3
Bisimulations

The operation c of a coalgebra $c\colon X \to F(X)$ gives us information about its states in X. It may allow us to observe certain things, and it may allow us to 'modify states' or to 'move to successor states'. Typically for coalgebras, we can observe and modify, but we have no means for constructing new states. The behaviour of a state x is all that we can observe about x, either directly or indirectly (via its successor states). This behaviour is written as $\mathrm{beh}(x)$, where beh is the unique map to the final coalgebra (if any), as introduced in Definition 2.3.1.

In this situation it may happen that two states have the same behaviour. In that case we cannot distinguish them with the operations (of the coalgebras) that we have at our disposal. The two states need not be equal then, since the operations may give only limited access to the state space, and certain aspects may not be observable. When two states x, y are observationally indistinguishable, they are called *bisimilar*. This is written as $x \leftrightarroweq y$.

The bisimilarity relation, for a given coalgebra (or pair of coalgebras), is introduced as the union of all bisimulations. A bisimulation is a relation on state spaces, which is maintained by coalgebra transitions and leads to equal observations. Bisimulations were first introduced by Park [371] for automata, as mutual simulations – building on an earlier notion of simulation between programs [353]. Park proved that if the initial states of two deterministic automata are related by a bisimulation, then they accept the same sets of inputs (see also Corollary 3.4.4 below). Indeed, bisimulations form a crucial tool for stepwise reasoning – as in induction arguments.

Bisimilarity is the main topic of the present chapter, but only for coalgebras of Kripke polynomial functors. In the next chapter more general functors will be considered, but for reasons of simplicity we prefer to first study the concrete situation for polynomial functors. Bisimulation will be introduced – like congruence – via a technique called relation lifting. These liftings can

be defined by induction on the structure of polynomial functors. In a related manner the notion of invariance will arise in the Chapter 6 via predicate lifting. The first part of this chapter concentrates on some basic properties of relation lifting and of bisimulation relations; these properties will be used frequently. The coinduction proof principle in Section 3.4 is a basic result in the theory of coalgebras. It says that two states have the same behaviour if and only if they are contained in a bisimulation relation. Coinduction via bisimulations is illustrated in a simple process calculus in Section 3.5.

3.1 Relation Lifting, Bisimulations and Congruences

This section will introduce the technique of relation lifting from [217, 218] and use it to define the notions of bisimulation for coalgebras, and congruence for algebras. Many elementary results about relation lifting are provided. Alternative ways for introducing bisimulations will be discussed later in Section 3.3.

We start by motivating the need for relation lifting. Consider a sequence coalgebra $c\colon X \to 1 + (A \times X)$, as in Section 1.2. A bisimulation for this coalgebra is a relation $R \subseteq X \times X$ on its state space which is 'closed under c'. What this means is that if R holds for states x, y, then either both x and y have no successor states (i.e. $c(x) = \kappa_1(*) = c(y)$), or they both have successor states which are again related by R and their observations are the same: $c(x) = \kappa_2(a, x')$, $c(y) = \kappa_2(b, y')$, with $a = b$ and $R(x', y')$.

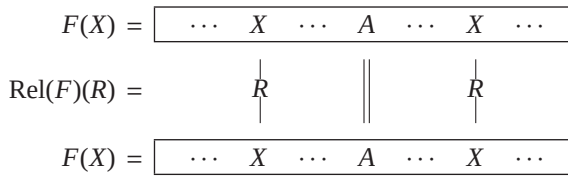
One way to express such closure properties uniformly is via a 'lifting' of R from a relation on X to a relation R' on $1 + (A \times X)$ so that this closure can be expressed simply as

$$R(x, y) \implies R'(c(x), c(y)).$$

This idea works if we take $R' \subseteq (1 + (A \times X)) \times (1 + (A \times X))$ to be

$$R' = \{\kappa_1(*), \kappa_1(*)\} \cup \{(\kappa_2(a, x), \kappa_2(b, y)) \mid a = b \wedge R(x, y)\}.$$

The general idea of relation lifting applies to a polynomial functor F. It is a transformation of a relation $R \subseteq X \times X$ to a relation $R' \subseteq F(X) \times F(X)$, which will be defined by induction on the structure of F. We shall use the notation $\mathrm{Rel}(F)(R)$ for R' above. Briefly, the lifted relation $\mathrm{Rel}(F)(R)$ uses equality on occurrences of constants A in F, and R on occurrences of the state space X, as suggested in



Actually, it will be convenient to define relation lifting slightly more generally and to allow different state spaces. Thus, it applies to relations $R \subseteq X \times Y$ and yields a relation $\mathrm{Rel}(F)(R) \subseteq F(X) \times F(Y)$.

Definition 3.1.1 (Relation lifting) Let $F\colon \mathbf{Sets} \to \mathbf{Sets}$ be a polynomial functor, and let X, Y be arbitrary sets. The mapping $\mathrm{Rel}(F)$ which sends a relation $R \subseteq X \times Y$ to a 'lifted' relation $\mathrm{Rel}(F)(R) \subseteq F(X) \times F(Y)$ is defined by induction on the structure of the functor F, following the points in Definition 2.2.1.

1. If F is the identity functor, then
$$\mathrm{Rel}(F)(R) \;=\; R.$$
2. If F is a constant functor $X \mapsto A$, then
$$\mathrm{Rel}(F)(R) \;=\; \mathrm{Eq}(A) \;=\; \{(a, a) \mid a \in A\}.$$
3. If F is a product $F_1 \times F_2$, then
$$\mathrm{Rel}(F)(R) \;=\; \{\, ((u_1, u_2), (v_1, v_2)) \mid \mathrm{Rel}(F_1)(R)(u_1, v_1) \wedge \mathrm{Rel}(F_2)(R)(u_2, v_2) \,\}.$$
4. If F is a set-indexed coproduct $\coprod_{i \in I} F_i$, then:
$$\mathrm{Rel}(F)(R) \;=\; \bigcup\nolimits_{i \in I} \{(\kappa_i(u), \kappa_i(v)) \mid \mathrm{Rel}(F_i)(R)(u, v)\}.$$
5. If F is an exponent G^A, then
$$\mathrm{Rel}(F)(R) \;=\; \{(\, f, g) \mid \forall a \in A.\, \mathrm{Rel}(G)(R)(\, f(a), g(a))\}.$$
6. If F is a powerset $\mathcal{P}(G)$, then
$$\mathrm{Rel}(F)(R) \;=\; \{\, (U, V) \mid \forall u \in U.\, \exists v \in V.\, \mathrm{Rel}(G)(R)(u, v) \wedge \forall v \in V.\, \exists u \in U.\, \mathrm{Rel}(G)(R)(u, v) \,\}.$$

This same formula will be used in case F is a *finite* powerset $\mathcal{P}_{\mathrm{fin}}(G)$.

In the beginning of Section 3.3 we shall see that relation lifting can also be defined directly via images. The above inductive definition may seem more cumbersome but gives us a better handle on the different cases. Also, it better emphasises the relational aspects of lifting and the underlying logical

infrastructure (such as finite conjunctions and disjunctions, and universal and existential quantification). This is especially relevant in more general settings, such as in [218, 45, 150].

Relation lifting with respect to a functor is closely related to so-called logical relations. These are collections of relations $(R_\sigma)_\sigma$ indexed by types σ, in such a way that $R_{\sigma\to\tau}$ and $R_{\sigma\times\tau}$ are determined by R_σ and R_τ. Similarly, we use collections of relations $(\mathrm{Rel}(F)(R))_F$ indexed by polynomial functors F, which are also structurally determined. Logical relations were originally introduced in the context of semantics of (simply) typed lambda calculus ([442, 147, 382]); see [356, chapter 8] for an overview. They are used for instance for definability, observational equivalence and data refinement.

In the next section we shall see various elementary properties of relation lifting. But first we show what it is used for: bisimulation for coalgebras and congruence for algebras. The definitions we use are *generic* or *polytypic*, in the sense that they apply uniformly to (co)algebras of an arbitrary polynomial functor.

Definition 3.1.2 Let $F\colon \mathbf{Sets} \to \mathbf{Sets}$ be a polynomial functor.

1. A **bisimulation** for coalgebras $c\colon X \to F(X)$ and $d\colon Y \to F(Y)$ is a relation $R \subseteq X \times Y$ which is 'closed under c and d':

$$(x, y) \in R \implies (c(x), d(y)) \in \mathrm{Rel}(F)(R)$$

   for all $x \in X$ and $y \in Y$. Equivalently:

$$R \subseteq (c \times d)^{-1}(\mathrm{Rel}(F)(R))$$

   or, by (2.15),

$$\coprod_{c\times d}(R) \subseteq \mathrm{Rel}(F)(R).$$

2. A **congruence** for algebras $a\colon F(X) \to X$ and $b\colon F(Y) \to Y$ is a relation $R \subseteq X \times Y$ which is also 'closed under a and b':

$$(u, v) \in \mathrm{Rel}(F)(R) \implies (a(u), b(v)) \in R.$$

   That is,

$$\mathrm{Rel}(F)(R) \subseteq (a \times b)^{-1}(R) \quad \text{or equivalently} \quad \coprod_{a\times b}(\mathrm{Rel}(F)(R)) \subseteq R.$$

Often we are interested in bisimulations $R \subseteq X \times X$ on a *single* coalgebra $c\colon X \to F(X)$. We then use the definition with $d = c$. Similarly for congruences.

Notice that we require only that a congruence is closed under the (algebraic) operations and not that it is an equivalence relation. This minor deviation from

standard terminology is justified by the duality we obtain between bisimulations and congruences. We shall use the following explicit terminology.

Definition 3.1.3 A **bisimulation equivalence** is a bisimulation on a single coalgebra which is an equivalence relation. Similarly, a **congruence equivalence** is a congruence on a single algebra which is an equivalence relation.

We continue with several examples of the notions of bisimulation and congruence for specific functors.

Bisimulations for Deterministic Automata

Consider a deterministic automaton $\langle \delta, \epsilon \rangle \colon X \to X^A \times B$. As we have seen in Section 2.2, it is a coalgebra for the functor $F = \mathrm{id}^A \times B$. Relation lifting for this functor yields for a relation $R \subseteq X \times X$ a new relation $\mathrm{Rel}(F)(R) \subseteq (X^A \times B) \times (X^A \times B)$, given by

$$\mathrm{Rel}(F)(R)\big((f_1, b_1), (f_2, b_2)\big) \iff \forall a \in A.\, R(f_1(a), f_2(a)) \wedge b_1 = b_2.$$

Thus, a relation $R \subseteq X \times X$ is a bisimulation with respect to the (single) coalgebra $\langle \delta, \epsilon \rangle \colon X \to X^A \times B$ if, for all $x, y \in X$,

$$R(x,y) \implies \mathrm{Rel}(F)(R)\big((\delta(x), \epsilon(x)), (\delta(y), \epsilon(y))\big).$$

That is,

$$R(x,y) \implies \forall a \in A.\, R(\delta(x)(a), \delta(y)(a)) \wedge \epsilon(x) = \epsilon(y).$$

That is, in transition notation:

$$R(x,y) \implies \begin{cases} x \xrightarrow{a} x' \wedge y \xrightarrow{a} y' \text{ implies } R(x', y') \\ x \downarrow b \wedge y \downarrow c \text{ implies } b = c. \end{cases}$$

Thus, once two states are in a bisimulation R, they remain in R and give rise to the same direct observations. This makes them observationally indistinguishable.

Bisimulations for Non-deterministic Automata

Next, consider a non-deterministic automaton $\langle \delta, \epsilon \rangle \colon X \to \mathcal{P}(X)^A \times B$, as coalgebra for the functor $F = \mathcal{P}(\mathrm{id})^A \times B$. Relation lifting for this functor is slightly more complicated: it sends a relation $R \subseteq X \times X$ to the relation $\mathrm{Rel}(F)(R) \subseteq (\mathcal{P}(X)^A \times B) \times (\mathcal{P}(X)^A \times B)$ given by

$$\begin{aligned} \mathrm{Rel}(F)(R)((f_1, b_1), (f_2, b_2)) \iff \forall a \in A.\ & \forall x \in f_1(a).\, \exists y \in f_2(a).\, R(x,y) \\ & \wedge \forall y \in f_2(a).\, \exists x \in f_1(a).\, R(x,y) \\ & \wedge b_1 = b_2. \end{aligned}$$

Thus, $R \subseteq X \times X$ is a bisimulation if for all $x, y \in X$ with $R(x, y)$,

- $x \xrightarrow{a} x'$ implies there is a y' with $y \xrightarrow{a} y'$ and $R(x', y')$;
- $y \xrightarrow{a} y'$ implies there is an x' with $x \xrightarrow{a} x'$ and $R(x', y')$;
- $x \downarrow b$ and $y \downarrow c$ implies $b = c$.

This corresponds to the standard notion of bisimulation used in the theory of automata/transition systems.

Congruences for Monoids

Recall that a monoid is a set M carrying an associative operation $+ \colon M \times M \to M$ with a unit element $0 \in M$. These two operations $+$ and 0, but not the relevant monoid equations, can be captured as an algebra:

$$1 + (M \times M) \xrightarrow{[0,+]} M$$

of the functor $F(X) = 1 + (X \times X)$. Relation lifting for F is described by

$$\mathrm{Rel}(F)(R) = \{(\kappa_1(*), \kappa_1(*))\} \cup \{(\kappa_2(x, x'), \kappa_2(y, y')) \mid R(x, y) \wedge R(x', y')\}.$$

Hence a relation $R \subseteq M \times M$ on the carrier of the monoid is a congruence if:

$$\mathrm{Rel}(F)(R)(u, v) \implies R([0, +](u), [0, +](v)).$$

This amounts to

$$R(0, 0) \qquad \text{and} \qquad R(x, y) \wedge R(x', y') \implies R(x + x', y + y').$$

Thus a congruence, like a bisimulation, is closed under the operations.

Congruences in a Binary Induction Proof Principle

We have already discussed the usual 'unary' induction proof principle for natural numbers in Example 2.4.4, expressed in terms of predicates, which are assumed to be closed under the operations. Later, in Section 6.1 we shall encounter it in full generality, stating that every invariant on an initial algebra is the truth predicate.

There is also a less well-known binary version of the induction proof principle, expressed in terms of congruences. It was first formulated as such for the natural numbers in [414] and further generalised in [218]. It also appeared in the derivations of induction and coinduction principles in [385] in the context of a formal logic for parametric polymorphism.

At this stage we only formulate this binary version, and postpone the proof. It can be given in various ways (see Exercises 3.3.2 and 6.2.2) but requires some properties of relation lifting which are still to come.

Theorem 3.1.4 (Binary induction proof principle) *Every congruence on an initial algebra contains the equality relation.*

This binary version of induction is the dual of a coinduction principle; see Corollary 3.4.2.

Bisimulations as Congruences

The so-called structural operational semantics (SOS) introduced by Plotkin is a standard technique in the semantics of programming languages to define the operational behaviour of programs. The latter are seen as elements of the initial algebra $F(\mathsf{Prog}) \xrightarrow{\cong} \mathsf{Prog}$ of a suitable functor F describing the signature of operations of the programming language, as in Example 2.4.2. A transition relation is then defined on top of the set of programs Prog, as the least relation closed under certain rules. This transition structure may be understood as coalgebra $\mathsf{Prog} \to G(\mathsf{Prog})$, for an appropriate functor G – which is often the functor $\mathcal{P}(\mathrm{id})^A$ for labelled transition systems (see [452, 451]); the transition structure is then given by transitions $p \xrightarrow{a} q$ describing an a-step between programs $p, q \in \mathsf{Prog}$.

The transition structure gives rise to certain equivalences for programs, such as bisimilarity (see below), trace equivalence or other equivalences; see [158]. These equivalences are typically bisimulation equivalences. An important issue in this setting is: are these bisimulation equivalences also *congruences* for the given algebra structure $F(\mathsf{Prog}) \xrightarrow{\cong} \mathsf{Prog}$? This is a basic requirement to make the equivalence a reasonable one for the kind of programs under consideration, because congruence properties are essential in reasoning with the equivalence. In this setting given by a bialgebra $F(\mathsf{Prog}) \to \mathsf{Prog} \to G(\mathsf{Prog})$, the two fundamental notions of this section (bisimulation and congruence) are thus intimately related. This situation will be investigated further in Section 5.5 in relation to distributive laws.

