An Introduction to the Theory of Coalgebras

Dirk Pattinson
Institut für Informatik, LMU München

Contents

Preface

In these notes, we give an introduction to the theory of coalgebras.

Our claim is, that coalgebras, for an endofunctor on the category of sets and functions, provide a general framework, which allows to model a vast range of state based systems. In the first chapter, we substantiate this claim by discussing several examples, from which we derive the general definitions. The second chapter presents various structure theoretic results and lays the foundations for the applications in the subsequent chapters.

We then discuss two applications of the general theory of coalgebras in detail: definitions and proofs by coinduction, and modal logics for coalgebras.

The choice of topics is of course subjective and biased towards the authors own interests. There are many other topics, which could not be included in these notes. Among the omitted material are: The study of expressivity for coalgebraic logics on a model class level, which provides co-Birkhoff type theorems [5, 23, 31]. Also, we have not discussed applications to coalgebraic specification [17, 14, 50, 40]. Coalgebras have also been used as models for object oriented programs [47, 26]. The structure theory for coalgebras we have developed is relative to the base category of sets and functions; a categorical account of structure present in categories of coalgebras can be found in [46, 61, 30]. Another branch which we have regretfully not included is the use of coalgebras in semantics of programming [57, 56]. mention the use of

Chapter 1

Introduction

1.1 State Based Systems

This section introduces the concept of coalgebras, which we understand as state based systems, by means of several examples. The main purpose is to convince the reader that the notion of coalgebra captures a large class of systems, and to provide enough illustrative material, on which the reader can use the concepts and ideas presented in later chapters. Before going into concrete examples, we clarify, in informal terms, what what we mean when talking about "state based systems".

1.1.1 Informal Definition

One of the claims of the theory of coalgebras is, that many different types of state based systems arise naturally as coalgebras. In general, state based systems meet the following criteria:

(i) The behaviour of the system depends on an internal state, which is invisible for the user of the system.

(ii) The system is reactive, i.e. not necessarily terminating, and interacts with its environment.

(iii) The system comes with a set of operations, through which this interaction takes place.

This is, of course, not a formal definition. We will argue in these notes, that the notion of coalgebra for a functor is an adequate formalisation of state based systems. Before going into the theory of coalgebras, we give some concrete examples of state based systems.

1.1.2 Stream Automata

Stream automata are a particularly simple class of systems, which have a display, and a button. In the display we can see a data element d (which belongs to some fixed set D of data). Whenever we push the button, the system updates its display, and we can see a (not necessarily new) data element $d' \in D$. Hence a stream automaton with a set S of internal states can be described by a pair of functions, which determine the observable data element and the successor state, respectively. Formally, we have

Definition 1.1.1. A *stream automaton* over a set D (of data) is a triple $(S, \mathsf{hd}, \mathsf{tl})$ where S is a set (the state space), and $\mathsf{hd} : S \to D$, $\mathsf{tl} : S \to S$ are functions.

The name "stream automaton" refers to the fact that, given some internal state $s \in S$, we can observe an infinite stream of data elements $(\mathsf{hd}(s), \mathsf{hd} \circ \mathsf{tl}(s), \mathsf{hd} \circ \mathsf{tl} \circ \mathsf{tl}(s), \dots)$. Here hd stands for "head", producing the first element of the stream and tl is short for "tail". We can combine these two functions into one by using cartesian products:

Notation 1.1.2. If A, B are sets, we denote their *cartesian product* $\{(a, b) \mid a \in A, b \in B$ by $A \times B$. If C is another set and $f : C \to A$, $g : C \to B$ are functions, we define $\langle f, g\rangle : C \to A \times B$ by $\langle f, g\rangle(c) = (f(a), g(a))$. Projections are denoted by $\pi_1 : A \times B \to A$ (resp. $\pi_2 : A \times B \to B$), where $\pi_1(a, b) = a$ (resp. $\pi_2(a, b) = b$). If $f : A \to B$ and $g : C \to D$, we put $f \times g = \langle f \circ \pi_1, g \circ \pi_2\rangle : A \times C \to B \times D$ (i.e. $(f \times g)(a, c) = (f(a), g(c))$).

We will often make use of the following easy, but useful lemma, the proof of which is relegated to the exercises.

Lemma 1.1.3 (Universal Property of the Cartesian Product). *Suppose $f : C \to A$, $g : C \to B$ are functions. Then $\langle f, g\rangle$ is the unique function satisfying $\langle f, g\rangle \circ \pi_1 = f$ and $\langle f, g\rangle \circ \pi_2 = g$.*

Using cartesian products, we can now express the dynamics of a stream automaton as a single function

$$\langle \mathsf{hd}, \mathsf{tl}\rangle : S \to D \times S.$$

In this example, the first prototypical feature of coalgebras is already present: We *observe* hidden states $s \in S$ by means of a function with *domain* S; on a formal level, this can be seen as dual to the case of algebras, where one *constructs* elements (of some set A) by means of functions with *codomain* A.

1.1.3 Mealy Machines

Our next example of a system, Mealy machines, can be seen as stream automata, which also accepts user input. Recall that a Mealy machine with input alphabet I and output alphabet O is just a deterministic finite automaton, which produces a letter of the output alphabet O in every transition step. Given an input alphabet I and an output alphabet O, the formalisation of Mealy machines is the following:

Definition 1.1.4. A *Mealy machine* is a triple $(S, \mathsf{next}, \mathsf{out})$, where S is the state set of the automaton, $\mathsf{next} : I \times S \to S$ is the transition function and $\mathsf{out} : I \times S \to O$ is the output function.

In general (see e.g. [25, 55]), a Mealy machine also has an initial state; since the initial state is not important for our purposes, it is omitted in the above definition.

