in the category of λ-bialgebras.

Lemma 3.5.1. *Let $\lambda\colon TB \Rightarrow BT$ be a distributive law (between functors). The final coalgebra (Z, ζ) lifts to a final λ-bialgebra.*

Similarly, if T has an initial algebra (A, κ) then we can lift it to an initial λ-bialgebra, see, e.g., [Kli11] for details. Instead of spelling this out, we will consider initial bialgebras and their properties for a more general type of distributive law in the next subsection.

3.5.1 Distributive laws of monads over (copointed) functors

Let $\mathcal{T} = (T, \eta, \mu)$ be a monad. A distributive law of $\mathcal{T}$ over B is a natural transformation $\lambda\colon TB \Rightarrow BT$ such that the following diagrams commute:

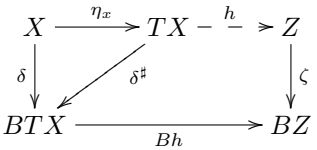


A distributive law λ as above induces a lifting $\overline{B}\colon \mathcal{T}\text{-Alg} \to \mathcal{T}\text{-Alg}$ of B to the category of Eilenberg-Moore algebras for the monad $\mathcal{T}$. In fact, there is a one-to-one correspondence between distributive laws λ as above and liftings of B to $\mathcal{T}$-Alg [Joh75, TP97].

Suppose λ is a distributive law of $\mathcal{T}$ over B. Any coalgebra $\delta\colon X \to BTX$ can then be extended to a homomorphism $\delta^\sharp\colon (TX, \mu_X) \to \overline{B}(TX, \mu_X)$ so that $\delta^\sharp \circ \eta_X = \delta$. Notice that $\delta^\sharp$ is defined by

$$TX \xrightarrow{T\delta} TBTX \xrightarrow{\lambda_{TX}} BTTX \xrightarrow{B\mu_X} BTX \tag{3.13}$$

This yields another lifting $\widehat{T}\colon TB\text{-coalg} \to B\text{-coalg}$ of T. Now, consider the coinductive extension below:



(3.14)

Since the final B-coalgebra lifts to a final $\overline{B}$-coalgebra (similar to Lemma 3.5.1), the coinductive extension h is an algebra homomorphism. This can be interpreted as stating that the semantics is compositional, in the sense that behavioural equivalence on $\delta^\sharp$ is a congruence.

The above use of distributive laws to turn TB-coalgebras into $\overline{B}$-coalgebras (and obtaining a semantics from the coinductive extension) is sometimes interpreted as a general way of solving corecursive equations (e.g., [Bar04, Jac06b]); it is also called the *generalized powerset construction* [SBBR13, JSS12].

Example 3.5.2 ([SBBR13, JSS12])**.** In Section 3.1.1 we have seen informally how to determinize non-deterministic automata. This construction arises from a distributive law $\lambda\colon \mathcal{P}_\omega(2 \times \mathsf{Id}^A) \Rightarrow 2 \times (\mathcal{P}-)^A$ of the powerset monad over the functor $2 \times \mathsf{Id}^A$, given by

$$\lambda_X(S) = \left(\bigvee_{(o,t)\in S} o, \lambda a. \bigcup_{(o,t)\in S} t(a) \right).$$

Spelling out the details of the construction in Equation 3.13 yields the classical powerset construction, as in Section 3.1.1. The composition $h \circ \eta_X$ in (3.14) is the usual language semantics of non-deterministic automata (obtained via determinization).

Similarly, the determinization of weighted automata arises from a distributive law $\lambda\colon \mathcal{M}B \Rightarrow B\mathcal{M}$ where $BX = \mathbb{S} \times X^A$ and λ is defined by

$$\lambda_X \left(\sum r_i(o_i, t_i) \right) = \left(\sum r_i \cdot o_i, \lambda a. \sum r_i \cdot t_i(a) \right) .$$

The composition $h \circ \eta_X$ in (3.14) maps a state to the weighted language that it accepts, see [JSS12, BBB$^+$12].