There is a whole line of research about suitable syntactic formats for SOS rules, ensuring that certain bisimulation equivalences are also congruences. See [175] for a basic reference. We shall use a more categorical perspective, first in Section 3.5, and later in Section 5.5, following [452, 451, 58]; see [298] for an overview of the coalgebraic approach.

Definition 3.1.5 Let $X \xrightarrow{c} F(X)$ and $Y \xrightarrow{d} F(Y)$ be two coalgebras of a polynomial functor F. The **bisimilarity** relation $\leftrightarrows$ is the union of all bisimulations:

$$x \leftrightarrows y \iff \exists R \subseteq X \times Y.\, R \text{ is a bisimulation for } c \text{ and } d, \text{ and } R(x, y).$$

As a result of Proposition 3.2.6.3 in the next section, this union is a bisimulation itself, so that $\leftrightarrows$ can be characterised as the greatest bisimulation.

Sometimes we write ${}_c\leftrightarrows_d$ for $\leftrightarrows$ to make the dependence on the coalgebras c and d explicit.

Bisimilarity formalises the idea of observational indistinguishability. It will be an important topic in the remainder of this chapter.

Exercises

3.1.1 Use the description (2.17) of a list functor $F^\star$ to show that:

$$\mathrm{Rel}(F^\star)(R) = \{(\langle u_1, \ldots, u_n\rangle, \langle v_1, \ldots, v_n\rangle) \mid \forall i \leq n.\, \mathrm{Rel}(F)(R)(u_i, v_i)\}.$$

3.1.2 Unfold the definition of bisimulation for various kind of tree coalgebras, like $X \to 1 + (A \times X \times X)$ and $X \to (A \times X)^\star$.

3.1.3 Do the same for classes in object-oriented languages (see (1.10)), described as coalgebras of a functor in Exercise 2.3.6.3.

3.1.4 Note that the operations of a vector space V (over $\mathbb{R}$), namely zero, addition, inverse and scalar multiplication, can be captured as an algebra $1 + (V \times V) + V + (\mathbb{R} \times V) \longrightarrow V$. Investigate then what the associated notion of congruence is.

3.1.5 We have described relation lifting on a coproduct functor $F = F_1 + F_2$ in Definition 3.1.1 as

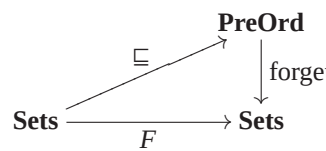
$$\mathrm{Rel}(F_1 + F_2)(R) = \coprod_{\kappa_1 \times \kappa_1}(\mathrm{Rel}(F_1)(R)) \cup \coprod_{\kappa_2 \times \kappa_2}(\mathrm{Rel}(F_2)(R)).$$

Prove that it can also be defined in terms of products $\prod$ and intersection $\cap$ as

$$\mathrm{Rel}(F_1 + F_2)(R) = \prod_{\kappa_1 \times \kappa_1}(\mathrm{Rel}(F_1)(R)) \cap \prod_{\kappa_2 \times \kappa_2}(\mathrm{Rel}(F_2)(R)),$$

where for a function $f\colon X \to Y$ the map $\prod_f\colon \mathcal{P}(X) \to \mathcal{P}(Y)$ is described in Exercise 2.1.12.

3.1.6 In this text we concentrate on *bi*simulations. There is also the notion of simulation, which can be defined via an order on a functor; see [446, 232]. For a functor $F\colon \mathbf{Sets} \to \mathbf{Sets}$ such an order consists of a functor $\sqsubseteq$ as in the diagram below.



Given such an order we define 'lax' relation lifting $\mathrm{Rel}_{\sqsubseteq}(F)$ as $R \mapsto \sqsubseteq \circ \mathrm{Rel}(F)(R) \circ \sqsubseteq$. A relation $R \subseteq X \times Y$ is then a **simulation** on coalgebras $X \xrightarrow{c} F(X)$, $Y \xrightarrow{d} F(Y)$ if $R \subseteq (c \times d)^{-1}(\mathrm{Rel}_{\sqsubseteq}(F)(R))$. Similarity is then the union of all simulations.

1. Investigate what it means to have an order as described in the above diagram.
2. Describe on the functor $L = 1 + (A \times (-))$ a 'flat' order, and on the powerset functor $\mathcal{P}$ the inclusion order, as in the diagram. Check what the associated notions of simulation are.
3. Prove that similarity on the final coalgebra A^∞ of the functor L with order as in (2) is the prefix order given by $\sigma \leq \tau$ iff $\sigma \cdot \rho = \tau$ for some $\rho \in A^\infty$; see [232, example 5.7].

3.2 Properties of Bisimulations

This section is slightly technical, and possibly also slightly boring. It starts by listing various elementary properties of relation lifting, and subsequently uses these properties to prove standard results about bisimulations and bisimilarity.

First there are three lemmas about relation lifting.

Lemma 3.2.1 *Let* $F\colon \mathbf{Sets} \to \mathbf{Sets}$ *be a polynomial functor. Relation lifting* $\mathrm{Rel}(F)$ *with respect to* F *satisfies the following basic properties.*

1. *It preserves the equality relation:*
$$\mathrm{Rel}(F)(\mathrm{Eq}(X)) = \mathrm{Eq}(F(X)).$$
2. *It commutes with reversal of relations:*
$$\mathrm{Rel}(F)(R^\dagger) = \mathrm{Rel}(F)(R)^\dagger.$$
3. *It is monotone:*
$$R \subseteq S \implies \mathrm{Rel}(F)(R) \subseteq \mathrm{Rel}(F)(S).$$
4. *It preserves relation composition*
$$\mathrm{Rel}(F)(R \circ S) = \mathrm{Rel}(F)(R) \circ \mathrm{Rel}(F)(S).$$
5. *It preserves reflexivity, symmetry and transitivity, and thus, if* R *is an equivalence relation, then so is* $\mathrm{Rel}(F)(R)$.

Proof The first four statements (1)–(4) are proved by induction on the structure of F, following the cases in Definition 3.1.1. The case in (4) where

F is an exponent G^A requires the axiom of choice (AC), as will be illustrated: assume, as induction hypothesis (IH), that the functor G preserves composition of relations, then so does the exponent G^A, since:

$$\begin{aligned}
&\mathrm{Rel}(G^A)(R \circ S)(f, g)\\
&\underset{\text{(IH)}}{\iff} \forall a \in A.\, \mathrm{Rel}(G)(R \circ S)(f(a), g(a))\\
&\iff \forall a \in A.\, (\mathrm{Rel}(G)(R) \circ \mathrm{Rel}(G)(S))(f(a), g(a))\\
&\underset{\text{(AC)}}{\iff} \forall a \in A.\, \exists z.\, \mathrm{Rel}(G)(R)(f(a), z) \wedge \mathrm{Rel}(G)(S)(z, g(a))\\
&\iff \exists h.\, \forall a \in \mathrm{Rel}(G)(R)(f(a), h(a)) \wedge \mathrm{Rel}(G)(S)(h(a), g(a))\\
&\iff \exists h.\, \mathrm{Rel}(G^A)(R)(f, h) \wedge \mathrm{Rel}(G^A)(S)(h, g)\\
&\iff (\mathrm{Rel}(G^A)(R) \circ \mathrm{Rel}(G^A)(S))(f, g).
\end{aligned}$$

The last statement (5) follows from the previous ones:

- If the relation R is reflexive, i.e. satisfies $\mathrm{Eq}(X) \subseteq R$, then $\mathrm{Eq}(F(X)) = \mathrm{Rel}(F)(\mathrm{Eq}(X)) \subseteq \mathrm{Rel}(F)(R)$, so that $\mathrm{Rel}(F)(R)$ is also reflexive.
- If R is symmetric, that is, if $R \subseteq R^{-1}$, then $\mathrm{Rel}(F)(R) \subseteq \mathrm{Rel}(F)(R^{-1}) = \mathrm{Rel}(F)(R)^{-1}$, so that $\mathrm{Rel}(F)(R)$ is symmetric as well.
- If R is transitive, i.e. $R \circ R \subseteq R$, then $\mathrm{Rel}(F)(R) \circ \mathrm{Rel}(F)(R) = \mathrm{Rel}(F)(R \circ R) \subseteq \mathrm{Rel}(F)(R)$, so that $\mathrm{Rel}(F)(R)$ is also transitive. ❑

We proceed with a similar lemma, about relation lifting and (inverse and direct) images.

Lemma 3.2.2 *Let $F\colon$ **Sets** $\to$ **Sets** be a Kripke polynomial functor again, and let $f\colon X \to Z$ and $g\colon Y \to W$ be arbitrary functions.*

1. *Relation lifting commutes with inverse images: for $R \subseteq Z \times W$,*

$$\mathrm{Rel}(F)((f \times g)^{-1}(R)) = (F(f) \times F(g))^{-1}(\mathrm{Rel}(F)(R)).$$

2. *Relation lifting also commutes with direct images: for $R \subseteq X \times Y$,*

$$\mathrm{Rel}(F)(\textstyle\coprod_{f\times g}(R)) = \coprod_{F(f)\times F(g)}(\mathrm{Rel}(F)(R)).$$

Proof Both equations are proved by induction on the structure of F. We leave (1) to the reader. Once (1) is established, it can be used to prove the direction ($\supseteq$) of (2), using the Galois connection relating direct and inverse images in (2.15):

$$\begin{aligned}
&\textstyle\coprod_{F(f)\times F(g)}(\mathrm{Rel}(F)(R)) \subseteq \mathrm{Rel}(F)(\coprod_{f\times g}(R))\\
&\iff \mathrm{Rel}(F)(R) \subseteq (F(f) \times F(g))^{-1}\mathrm{Rel}(F)(\textstyle\coprod_{f\times g}(R))\\
&\qquad\qquad = \mathrm{Rel}(F)((f \times g)^{-1} \textstyle\coprod_{f\times g}(R)).
\end{aligned}$$

This second inclusion is obtained from monotonicity of relation lifting (see Lemma 3.2.1(3)), using that $R \subseteq (f \times g)^{-1} \coprod_{f\times g}(R)$.

The inclusion ($\subseteq$) of (2) is proved by induction on the structure of the functor F. This requires the axiom of choice to handle the exponent functor case, as in the proof of point (4) in the previous lemma. The powerset case $F = \mathcal{P}G$ is most complicated, and will be described in detail.

$$
\begin{aligned}
&\mathrm{Rel}(\mathcal{P}G)(\textstyle\coprod_{f\times g}(R))(U,V)\\
&\iff \forall x \in U.\, \exists y \in V.\, \mathrm{Rel}(G)(\textstyle\coprod_{f\times g}(R))(x,y)\\
&\qquad \wedge\ \forall y \in V.\, \exists x \in U.\, \mathrm{Rel}(G)(\textstyle\coprod_{f\times g}(R))(x,y)\\
&\overset{\text{(IH)}}{\iff} \forall x \in U.\, \exists y \in V.\, \textstyle\coprod_{G(f)\times G(g)} \mathrm{Rel}(G)(R)(x,y)\\
&\qquad \wedge\ \forall y \in V.\, \exists x \in U.\, \textstyle\coprod_{G(f)\times G(g)} \mathrm{Rel}(G)(R)(x,y)\\
&\iff \forall x \in U.\, \exists y \in V.\, \exists u, v.\, G(f)(u) = x \wedge G(g)(v) = y\\
&\qquad\qquad \wedge\ \mathrm{Rel}(G)(R)(u,v)\\
&\qquad \wedge\ \forall y \in V.\, \exists x \in U.\, \exists u, v.\, G(f)(u) = x \wedge G(g)(v) = y\\
&\qquad\qquad \wedge\ \mathrm{Rel}(G)(R)(u,v)\\
&\Longrightarrow \forall u \in U'.\, \exists v \in V'.\, \mathrm{Rel}(G)(R)(u,v)\\
&\qquad \wedge\ \forall v \in V'.\, \exists u \in U'.\, \mathrm{Rel}(G)(R)(u,v), \quad \text{where}\\
&\qquad\qquad U' = \{u \mid G(f)(u) \in U \wedge \exists v.\, G(g)(v) \in V \wedge \mathrm{Rel}(G)(R)(u,v)\}\\
&\qquad\qquad V' = \{v \mid G(g)(v) \in V \wedge \exists u.\, G(f)(u) \in U \wedge \mathrm{Rel}(G)(R)(u,v)\}\\
&\iff \exists U', V'.\, \mathcal{P}(G(f))(U') = U \wedge \mathcal{P}(G(f))(V')\\
&\qquad\qquad\qquad = V \wedge \mathrm{Rel}(\mathcal{P}(G))(R)(U',V')\\
&\iff \textstyle\coprod_{\mathcal{P}(G(f))\times\mathcal{P}(G(g))}(\mathrm{Rel}(\mathcal{P}G)(R))(U,V). \qquad \square
\end{aligned}
$$

Below we show in a diagram why $\mathrm{Rel}(F)$ is called relation lifting. The term 'relator' is often used in this context for the lifting of F; see for instance [446] (and Definition 4.4.5, where a more general description of the situation is given). Additionally, the next result shows that bisimulations are coalgebras themselves. It involves a category **Rel** with relations *as objects*. This **Rel** should not be confused with the category **SetsRel**, from Example 1.4.2.4, which has relations *as morphisms*.

Definition 3.2.3 We write **Rel** for the category with binary relations $R \subseteq X \times Y$ as objects. A morphism $(R \subseteq X \times Y) \longrightarrow (S \subseteq U \times V)$ consists of two functions $f\colon X \to U$ and $g\colon Y \to V$ with $R(x,y) \implies S(f(x), g(y))$ for all $x \in X, y \in Y$. The latter amounts to the existence of the necessarily unique dashed map in:

$$
\begin{array}{ccc}
R & \dashrightarrow & S\\
\downarrow & & \downarrow\\
X \times Y & \xrightarrow[f\times g]{} & U \times V
\end{array}
$$

Equivalently, $R \subseteq (f \times g)^{-1}(S)$, or $\coprod_{f\times g}(R) \subseteq S$, by the correspondence (2.15).

Lemma 3.2.4 *Consider a Kripke polynomial functor F.*

1. *Relation lifting forms a functor:*

$$\begin{array}{ccc} \mathbf{Rel} & \xrightarrow{\mathrm{Rel}(F)} & \mathbf{Rel} \\ \downarrow & & \downarrow \\ \mathbf{Sets} \times \mathbf{Sets} & \xrightarrow[F \times F]{} & \mathbf{Sets} \times \mathbf{Sets} \end{array}$$

   where the vertical arrows are the obvious forgetful functors.
2. *A bisimulation is a* $\mathrm{Rel}(F)$*-coalgebra in this category* **Rel**. *Similarly, a congruence is a* $\mathrm{Rel}(F)$*-algebra in* **Rel**.

Proof 1. We show that if a pair $(f, g)\colon R \to S$ is a morphism in **Rel** – where $f\colon X \to U$ and $g\colon Y \to V$ – then $(F(f), F(g))\colon \mathrm{Rel}(F)(R) \to \mathrm{Rel}(F)(S)$ is also a morphism in **Rel**. From the inclusion $R \subseteq (f \times g)^{-1}(S)$ we obtain by monotony and preservation of inverse images by relation lifting:

$$\mathrm{Rel}(F)(R) \subseteq \mathrm{Rel}(F)((f \times g)^{-1}(S)) = (F(f) \times F(g))^{-1}\mathrm{Rel}(F)(S).$$

   This means that $(F(f), F(g))$ is a morphism $\mathrm{Rel}(F)(R) \to \mathrm{Rel}(F)(S)$ in **Rel**.

2. A $\mathrm{Rel}(F)$-coalgebra $R \to \mathrm{Rel}(F)(R)$ in **Rel**, for $R \subseteq X \times Y$, consists of two underlying maps $c\colon X \to F(X)$ and $d\colon Y \to F(Y)$ with

$$\begin{array}{ccc} R & \dashrightarrow & \mathrm{Rel}(F)(R) \\ \downarrow & & \downarrow \\ X \times Y & \xrightarrow[c \times d]{} & F(X) \times F(Y) \end{array}$$

   This says that R is a relation which is closed under the F-coalgebras c, d, i.e. that R is a bisimulation for c, d.

   In the same way congruences are $\mathrm{Rel}(F)$-algebras in the category **Rel**. ❑

The next result lists several useful preservation properties of relation lifting.