In the case of stream automata, we have remarked that it is a typical feature of state based systems, that one can observe the state space by means of a function with the state space as domain. In the case of Mealy machines however, we deal with two functions whose codomain is the cartesian product $S \times I$ of state space and input alphabet. This difference only exists at the surface, and we can convert the functions next and out to a single function with domain S, using a technique known as *currying*. Given a binary function $f : A \times B \to C$ and $a \in A$, the assignment $f(a, -) : b \mapsto f(a, b)$ defines a function $B \to C$. This allows us to view f as a unary function, taking values in the set of all functions from $B \to C$. We denote the function space as follows:

Notation 1.1.5. If A, B are sets, then B^A denotes the set of all functions from A to B. When defining a particular element $f \in B^A$, we often use λ-notation, that is, if $t(a)$ is an expression denoting an element of B with a free parameter a, which stands for an element of a, we write $\lambda a.t(a)$ for the function $a \mapsto t(a)$; for example $\lambda x.x + 1$ denotes the successor function. If $f : A \times B \to C$ is a function, we write $\mathsf{curry}(f)$ for the function $a \mapsto \lambda b.f(a, b)$. Note that $\mathsf{curry}(f) : A \to C^B$.

Using cartesian products, we can express the transition function of a Mealy automaton as a function

$$\langle \mathsf{out}, \mathsf{next} \rangle : S \times I \to O \times S$$

and, using currying as explained above, as a unary function

$$\mathsf{curry}(\langle \mathsf{out}, \mathsf{next} \rangle) : S \to (O \times S)^I$$

After these rearrangements, we can view Mealy automata as pairs (S, f), where S is a set of states and f is a function with domain S, in accordance with our above remark that one can observe states via a function with domain S.

1.1.4 Automata with Error Conditions

We now extend the previous example to incorporate the failure to perform a certain transition. In this context, it is helpful to think of a deterministic automaton not in the context of formal languages, but to consider the automaton as a vending machines, where we the input is e.g. coins and the machine serves drinks, say. In a setting which does not incorporate failures, we can use the same model as for the Mealy machine: If the machine is some internal state s, an input gives us both a new state s' and an output, depending on the current state (i.e. s) and the input.

We now add the possibility of failures to the picture. Suppose we want the machine to signal that it is (due to some internal error, or because it ran out of merchandise) no longer able to serve our requests. Let's furthermore assume that the machine realises, that it ran out of merchandise after the last item was sold (that is, we do not consider erroneous vending attempts). In addition to our set I of input tokens, the set O of outputs, and a set S of internal states, we also fix a set E of errors (e.g. $E = \{\mathsf{out\ of\ change}, \mathsf{our\ of\ merchandise}\}$). Since we want to design the machine such that it does not allow for failed vending attempts, the output function

$$\mathsf{out} : S \times I \to O$$

is the same as for Mealy machines. The difference comes with the function which determines the next state. If the automaton decides to stop working, we have to produce an error condition $e \in E$; otherwise a successor state; note that as soon as we produce a successor state, we can apply out to initiate a new vending action. Assuming that S and E are disjoint sets, we can therefore put

$$\mathsf{next} : S \times I \to S \cup E.$$

The assumption of disjointness of two sets is tedious, most of all because it requires a lot of bookkeeping. Using coproducts, we can incorporate the assumption of disjointness into the codomain of next.

Notation 1.1.6. Suppose A, B are sets. We denote their *coproduct* (sometimes also called *disjoint union*) $\{(a, 0) \mid a \in A\} \cup \{(b, 1) \mid b \in B\}$ by $A + B$. If C is another set and $f : A \to C$ and $g : B \to C$ are functions, we define $[f, g] : A + B \to C$ by $(f + g)(x, i) = f(x)$ if $i = 0$ and $(f + g)(x, i) = g(x)$ if $i = 1$. The canonical injections are denoted by $\mathrm{in}_l : A \to A + B$ (resp.

$\text{in}_r : B \to A + B$) where $\text{in}_l(a) = (a, 0)$ (resp. $\text{in}_r(b) = (b, 1)$). If $f : A \to B$ and $g : C \to D$, we put $f + g = [\text{in}_l \circ f, \text{in}_r \circ g] : A + C \to B + D$, that is $(f + g)(a, 0) = (f(a), 0)$ and $(f + g)(c, 1) = (g(c), 1)$.

As with cartesian products, the following universal property often simplifies proofs:

Lemma 1.1.7 (Universal Property of Coproducts). *Suppose $f : A \to C$ and $g : B \to C$ are functions. Then $[f, g]$ is the unique function satisfying $[f, g] \circ \text{in}_l = f$ and $[f, g] \circ \text{in}_r = g$.*

Exercise 1.5.2 asks you to prove this lemma.

Using coproducts, we drop the assumption of disjointness and model the next state function of the automaton with error conditions as

$$\mathsf{out} : S \times I \to O + E.$$

Combining out and next into a single function, we obtain a binary function

$$\langle \mathsf{out}, \mathsf{next} \rangle : S \times I \to O \times (S + E) \tag{1.1}$$

which we can view, via currying, as a unary function

$$\text{curry}(\langle \mathsf{out}, \mathsf{next} \rangle) : S \to (O \times (S + E))^I. \tag{1.2}$$

Thus also automaton with error conditions can be modelled as a unary function with the state space as codomain.

1.1.5 Kripke Models

In both of the above examples, the automata were deterministic, in the sense that the successor state is determined by a function. In the case of Kripke Models, the successor state is not determined by a transition function, but by a transition relation. This way, every state can have 0, 1 or more successors.

Definition 1.1.8. A *Kripke model* over a set A (of atomic propositions) is a triple (S, R, V) where S is a set (of states, or worlds), $R \subseteq S \times S$ is the accessibility relation and $V : A \to \mathcal{P}(W)$ is a valuation.

Given a Kripke model (S, R, V) over a set A of propositions, one generally thinks of S as a set of worlds, and the relation R determines the evolution of one world into another. The valuation determines, for each proposition $a \in A$, the set of worlds, where this proposition is satisfied. If, for example a corresponds to "it's raining", then $V(a)$ is the set of all worlds where it rains.