There is yet another type of distributive law, which is particularly suitable for operational semantics, as we will see below. To define it we need the notion of a *copointed* functor, which is a pair (B, ϵ) where $B\colon \mathcal{C} \to \mathcal{C}$ is an endofunctor and $\epsilon\colon B \Rightarrow \mathsf{Id}$ a natural transformation. A coalgebra for a copointed functor (B, ϵ) is a B-coalgebra (X, δ) such that $\epsilon_X \circ \delta = \mathsf{id}$. We will frequently consider copointed functors $(B \times \mathsf{Id}, \pi_2)$; such a functor is called the *cofree* copointed functor for B. It is easy to see that B-coalgebras are in one-to-one correspondence to coalgebras for $(B \times \mathsf{Id}, \pi_2)$. Now, a distributive law of a monad (T, η, μ) over a copointed functor (B, ϵ) is a distributive law λ of (T, η, μ) over B such that, additionally, the axiom

$$\begin{array}{ccc} TB & & \\ {\scriptstyle\lambda}\Downarrow & \searrow^{T\epsilon} & \\ BT & \xRightarrow[\epsilon T]{} & T \end{array}$$

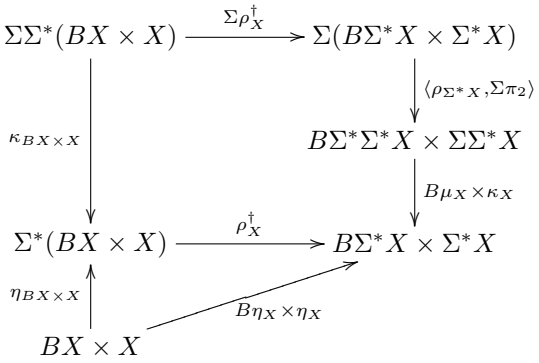
is satisfied. (We note that this can be further generalized by considering distributive laws of monads over comonads; for a formal definition see, e.g., [Kli11].)

3.5.2 Abstract GSOS

In this section we consider *abstract GSOS*, which provides specification formats for defining operations on coalgebras, and allows to study operational semantics in a general fashion. It is a generalization of *GSOS*, which is a syntactic format for transition system specifications (see Example 3.5.4 below). An abstract GSOS specification of Σ over B is a natural transformation $\rho\colon \Sigma(B \times \mathsf{Id}) \Rightarrow B\Sigma^*$. The following result [TP97] states that distributive laws of monad over copointed functor can be presented by abstract GSOS specifications.

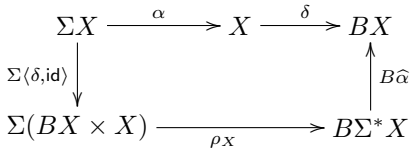
Lemma 3.5.3. *There is a one-to-one correspondence between abstract GSOS specifications ρ of Σ over B and distributive laws $\rho^\dagger$ of the free monad Σ^* over the cofree copointed functor $B \times \mathsf{Id}$.*

For a full proof, see [LPW04, Bar04]. Given ρ, $\rho^\dagger$ is defined on a component X using initiality:



(3.15)

A *model* of ρ is a triple (X, α, δ) where $\alpha\colon \Sigma X \to X$ is a Σ-algebra and $\delta\colon X \to BX$ a B-coalgebra, such that the diagram



commutes. There is a one-to-one correspondence between models for ρ and $\rho^\dagger$-bialgebras; more precisely, a triple (X, α, δ) is a model of ρ iff $(X, \widehat{\alpha}, \langle \delta, \mathsf{id} \rangle)$ is a $\rho^\dagger$-bialgebra. Based on this correspondence, it is easy to establish that behavioural equivalence on (the coalgebra part of) any ρ-model is a congruence. The ρ-model corresponding to the initial $\rho^\dagger$-bialgebra is sometimes referred to as the *operational model* of ρ. We consider a few examples of abstract GSOS for particular choices of the behaviour functor B; for many other instances of abstract GSOS, see (the references in) [Kli11].

Example 3.5.4. Abstract GSOS is a generalization of *GSOS*, a format for transition system specifications introduced in [BIM95]. Given a signature, a *GSOS* rule for an operator σ of arity n is of the form

$$\frac{\{x_{i_j} \xrightarrow{a_j} y_j\}_{j=1..m} \qquad \{x_{i_k} \overset{b_k}{\not\to}\}_{k=1..l}}{\sigma(x_1, \ldots, x_n) \xrightarrow{c} t} \tag{3.16}$$

where m is the number of positive premises, l is the number of negative premises, and $a_1, \ldots, a_m, b_1, \ldots, b_l, c \in A$ are labels. The variables $x_1, \ldots, x_n$, $y_1, \ldots, y_m$ are pairwise distinct, and t is a term over these variables.