Lemma 3.2.5 *Relation lifting* $\mathrm{Rel}(F)$ *for a given Kripke polynomial functor* $F\colon \mathbf{Sets} \to \mathbf{Sets}$ *preserves the following:*

1. ***Kernel*** *relations, given for an arbitrary function* $f\colon X \to Y$ *by:*

$$\begin{aligned} \mathrm{Ker}(f) &= \{(x_1, x_2) \in X \times X \mid f(x_1) = f(x_2)\} \\ &= (f \times f)^{-1}(\mathrm{Eq}(Y)) \end{aligned}$$

   in

$$\mathrm{Rel}(F)(\mathrm{Ker}(f)) = \mathrm{Ker}(F(f)).$$

2. ***Graph*** *relations given for* $f : X \to Y$ *by:*

$$\begin{aligned}\mathrm{Graph}(f) &= \{(x,y) \in X \times Y \mid f(x) = y\} \\ &= (f \times \mathrm{id}_Y)^{-1}(\mathrm{Eq}(Y))\end{aligned}$$

in

$$\mathrm{Rel}(F)(\mathrm{Graph}(f)) = \mathrm{Graph}(F(f)).$$

3. ***Images*** *of tuples: for* $X \xleftarrow{f} Z \xrightarrow{g} Y$,

$$\begin{aligned}\mathrm{Im}(\langle f, g\rangle) &= \{(x,y) \in X \times Y \mid \exists z \in Z.\ f(z) = x \wedge g(z) = y\} \\ &= \coprod_{f \times g}(\mathrm{Eq}(Z))\end{aligned}$$

in

$$\mathrm{Rel}(F)(\mathrm{Im}(\langle f, g\rangle)) = \mathrm{Im}(\langle F(f), F(g)\rangle).$$

4. ***Pullback*** *relations of spans: for* $X \xrightarrow{f} Z \xleftarrow{g} Y$,

$$\begin{aligned}\mathrm{Eq}(f, g) &= \{(x,y) \in X \times Y \mid f(x) = g(y)\} \\ &= (f \times g)^{-1}(\mathrm{Eq}(Z))\end{aligned}$$

in

$$\mathrm{Rel}(F)(\mathrm{Eq}(f, g)) = \mathrm{Eq}(F(f), F(g)).$$

Proof 1. By the results from the previous two lemmas:

$$\begin{aligned}\mathrm{Rel}(F)(\mathrm{Ker}(f)) &= \mathrm{Rel}(F)((f \times f)^{-1}(\mathrm{Eq}(Y))) \\ &= (F(f) \times F(f))^{-1}(\mathrm{Rel}(F)(\mathrm{Eq}(Y))) \\ &= (F(f) \times F(f))^{-1}(\mathrm{Eq}(F(Y))) \\ &= \mathrm{Ker}(F(f)).\end{aligned}$$

2. Similarly,

$$\begin{aligned}\mathrm{Rel}(F)(\mathrm{Graph}(f)) &= \mathrm{Rel}(F)((f \times \mathrm{id}_Y)^{-1}(\mathrm{Eq}(Y))) \\ &= (F(f) \times \mathrm{id}_{F(Y)})^{-1}(\mathrm{Rel}(F)(\mathrm{Eq}(Y))) \\ &= (F(f) \times \mathrm{id}_{F(Y)})^{-1}(\mathrm{Eq}(F(Y))) \\ &= \mathrm{Graph}(F(f)).\end{aligned}$$

3. And

$$\begin{aligned}\mathrm{Rel}(F)(\mathrm{Im}(\langle f, g\rangle)) &= \mathrm{Rel}(F)(\coprod_{f \times g}(\mathrm{Eq}(Z))) \\ &= \coprod_{F(f) \times F(g)}(\mathrm{Rel}(F)(\mathrm{Eq}(Z))) \\ &= \coprod_{F(f) \times F(g)}(\mathrm{Eq}(F(Z))) \\ &= \mathrm{Im}(\langle F(f), F(g)\rangle)\end{aligned}$$

4. Finally,

$$\begin{aligned}\mathrm{Rel}(F)(\mathrm{Eq}(f,g)) &= \mathrm{Rel}(F)((f\times g)^{-1}(\mathrm{Eq}(Z)))\\ &= (F(f)\times F(g))^{-1}(\mathrm{Rel}(F)(\mathrm{Eq}(Z)))\\ &= (F(f)\times F(g))^{-1}(\mathrm{Eq}(F(Z)))\\ &= \mathrm{Eq}(F(f),F(g)).\end{aligned}$$ ❑

Once these auxiliary results are in place, the next two propositions establish a series of standard facts about bisimulations and bisimilarity; see [411, section 5]. We begin with closure properties.

Proposition 3.2.6 *Assume coalgebras* $X \xrightarrow{c} F(X)$, $X' \xrightarrow{c'} F(X')$, *and* $Y \xrightarrow{d} F(Y)$, $Y' \xrightarrow{d'} F(Y')$ *of a Kripke polynomial functor* F. *Bisimulations are closed under the following:*

1. *Reversal: if* $R \subseteq X \times Y$ *is a bisimulation, then so is* $R^\dagger \subseteq Y \times X$.
2. *Composition: if* $R \subseteq X \times X'$ *and* $S \subseteq X' \times Y$ *are bisimulations, then so is* $(S \circ R) \subseteq X \times Y$.
3. *Arbitrary unions: if* $R_i \subseteq X \times Y$ *is a bisimulation for each* $i \in I$, *then so is* $\bigcup_{i\in I} R_i \subseteq X \times Y$.
4. *Inverse images: for homomorphisms* $f\colon X \to Y$ *and* $f'\colon X' \to Y'$, *if* $R \subseteq Y \times Y'$ *is a bisimulation, then so is* $(f \times f')^{-1}(R) \subseteq X \times X'$.
5. *Direct images: for homomorphisms* $f\colon X \to Y$ *and* $f'\colon X' \to Y'$, *if* $R \subseteq X \times X'$ *is a bisimulation, then so is* $\coprod_{f\times f'}(R) \subseteq Y \times Y'$.

Proof 1. If the relation R is a bisimulation, i.e. if $R \subseteq (c \times d)^{-1}(\mathrm{Rel}(F)(R))$, then

$$\begin{aligned}R^\dagger &\subseteq \Big((c\times d)^{-1}(\mathrm{Rel}(F)(R))\Big)^\dagger\\ &= (d\times c)^{-1}\Big(\mathrm{Rel}(F)(R)^\dagger\Big)\\ &= (d\times c)^{-1}(\mathrm{Rel}(F)(R^\dagger)) \quad \text{by Lemma 3.2.1.2.}\end{aligned}$$

2. Assume $R \subseteq (c \times e)^{-1}(\mathrm{Rel}(F)(R))$ and $S \subseteq (e \times d)^{-1}(\mathrm{Rel}(F)(S))$. If $(x,y) \in (S \circ R)$, say with $R(x,w)$ and $S(w,y)$, then by assumption $(c(x),e(w)) \in \mathrm{Rel}(F)(R)$ and $(e(w),d(y)) \in \mathrm{Rel}(F)(S)$. Hence $(c(x),d(y)) \in (\mathrm{Rel}(F)(S) \circ \mathrm{Rel}(F)(R)) = \mathrm{Rel}(F)(S \circ R)$, by Lemma 3.2.1.4. We have thus proved the inclusion $(S \circ R) \subseteq (c \times d)^{-1}(\mathrm{Rel}(F)(S \circ R)$, and thus that $S \circ R$ is a bisimulation.
3. Assume $R_i \subseteq (c \times d)^{-1}(\mathrm{Rel}(F)(R_i))$, for each $i \in I$. Then

$$\begin{aligned}\textstyle\bigcup_{i\in I} R_i &\subseteq \textstyle\bigcup_{i\in I}(c\times d)^{-1}(\mathrm{Rel}(F)(R_i))\\ &= (c\times d)^{-1}\big(\textstyle\bigcup_{i\in I}\mathrm{Rel}(F)(R_i)\big) && \text{inverse image preserves unions}\\ &\subseteq (c\times d)^{-1}\big(\mathrm{Rel}(F)(\textstyle\bigcup_{i\in I} R_i)\big) && \text{by monotony of relation lifting}\\ & && \text{(and of inverse images).}\end{aligned}$$

4. If $R \subseteq (d \times d')^{-1}(\mathrm{Rel}(F)(R))$, then

$$\begin{array}{ll} (f \times f')^{-1}(R) & \\ \subseteq (f \times f')^{-1}(d \times d')^{-1}(\mathrm{Rel}(F)(R)) & \\ = (c \times c')^{-1}(F(f) \times F(f'))^{-1}(\mathrm{Rel}(F)(R)) & f, f' \text{ are homomorphisms} \\ = (c \times c')^{-1}(\mathrm{Rel}(F)((f \times f')^{-1}(R))) & \text{by Lemma 3.2.2.1.} \end{array}$$

5. If $\coprod_{c \times c'}(R) \subseteq \mathrm{Rel}(F)(R)$, then:

$$\begin{array}{ll} \coprod_{d \times d'} \coprod_{f \times f'}(R) & \\ = \coprod_{F(f) \times F(f')} \coprod_{c \times c'}(R) & \text{since } f, f' \text{ are homomorphisms} \\ \subseteq \coprod_{F(f) \times F(f')}(\mathrm{Rel}(F)(R)) & \text{by assumption} \\ = \mathrm{Rel}(F)(\coprod_{f \times f'}(R)) & \text{by Lemma 3.2.2.2.} \end{array}$$ ❑

Proposition 3.2.7 *Let $X \xrightarrow{c} F(X)$, $Y \xrightarrow{d} F(Y)$ and $Z \xrightarrow{e} F(Z)$ be three coalgebras of a Kripke polynomial functor F.*

1. *An arbitrary function $f\colon X \to Y$ is a homomorphism of coalgebras if and only if its graph relation* $\mathrm{Graph}(f)$ *is a bisimulation.*
2. *The equality relation* $\mathrm{Eq}(X)$ *on X is a bisimulation equivalence. More generally, for a homomorphism $f\colon X \to Y$, the kernel relation* $\mathrm{Ker}(f)$ *is a bisimulation equivalence.*
3. *For two homomorphisms $X \xleftarrow{f} Z \xrightarrow{g} Y$ the image of the tuple* $\mathrm{Im}(\langle f, g\rangle) \subseteq X \times Y$ *is a bisimulation.*
4. *For two homomorphisms $X \xrightarrow{f} Z \xleftarrow{g} Y$ in the opposite direction, the pullback relation* $\mathrm{Eq}(f, g) \subseteq X \times Y$ *is a bisimulation.*

Proof By using Lemma 3.2.1.

1. Because

$$\begin{array}{ll} \mathrm{Graph}(f) \text{ is a bisimulation} & \\ \iff \mathrm{Graph}(f) \subseteq (c \times d)^{-1}(\mathrm{Rel}(F)(\mathrm{Graph}(f))) & \\ \qquad = (c \times d)^{-1}(\mathrm{Graph}(F(f))) & \text{by Lemma 3.2.5.2} \\ \iff \forall x, y.\ f(x) = y \Rightarrow F(f)(c(x)) = d(y) & \\ \iff \forall x.\ F(f)(c(x)) = d(f(x)) & \\ \iff f \text{ is a homomorphism of coalgebras from } c \text{ to } d. & \end{array}$$

2. The fact that equality is a bisimulation follows from Lemma 3.2.1.1. Further, the kernel $\mathrm{Ker}(f)$ is a bisimulation because it can be written as $\mathrm{Graph}(f) \circ \mathrm{Graph}(f)^{-1}$, which is a bisimulation by point (1) and by Proposition 3.2.6.1, 3.2.6.2.
3. We wish to prove an inclusion:

$$\mathrm{Im}(\langle f, g\rangle) \subseteq (c \times d)^{-1}(\mathrm{Rel}(F)(\mathrm{Im}(\langle f, g\rangle))) = (c \times d)^{-1}(\mathrm{Im}(\langle F(f), F(g)\rangle)),$$

where we used Lemma 3.2.5.3 for the last equation. This means that we have to prove: for each $z \in Z$ there is a $w \in F(Z)$ with $c(f(z)) = F(f)(w)$ and $d(f(z)) = F(g)(w)$. But clearly we can take $w = e(z)$.

4. We now seek an inclusion:

$$\mathrm{Eq}(f,g) \subseteq (c \times d)^{-1}(\mathrm{Rel}(F)(\mathrm{Eq}(f,g))) = (c \times d)^{-1}(\mathrm{Eq}(F(f), F(g))),$$

via Lemma 3.2.5.4. But this amounts to $f(x) = g(y) \Rightarrow F(f)(c(x)) = G(f)(d(y))$, for all $x \in X, y \in Y$. The implication holds because both f and g are homomorphisms. ❑

We can now also establish some elementary properties of bisimilarity.

Proposition 3.2.8 *Let $F\colon$ **Sets** $\to$ **Sets** be a Kripke polynomial functor, with coalgebras $X \xrightarrow{c} F(X)$, $X' \xrightarrow{c'} F(X')$, and $Y \xrightarrow{d} F(Y)$, $Y' \xrightarrow{d'} F(Y')$.*

1. *The bisimilarity relation ${}_c\!\leftrightarrow\!\!\underline{\ }_d \subseteq X \times Y$ is a bisimulation; it is the greatest among all bisimulations between c and d.*
2. $({}_c\!\leftrightarrow\!\!\underline{\ }_d)^\dagger \subseteq {}_d\!\leftrightarrow\!\!\underline{\ }_c$ *and* ${}_c\!\leftrightarrow\!\!\underline{\ }_e \circ {}_e\!\leftrightarrow\!\!\underline{\ }_d \subseteq {}_c\!\leftrightarrow\!\!\underline{\ }_d$.
3. *The bisimilarity relation ${}_c\!\leftrightarrow\!\!\underline{\ }_c \subseteq X \times X$ for a single coalgebra is a bisimulation equivalence.*
4. *For homomorphisms of coalgebras $f\colon X \to Y$ and $f'\colon X' \to Y'$ one has, for $x \in X, x' \in X'$,*

$$f(x)\ {}_d\!\leftrightarrow\!\!\underline{\ }_{d'}\ f'(x') \iff x\ {}_c\!\leftrightarrow\!\!\underline{\ }_{c'}\ x'.$$

Proof 1. By Proposition 3.2.6.2.

2. The first inclusion follows from Proposition 3.2.6.1 and the second one from Proposition 3.2.6.2.
3. The bisimilarity relation ${}_c\!\leftrightarrow\!\!\underline{\ }_c \subseteq X \times X$ is reflexive, because the equality relation $\mathrm{Eq}(X) \subseteq X \times X$ is a bisimulation; see Proposition 3.2.6.2. Symmetry and transitivity of ${}_c\!\leftrightarrow\!\!\underline{\ }_c$ follow from point (2).
4. Since ${}_d\!\leftrightarrow\!\!\underline{\ }_{d'} \subseteq Y \times Y'$ is a bisimulation, so is $(f \times f')^{-1}({}_d\!\leftrightarrow\!\!\underline{\ }_{d'}) \subseteq X \times X'$, by Proposition 3.2.6.4. Hence $(f \times f')^{-1}({}_d\!\leftrightarrow\!\!\underline{\ }_{d'}) \subseteq {}_c\!\leftrightarrow\!\!\underline{\ }_{c'}$, which corresponds to the implication ($\Longrightarrow$).

   Similarly we obtain an inclusion $\coprod_{f \times f'}({}_c\!\leftrightarrow\!\!\underline{\ }_{c'}) \subseteq {}_d\!\leftrightarrow\!\!\underline{\ }_{d'}$ from Proposition 3.2.6.5, which yields ($\Longleftarrow$). ❑

Exercises

3.2.1 1. Prove that relation lifting $\mathrm{Rel}(F)$ for an *exponent* polynomial functor F (hence without powersets $\mathcal{P}$) preserves non-empty intersections of relations: for $I \neq \emptyset$,

$$\mathrm{Rel}(F)(\textstyle\bigcap_{i\in I} R_i) \;=\; \bigcap_{i\in I} \mathrm{Rel}(F)(R_i).$$

2. Assume now that F is now a simple polynomial functor (also without exponents $(-)^A$, for A infinite). Prove that relation lifting $\mathrm{Rel}(F)$ preserves unions of ascending chains of relations: if $S_0 \subseteq S_1 \subseteq S_2 \subseteq \cdots$, then

$$\mathrm{Rel}(F)(\textstyle\bigcup_{n\in\mathbb{N}} S_n) \;=\; \bigcup_{n\in\mathbb{N}} \mathrm{Rel}(F)(S_n).$$

Which additional closure properties hold for bisimulations for coalgebras of such functors?