For the purpose of these notes, it's probably best to think of a Kripke model as a non-deterministic system, which changes state non-deterministically. The valuation allows us to make assertions about the current state.

philosophical un
pinning, importa
in model-checking

Also the structure of Kripke models can be expressed using a function out of the state space: We have to represent the transition relation $R \subseteq S \times S$ and the valuation $A \to \mathcal{P}(S)$ as functions with domain S. Since there is a one-to-one correspondence

$$\{\text{relations } R \subseteq A \times B\} \leftrightarrow \{\text{functions } f : A \to \mathcal{P}(B)\},$$

given by

$$R \mapsto (a \mapsto \{b \in B \mid (a, b) \in R\}$$

we can equivalently view the transition structure of a Kripke model as a function $\mathsf{next} : S \to \mathcal{P}(S)$. A similar trick can also be applied to the valuation function: Instead of associating a set of worlds to each proposition, we can equivalently map a world to the set of propositions it satisfies, obtaining a function $\mathsf{propn} : S \to \mathcal{P}(S)$, mapping $s \mapsto \{a \in A \mid s \in V(a)\}$. Combining both functions, we obtain a representation of Kripke models as pair (S, f) where S is the state space and the dynamics is given by

$$\langle \mathsf{next}, \mathsf{prop} \rangle : S \to \mathcal{P}(S) \times \mathcal{P}(A)$$

with the state space as codomain.

1.1.6 Coalgebras

We have seen in the examples, that different types of state based systems can be modelled by a set (of states) and a function, whose domain is the state space. As the above examples show, the codomain of this function determines the type of system. Being interested in a general theory of systems, we consider the type as a parameter of our definition. This is done by considering the type as an operation on sets, assigning a set TX to every set X.

Definition 1.1.9. Suppose T is an operation on sets. A T-*coalgebras* is a pair (C, γ) where C is a set (of states) and $\gamma : C \to TC$ is a (transition) function.

We have already seen three examples:

- *stream automata* are T-coalgebras for $TX = D \times X$
- *Mealy machines* are T-coalgebras for $TX = (O \times X)^I$
- *Kripke Models* are T-coalgebras for $TX = \mathcal{P}(X) \times \mathcal{P}(A)$
- *Automata with Errors* are T-coalgebras for $TX = (O \times (S + E)^I$.

Three more types of systems are treated in the exercises.

1.2 Homomorphisms of Systems

The easiest way to describe the relation between different systems (coalgebras) is via *homomorphisms.* Just as in universal algebra, a homomorphism is a structure preserving map. This section takes up some of the examples discussed previously and motivates the definition of coalgebra-homomorphism.

1.2.1 Morphisms of Stream Automata

Let's start with the simples example, and consider homomorphisms between stream automata over a set D of data. With the idea at the back of our mind that a homomorphism is a structure preserving function, the obvious definition of homomorphism is the following:

Definition 1.2.1. Suppose $(S, \mathsf{hd}, \mathsf{tl})$ and $(S, \mathsf{hd}', \mathsf{tl}'$ are stream automata. A function $f : S \to S'$ is a *homomorphism of stream automata*, if

- $\mathsf{hd}(s) = \mathsf{hd}'(f(s))$
- $f \circ \mathsf{tl}(s) = \mathsf{tl}'(f(s))$

for all $s \in S$.

In view of our general definition of coalgebras as given by a unary function with the state space as domain, we can rephrase this definition as follows:

Fact 1.2.2. *A function $f : S \to S'$ is a homomorphism between the stream automata $(S, \mathsf{hd}, \mathsf{tl})$ and $(S', \mathsf{hd}', \mathsf{tl}')$ iff $D \times f \circ \langle \mathsf{hd}, \mathsf{tl}\rangle = \langle \mathsf{hd}', \mathsf{tl}'\rangle \circ f$.*

Here, as in the sequel, we use the same symbol for a set and the identity function on that set:

Notation 1.2.3. If A is a set, then we also use A to denote the identity function on A. In particular, if f is a function, $A \times f$ stands for $\mathrm{id}_A \times f$.

The homomorphism condition on f is often expressed diagrammatically by requiring the diagram

$$\begin{tikzcd}
S \arrow[r, "f"] \arrow[d, "\sigma"'] & S' \arrow[d, "\sigma'"] \\
D \times S \arrow[r, "D\times f"'] & D \times S',
\end{tikzcd}$$

where $\sigma = \langle \mathsf{hd}, \mathsf{tl}\rangle$ and $\sigma' = \langle \mathsf{hd}', \mathsf{tl}'\rangle$ to commute (i.e. all paths from any node to another yield the same function).

The proof of the above fact is straightforward, and therefore omitted, although the reader is encouraged to give a proof, which only uses the universal property of cartesian products (Lemma 1.1.3).

The fact that homomorphisms are compatible with the transition structure of stream automata entails, that homomorphisms preserve observable behaviour. In the case of a stream automaton $(S, \mathsf{hd}, \mathsf{tl})$, the only thing we can observe from a state $s \in S$ is the induced sequence $(\mathsf{hd}(s), \mathsf{hd} \circ \mathsf{tl}(s), \mathsf{hd} \circ \mathsf{tl} \circ \mathsf{tl}(s), \dots)$ of data elements. In this sense, a homomorphism preserves observable behaviour:

Lemma 1.2.4. *Suppose* $(S, \mathsf{hd}, \mathsf{tl})$ *and* $(S', \mathsf{hd}', \mathsf{tl}')$ *are stream automata and* $f : S \to S'$ *is a morphism of stream automata. Then*

$$\mathsf{hd} \circ \mathsf{tl}^n(s) = \mathsf{hd}' \circ \mathsf{tl}'^n(f(s))$$

for all $s \in S$.