GSOS rules for a signature Σ induce abstract GSOS specifications of Σ over the functor $BX = (\mathcal{P}_\omega X)^A$ of labelled transition systems. Conversely, every GSOS

specification arises from an abstract GSOS specification. This correspondence was first observed in [TP97], and proved in detail in [Bar04]; see also [Kli11] for a detailed explanation. The unique ρ-model on the initial algebra corresponds to the supported model of a GSOS specification. The well-known fact that bisimilarity on the supported model of a GSOS specification is a congruence, thus follows from the abstract underlying theory of distributive laws.

A simple example of a GSOS specification is given by the parallel composition operation. Let $A = N \cup \overline{N} \cup \{\tau\}$ where N is a set of labels and $\overline{N} = \{\overline{a} \mid a \in N\}$; we let $\overline{\overline{a}} = a$. The parallel composition is then defined by the following rules:

$$\frac{x \xrightarrow{a} x'}{x|y \xrightarrow{a} x'|y} \qquad \frac{y \xrightarrow{a} y'}{x|y \xrightarrow{a} x|y'} \qquad \frac{x \xrightarrow{a} x' \quad y \xrightarrow{\overline{a}} y'}{x|y \xrightarrow{\tau} x'|y'}$$

We define this as an abstract GSOS specification $\rho\colon \Sigma((\mathcal{P}_\omega -)^A \times \mathsf{Id}) \Rightarrow \mathcal{P}_\omega(\Sigma -)^A$, where $\Sigma X = X \times X$ (we model a binary operator). Then, on a component X, $\rho_X\colon ((\mathcal{P}_\omega X)^A \times X) \times ((\mathcal{P}_\omega X)^A \times X) \to (\mathcal{P}_\omega(\Sigma^* X))^A$ is given by

$$\begin{aligned}\rho(f,x,g,y)(\tau) =& \{(x'|y) \mid x' \in f(\tau)\} \cup \{(x|y') \mid y' \in g(\tau)\} \\ & \cup \{(x'|y') \mid x' \in f(a) \text{ and } y' \in g(\overline{a})\}\end{aligned}$$

and for any $a \in A$ with $a \neq \tau$:

$$\rho(f,x,g,y)(a) = \{(x'|y) \mid x' \in f(a)\} \cup \{(x|y') \mid y' \in g(a)\}\,.$$

Notice that the carrier of the operational model of ρ is empty; this is because we did not add any constants to the signature and the specification. If we do, then the operational model will be a transition system whose states are the terms built from these constants and the parallel operator, and where the behaviour of a term $p|q$ is dictated by the GSOS rules above.

Example 3.5.5. Behavioural differential equations for streams, such as those defined in Section 3.1.1, can be presented by abstract GSOS specifications of Σ over $BX = A \times X$, where Σ is the signature functor representing the syntax. The precise format and definitions are well explained in [HKR14]. For example, to define the operations of sum, convolution product and the constants $[r]$ we take $\Sigma X = X \times X + X \times X + \mathbb{R}$ and define $\rho\colon \Sigma((\mathbb{R} \times \mathsf{Id}) \times \mathsf{Id}) \Rightarrow \mathbb{R} \times \Sigma^*$ by cases:

$$\begin{aligned}\rho_X^{[r]} &= (r, [0]) \\ \rho_X^{+}((a,x',x),(b,y',y)) &= (a+b, x'+y') \\ \rho_X^{\times}((a,x',x),(b,y',y)) &= (a \cdot b, (x' \times y) + (a \times y'))\end{aligned}$$

The shuffle product is also easily defined this way. The shuffle inverse is a bit more problematic, since it is not always defined. One ad-hoc way of solving this is by just assigning it some fixed constant value in those cases.

The operational model of ρ then consists of the closed terms over Σ, and its coalgebra structure is defined by induction according to ρ. The coinductive extension yields the semantics, and this is compositional with respect to the algebraic structure induced on the final coalgebra.

Similarly, the format of behavioural differential equations for deterministic automata presented in Definition 2.3.3 induces GSOS specifications for the functor $BX = 2 \times X^A$. We did not formally prove the converse, i.e., that every such specification arises from an abstract GSOS specification, although this seems rather likely. The format of behavioural differential equations in Definition 2.3.3 thus consititutes a concrete, syntactic presentation of GSOS specifications for deterministic automata.

The GSOS format for streams and the one for deterministic automata give rise to specific types of behavioural differential equations. BDEs can be defined more generally, for instance involving second derivative, which can not be expressed in these formats. The advantage of (abstract) GSOS is that it is quite expressive and covers most examples encountered in the literature, while these specifications are still well-behaved: they give rise to a compositional semantics, and have distributive laws and bialgebras as a solid underlying mathematical theory.