3.2.2 1. Check that if $\leq$ is a preorder on a set X, then $\mathrm{Rel}(F)(\leq)$ is also a preorder on $F(X)$.
2. Prove the same with 'poset' instead of 'preorder', for an exponent polynomial functor F (without powerset).
3. Prove that a Galois connection $f \dashv g$ in

$$(X, \leq_X) \underset{g}{\overset{f}{\leftrightarrows}} (Y, \leq_Y)$$

yields a 'lifted' Galois connection $F(f) \dashv F(g)$ in

$$(F(X), \mathrm{Rel}(F)(\leq_X)) \underset{F(g)}{\overset{F(f)}{\leftrightarrows}} (F(Y), \mathrm{Rel}(F)(\leq_Y)).$$

3.2.3 Check that, in analogy with Proposition 3.2.6, congruences are closed under inverses, composition, arbitrary intersections and inverse and direct images.

3.2.4 Prove the following analogue of Proposition 3.2.7. Let F be a polynomial functor, with algebras $F(X) \xrightarrow{a} X$, $F(Y) \xrightarrow{b} Y$ and $F(Z) \xrightarrow{c} Z$.

1. A function $f\colon X \to Y$ is a homomorphism of algebras if and only if its graph relation $\mathrm{Graph}(f) \subseteq X \times Y$ is a congruence.
2. The kernel relation $\mathrm{Ker}(f)$ of an algebra homomorphism $f\colon X \to Y$ is always a congruence equivalence.
3. The image $\mathrm{Im}(\langle f, g\rangle) \subseteq X \times Y$ of a pair of algebra homomorphisms $X \xleftarrow{f} Z \xrightarrow{g} Y$ is a congruence.
4. The pullback relation $\mathrm{Eq}(f, g) \subseteq X \times Y$ of a span of algebra homomorphisms $X \xrightarrow{f} Z \xleftarrow{g} Y$ is a congruence.

3.2.5 Let $R \subseteq X \times X$ be an arbitrary relation on the state space of a coalgebra, and let $\overline{R}$ be the least equivalence relation containing R. Prove that if R is a bisimulation, then $\overline{R}$ is a bisimulation equivalence. *Hint*: Write $\overline{R}$ as union of iterated compositions R^n for $n \in \mathbb{N}$.

3.2.6 Check that lax relation lifting as introduced in Exercise 3.1.6 forms a functor in

$$\begin{array}{ccc} \mathbf{Rel} & \xrightarrow{\mathrm{Rel}_{\sqsubseteq}(F)} & \mathbf{Rel} \\ \downarrow & & \downarrow \\ \mathbf{Sets} \times \mathbf{Sets} & \xrightarrow[F \times F]{} & \mathbf{Sets} \times \mathbf{Sets} \end{array}$$

and that simulations are coalgebras of this functor $\mathrm{Rel}_{\sqsubseteq}(F)$ – as in Lemma 3.2.4.

3.2.7 This exercise describes a simple characterisation from [154] of when a function is definable by induction. The analogue for coinduction is in Exercise 6.2.4.

Consider an initial algebra $F(A) \xrightarrow{\cong} A$ of a polynomial functor F, where $A \neq \emptyset$. Prove that a function $f\colon A \to X$ is defined by initiality (i.e. $f = \mathrm{int}_a$ for some algebra $a\colon T(X) \to X$ on its codomain) if and only its kernel $\mathrm{Ker}(f)$ is a congruence.

Hint: Extend the induced map $F(A)/\mathrm{Ker}(F(f)) \to X$ along the inclusion $F(A)/\mathrm{Ker}(F(f)) \rightarrowtail F(X)$ by using an arbitrary element in $F(A)/\mathrm{Ker}(F(f))$ obtained from $A \neq \emptyset$.

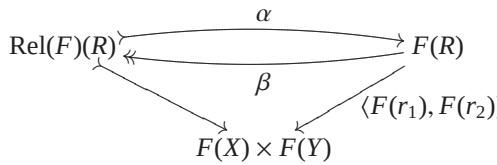
3.3 Bisimulations as Spans and Cospans

This section continues the investigation of bisimulations and focusses specifically on the relation between the definition of bisimulation used here, given in terms of relation lifting, and an earlier definition given by Aczel and Mendler; see [9, 12]. One of the main results is that these definitions are equivalent, as will be shown below in Theorem 3.3.2.

The first lemma below establishes an important technical relationship which forms the basis for the subsequent theorem. The lemma uses that relations can be considered as sets themselves. From a logical perspective this involves a form of comprehension; see e.g. [239, chapter 4, section 6] or [218, 45, 150].

Lemma 3.3.1 *Let $F\colon$ **Sets** $\to$ **Sets** be a Kripke polynomial functor and $R \subseteq X \times Y$ be an arbitrary relation, written via explicit functions $\langle r_1, r_2 \rangle\colon R \rightarrowtail X \times Y$. The lifted relation* $\mathrm{Rel}(F)(R)$*, considered as a set, is a* ***retract*** *of $F(R)$:*

there are functions $\alpha\colon \mathrm{Rel}(F)(R) \to F(R)$ *and* $\beta\colon F(R) \to \mathrm{Rel}(F)(R)$ *with* $\beta \circ \alpha = \mathrm{id}_{\mathrm{Rel}(F)(R)}$. *Moreover, these* α *and* β *make the following triangle commute:*



This means that $\mathrm{Rel}(F)(R) \rightarrowtail F(X) \times F(Y)$ *is the image of* $F(R) \to F(X) \times F(Y)$.

Proof The functions α and β are constructed by induction on the structure of F. In the two base cases where F is the identity functor or a constant functor, α and β are each other's inverses. We shall consider two induction steps, for product and powerset.

If F is a product $F_1 \times F_2$, we assume appropriate functions $\alpha_i\colon \mathrm{Rel}(F_i)(R) \to F_i(R)$ and $\beta_i\colon F_i(R) \to \mathrm{Rel}(F_i)(R)$, for $i = 1, 2$. The aim is to construct functions α, β in

$$\left(\begin{array}{l}\{((u_1,u_2),(v_1,v_2)) \mid \mathrm{Rel}(F_1)(R)(u_1,v_1) \\ \qquad\qquad \wedge\ \mathrm{Rel}(F_2)(R)(u_2,v_2)\}\end{array}\right) \underset{\beta}{\overset{\alpha}{\rightleftarrows}} F_1(R) \times F_2(R).$$

The definitions are obvious:

$$\begin{aligned}\alpha((u_1,u_2),(v_1,v_2)) &= (\,\alpha_1(u_1,v_1),\ \alpha_2(u_2,v_2)\,)\\ \beta(w_1,w_2) &= (\,(\pi_1\beta_1(w_1),\pi_1\beta_2(w_2)),\ (\pi_2\beta_1(w_1),\pi_2\beta_2(w_2))\,).\end{aligned}$$

If F is a powerset $\mathcal{P}(F_1)$, we may assume functions $\alpha_1\colon \mathrm{Rel}(F_1)(R) \to F_1(R)$ and $\beta_1\colon F_1(R) \to \mathrm{Rel}(F_1)(R)$ as in the lemma. We have to construct

$$\left(\begin{array}{l}\{(U,V) \mid \forall u \in U.\, \exists v \in V.\, \mathrm{Rel}(F_1)(R)(u,v) \\ \qquad\qquad \wedge\ \forall v \in V.\, \exists u \in U.\, \mathrm{Rel}(F_1)(R)(u,v)\}\end{array}\right) \underset{\beta}{\overset{\alpha}{\rightleftarrows}} \mathcal{P}(F_1(R)).$$

In this case we define:

$$\begin{aligned}\alpha(U,V) &= \{\alpha_1(u,v) \mid u \in U \wedge v \in V \wedge \mathrm{Rel}(F_1)(R)(u,v)\},\\ \beta(W) &= (\{\pi_1\beta_1(w) \mid w \in W\}, \{\pi_2\beta_1(w) \mid w \in W\}).\end{aligned}$$

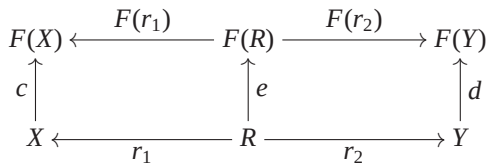
Then, using that $\beta_1 \circ \alpha_1 = \mathrm{id}$ holds by assumption, we compute:

$$\begin{aligned}(\beta \circ \alpha)(U,V) &= (\{\pi_1\beta_1\alpha_1(u,v) \mid u \in U \wedge v \in V \wedge \mathrm{Rel}(F)(R)(u,v)\},\\ &\qquad \{\pi_2\beta_1\alpha_1(u,v) \mid u \in U \wedge v \in V \wedge \mathrm{Rel}(F)(R)(u,v)\})\\ &= (\{u \in U \mid \exists v \in V.\, \mathrm{Rel}(F_1)(R)(u,v)\},\\ &\qquad \{v \in V \mid \exists u \in U.\, \mathrm{Rel}(F_1)(R)(u,v)\})\\ &= (U,V).\end{aligned}$$

❑

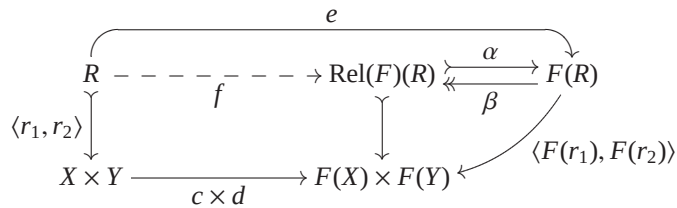
The formulation of bisimulation that we are using here relies on relation lifting; see Definition 3.1.2 and Lemma 3.2.4.2, where bisimulations for coalgebras of a functor F are described as Rel(F)-coalgebras. This comes from [218]. An earlier definition was introduced by Aczel and Mendler; see [9, 12]. With the last lemma we can prove the equivalence of these definitions.

Theorem 3.3.2 *Let $X \xrightarrow{c} F(X)$ and $Y \xrightarrow{d} F(Y)$ be two coalgebras of a Kripke polynomial functor F. A relation $\langle r_1, r_2\rangle \colon R \rightarrowtail X \times Y$ is a bisimulation for c and d if and only if R is an **Aczel–Mendler bisimulation**: R itself is the carrier of some coalgebra $e \colon R \to F(R)$, making the legs r_i homomorphisms of coalgebras, as in*



Thus, briefly: R carries a Rel(F)*-coalgebra in the category* **Rel** *if and only if R carries an F-coalgebra in* **Sets** *making the diagram commute.*

Proof In Lemma 3.2.4.2 we already saw that $R \subseteq X \times Y$ is a bisimulation according to Definition 3.1.2 if and only if it carries a Rel(F)-coalgebra; that is, if and only if the function $c \times d \colon X \times Y \to F(X) \times F(Y)$ restricts to a necessarily unique function $f \colon R \to \mathrm{Rel}(F)(R)$, making the square on the left below commute:

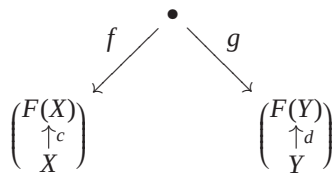


The functions α, β in the triangle on the right form the retract from the previous lemma. Then, if R is a bisimulation according to Definition 3.1.2, there is a function f as indicated, so that $\alpha \circ f \colon R \to F(R)$ yields an F-coalgebra on R making the legs r_i homomorphisms of coalgebras. Conversely, if there is a coalgebra $e \colon R \to F(R)$ making the r_i homomorphisms, then $\beta \circ e \colon R \to \mathrm{Rel}(F)(R)$ shows that $R \subseteq (c \times d)^{-1}(\mathrm{Rel}(F)(R))$. ❑

This result gives rise to another two alternative characterisations of bisimilarity. For non-polynomial functors on categories other than **Sets** these different descriptions may diverge; see [436] for more information.

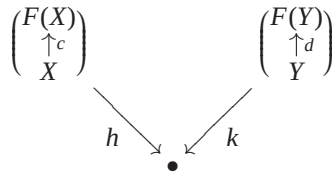
Theorem 3.3.3 *Assume two elements $x \in X$ and $y \in Y$ of coalgebras $X \xrightarrow{c} F(X)$ and $Y \xrightarrow{d} F(Y)$ of a Kripke polynomial functor $F\colon$ **Sets** $\to$ **Sets**. The following statements are equivalent:*

1. *x, y are bisimilar: $x \;{}_c\!\leftrightarrow\!{}_d\; y$.*
2. *There is a span of coalgebra homomorphisms:*



with $x = f(z)$ and $y = g(z)$, for some element z.

3. *x, y are **behaviourally equivalent**: there is a cospan of coalgebra homomorphisms:*



with $h(x) = k(y)$.

For the proof we recall from Exercise 2.1.14 that the category **Sets** has coequalisers and that categories of coalgebras **CoAlg**(F) of endofunctors of **Sets** then also have such coequalisers.

Proof (1) $\Rightarrow$ (2). If x and y are bisimilar, then they are contained in some bisimulation $R \subseteq X \times Y$. By the previous result, this relation carries a coalgebra structure making the two legs $X \leftarrow R \rightarrow Y$ homomorphisms, and thus a span of coalgebras.

(2) $\Rightarrow$ (3). Assume there is a span of coalgebra homomorphisms $f\colon W \to X$, $g\colon W \to Y$ as described above in point (2), with $x = f(z)$ and $y = g(z)$, for some $z \in W$. One obtains a cospan as in (3) by taking the pushout of f, g in the category **CoAlg**(F). This notion of pushout has not been discussed yet, but it can be described in terms of notions that we have seen. We form the coequaliser in **CoAlg**(F) in

$$\begin{pmatrix} F(W) \\ \uparrow \\ W \end{pmatrix} \underset{\kappa_2 \circ g}{\overset{\kappa_1 \circ f}{\rightrightarrows}} \begin{pmatrix} F(X+Y) \\ \uparrow \\ X+Y \end{pmatrix} \xrightarrow{\;q\;} \begin{pmatrix} F(Q) \\ \uparrow \\ Q \end{pmatrix}$$

where the coalgebra in the middle on $X + Y$ is $[F(\kappa_1) \circ c, F(\kappa_2) \circ d]$, as in Proposition 2.1.5. Then we take $h = q \circ \kappa_1$ and $k = q \circ \kappa_2$, so that:

$$\begin{aligned} h(x) \;=\; q(\kappa_1 x) &= q(\kappa_1 f(z)) \\ &= q(\kappa_2 g(z)) \qquad \text{since } q \text{ is coequaliser} \\ &= q(\kappa_2 y) \;=\; k(y). \end{aligned}$$

(3) $\Rightarrow$ (1). Assuming a cospan h, k of coalgebra homomorphisms with $h(x) = k(y)$ we take the pullback relation $\mathrm{Eq}(h, k) \subseteq X \times Y$. It is a bisimulation by Proposition 3.2.7 and it contains x, y by assumption. Hence $x \;{}_c\!\leftrightarrow_d\; y$. ❑

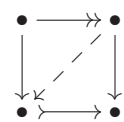
We postpone a discussion of the different formulations of the notion of bisimulation to the end of this section. At this stage we shall use the new 'Aczel–Mendler' bisimulations in the following standard result – specifically in its point (2) – about monos and epis in categories of coalgebras.

Theorem 3.3.4 *Let $F\colon$ **Sets** $\to$ **Sets** be a Kripke polynomial functor.*

1. *For a coalgebra $c\colon X \to F(X)$ and a bisimulation equivalence $R \subseteq X \times X$, the quotient set X/R carries a unique quotient coalgebra structure, written as $c/R\colon X/R \to F(X/R)$, making the canonical quotient map $[-]_R\colon X \twoheadrightarrow X/R$ a homomorphism of coalgebras, as in*

$$\begin{array}{ccc} F(X) & \xrightarrow{F([-]_R)} & F(X/R) \\ {\scriptstyle c}\uparrow & & \uparrow{\scriptstyle c/R} \\ X & \xrightarrow[[-]_R]{} & X/R \end{array}$$

2. *A homomorphism of coalgebras $f\colon X \to Y$ from $X \xrightarrow{c} F(X)$ to $Y \xrightarrow{d} F(Y)$ is a monomorphism/epimorphism in the category* **CoAlg**(F) *if and only if f is an injective/surjective function between the underlying sets.*
3. *Every homomorphism of coalgebras factors as an epimorphism followed by a monomorphism in* **CoAlg**(F). *This factorisation is essentially unique because of the following 'diagonal-fill-in' property. For each commuting square of coalgebra homomorphisms as below, there is a unique diagonal homomorphism making both triangles commute:*



This means that monomorphisms and epimorphisms in a category of coalgebras **CoAlg**(F) *form a so-called factorisation system; see [56, 36, 146] and Section 4.3.*

As an aside, the result (2) that a monomorphism between coalgebras is an injective function holds for Kripke polynomial functors but not for arbitrary functors, as shown in [184].