In the formulation of the lemma, we have used the following terminology to express the iteration of endofunctions:

Notation 1.2.5. Suppose $f : A \to A$ is an endofunction. For $n \in \mathbb{N}$, the function f^n is defined inductively by $f^0 = \mathrm{id}_A$ and $f^n = f \circ f^{n-1} f$ for $n > 0$.

This allows us to use induction in the proof of Lemma 1.2.4:

Proof of Lemma 1.2.4. We use induction on n to show that

$$f \circ \mathsf{tl}^n = \mathsf{tl}'^n \circ f \tag{1.3}$$

if $f : S \to S'$ is a morphism of stream automata $(S, \mathsf{hd}, \mathsf{tl}) \to (S', \mathsf{hd}', \mathsf{tl}')$. For $n = 0$, there is nothing to show. Now suppose $0 < n$ and pick some $s \in S$. We have

$$\begin{aligned} f \circ \mathsf{tl}^n(s) &= f \circ \mathsf{tl} \circ \mathsf{tl}^{n-1}(s) && \text{Def'n 1.2.5} \\ &= \mathsf{tl}' \circ f \circ \mathsf{tl}^{n-1}(s) && \text{Def'n 1.2.1} \\ &= \mathsf{tl}' \circ \mathsf{tl}'^{n-1} \circ f(s) && \text{Def'n 1.2.5} \\ &= \mathsf{tl}'^n \circ f(s). \end{aligned}$$

Identity 1.3 now entails that

$$\begin{aligned} \mathsf{hd}' \circ \mathsf{tl}'^n f(s) &= \mathsf{hd}' \circ f \circ \mathsf{tl}^n(s) && \text{Equation 1.3} \\ &= \mathsf{hd} \circ \mathsf{tl}^n(s) && \text{Def'n 1.2.1} \end{aligned}$$

which proves the claim. □

Note that this is not a characterisation of morphisms of stream automata; in Exercise 1.5.6 you are asked to find an example of a behaviour preserving function, which is not a homomorphism.

1.2.2 Morphisms of Mealy Machines

Let's now consider homomorphisms of Mealy machines. The requirement that homomorphisms be compatible with the transition structure of Mealy machines leads us to the following:

Definition 1.2.6. Suppose $(S, \mathsf{next}, \mathsf{out})$ and $(S, \mathsf{next}', \mathsf{out}')$ are Mealy machines. A function $f : S \to S'$ is a homomorphism of Mealy machines, if

(*i*) $\mathsf{out}(s, i) = \mathsf{out}'(f(s), i)$

(*ii*) $f \circ \mathsf{next}(s, i) = \mathsf{next}'(f(s), i)$

for all $s \in S$ and $i \in I$.

Let's briefly contemplate on this definition. The first requirement says that – given the same input – the states s and $f(s)$ produce the same output. The second equation says that f is compatible with next (again, given the same inputs). This becomes more apparent when considering the functions $\mathsf{next}_i : S \to S$, $s \mapsto \mathsf{next}(i, s)$ (and $\mathsf{next}'_i : s' \mapsto \mathsf{next}'(i, s)$) for some fixed $i \in I$. Note that $\mathsf{next}_i = \mathsf{curry}(\mathsf{next})(i)$ (and of course $\mathsf{next}' = \mathsf{curry}(\mathsf{next}')(i)$). Using this notation, the second requirement we put on a homomorphism of Mealy machines reads $f \circ \mathsf{next}_i = \mathsf{next}'_i \circ f$ (for all $i \in I$), which is, except for the universally quantifies variable i, the requirement we have encountered in the definition of homomorphisms for stream automata, cf. Definition 1.2.1.

We now claim that the definition of homomorphisms of Mealy machines also formally follows the same pattern as the definition of homomorphisms for stream automata. To make this precise, we use

Notation 1.2.7. If $f : A \to B$ is a function and C is a set, $f^C : A^C \to B^C$ is the function given by $h \mapsto f \circ h$.

Lemma 1.2.8. *Suppose $(S, \mathsf{next}, \mathsf{out})$ and $(S, \mathsf{next}', \mathsf{out}')$ are Mealy machines. A function $f : S \to S'$ is a homomorphism of Mealy machines, if the diagram*

$$\begin{tikzcd}
S \arrow[r, "f"] \arrow[d, "\sigma"'] & S' \arrow[d, "\sigma'"] \\
(O \times S)^I \arrow[r, "(O\times f)^I"'] & (O \times S')^I,
\end{tikzcd}$$

where $\sigma = \mathsf{curry}(\langle \mathsf{out}, \mathsf{next}\rangle)$ and $\sigma' = \mathsf{curry}\langle \mathsf{out}', \mathsf{next}'\rangle$, commutes.

Proof. Note that, unravelling Notation 1.2.7, we have

$$(O \times f)^I \circ \sigma(s) = (O \times f)^I(\sigma(s)) = (O \times f) \circ \sigma(s). \tag{1.4}$$

Using this identity, we obtain the following chain of equivalences:

	f is a morphism of Mealy machines	
iff	$(\mathsf{out}(s,i), f \circ \mathsf{next}(s,i)) = (\mathsf{out}'(f(s),i), \mathsf{next}'(f(s),i))$	Def'n 1.2.6
iff	$(\mathrm{id}_O \times f) \circ \langle \mathsf{out}, \mathsf{next} \rangle (s,i) = \langle \mathsf{out}', \mathsf{next}' \rangle (f(s), i)$	Not'n 1.1.2
iff	$(\mathrm{id}_O \times f)(\sigma(s)(i)) = (\sigma' \circ f)(s)(i)$	Def'n of σ
iff	$((O \times f) \circ \sigma(s))(i) = (\sigma' \circ f)(s)(i)$	Def'n of $\circ$
iff	$((O \times f)^I \circ \sigma)(s)(i) = (\sigma' \circ f)(s)(i)$	Equation 1.4
iff	the diagram in Lemma 1.2.8 commutes	

□

1.2.3 Morphisms of Coalgebras: The Common Pattern

Let's try to find the formal similarities between morphisms of stream automata and morphisms of Mealy machines. We have seen that both steam automata and Mealy machines arise as coalgebras, the former as coalgebras for the operation $TX = D \times X$, the latter as coalgebras for $TX = (O \times X)^I$.