Proof 1. It suffices to prove that $F([-]_R) \circ c$ is constant on R. But this is obvious:

$$\begin{aligned} R &\subseteq (c \times c)^{-1}(\mathrm{Rel}(F)(R)) && \text{since } R \text{ is a bisimulation} \\ &= (c \times c)^{-1}(\mathrm{Rel}(F)(\mathrm{Ker}([-]_R))) \\ &= (c \times c)^{-1}(\mathrm{Ker}(F([-]_R))) && \text{by Lemma 3.2.5.1} \\ &= \mathrm{Ker}(F([-]_R) \circ c). \end{aligned}$$

2. It is standard that if a coalgebra homomorphism f is injective/surjective, then it is a monomorphism/epimorphism in **Sets** – see also Exercise 2.5.8 – and hence also in **CoAlg**(F). Conversely, assume first that f is a monomorphism in **CoAlg**(F). The kernel $\langle r_1, r_2 \rangle \colon \mathrm{Ker}(f) \rightarrowtail X \times X$ is a bisimulation by Proposition 3.2.7.2. Hence it carries an F-coalgebra structure by the previous theorem, making the r_i homomorphisms. From $f \circ r_1 = f \circ r_2$, we can conclude that $r_1 = r_2$, since f is a monomorphism in **CoAlg**(F). But $r_1 = r_2$ yields that f is injective:

$$\begin{aligned} f(x_1) = f(x_2) &\Longrightarrow (x_1, x_2) \in \mathrm{Ker}(f) \\ &\Longrightarrow x_1 = r_1(x_1, x_2) = r_2(x_1, x_2) = x_2. \end{aligned}$$

Next, assume that f is an epimorphism in **CoAlg**(F). There is a short categorical argument that tells that f is then an epi in **Sets**, and thus surjective: the forgetful functor **CoAlg**$(F) \to$ **Sets** creates colimits; see [411, proposition 4.7]. However, we spell out the argument. That f is epi in **CoAlg**(F) can be reformulated as: the following diagram is a coequaliser in **CoAlg**(F):

$$\begin{pmatrix} F(X) \\ \uparrow c \\ X \end{pmatrix} \underset{\kappa_2 \circ f}{\overset{\kappa_1 \circ f}{\rightrightarrows}} \begin{pmatrix} F(X+Y) \\ \uparrow \\ X+Y \end{pmatrix} \xrightarrow{[\mathrm{id}, \mathrm{id}]} \begin{pmatrix} F(Y) \\ \uparrow d \\ Y \end{pmatrix}$$

where the coalgebra in the middle is $[F(\kappa_1) \circ d, F(\kappa_2) \circ d]$, as in Proposition 2.1.5. Since in **Sets** a map is surjective if and only if it is an epimorphism – see Exercise 2.5.8 or any basic textbook in category theory – it suffices to show that f is an epimorphism in **Sets**. Thus we construct the coequaliser $q \colon Y + Y \twoheadrightarrow Q$ in **Sets** of $\kappa_1 \circ f, \kappa_2 \circ f \colon X \to Y + Y$.

By Exercise 2.1.14 this quotient set Q carries a unique coalgebra $Q \to F(Q)$ making q not only a homomorphism in $\mathbf{CoAlg}(F)$ but also a coequaliser of the homomorphisms $\kappa_1 \circ f, \kappa_2 \circ f$. Since coequalisers are determined up-to-isomorphism there is an isomorphism $\varphi\colon Y \xrightarrow{\cong} Q$ in $\mathbf{CoAlg}(F)$ with $\varphi \circ [\mathrm{id}, \mathrm{id}] = q$. This means that the codiagonal $[\mathrm{id}, \mathrm{id}]$ is also the coequaliser of $\kappa_1 \circ f, \kappa_2 \circ f$ in **Sets**, and thus that f is an epimorphism in **Sets** (and hence surjective).

3. Given a homomorphism of coalgebras $f\colon X \to Y$, we know by Proposition 3.2.7.2 that the kernel $\mathrm{Ker}(f)$ is a bisimulation equivalence. Hence by point (1) the quotient $X/\mathrm{Ker}(f)$ carries a coalgebra structure and yields a standard factorisation (in **Sets**):

$$\Big(X \xrightarrow{\ f\ } Y\Big) = \Big(X \xrightarrow{\ e\ } X/\mathrm{Ker}(f) \xrightarrow{\ m\ } Y\Big).$$

The map $e\colon X \twoheadrightarrow X/\mathrm{Ker}(f)$ is by definition of the unique coalgebra structure c' on $X/\mathrm{Ker}(f)$ – see point (1) – a homomorphism of coalgebras in

$$\begin{array}{ccccc} F(X) & \xrightarrow{F(e)} & F(X/\mathrm{Ker}(f)) & \xrightarrow{F(m)} & F(Y) \\ {\scriptstyle c}\uparrow & & \uparrow{\scriptstyle c' = c/\mathrm{Ker}(f)} & & \uparrow{\scriptstyle d} \\ X & \xrightarrow[e]{} & X/\mathrm{Ker}(f) & \xrightarrow[m]{} & Y \end{array}$$

Not only the square on the left, but also the one on the right commutes; using that e is an epimorphism: $d \circ m = F(m) \circ c'$ follows from

$$\begin{aligned} d \circ m \circ e &= d \circ f \\ &= F(f) \circ c \\ &= F(m) \circ F(e) \circ c \\ &= F(m) \circ c' \circ e. \end{aligned}$$

Thus, $m\colon X/\mathrm{Ker}(f) \rightarrowtail Y$ is also a homomorphism of coalgebras. Hence we have a factorisation $X \twoheadrightarrow X/\mathrm{Ker}(f) \rightarrowtail Y$ of f in the category $\mathbf{CoAlg}(F)$.

Regarding the diagonal-fill-in property, the diagonal is defined via the surjectivity of the top arrow. Hence this diagonal is a homomorphism of coalgebras. ❑

3.3.1 Comparing Definitions of Bisimulation

We started this chapter by introducing bisimulations via the logical technique of relation lifting, and later showed equivalence to quite different formulations in terms of (co)spans (Theorems 3.3.2 and 3.3.3). We shall discuss some differences between these 'logical' and 'span-based' approaches.

1. The logical approach describes bisimulations as relations with a special *property*, whereas the (co)span-based approaches rely on the existence of certain (coalgebra) *structure*. This aspect of the logical approach is more appropriate, because it is in line with the idea that bisimulations are special kinds of relations. If, in the Aczel–Mendler approach, the coalgebra structure is necessarily unique, existence of this structure also becomes a property. But uniqueness is neither required nor guaranteed. This is slightly unsatisfactory.
2. Relation lifting has been defined by induction on the structure of Kripke polynomial functors. Therefore, the logical approach (so far) applies only to such a limited collection of functors. Later, in Section 4.4 we will extend such lifting to functors on categories equipped with a factorisation system. But (co)span-based approaches apply more generally, without such factorisation structure. However, the dependence on such structure may also be seen as an advantage, as will be argued next.
3. Relation lifting is a logical technique which is not restricted to the standard classical logic of sets but may be defined for more general (categorical) logics, in terms of factorisation systems (see Section 4.4) or in terms of 'indexed' or 'fibred' preorders; see [218, 239]. For instance, one may wish to consider topological spaces with different logics, for instance with predicates given by the subsets which are closed, open, or both (clopen). Each of those preorders of predicates has different algebraic/logical properties. Thus, the logical approach is more general (or flexible) in another, logical dimension.
4. The cospan formulation (or behavioural equivalence) is essentially equivalent to equality on the final coalgebra. But the convenient aspect of behavioural equivalence is that it makes also makes sense in contexts where there is no final coalgebra.
5. Bisimulation relations can be constructed iteratively, by adding pairs until the relation is appropriately closed. This is done for instance in tools such as CIRC [338]. Such techniques are not available for behavioural equivalence.

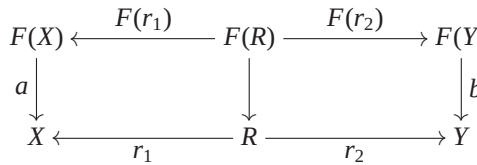
There is no clear answer as to which approach is 'the best'. In more general situations – functors other than the polynomial ones, on categories other than **Sets** – the various notions may diverge (see [436]) and one may be better than the other. For instance, in coalgebraic modal logic the cospan-based approach, using behavioural equivalence (see Theorem 3.3.3.3), seems to work best. We have chosen to start from the logical approach because we consider it to be more intuitive and easier to use in concrete examples. However, from now on we shall freely switch between the different approaches and use whatever is most convenient in a particular situation.

In Chapter 6 on invariants we shall encounter the same situation. There is a logical definition based on 'predicate lifting'. It leads to a notion of invariant, which, in the classical logic of sets, is equivalent to the notion of subcoalgebra; see Theorem 6.2.5. The latter is again defined in terms of structure and applies to more general functors.

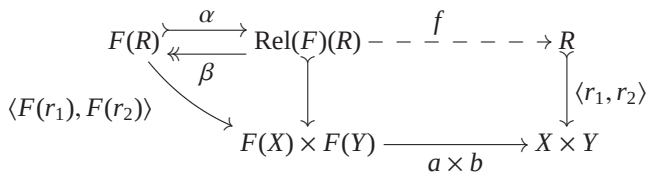
3.3.2 Congruences and Spans

We conclude this section with two basic results for algebra. The first one is an analogue of the Aczel–Mendler formulation of bisimulations (see Theorem 3.3.2) for congruences on algebras. The second one involves quotient algebras; it requires an auxiliary technical result about quotients in **Sets**.

Theorem 3.3.5 *Assume two algebras $a\colon F(X) \to X$ and $b\colon F(Y) \to Y$ of a Kripke polynomial functor F. A relation $R \subseteq X \times Y$ is a congruence relation if and only it carries a (necessarily unique) algebra structure $F(R) \to R$ itself, making the two legs $\langle r_1, r_2\rangle\colon R \rightarrowtail X \times Y$ of the inclusion homomorphisms of algebras, as in*



Proof Recall the retract (α, β), with $\beta \circ \alpha = \mathrm{id}$, from Lemma 3.3.1 in



Assume R is a congruence, say via $f\colon \mathrm{Rel}(F)(R) \to R$ as above. Then $f \circ \alpha\colon F(R) \to R$ is an algebra such that the r_i are homomorphisms. Conversely, if an algebra $c\colon F(R) \to R$ making the r_i morphisms of algebras exists, then $c \circ \beta$ ensures $\mathrm{Rel}(F)(R) \leq (a \times b)^{-1}(R)$. ❑

Theorem 3.3.4 describes coalgebras on quotients in the category **Sets**. Below we use some special properties of sets, like the axiom of choice and the fact that each equivalence relation is 'effective': it is equal to the kernel of its quotient map.

Lemma 3.3.6 *Let $F\colon$ **Sets** $\to$ **Sets** be an arbitrary endofunctor. If $q\colon X \twoheadrightarrow X/R$ is the quotient map of an equivalence relation $\langle r_1, r_2\rangle\colon R \hookrightarrow X \times X$, then we have a coequaliser diagram:*

$$F(R) \underset{F(r_2)}{\overset{F(r_1)}{\rightrightarrows}} F(X) \xrightarrow{F(q)} \!\!\!\!\rightarrow F(X/R).$$

Proof The quotient map $q\colon X \twoheadrightarrow X/R$ sends an element $x \in X$ to its equivalence class $q(x) = [x]_R = \{x' \in X \mid R(x, x')\}$ with respect to the relation R. It is surjective, so by the axiom of choice we may assume a section $s\colon X/R \to X$ with $q \circ s = \mathrm{id}$ – as noted at the end of Section 2.1. This section s chooses a representative in each equivalence class. Since $q(x) = q(s(q(x)))$ we get $R(x, s(q(x)))$, for each $x \in X$. Thus we can define a function $d\colon X \to R$ by $d(x) = (x, s(q(x)))$; by construction it satisfies $r_1 \circ d = \mathrm{id}$ and $r_2 \circ d = s \circ q$.

We now turn to the diagram in the lemma. First, we know that the map $F(e)$ is surjective, because it has $F(s)$ as a section: $F(q) \circ F(s) = F(q \circ s) = \mathrm{id}$. Next, assume we have a map $f\colon F(X) \to Y$ with $f \circ F(r_1) = f \circ F(r_2)$. Then

$$\begin{aligned} f \;=\; f \circ \mathrm{id}_{F(X)} &= f \circ F(r_1) \circ F(d) \\ &= f \circ F(r_2) \circ F(d) \qquad \text{by assumption about } f. \\ &= f \circ F(s) \circ F(q) \end{aligned}$$

This shows that $f' = f \circ F(s)\colon F(X/R) \to Y$ is the required mediating map: $f' \circ F(q) = f \circ F(s) \circ F(q) = f$. Clearly, f' is unique with this property, because $F(q)$ is an epimorphism (i.e. is surjective). ❑

Proposition 3.3.7 *Let $R \subseteq X \times X$ be a congruence equivalence relation (*i.e. *both a congruence and an equivalence relation) on an algebra $F(X) \to X$. The quotient X/R carries a unique algebra structure $a/R\colon F(X/R) \to X/R$ making the quotient map $q\colon X \to X/R$ a homomorphism of algebras.*

Proof Since R is a quotient we may assume by Theorem 3.3.5 an algebra $c\colon F(R) \to R$, as on the left in

$$\begin{array}{ccccc} F(R) & \overset{F(r_1)}{\underset{F(r_2)}{\rightrightarrows}} & F(X) & \xrightarrow{F(q)} & F(X/R) \\ {\scriptstyle c}\downarrow & & {\scriptstyle a}\downarrow & & \downarrow{\scriptstyle a/R} \\ R & \overset{r_1}{\underset{r_2}{\rightrightarrows}} & X & \xrightarrow[q]{} & X/R \end{array}$$

The algebra structure a/R on the right exists because the top row is a coequaliser by Lemma 3.3.6 and

$$\begin{aligned} q \circ a \circ F(r_1) &= q \circ r_1 \circ c \\ &= q \circ r_2 \circ c \qquad \text{since } q \text{ is coequaliser.} \\ &= q \circ a \circ F(r_2) \end{aligned}$$

❑

The last quotient result is of course well known in universal algebra. Here we obtain it in a uniform manner, for many functors (seen as signatures).

Exercises

3.3.1 Assume a homomorphism of coalgebras $f : X \to Y$ has two factorisations $X \twoheadrightarrow U \rightarrowtail Y$ and $X \twoheadrightarrow V \rightarrowtail Y$. Prove that the diagonal-fill-in property of Theorem 3.3.4.3 yields a unique isomorphism $U \xrightarrow{\cong} V$ commuting with the mono- and epimorphisms and with the coalgebra structures.

3.3.2 Use Theorem 3.3.5 to prove the binary induction proof principle in Theorem 3.1.4: every congruence on an initial algebra is reflexive.

3.3.3 Assume $f : X \to Y$ is an epimorphism in a category **CoAlg**(F), for a Kripke polynomial functor F on **Sets**. Prove that f is the coequaliser in **CoAlg**(F) of its own kernel pair $p_1, p_2 : \mathrm{Ker}(f) \to X$. *Hint*: Use that this property holds in **Sets** and lift it to **CoAlg**(F).

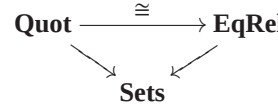
3.3.4 Formally, a **quotient** of an object X in a category is an equivalence class of epimorphisms $X \twoheadrightarrow \bullet$. Two epis $e : X \twoheadrightarrow U$ and $d : X \twoheadrightarrow V$ are equivalent when there is a necessarily unique isomorphism $h : U \to V$ with $h \circ e = d$. As for subobjects – which are equivalence classes of monomorphisms – we often confuse a quotient with an epi that represents it.

We now concentrate on the category **Sets** and form a category **Quot** with

objects	quotients $X \twoheadrightarrow U$
morphisms	from $(X \overset{e}{\twoheadrightarrow} U)$ to $(Y \overset{d}{\twoheadrightarrow} V)$ that are maps $f : X \to Y$ for which there is a necessarily unique map:

$$\begin{array}{ccc} U & \dashrightarrow & V \\ {\scriptstyle e}\uparrow\!\uparrow & & \uparrow\!\uparrow{\scriptstyle d} \\ X & \xrightarrow[f]{} & Y \end{array}$$

Use that quotients are effective in **Sets** to prove that there is an isomorphism of categories:



where **EqRel** $\hookrightarrow$ **Rel** is the subcategory of equivalence relations $R \rightarrowtail X \times X$.

3.4 Bisimulations and the Coinduction Proof Principle

We have already seen that states of final coalgebras coincide with behaviours and that bisimilarity captures observational indistinguishability. Hence the following fundamental result does not come as a surprise: states are bisimilar if and only if they have the same behaviour, i.e. become equal when mapped to the final coalgebra. This insight has been important in the development of the field of coalgebra.

Theorem 3.4.1 ([453, 411]) *Let* $F\colon$ **Sets** $\to$ **Sets** *be a (finite) polynomial functor which has a final coalgebra* $\zeta\colon Z \xrightarrow{\cong} F(Z)$. *Let* $c\colon X \to F(X)$ *and* $d\colon Y \to F(Y)$ *be arbitrary coalgebras, with associated behaviour homomorphisms* $\mathrm{beh}_c\colon X \to Z$ *and* $\mathrm{beh}_d\colon Y \to Z$ *given by finality. Two states* $x \in X$ *and* $y \in Y$ *are then bisimilar if and only if they have the same behaviour:*

$$x \;{}_c\!\underline{\leftrightarrow}_d\; y \iff \mathrm{beh}_c(x) = \mathrm{beh}_d(y).$$

In particular, bisimilarity ${}_\zeta\!\underline{\leftrightarrow}_\zeta \subseteq Z \times Z$ *on the final coalgebra is equality.*

Proof ($\Leftarrow$) This is easy since we know by Proposition 3.2.7.4 that the pullback relation $\mathrm{Eq}(\mathrm{beh}_c, \mathrm{beh}_d) = \{(x,y) \mid \mathrm{beh}_c(x) = \mathrm{beh}_d(y)\} \subseteq X \times Y$ of two homomorphisms is a bisimulation. Hence it is included in the greatest bisimulation ${}_c\!\underline{\leftrightarrow}_d$.

($\Rightarrow$) The bisimilarity relation ${}_c\!\underline{\leftrightarrow}_d$ is itself a bisimulation, so it carries by Theorem 3.3.2 a coalgebra structure $e\colon ({}_c\!\underline{\leftrightarrow}_d) \to F({}_c\!\underline{\leftrightarrow}_d)$ making the two legs r_i of the relation $\langle r_1, r_2\rangle : ({}_c\!\underline{\leftrightarrow}_d) \rightarrowtail X \times Y$ homomorphisms. By finality we then get $\mathrm{beh}_c \circ r_1 = \mathrm{beh}_d \circ r_2$, yielding the required result. ❑

This result gives rise to an important proof method for establishing that two states have the same behaviour. This method is often referred to as the coinduction proof principle and goes back to [354]. It corresponds to

the uniqueness part of the unique existence property of behaviour maps in Definition 2.3.1.

Corollary 3.4.2 (Coinduction proof principle) *Two states have the same behaviour if and only if there is a bisimulation that contains them.*

Consequently, every bisimulation on a final coalgebra is contained in the equality relation. ❑

The second formulation in this corollary is sometimes called 'internal full abstractness'. It is the dual of the binary induction principle in Theorem 3.1.4, stating that on an initial algebra each congruence contains the equality relation (i.e. is reflexive).

As we shall see in Example 3.4.5 below, it may sometimes require a bit of ingenuity to produce an appropriate bisimulation. The standard way to find such a bisimulation is to start with the given equation as relation and close it off with successor states until no new elements appear. In that case one has only 'circularities'. Formalising this approach led to what is sometimes called circular rewriting; see e.g. [163], implemented in the tool CIRC [338].

Corollary 3.4.3 *Call a coalgebra $c\colon X \to F(X)$* ***observable*** *if its bisimilarity relation ${}_c\underline{\leftrightarrow}_c$ is equality on X. This is equivalent to a generalisation of the formulation used in Exercise 2.5.14 for deterministic automata: the associated behaviour map* $\mathrm{beh}_c\colon X \to Z$ *to the final coalgebra* $Z \xrightarrow{\cong} F(Z)$*, if any, is injective.* ❑

Observable coalgebras are called *simple* in [411]. Coalgebras can always be forced to be observable via quotienting; see Exercise 3.4.1 below.

Bisimilarity is closely related to equivalence of automata, expressed in terms of equality of accepted languages (see Corollary 2.3.6). This result, going back to [371], will be illustrated next.

Corollary 3.4.4 *Consider two deterministic automata* $\langle \delta_i, \epsilon_i \rangle\colon S_i \to S_i^A \times \{0,1\}$ *with initial states* $s_i \in S_i$*, for* $i = 1, 2$*. These states* s_1, s_2 *are called* ***equivalent*** *if they accept the same language. The states* s_1 *and* s_2 *are then equivalent if and only if they are bisimilar.*

Proof The accepted languages are given by the behaviours $\mathrm{beh}_{\langle \delta_i, \epsilon_i \rangle}(s_i) \in \mathcal{P}(A^\star)$ of the initial states; see Corollary 2.3.6.2. ❑

Early on in Section 1.2 we already saw examples of coinductive reasoning for sequences. Here we continue to illustrate coinduction with (regular) languages.

Example 3.4.5 (Equality of regular languages [408]) In Corollary 2.3.6.2 we have seen that the set $\mathcal{L}(A) = \mathcal{P}(A^\star)$ of languages over an alphabet A forms a final coalgebra, namely for the deterministic automaton functor $X \mapsto X^A \times \{0, 1\}$. We recall that the relevant coalgebra structure on $\mathcal{L}(A)$ is given on a language $L \subseteq A^\star$ by

$$\begin{array}{ll} L \xrightarrow{a} L_a & \text{where } L_a = \{\sigma \in A^\star \mid a \cdot \sigma \in L\} \text{ is the } a\text{-derivative of } L \\ L\!\downarrow 1 \iff \langle\rangle \in L & \text{which may simply be written as } L\!\downarrow. \end{array}$$

The subset $\mathcal{R}(A) \subseteq \mathcal{L}(A)$ of so-called **regular languages** is built up inductively from constants

$$0 = \emptyset, \qquad 1 = \{\langle\rangle\}, \qquad \{a\}, \text{ for } a \in A, \text{ usually written simply as } a,$$

and the three operations of **union**, **concatenation** and **Kleene star**:

$$\begin{array}{rcl} K + L & = & K \cup L \\ KL & = & \{\sigma \cdot \tau \mid \sigma \in K \wedge \tau \in L\} \\ K^* & = & \bigcup_{n \in \mathbb{N}} K^n, \quad \text{where } K^0 = 1 \text{ and } K^{n+1} = KK^n. \end{array}$$

See also Exercise 3.4.4 below. For example, the regular language $a(a + b)^* b$ consists of all (finite) words consisting of letters a, b only, that start with an a and end with a b. Regular languages can be introduced in various other ways, for example as the languages accepted by deterministic and non-deterministic automata with a finite state space (via what is called Kleene's theorem [294], or [430] for coalgebraic versions), or as the languages generated by regular grammars. Regular languages (or expressions) are used in many situations, such as lexical analysis (as patterns for tokens) or text editing and retrieval (for context searches). Regular expressions (see Exercise 3.4.4) are often used as search strings in a Unix/Linux environment; for example in the command `grep`, for 'general regular expression parser'.

An important topic is proving equality of regular languages. There are several approaches, namely via unpacking the definitions, via algebraic reasoning using a complete set of laws (see [309] and also [246]) or via minimalisation of associated automata. A fourth, coinductive approach is introduced in [408] using bisimulations. It is convenient and will be illustrated here.

Recall from Section 3.1 that a relation $R \subseteq \mathcal{L}(A) \times \mathcal{L}(A)$ is a bisimulation if for all languages L, K,

$$R(L, K) \implies \begin{cases} R(L_a, K_a) & \text{for all } a \in A \\ L\!\downarrow \text{ iff } K\!\downarrow. \end{cases}$$

The coinduction proof principle then says, for $L, K \in \mathcal{L}(A)$,

$$L = K \iff \text{there is a bisimulation } R \subseteq \mathcal{L}(A) \times \mathcal{L}(A) \text{ with } R(L, K). \qquad (3.1)$$

We will use this same principle for the subset $\mathcal{R}(A) \subseteq \mathcal{L}(A)$ of regular languages. This is justified by the fact that the inclusion map $\mathcal{R}(A) \hookrightarrow \mathcal{L}(A)$ is the unique homomorphism of coalgebras obtained by finality in

$$\begin{array}{ccc} \mathcal{R}(A)^A \times 2 & - - - - - \to & \mathcal{L}(A)^A \times 2 \\ \uparrow & & \uparrow \cong \\ \mathcal{R}(A) & - - - - - - - \to & \mathcal{L}(A) = \mathcal{P}(A^\star) \end{array}$$

This works because the coalgebra structure on $\mathcal{L}(A)$ restricts to $\mathcal{R}(A)$. This coalgebra on $\mathcal{R}(A)$ is given by the following rules for (Brzozowski) derivatives L_a and termination $L\!\downarrow$:

$$\begin{array}{rcll} 0_a &=& 0 & \neg(0\!\downarrow) \\ 1_a &=& 0 & 1\!\downarrow \\ b_a &=& \begin{cases} 1 & \text{if } b = a \\ 0 & \text{otherwise} \end{cases} & \neg(b\!\downarrow) \\ (K+L)_a &=& K_a + L_a & K + L\!\downarrow \text{ iff } K\!\downarrow \text{ or } L\!\downarrow \\ (KL)_a &=& \begin{cases} K_a L + L_a & \text{if } K\!\downarrow \\ K_a L & \text{otherwise} \end{cases} & KL\!\downarrow \text{ iff } K\!\downarrow \wedge L\!\downarrow \\ (K^*)_a &=& K_a K^* & K^*\!\downarrow . \end{array}$$

We shall illustrate the use of the coinduction proof principle (3.1) for establishing equality of regular languages via two examples.

1. For an arbitrary element a in the alphabet A one has

$$(1 + a)^* \;=\; a^*.$$

As candidate bisimulation R in (3.1) we take

$$R \;=\; \{((1 + a)^*, a^*)\} \cup \{(0, 0)\}.$$

The termination requirement obviously holds, so we concentrate on derivatives. First, for a itself:

$$((1 + a)^*)_a = (1 + a)_a\,(1 + a)^* = (1_a + a_a)\,(1 + a)^* = (0 + 1)\,(1 + a)^* = (1 + a)^*$$

and similarly

$$(a^*)_a \;=\; a_a\, a^* \;=\; 1\, a^* \;=\; a^*.$$

Hence the pair of a-derivatives $(((1 + a)^*)_a, (a^*)_a) = ((1 + a)^*, a^*)$ is again in the relation R. Likewise, $(0_a, 0_a) = (0, 0) \in R$. And for an element $b \neq a$ we similarly have $(((1 + a)^*)_b, (a^*)_b) = (0, 0) \in R$. This shows that R is a bisimulation, and completes the proof. The reader may wish to compare it with an alternative proof of equality, using the definition of Kleene star $(-)^*$.

2. Next we restrict ourselves to an alphabet $A = \{a, b\}$ consisting of two (different) letters only. Consider the two languages

$$\begin{aligned} E(b) &= \{\sigma \in A^\star \mid \sigma \text{ contains an even number of } bs\} \\ O(b) &= \{\sigma \in A^\star \mid \sigma \text{ contains an odd number of } bs\}. \end{aligned}$$

(We consider $0 \in \mathbb{N}$ to be even.) Using the definitions of derivative and termination, it is not hard to see that

$$\begin{array}{lll} E(b)_a = E(b) & E(b)_b = O(b) & E(b)\downarrow \\ O(b)_a = O(b) & O(b)_b = E(b) & \neg\, O(b)\downarrow. \end{array}$$

Our aim is to prove the equality

$$E(b) = a^* + a^*b(a + ba^*b)^*ba^*$$

via coinduction. This requires by (3.1) a bisimulation R containing both sides of the equation. We take

$$\begin{aligned} R &= \{(E(b), K)\} \cup \{(O(b), L)\} \text{ where} \\ K &= a^* + a^*b(a + ba^*b)^*ba^* \quad \text{(the right-hand side of the equation)} \\ L &= (a + ba^*b)^*ba^*. \end{aligned}$$

The computations that show that R is a bisimulation use the above computation rules for derivatives plus some obvious properties of the regular operations (such as associativity, commutativity, $X + 0 = X$ and $1X = X$):

$$\begin{aligned} K_a &= (a^*)_a + (a^*)_a b(a + ba^*b)^*ba^* + (b(a + ba^*b)^*ba^*)_a \\ &= a^* + a^*b(a + ba^*b)^*ba^* + b_a(a + ba^*b)^*ba^* \\ &= K + 0(a + ba^*b)^*ba^* \\ &= K \\ K_b &= (a^*)_b + (a^*)_b b(a + ba^*b)^*ba^* + (b(a + ba^*b)^*ba^*)_b \\ &= 0 + 0b(a + ba^*b)^*ba^* + b_b(a + ba^*b)^*ba^* \\ &= 0 + 0 + 1(a + ba^*b)^*ba^* \\ &= L \\ L_a &= ((a + ba^*b)^*)_a ba^* + (ba^*)_a \\ &= (a + ba^*b)_a(a + ba^*b)^*ba^* + b_a a^* \\ &= (a_a + b_a a^*b)(a + ba^*b)^*ba^* + 0a^* \\ &= (1 + 0a^*b)(a + ba^*b)^*ba^* + 0 \\ &= L \\ L_b &= ((a + ba^*b)^*)_b ba^* + (ba^*)_b \\ &= (a + ba^*b)_b(a + ba^*b)^*ba^* + b_b a^* \\ &= (a_b + b_b a^*b)(a + ba^*b)^*ba^* + 1a^* \\ &= a^*b(a + ba^*b)^*ba^* + a^* \\ &= K. \end{aligned}$$

This shows that $(U, V) \in R$ implies both $(U_a, V_a) \in R$ and $(U_b, V_b) \in R$. Further:

$$\begin{aligned} K\downarrow &\iff a^*\downarrow \ \vee \ a^*b(a + ba^*b)^*ba^*\downarrow \\ &\iff \text{true} \\ L\downarrow &\iff (a + ba^*b)^*\downarrow \ \wedge \ ba^*\downarrow \\ &\iff \text{true} \ \wedge \ b\downarrow \ \wedge \ a^*\downarrow \\ &\iff \text{true} \ \wedge \ \text{false} \ \wedge \ \text{true} \\ &\iff \text{false}. \end{aligned}$$

This shows that R is a bisimulation. As a result we obtain $E(b) = K$, as required, but also $O(b) = L$.

This concludes the example. For more information, see [408, 410, 412].

There are many more examples of coinductive reasoning in the literature, in various areas: non-well-founded sets [9, 60], processes [355], functional programs [172], streams [412, 215] (with analytic functions as special case [376]), datatypes [214], domains [133, 380] etc.

Exercises

3.4.1 Check that for an arbitrary coalgebra $c\colon X \to F(X)$ of a Kripke polynomial functor, the induced quotient coalgebra $c/\underline{\leftrightarrow} : X/\underline{\leftrightarrow} \to F(X/\underline{\leftrightarrow})$ is observable – using both Theorem 3.3.4.1 and Proposition 3.2.8.3. Is the mapping $c \mapsto c/\underline{\leftrightarrow}$ functorial?

Note that since the canonical map $[-]\colon X \twoheadrightarrow X/\underline{\leftrightarrow}$ is a homomorphism, its graph is a bisimulation. Hence a state $x \in X$ is bisimilar to its equivalence class $[x] \in X/\underline{\leftrightarrow}$. This means that making a coalgebra observable does not change the behaviour.

3.4.2
1. ([411, theorem 8.1]) Prove that a coalgebra c is observable (or simple) if and only if it has no proper quotients: every epimorphism $c \twoheadrightarrow d$ is an isomorphism. *Hint*: Consider the kernel of such a map.
2. Conclude that there is at most one homomorphism to an observable coalgebra.