Note that Notation 1.2.3 and 1.2.7 (purposely) extend both the operations to functions, that is, if $f : A \to B$ is a function, then $Tf : TA \to TB$, for both $TX = D \times X$ and $TX = (O \times X)^I$.

Now, if (S, σ) and (S', σ') are coalgebraic representations of either stream automata (cf. Section 1.1.2) or Mealy machines (given in Section 1.1.3), we have seen that a function $f : S \to S'$ is a homomorphism of the respective structures, iff

$$Tf \circ \sigma = \sigma' \circ f.$$

This is the key observation, which gives rise to a general definition of morphisms for coalgebras, but it requires the extension of T to functions.

Let's see what we need to require from this extension to functions, in order to obtain a reasonable notion of morphisms. If we think of morphisms as structure preserving functions, we have the following two minimal requirements:

(*i*) The identity function is a morphism

(*ii*) The composition of two morphisms is a morphism.

(Note that also morphisms of algebraic structures meet both these requirements, see e.g. [60].)

Now suppose that T is an operation on sets, which extends to functions, i.e. if $f : A \to B$ is a function, then so is $Tf : TA \to TB$. We discuss the requirements which T has to meet in order to guarantee both assumptions above.

For the identity map, let (S, σ) be a T-coalgebra. The identity function id_S is a morphism if the square

$$\begin{tikzcd}
S \arrow[r, "\mathrm{id}_S"] \arrow[d, "\sigma"'] & S \arrow[d, "\sigma"] \\
TS \arrow[r, "T\mathrm{id}_S"'] & TS
\end{tikzcd}$$

commutes. This is guaranteed if we require T to preserve identity morphisms, that is

$$T\mathrm{id}_S = \mathrm{id}_{TS}$$

for all sets S.

For compositionality, assume that (S, σ), (S', σ') and (S'', σ'') are T-coalgebras and $f : S \to S'$, $g : S' \to S''$ are morphisms. Consider the following diagram:

$$\begin{tikzcd}
S \arrow[r, "f"] \arrow[rr, bend left, "g\circ f"] \arrow[d, "\sigma"'] & S' \arrow[r, "g"] \arrow[d, "\sigma"'] & S'' \arrow[d, "\sigma''"'] \\
TS \arrow[r, "Tf"'] \arrow[rr, bend right, "T(g\circ f)"'] & TS' \arrow[r, "Tg"'] & TS''
\end{tikzcd}$$

The inner two squares of the diagram commute, since we have assumed both f and g to be homomorphisms. We have that $g \circ f$ is a homomorphism, iff the the rectangle constituted by the curved arrows commutes. This is certainly the case if T preserves function composition, that is,

$$T(g \circ f) = Tg \circ Tf$$

whenever f, g are composable functions.

Both conditions, preservation of identities and preservation of compositions, put together, say that we have to require that T is a functor:

Definition 1.2.9. A *functor on sets* is an operation T such that

(i) TA is a set, whenever A is a set (*ii*) Tf is a function, whenever f is a function

subject to the following conditions:

(*i*) $Tf : TA \to TB$, for all functions $f : A \to B$

(*ii*) $T\mathrm{id}_A = \mathrm{id}_{TA}$ for all sets A

(*iii*) $T(g \circ f) = Tg \circ Tf$ whenever $f : A \to B$ and $g : B \to C$ are composable functions.

We write $T : \mathsf{Set} \to \mathsf{Set}$ to indicate that T is a functor on sets.

We now require for the rest of these notes that the operation T, which describes the type of system under consideration, is a functor on sets. The reader is invited to check that the operations we have considered in our discussion of coalgebra-morphism, that is, $TX = D \times X$ and $TX = (O \times X)^I$ are indeed functors (Exercise 1.5.7). Using that T can also be applied to functions, we obtain the following definition of coalgebra homomorphism:

Definition 1.2.10. Suppose $T : \mathsf{Set} \to \mathsf{Set}$ is a functor and (S, σ), (S', σ') are T-coalgebras. A function $f : S \to S'$ is a *homomorphism* from (S, σ) to (S, σ') (denoted by $f : (S, \sigma) \to (S, \sigma')$, if $Tf \circ \sigma = \sigma' \circ f$.

Since the word "homomorphism" is quite long, we will often simply speak of morphisms of coalgebras. Our above discussion about the required properties of T then proves the following:

Proposition 1.2.11. *Suppose* $T : \mathsf{Set} \to \mathsf{Set}$.

(i) If (S, σ) *is a* T*-coalgebra, then* $\mathrm{id}_S : (S, \sigma) \to (S, \sigma)$ *is a coalgebra morphism.*

(ii) If also (S', σ') *and* (S'', σ'') *are* T*-coalgebras and* $f : (S, \sigma) \to (S', \sigma')$ *and* $g : (S', \sigma') \to (S'', \sigma'')$ *are morphisms, then so is* $g \circ f : (S, \sigma) \to (S'', \sigma'')$.

Proof. Immediate by the fact that T is a functor, see the discussion at the beginning of this section. □

1.2.4 Morphisms of Kripke Models

In the previous section, we have examined morphisms of stream automata and Mealy machines in order to distill the general pattern for the definition of coalgebra-homomorphism. We now go the other way round and instantiate Definition 1.2.10 with the case of Kripke models (Section 1.1.5).