3.4.3 Prove the following analogue of Theorem 3.4.1 for two arbitrary algebras $a\colon F(X) \to X$ and $b\colon F(Y) \to Y$ of a Kripke polynomial functor F, with an initial algebra $F(A) \xrightarrow{\cong} A$.

> Two elements $x \in X$ and $y \in Y$ are interpretations $x = \mathrm{int}_a(t)$ and $y = \mathrm{int}_b(t)$ of the same element $t \in A$ if and only if the pair (x, y) is in each congruence relation $R \subseteq X \times Y$.

Conclude that for coalgebras c, d and algebras a, b:

$$\begin{aligned}\mathrm{Eq}(\mathrm{beh}_c, \mathrm{beh}_d) &\stackrel{\text{def}}{=} (\mathrm{beh}_c \times \mathrm{beh}_d)^{-1}(\mathrm{Eq}) \\ &= \bigcup\{R \mid R \text{ is a bisimulation on the carriers of } c, d\} \\ \mathrm{Im}(\langle \mathrm{int}_a, \mathrm{int}_b\rangle) &\stackrel{\text{def}}{=} \coprod_{\mathrm{int}_a \times \mathrm{int}_b}(\mathrm{Eq}) \\ &= \bigcap\{R \mid R \text{ is a congruence on the carriers of } a, b\}.\end{aligned}$$

3.4.4 Fix an alphabet A and consider the simple polynomial functor

$$\mathcal{R}(X) = 1 + 1 + A + (X \times X) + (X \times X) + X.$$

1. Show that the initial algebra RE of $\mathcal{R}$ is the set of **regular expressions**, given by the BNF syntax:

$$E ::= 0 \mid 1 \mid a \mid E + E \mid EE \mid E^*$$

where $a \in A$.

2. Define an interpretation map $\mathrm{int}\colon RE \to \mathcal{P}(A^\star) = \mathcal{L}(A)$ by initiality, whose image contains precisely the regular languages. In order to do so one needs an $\mathcal{R}$-algebra structure on the final coalgebra $\mathcal{L}(A)$; see also [246].

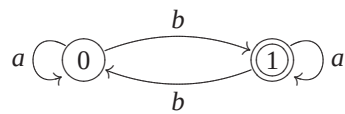
3.4.5 Use coinduction to prove the equation

$$(a + b)^* = (a^*b)^*a^* \qquad \text{for alphabet } A = \{a, b, c\}.$$

3.4.6 (From [408]) Prove the following equality of regular languages (over the alphabet $\{a, b\}$) by coinduction:

$$((b^*a)^*ab^*)^* = 1 + a(a + b)^* + (a + b)^*aa(a + b)^*.$$

3.4.7 Prove that the language $K = a^* + a^*b(a + ba^*b)^*ba^*$ of words with an even numbers of bs from Example 3.4.5 is the language that is accepted by the following finite deterministic automaton



with 1 both as initial and as final state. More formally, this automaton is $\langle \delta, \epsilon\rangle\colon \{0, 1\} \to \{0, 1\}^{\{a,b\}} \times 2$ with

$$\begin{aligned}\delta(0)(a) &= 0 & \delta(0)(b) &= 1 & \epsilon(0) &= 0 \\ \delta(1)(a) &= 1 & \delta(1)(b) &= 0 & \epsilon(1) &= 1.\end{aligned}$$

3.5 Process Semantics

This section will introduce a semantics for processes using final coalgebras for the finite powerset functor $\mathcal{P}_{\text{fin}}$. They capture the behaviour of so-called finitely branching transition systems. This section forms an illustration of many of the ideas we have seen so far, such as behavioural interpretations via finality and compositional interpretations via initiality. Also, we shall see how the coalgebraic notion of bisimilarity forms a congruence – an algebraic notion. The material in this section builds on [453, 417], going back to [407]. It will be put in a broader context via distributive laws in Section 5.5.

A first, non-trivial question is: what is a process? Usually one understands it as a running program. Thus, a sequential program, transforming input to output, when in operation forms a process. But typical examples of processes are programs that are meant to be running 'forever', such as operating systems or controllers. Often they consist of several processes that run in parallel, with appropriate synchronisation between them. The proper way to describe such processes is not via input-output relations, as for sequential programs. Rather, one looks at their behaviour, represented as suitable (infinite) trees.

Let us start with the kind of example that is often used to introduce processes. Suppose we wish to describe a machine that can change 5- and 10-euro notes into 1- and 2-euro coins. We shall simply use '5' and '10' as input labels. And as output labels we use pairs (i, j) to describe the return of i 2-euro coins and j 1-euro coins. Also there is a special output action empty that indicates that the machine does not have enough coins left. Our abstract description will not determine which combination of coins is returned, but only gives the various options as a non-deterministic choice. Pictorially this yields a 'state-transition' diagram as in Figure 3.1. Notice that the machine can make a '5' or '10' transition only if it can return a corresponding change. Otherwise, it can do only an 'empty' step.

In this section we shall describe such transition systems as coalgebras, in particular as coalgebras of the functor $X \mapsto \mathcal{P}_{\text{fin}}(X)^A$, for various sets A of 'labels' or 'actions'. As usual, for states $s, s' \in S$ and actions $a \in A$ we write $s \xrightarrow{a} s'$ for $s' \in c(s)(a)$, where $c\colon S \to \mathcal{P}_{\text{fin}}(S)^A$ is our coalgebra. Note that for each state $s \in S$ and input $a \in A$ there are only finitely many successor states s' with $s \xrightarrow{a} s'$. Therefore, such transition systems are often called **finitely branching**.

In the example of Figure 3.1, the set of labels is

$$E = \{5, 10, (2,1), (1,3), (0,5), (5,0), (4,2), (3,4), (2,6), (1,8), (0,10), \mathsf{empty}\}.$$

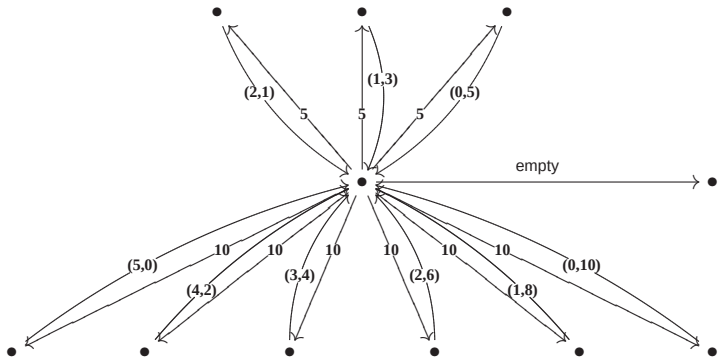


Figure 3.1 Transition diagram of a machine for changing 5- and 10-euro notes into coins.

And the set of states is

$$S = \{s_0, s_1, s_2, s_3, s_4, s_5, s_6, s_7, s_8, s_9, s_{10}\},$$

with 'change' coalgebra structure $\mathsf{ch}\colon S \to \mathcal{P}_{\mathrm{fin}}(S)^E$ given by

$$\begin{aligned}
\mathsf{ch}(s_0) &= \lambda a \in E.\begin{cases} \{s_1, s_2, s_3\} & \text{if } a = 5 \\ \{s_4, s_5, s_6, s_7, s_8, s_9\} & \text{if } a = 10 \\ \{s_{10}\} & \text{if } a = \mathsf{empty} \\ \emptyset & \text{otherwise} \end{cases} \\
\mathsf{ch}(s_1) &= \lambda a \in E.\begin{cases} \{s_0\} & \text{if } a = (2,1) \\ \emptyset & \text{otherwise} \end{cases} \\
\mathsf{ch}(s_2) &= \lambda a \in E.\begin{cases} \{s_0\} & \text{if } a = (1,3) \\ \emptyset & \text{otherwise} \end{cases} \\
&\text{etc.}
\end{aligned} \tag{3.2}$$

Since the functor $X \mapsto \mathcal{P}_{\mathrm{fin}}(X)^A$ is a finite Kripke polynomial functor, we know by Theorem 2.3.9 that it has a final coalgebra. In this section we shall write this final coalgebra as

$$Z_A \xrightarrow[\cong]{\zeta_A} \mathcal{P}_{\mathrm{fin}}(Z_A)^A, \tag{3.3}$$

where the set of inputs A is a parameter. We do not really care what these sets Z_A actually look like, because we shall use only their universal properties and do not wish to depend on a concrete representation. However, concrete descriptions in terms of certain finitely branching trees modulo bisimilarity may be given; see [453, section 4.3] (following [55]). In this context we shall call the elements of carrier Z_A of the final coalgebra **processes**.

Our change machine coalgebra ch from Figure 3.1 with its set of actions E thus gives rise to a behaviour map:

$$\begin{array}{ccc} \mathcal{P}_{\text{fin}}(S)^E & \xrightarrow{\mathcal{P}_{\text{fin}}(\text{beh}_{\mathsf{ch}})^E} & \mathcal{P}_{\text{fin}}(Z_E)^E \\ {\scriptstyle \mathsf{ch}}\uparrow & \cong & \uparrow{\scriptstyle \zeta_E} \\ S & \xrightarrow[\text{beh}_{\mathsf{ch}}]{} & Z_E \end{array} \tag{3.4}$$

The behaviour function beh_{ch} turns the concrete states $s_0, \ldots, s_{10}$ of our change machine into abstract states of the final coalgebra Z_E – i.e. into processes – with the same behaviour.

3.5.1 Process Descriptions

Several languages have been proposed in the literature to capture processes, such as algebra of communicating processes (ACP) [67, 139], calculus of communicating systems (CCS) [354, 355] or communicating sequential processes (CSP) [223]. The goal of such languages is to study processes via axiom systems in which notions such as sequential and parallel execution, alternative choice and communication are formalised by means of algebraic operations and equations. It is not our aim to go into precise syntax and into the various differences between these formalisms. Instead we concentrate on the semantics and describe only a few basic operations on processes, showing how they can be interpreted in a final coalgebra of the form (3.3). As an example, the above change machine could be described via a (recursive) process equation:

$$\begin{aligned} \mathsf{CH} = {} & 5 \cdot (2,1) \cdot \mathsf{CH} + 5 \cdot (1,3) \cdot \mathsf{CH} + 5 \cdot (0,5) \cdot \mathsf{CH} + \\ & 10 \cdot (5,0) \cdot \mathsf{CH} + 10 \cdot (4,2) \cdot \mathsf{CH} + 10 \cdot (3,4) \cdot \mathsf{CH} + \\ & 10 \cdot (2,6) \cdot \mathsf{CH} + 10 \cdot (1,8) \cdot \mathsf{CH} + 10 \cdot (0,10) \cdot \mathsf{CH} + \\ & \mathsf{empty} \cdot 0. \end{aligned} \tag{3.5}$$

Process algebras can be understood more generally as providing a convenient syntax for describing various kinds of transition systems; see Example 3.5.1 below.

In the following we fix an arbitrary set A of actions, and we consider the associated final coalgebra $Z_A \xrightarrow{\cong} \mathcal{P}_{\text{fin}}(Z_A)^A$ as in (3.3). We shall describe a collection of operations (such as + and $\cdot$ above) on processes, understood as elements of Z_A.

The Null Process

One can define a trivial process which does nothing. It is commonly denoted as 0 and is defined as

$$0 \stackrel{\text{def}}{=} \zeta_A^{-1}(\lambda a \in A.\, \emptyset).$$

This means that there are no successor states of the process $0 \in Z_A$ – since by construction the set of successors $\zeta_A(0)(a)$ is empty, for each label $a \in A$.

Sum of Two Processes

Given two processes $z_1, z_2 \in Z_A$, one can define a sum process $z_1 + z_2 \in Z_A$ via the union operation $\cup$ on subsets:

$$z_1 + z_2 \stackrel{\text{def}}{=} \zeta_A^{-1}(\lambda a \in A.\, \zeta_A(z_1)(a) \cup \zeta_A(z_2)(a)).$$

Then:

$$\begin{aligned}(z_1 + z_2) \stackrel{a}{\longrightarrow} w &\iff w \in \zeta_A(z_1 + z_2)(a) \\ &\iff w \in \zeta_A(z_1)(a) \cup \zeta_A(z_2)(a) \\ &\iff w \in \zeta_A(z_1)(a) \ \text{ or } \ w \in \zeta_A(z_2)(a) \\ &\iff z_1 \stackrel{a}{\longrightarrow} w \ \text{ or } \ z_2 \stackrel{a}{\longrightarrow} w.\end{aligned}$$

This means that there is an a-transition out of the sum process $z_1 + z_2$ if and only if there is an a-transition out of either z_1 or z_2. In the process literature one usually encounters the following two rules:

$$\frac{z_1 \stackrel{a}{\longrightarrow} w}{z_1 + z_2 \stackrel{a}{\longrightarrow} w} \qquad \frac{z_2 \stackrel{a}{\longrightarrow} w}{z_1 + z_2 \stackrel{a}{\longrightarrow} w}. \tag{3.6}$$

It is not hard to see that the structure $(Z_A, +, 0)$ is a commutative monoid with idempotent operation: $z + z = z$ for all $z \in Z_A$.

Prefixing of Actions

Given a process $z \in Z_A$ and an action $b \in A$ there is a process $b \cdot z$ which first performs b and then continues with z. It can be defined as

$$\begin{aligned}b \cdot z &\stackrel{\text{def}}{=} \zeta_A^{-1}(\lambda a \in A.\, \text{if } a = b \text{ then } \{z\} \text{ else } \emptyset) \\ &= \zeta_A^{-1}(\lambda a \in A.\, \{z \mid a = b\}).\end{aligned}$$

We then have, for $w \in Z_A$ and $b \in A$,

$$\begin{aligned}(b \cdot z) \stackrel{a}{\longrightarrow} w &\iff w \in \zeta_A(b \cdot z)(a) \\ &\iff w \in \{z \mid a = b\} \\ &\iff a = b \wedge w = z.\end{aligned}$$

This gives the standard rule:

$$\frac{}{b \cdot z \xrightarrow{\ b\ } z}\ . \tag{3.7}$$

Example 3.5.1 (Change machine, continued) Having defined prefixing and sum we can verify the validity of the change machine equation (3.5), for the interpretation $\mathsf{CH} = \mathrm{beh}_{\mathsf{ch}}(s_0) \in Z_E$ from (3.2). First we note that

$$\begin{aligned}
(2,1) \cdot \mathsf{CH} &= \zeta_E^{-1}\big(\lambda a \in E.\, (\mathsf{if}\ a = (2,1)\ \mathsf{then}\ \{\mathrm{beh}_{\mathsf{ch}}(s_0)\}\ \mathsf{else}\ \emptyset)\big) \\
&= \zeta_E^{-1}\big(\lambda a \in E.\, \mathcal{P}_{\mathrm{fin}}(\mathrm{beh}_{\mathsf{ch}})(\mathsf{if}\ a = (2,1)\ \mathsf{then}\ \{s_0\}\ \mathsf{else}\ \emptyset)\big) \\
&= \zeta_E^{-1}\big(\lambda a \in E.\, \mathcal{P}_{\mathrm{fin}}(\mathrm{beh}_{\mathsf{ch}})(\mathsf{ch}(s_1)(a))\big) \\
&= \zeta_E^{-1}\big(\lambda a \in E.\, \zeta_E(\mathrm{beh}_{\mathsf{ch}}(s_1))(a)\big) \\
&\qquad\qquad \text{since } \mathrm{beh}_{\mathsf{ch}} \text{ is a map of coalgebras in (3.4)} \\
&= \zeta_E^{-1}\big(\zeta_E(\mathrm{beh}_{\mathsf{ch}}(s_1))\big) \\
&= \mathrm{beh}_{\mathsf{ch}}(s_1).
\end{aligned}$$

Similar equations can be derived for the states $s_2, \ldots, s_9$. And for s_{10} we have

$$\begin{aligned}
\mathrm{beh}_{\mathsf{ch}}(s_{10}) &= \zeta_E^{-1}\big(\lambda a \in E.\, \zeta_E(\mathrm{beh}_{\mathsf{ch}}(s_{10}))(a)\big) \\
&= \zeta_E^{-1}\big(\lambda a \in E.\, \mathcal{P}_{\mathrm{fin}}(\mathrm{beh}_{\mathsf{ch}})(\mathsf{ch}(s_{10})(a))\big) \\
&= \zeta_E^{-1}\big(\lambda a \in E.\, \mathcal{P}_{\mathrm{fin}}(\mathrm{beh}_{\mathsf{ch}})(\emptyset)\big) \\
&= \zeta_E^{-1}\big(\lambda a \in E.\, \emptyset\big) \\
&= \mathsf{0}.
\end{aligned}$$

Finally we can check that Equation (3.5) holds for the behaviour $\mathsf{CH} = \mathrm{beh}_{\mathsf{ch}}(s_0)$. The calculation is described in Figure 3.2, for each action $a \in E$.