Recall that Kripke models (S, R, V) over a set P of propositions are in 1-1 correspondence to coalgebras for $TX = \mathcal{P}(X) \times \mathcal{P}(P)$. In order to use the definition of coalgebra morphism, we first have to extend T to a functor, that is, we need to define the action of T on functions. We begin with the action of $\mathcal{P}$ on functions. If $f : A \to B$, what's a good definition of $\mathcal{P}(f) : \mathcal{P}(A) \to \mathcal{P}(B)$? The first (and only) thing, which comes to mind is *direct image*:

Notation 1.2.12. If $f : A \to B$ is a function and $\mathfrak{a} \subseteq A$, we denote by

$$f[\mathfrak{a}] = \{f(a) \mid a \in A\}$$

the *direct image* of $\mathfrak{a}$ under f.

We can now put

$$\mathcal{P}(f)(\mathfrak{a}) = f[\mathfrak{a}]$$

to obtain the desired definition of $\mathcal{P}$ on functions (would inverse image also be a possibility?). Using the abbreviation introduced in Notation 1.2.3, the action of T on functions is now given by

$$\mathcal{P}(f) \times \mathcal{P}(P) = \mathcal{P}(f) \times \mathrm{id}_{\mathcal{P}(P)} : \mathcal{P}(A) \times \mathcal{P}(P) \to \mathcal{P}(B) \times \mathcal{P}(P)$$

(recall that the set P is fixed once and for all).

Now suppose that (S, R, V), (S', R', V') are Kripke models, and (S, σ), (S', σ') are their coalgebraic representations (see Section 1.1.5, i.e. $\sigma(s) = (\{s' \in S \mid (s, s') \in R\}, \{p \in P \mid s \in V(p)\})$, analogously for (S', R', V')).

Instantiating Definition 1.2.10, we require for $f : S \to S'$ to be a morphism that

$$\begin{tikzcd}
S \arrow[r, "f"] \arrow[d, "\sigma"'] & S' \arrow[d, "\sigma'"] \\
\mathcal{P}(S) \times \mathcal{P}(P) \arrow[r, "\mathcal{P}(f)\times\mathcal{P}(P)"'] & \mathcal{P}(S') \to \mathcal{P}(P)
\end{tikzcd}$$

commutes. In terms of the relational presentation of Kripke models, this boils down to the following:

Lemma 1.2.13. *Suppose (S, R, V), (S', R', V') are Kripke models with associated coalgebraic representations (S, σ) and (S', σ'). A function $f : S \to S'$ is a coalgebra morphism $f : (S, \sigma) \to (S', \sigma')$ if and only if* *(i)* $s \in V(p)$ *iff* $f(s) \in V(p)$ *for all* $s \in S$ *and* $p \in P$

(ii) $(s_0, s_1) \in R \implies (f(s_0), f(s_1)) \in R'$ *for all* $s_0, s_1 \in S$

(iii) $(f(s_0), s_1') \in R' \implies \exists s_1 \in S_1 . f(s_1) = s_1' \& (s_0, s_1) \in R$ *for all* $s_0 \in S$, $s_1' \in S'$

Proof. We denote the first and second component of σ by σ_1 and σ', respectively and use the analogous notation for σ'. First note that f is a coalgebra morphism, iff

(i') $\sigma_2(s) = \sigma_2'(f(s))$

(ii') $\mathcal{P}(f) \circ \sigma_1 \subseteq f \circ \sigma_1'$.

(iii') $\mathcal{P}(f) \circ \sigma_1 \supseteq f \circ \sigma_1'$.

We show that $(x) \iff (x')$ for $x \in \{i, ii, iii\}$.

First note that

$$\begin{array}{lll} (i) & \iff & \forall s \in S \forall p \in P . p \in \sigma_1(s) \text{ iff } p \in \sigma_2(f(s)) \\ & \iff & \forall s \in S \forall p \in P . s \in V(p) \text{ iff } f(s) \in V'(p) \\ & \iff & (i') \end{array}$$

The second equivalence is due to

$$\begin{array}{lll} (ii) & \iff & \forall s_0 \in S . \{f(s_1) \mid (s_0, s_1) \in R\} \subseteq \{s_1' \in S' \mid (f(s_0), s_1') \in R'\} \\ & \iff & (ii'). \end{array}$$

Finally,

$$\begin{array}{lll} (iii) & \iff & \forall s_0 \in S . \{s_1' \in S' \mid (f(s_0), s_1') \in R'\} \subseteq \{f(s_1) \mid (s_0, s_1) \in R\} \\ & \iff & (iii'), \end{array}$$

which finishes the proof. □

This Lemma shows, that in the case of Kripke models, we obtain (yet another) familiar notion of morphism: coalgebra morphisms are precisely the p morphisms (in the terminology of [13, 19, 20]), or bounded morphisms (following [10]).

The case of morphisms between automata with error conditions, which involves the extension of $TX = (O \times (X + E))^I$ to functions, is treated in the exercises.

1.3 Behavioural Equivalence

When we have first introduced systems informally, we have emphasised that the internal state of the system is invisible for the user (this is the so-called "black box view" of a system). That is, we identify any two states, which cannot be distinguished from the outside. The study of this indistinguishability relation (called "behavioural equivalence") is one of the main concerns of a theory of systems, for the following two reasons:

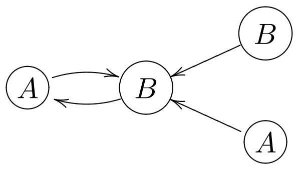
(i) If two states cannot be distinguished from the outside, we can substitute them for one another without affecting the behaviour of the system (i.e. the environment cannot distinguish between behaviourally equivalent states anyway).

(ii) When making assertions about the system, we want to be sure that we only talk about its visible behaviour, and not constrain its internal realisation.

This section introduces the general definition of behavioural equivalence and discusses some examples.