This example illustrates an approach for verifying that a transition system on an arbitrary state space satisfies a certain process equation:

1. Map the transition system to the final coalgebra via the behaviour map beh.
2. Check the equation for $\mathrm{beh}(s_0)$, where s_0 is a suitable (initial) state, using the above interpretations of the process combinators $+$, $\mathsf{0}$, $\cdot$ etc. on the final coalgebra.

3.5.2 A Simple Process Algebra

In the previous subsection we have seen several process combinators, described as functions on a terminal coalgebra $Z_A \xrightarrow{\cong} \mathcal{P}_{\mathrm{fin}}(Z_A)^A$. Next we shall consider these basic combinators as constructors of a very simple process language, often called basic process algebra (BPA); see [139, 451]. In the spirit of this text, the language of finite terms will be described as an initial algebra.

$$
\begin{aligned}
\zeta_E(\mathsf{CH})(a) &= \zeta_E(\mathrm{beh}_{\mathsf{ch}}(s_{10}))(a) \\
&= \mathcal{P}_{\mathrm{fin}}(\mathrm{beh}_{\mathsf{ch}})(\mathsf{ch}(s_0)(a)) \qquad \text{by (3.4)} \\
&= \begin{cases} \{\mathrm{beh}_{\mathsf{ch}}(s_1), \mathrm{beh}_{\mathsf{ch}}(s_2), \mathrm{beh}_{\mathsf{ch}}(s_3)\} & \text{if } a = 5 \\ \{\mathrm{beh}_{\mathsf{ch}}(s_4), \mathrm{beh}_{\mathsf{ch}}(s_5), \mathrm{beh}_{\mathsf{ch}}(s_6), \mathrm{beh}_{\mathsf{ch}}(s_7), \mathrm{beh}_{\mathsf{ch}}(s_8), \mathrm{beh}_{\mathsf{ch}}(s_9)\} & \text{if } a = 10 \\ \{\mathrm{beh}_{\mathsf{ch}}(s_{10})\} & \text{if } a = \mathsf{empty} \end{cases} \\
&= \{\, \mathrm{beh}_{\mathsf{ch}}(s_1), \mathrm{beh}_{\mathsf{ch}}(s_2), \mathrm{beh}_{\mathsf{ch}}(s_3) \mid a = 5\} \\
&\qquad \cup \{\, \mathrm{beh}_{\mathsf{ch}}(s_4), \mathrm{beh}_{\mathsf{ch}}(s_5), \mathrm{beh}_{\mathsf{ch}}(s_6), \mathrm{beh}_{\mathsf{ch}}(s_7), \mathrm{beh}_{\mathsf{ch}}(s_8), \mathrm{beh}_{\mathsf{ch}}(s_9) \mid a = 10\} \\
&\qquad \cup \{\, \mathsf{ch}(s_{10}) \mid a = \mathsf{empty}\} \\
&= \{\, \mathrm{beh}_{\mathsf{ch}}(s_1) \mid a = 5\} \cup \{\, \mathrm{beh}_{\mathsf{ch}}(s_2) \mid a = 5\} \cup \{\, \mathrm{beh}_{\mathsf{ch}}(s_3) \mid a = 5\} \\
&\qquad \cup \{\, \mathrm{beh}_{\mathsf{ch}}(s_4) \mid a = 10\} \cup \{\, \mathrm{beh}_{\mathsf{ch}}(s_5) \mid a = 10\} \cup \{\, \mathrm{beh}_{\mathsf{ch}}(s_6) \mid a = 10\} \\
&\qquad \cup \{\, \mathrm{beh}_{\mathsf{ch}}(s_7) \mid a = 10\} \cup \{\, \mathrm{beh}_{\mathsf{ch}}(s_8) \mid a = 10\} \cup \{\, \mathrm{beh}_{\mathsf{ch}}(s_9) \mid a = 10\} \cup \{\, \mathsf{ch}(s_{10}) \mid a = \mathsf{empty}\} \\
&= \zeta_E(5 \cdot (2,1) \cdot \mathsf{CH})(a) \cup \zeta_E(5 \cdot (1,3) \cdot \mathsf{CH})(a) \cup \zeta_E(5 \cdot (0,5) \cdot \mathsf{CH})(a) \\
&\qquad \cup \zeta_E(10 \cdot (5,0) \cdot \mathsf{CH})(a) \cup \zeta_E(10 \cdot (4,2) \cdot \mathsf{CH})(a) \cup \zeta_E(10 \cdot (3,4) \cdot \mathsf{CH})(a) \\
&\qquad \cup \zeta_E(10 \cdot (2,6) \cdot \mathsf{CH})(a) \cup \zeta_E(10 \cdot (1,8) \cdot \mathsf{CH})(a) \cup \zeta_E(10 \cdot (0,10) \cdot \mathsf{CH})(a) \cup \zeta_E(\mathsf{empty} \cdot 0)(a) \\
&= \zeta_E\Big(5 \cdot (2,1) \cdot \mathsf{CH} + 5 \cdot (1,3) \cdot \mathsf{CH} + 5 \cdot (0,5) \cdot \mathsf{CH} \\
&\qquad +10 \cdot (5,0) \cdot \mathsf{CH} + 10 \cdot (4,2) \cdot \mathsf{CH} + 10 \cdot (3,4) \cdot \mathsf{CH} + \\
&\qquad 10 \cdot (2,6) \cdot \mathsf{CH} + 10 \cdot (1,8) \cdot \mathsf{CH} + 10 \cdot (0,10) \cdot \mathsf{CH} + \mathsf{empty} \cdot 0\Big)(a).
\end{aligned}
$$

Figure 3.2 Calculation of the behaviour of the process CH.

For an arbitrary set A of actions, consider the simple polynomial functor $\Sigma_A \colon \mathbf{Sets} \to \mathbf{Sets}$ given by

$$\Sigma_A(X) \;=\; 1 + (A \times X) + (X \times X).$$

An algebra $\Sigma_A(X) \to X$ for this functor thus consists of three operations, which we call $0 \colon 1 \to X$ for null-process, $\cdot \colon A \times X \to X$ for prefixing and $+ \colon X \times X \to X$ for sum. We have seen that the final coalgebra $Z_A \xrightarrow{\cong} \mathcal{P}_{\mathrm{fin}}(Z_A)^A$ carries a Σ_A-algebra structure which we shall write as $\xi_A \colon \Sigma_A(Z_A) \to Z_A$. It is given by the structure described earlier (before Example 3.5.1). Thus we have a bialgebra of processes:

$$\Sigma_A(Z_A) \xrightarrow{\;\xi_A\;} Z_A \xrightarrow[\cong]{\;\zeta_A\;} \mathcal{P}_{\mathrm{fin}}(Z_A)^A .$$

The free Σ_A-algebra on a set V consists of the terms built up from the elements from V as variables. An interesting algebra is given by the free Σ_A-algebra on the final coalgebra Z_A. It consists of terms built out of processes, as studied in [407] under the phrase 'processes as terms'. Here we shall write P_A for the initial Σ_A-algebra, i.e. the free algebra on the empty set. The set P_A contains the 'closed' process terms, without free variables. They are built up from 0 and $a \in A$. We shall write this algebra as $\alpha \colon \Sigma_A(P_A) \xrightarrow{\cong} P_A$. It is not hard to see that the set P_A of process terms also carries a bialgebra structure:

$$\Sigma_A(P_A) \xrightarrow[\cong]{\;\alpha\;} P_A \xrightarrow{\;\beta\;} \mathcal{P}_{\mathrm{fin}}(P_A)^A .$$

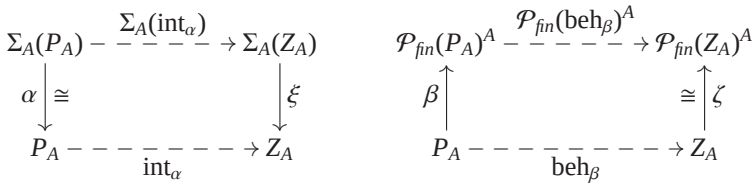
The coalgebra β is defined by induction, following the transition rules (3.6) and (3.7). For each $a \in A$,

$$\begin{aligned} \beta(0)(a) &= \emptyset \\ \beta(b \cdot s)(a) &= \{s \mid b = a\} \\ \beta(s + t)(a) &= \beta(s)(a) \cup \beta(t)(a). \end{aligned}$$

These definitions form a very simple example of a structural operations semantics (SOS): operational behaviour defined by induction on the structure of terms.

The next result shows that the denotational semantics given by initiality and operational semantics given by finality for process terms coincide.

Proposition 3.5.2 *In the above situation we obtain two maps $P_A \to Z_A$, one by initiality and one by finality:*



These two maps are equal, so that $\mathrm{int}_\alpha = \mathrm{beh}_\beta \colon P_A \to Z_A$ is a 'map of bialgebras' commuting with both the algebra and coalgebra structures. This proves in particular that the behavioural semantics beh_β of processes is compositional: it commutes with the term-forming operations.

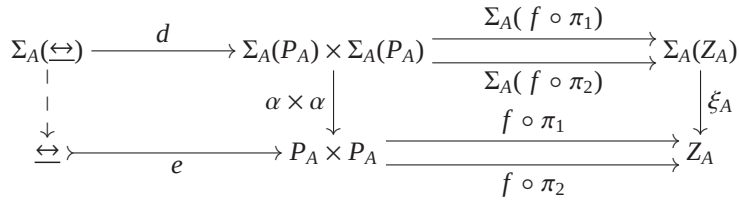
Proof By induction on the structure of a term $s \in P_A$ we prove that $\mathrm{beh}_\beta(s) = \mathrm{int}_\alpha(s)$, or equivalently, $\zeta(\mathrm{beh}_\beta(s))(a) = \zeta(\mathrm{int}_\alpha(s))(a)$, for all $a \in A$:

$$\begin{aligned}
\zeta(\mathrm{beh}_\beta(0))(a) &= \mathcal{P}_{\mathrm{fin}}(\mathrm{beh}_\beta)^A(\beta(0))(a) \\
&= \mathcal{P}_{\mathrm{fin}}(\mathrm{beh}_\beta)(\beta(0)(a)) \\
&= \mathcal{P}_{\mathrm{fin}}(\mathrm{beh}_\beta)(\emptyset) \\
&= \emptyset \\
&= \zeta(0)(a) \\
&= \zeta(\mathrm{int}_\alpha(0))(a) \\
\zeta(\mathrm{beh}_\beta(b \cdot s))(a) &= \mathcal{P}_{\mathrm{fin}}(\mathrm{beh}_\beta)(\beta(b \cdot s)(a)) \\
&= \mathcal{P}_{\mathrm{fin}}(\mathrm{beh}_\beta)(\{s \mid b = a\}) \\
&= \{\mathrm{beh}_\beta(s) \mid b = a\}) \\
&\overset{\text{(IH)}}{=} \{\mathrm{int}_\alpha(s) \mid b = a\}) \\
&= \zeta(b \cdot \mathrm{int}_\alpha(s))(a) \\
&= \zeta(\mathrm{int}_\alpha(b \cdot s))(a) \\
\zeta(\mathrm{beh}_\beta(s + t))(a) &= \mathcal{P}_{\mathrm{fin}}(\mathrm{beh}_\beta)(\beta(s + t)(a)) \\
&= \mathcal{P}_{\mathrm{fin}}(\mathrm{beh}_\beta)(\beta(s)(a) \cup \beta(t)(a)) \\
&= \mathcal{P}_{\mathrm{fin}}(\mathrm{beh}_\beta)(\beta(s)(a)) \cup \mathcal{P}_{\mathrm{fin}}(\mathrm{beh}_\beta)(\beta(t)(a)) \\
&= \zeta(\mathrm{beh}_\beta(s))(a) \cup \zeta(\mathrm{beh}_\beta(t))(a) \\
&\overset{\text{(IH)}}{=} \zeta(\mathrm{int}_\alpha(s))(a) \cup \zeta(\mathrm{int}_\alpha(t))(a) \\
&= \zeta(\mathrm{int}_\alpha(s) + \mathrm{int}_\alpha(t))(a) \\
&= \zeta(\mathrm{int}_\alpha(s + t))(a).
\end{aligned}$$

❑

Proposition 3.5.3 *Still in the above situation, the bisimilarity relation $\leftrightarrows$ on the set P_A of process terms is a congruence.*

Proof Consider the following diagram, where we abbreviate $f = \mathrm{beh}_\beta = \mathrm{int}_\alpha$:



The map e is the equaliser of $f \circ \pi_1$ and $f \circ \pi_2$, using Theorem 3.4.1. The map d is the pair $\langle \Sigma_A(\pi_1 \circ e), \Sigma(\pi_2 \circ e)\rangle$. We show that $(\alpha \times \alpha) \circ d$ equalises $f \circ \pi_1, f \circ \pi_2$. The dashed arrow then exists by the universal property of the equaliser e, making $\leftrightarrows$ a congruence (see Theorem 3.3.5). Thus, what remains is

$$\begin{aligned}
& f \circ \pi_1 \circ (\alpha \times \alpha) \circ d \\
&= \mathrm{int}_\alpha \circ \alpha \circ \pi_1 \circ d \\
&= \xi \circ \Sigma_A(\mathrm{int}_\alpha) \circ \Sigma_A(\pi_1 \circ e) \\
&= \xi \circ \Sigma_A(\mathrm{int}_\alpha) \circ \Sigma_A(\pi_2 \circ e) \qquad \text{since } e \text{ is equaliser and } \mathrm{beh}_\beta = \mathrm{int}_\alpha \\
&= \mathrm{int}_\alpha \circ \alpha \circ \pi_2 \circ d \\
&= f \circ \pi_2 \circ (\alpha \times \alpha) \circ d.
\end{aligned}$$

❑

This result shows that $s \leftrightarrows s'$ and $t \leftrightarrows t'$ implies $a \cdot s \leftrightarrows a \cdot s'$ and $s + t \leftrightarrows s' + t'$. Such congruence results are fundamental in process algebra, because they show that the algebraic operations for process formation preserve indistinguishability of behaviour. Later, in Section 5.5, this topic will be studied in greater generality (following [452, 451, 58, 298]). A more abstract view on the structure we find on the final coalgebra Z_A is elaborated in [207, 202, 204], where it is shown that 'outer' structure that can be formulated on coalgebras is mimicked by 'inner' structure on the final coalgebra.

The toy example (3.5) of this section illustrates the close connection between final coalgebras and process languages. This connection is further elaborated in [430] (and [428]) where locally final coalgebras of polynomial functors are described syntactically.

Exercises

3.5.1 Complete the definition of the coalgebra $\mathsf{ch}\colon S \to \mathcal{P}_{\mathrm{fin}}(S)^E$ in the beginning of this section.

3.5.2 Prove that $(Z_A, +, 0)$ is indeed a commutative monoid, as claimed above.

3.5.3 One can consider each action $a \in A$ as a process

$$\widehat{a} \stackrel{\text{def}}{=} a \cdot 0 \in Z_A.$$

It can do only an a-transition. Prove that this yields an injection $A \hookrightarrow Z_A$.

3.5.4 Consider the following alternative process equation for a euro change machine.

$$\begin{aligned} \mathsf{CH}' \;=\; & 5 \cdot \big((2,1) \cdot \mathsf{CH}' + (1,3) \cdot \mathsf{CH}' + (0,5) \cdot \mathsf{CH}'\big) \\ & + 10 \cdot \big((5,0) \cdot \mathsf{CH}' + (4,2) \cdot \mathsf{CH}' + (3,4) \cdot \mathsf{CH}' + \\ & \qquad (2,6) \cdot \mathsf{CH}' + (1,8) \cdot \mathsf{CH}' + (0,10) \cdot \mathsf{CH}'\big) \\ & + \mathsf{empty}. \end{aligned}$$

Examine the difference between CH in (3.5) and this CH', for instance by describing a suitable transition system which forms a model of this equation.

Are CH and CH' bisimilar?