1.3.1 From Stream Automata to a General Definition

Just to give an idea what behavioural equivalence means, consider the following stream automaton, where the states are represented as circles, and state transitions as arrows:



In this diagram, one would consider all states labelled with A behaviourally equivalent, whereas we can distinguish both states which have a B-label. This shows different states, which are behaviourally equivalent. In the concrete case of stream automata, we have already used an intuitive definition of behavioural equivalence: If $(S, \mathsf{hd}, \mathsf{tl})$ is a stream automaton, we have considered the behaviour associated with a state s as the infinite sequence $(\mathsf{hd}(s), \mathsf{hd} \circ \mathsf{tl}(s), \mathsf{hd} \circ \mathsf{tl}^2(s), \dots)$, and we can call two states behaviourally equivalent, if they have the same behaviour (in the above sense). They key observation now is, that the set of behaviours constitutes a stream automaton (Z, ζ_1, ζ_2) itself, where

$$Z = \{(d_0, d_1, d_2, \dots) \mid d_i \in D \text{ for all } d \geq 0\}$$

is the set of all infinite sequences over D, and

$$\zeta_1(d_0, d_1, d_2, \dots) = d_0 \quad \zeta_2(d_0, d_1, d_2, \dots) = (d_1, d_2, d_3, \dots).$$

That is, ζ_1 gives us the first element of the sequence (the head) and ζ_2 deletes the first element.

Furthermore, mapping a state to its behaviour is a homomorphism of systems:

Lemma 1.3.1. *Suppose* $(S, \mathsf{hd}, \mathsf{tl})$ *is a stream automaton and* $\mathsf{beh} : S \to Z$ *is defined by*

$$\mathsf{beh}(s) = (\mathsf{hd}(s), \mathsf{hd} \circ \mathsf{tl}(s), \mathsf{hd} \circ \mathsf{tl}^2(s), \dots).$$

Then beh *is a morphism of stream automata.*

Proof. We have $\zeta_1(\mathsf{beh}(s)) = \zeta_1(\mathsf{hd}(s), \mathsf{hd} \circ \mathsf{tl}(s), \mathsf{hd} \circ \mathsf{tl}^2(s), \dots) = \mathsf{hd}(s)$ and $\zeta_2(\mathsf{beh}(s)) = \zeta_2(\mathsf{hd}(s), \mathsf{hd} \circ \mathsf{tl}(s), \mathsf{hd} \circ \mathsf{tl}^2(s), \dots) = (\mathsf{hd} \circ \mathsf{tl}(s), \mathsf{hd} \circ \mathsf{tl}^2(s), \mathsf{hd} \circ \mathsf{tl}^3(s), \dots) = \mathsf{beh} \circ \mathsf{tl}(s)$. □

Actually, beh above is the only homomorphism $\mathsf{beh} : (S, \mathsf{hd}, \mathsf{tl}) \to (Z, \zeta_1, \zeta_2)$, but we don't make use of uniqueness at this point; Exercise 1.5.10 asks you to show it's uniqueness.

Now two states s and s' are behaviourally equivalent, if they have the same behaviour, that is, if $\mathsf{beh}(s) = \mathsf{beh}(s')$, that is, if they are identified by (the coalgebra morphism) beh. On the other hand, there's no need to require that the states be identified by this particular morphism, since every morphism preserves behaviour (Lemma 1.2.4). In other words: it doesn't hurt to require that s and s' can be identified by *arbitrary* coalgebra morphisms. We thus arrive at the following definition:

Definition 1.3.2. Suppose $(S, \sigma), (S', \sigma') \in \mathsf{CoAlg}(T)$. A pair of states $(s, s') \in S \times S'$ is *behaviourally equivalent* (denoted by $s \approx s'$), if there is $(S'', \sigma'') \in \mathsf{CoAlg}(T)$ and a pair of morphisms $f : (S, \sigma) \to (S'', \sigma'')$ and $g : (S', \sigma') \to (S'', \sigma'')$ such that $f(s) = g(s')$.

We briefly note the following easy consequences:

Proposition 1.3.3. *Suppose* $(S, \sigma), (S', \sigma') \in \mathsf{CoAlg}(T)$ *and* $(s, s') \in S \times S'$.

(i) $s \approx s$.

(ii) $s \approx s' \implies s' \approx s$.

Proof. Obvious. □

We leave transitivity to the next chapter, which then shows that behavioural equivalence is indeed an equivalence. For the moment, we contend ourselves with seeing, what this definition means for several examples.

In the context of stream automata, we have the following characterisation of behavioural equivalence:

Proposition 1.3.4. *Suppose* $(S, \mathsf{hd}, \mathsf{tl})$ *and* $(S', \mathsf{hd}', \mathsf{tl}')$ *are stream automata and* $(s, s') \in S \times S'$. *Then* $s \approx s'$ *iff* $\mathsf{beh}(s) = \mathsf{beh}(s')$.

Proof. If $s \approx s'$ then Lemma 1.2.4 shows $\mathsf{beh}(s) = \mathsf{beh}(s')$; the converse is the content of Lemma 1.3.1. □

The remainder of the chapter instantiates the notion of behavioural equivalence with more examples.

1.3.2 Behavioural Equivalence on Mealy Machines

We consider Mealy machines over the input and output alphabets I and O, respectively. From the viewpoint of automata theory, the main feature of a Mealy machine is, that it defines a (rather special) function from strings over I to strings over O. Suppose $(S, \mathsf{next}, \mathsf{out})$ is a Mealy machine. Ever state $s \in S$ defines a function $t(s) : I^* \to O^*$ as follows:

$$t(s)(\epsilon) = \epsilon \quad t(s)(i \cdot \vec{i}) = \mathsf{out}(i, s) \cdot t(\mathsf{next}(i, s))(\vec{i}) \text{ for } i \in I, \vec{i} \in I^*,$$

where we have used the following

Notation 1.3.5. If A is a set (or an alphabet), A^* denotes the set of finite sequences of elements of A (or the set of words over A). We denote the empty word by ϵ, and concatenation of $\vec{a}, \vec{b} \in A^*$ by $\vec{a} \cdot \vec{b}$; we often consider $A \subseteq A^*$.

So what's the observable behaviour of a (state of a) Mealy machine? In this section, we show that two states s, s' of Mealy machines are behaviourally equivalent, if and only if they define the same function, i.e. if $t(s) = t(s')$, which conforms with our intuitive idea of observable behaviour of Mealy machines.

Recall that every Mealy machine arises as T-coalgebra for $TX = (O \times X)^I$ (Section 1.1.3), and that we have shown in Lemma 1.2.8, that the coalgebraic notion of morphism coincides with homomorphisms of Mealy machines. Thus a pair (s, s') is behaviourally equivalent, iff they can be identified by a morphism of Mealy machines.

We begin with showing, that behavioural equivalence of two states implies equality of the induced function. By way of doing this, we also demonstrate a handy trick for proving statements involving behavioural equivalence: In order to show that a property is invariant under behavioural equivalence, we just have to show that it is stable under homomorphisms.

Lemma 1.3.6. *Suppose* $(S, \mathsf{out}, \mathsf{next})$, $(S', \mathsf{out}', \mathsf{next}')$ *are Mealy machines and* $s \in S$ *and* $s' \in S'$ *are behaviourally equivalent. Then* $t(s) = t(s')$.

Proof. As remarked above, it suffices to consider a morphism $f : S \to S'$ and show, that $t(s) = t(f(s))$. Having established that, we can argue as follows: If $s \approx s'$, then $f(s) = g(s')$ for some homomorphisms f and g, hence $t(s) = t(f(s)) = t(g(s')) = t(s')$.

We now show that $t(s) = t(f(s))$ for all $s \in S$, that is, $t(s)(i) = t(f(s))(i)$ by induction on i. For $i = \epsilon$, we have $t(s)(\epsilon) = t(f(s))(\epsilon)$ by definition of t. Now suppose $i = i_0 \cdot i_1$ with $i_0 \in I$ and $i_1 I^*$. Note that $f(\mathsf{next}(s, i_0)) = \mathsf{next}'(f(s), i_0)$, hence $t(\mathsf{next}'(f(s), i_0)) = t(f(\mathsf{next}(s, i_0)) = t(\mathsf{next}(s, i_0))$ by induction hypothesis. Now $t(s)(i_0 \cdot i_1) = \mathsf{out}(s, i_0) \cdot t(\mathsf{next}(s, i_0))(i_1) = \mathsf{out}'(f(s), i_0) \cdot t(\mathsf{next}'(f(s), i_0)) = t(f(s))(i_0 \cdot i_1)$, as claimed. $\square$

For the converse of the lemma, we use a technique similar to that used in the proof of Proposition 1.3.4: We isolate a notion of behaviour, put a transition structure on the set of all behaviours and show, that the function, which maps a state to its behaviour is a homomorphism. So what is to be regarded as the observable behaviour of a state of a Mealy machine? We have seen above, that every state s defines a function $t(s) : I^* \to O^*$. These functions can easily be seen to have the following two properties:

- They are *length preserving* that is, $\ell(i) = \ell(t(s)(i))$, for all $i \in I^*$, where ℓ denotes the length of a finite sequence of letters, and
- They are *prefix closed*, that is, if i_0 is a prefix of i_1 (and $i_0, i_1 \in I^*$), then $t(s)(i_0)$ is a prefix of $t(s)(i_1)$.

We denote the set of all functions, which have the above two properties, by $\mathcal{M}$, i.e.

$$\mathcal{M} = \{b : I^* \to O^* \mid b \text{ length preserving and prefix closed}\};$$

we think of $\mathcal{M}$ as the set of behaviours of Mealy machines.

Lemma 1.3.7. *Suppose* $(S, \mathsf{out}, \mathsf{next})$, $(S', \mathsf{out}', \mathsf{next}')$ *are Mealy machines and* $s \in S$ *and* $s' \in S'$ *satisfy* $t(s) = t(s')$. *Then* $s \approx s'$.

Proof. We put a transition structure on the set $\mathcal{M}$ of behaviours of Mealy machines as follows:

$$\mathsf{out}_M(b, i) = b(i) \quad \mathsf{next}_M(b, i) = \lambda j.\mathsf{tl}(b(i \cdot j))$$

This gives rise to a Mealy machine $(\mathcal{M}, \mathsf{out}_M, \mathsf{next}_M)$; the (partial) function tl deletes the first element from a finite sequence.

We claim that $t(b) = b$, i.e. $t(b)(i) = b(i)$ for all $i \in I^*$. This is proved by induction on i. (We have to take a little care here: in order to apply the induction hypothesis, we prove for all $i \in I^*$ and all $b \in \mathcal{M}$ that $t(b)(i) = b(i)$.) If $i = \epsilon$, we have $t(b)(i) = \epsilon$ by definition of t and $b(\epsilon) = \epsilon$ since b is length-preserving. Now suppose $i = i_0 \cdot i_1$ with $i_0 \in I$ and $i_1 \in I^*$. We obtain

$$\begin{aligned}
t(b)(i_0 \cdot i_1) &= \mathsf{out}_M(b, i_0) \cdot t(\mathsf{next}_M(b, i_0))(i_1) && \text{Def'n of } t \\
&= b(i_0) \cdot t(\lambda j.\mathsf{tl}(b(i_0 \cdot j)))(i_1) && \text{Def'n of } \mathsf{out}_M,\ \mathsf{next}_M \\
&= b(i_0) \cdot \lambda j.\mathsf{tl}(b(i_0 \cdot j))(i_1) && \text{Ind'n Hyp} \\
&= b(i_0) \cdot \mathsf{tl}(b(i_0 \cdot i_1)) && b \text{ is prefix closed} \\
&= b(i_0 \cdot i_1)
\end{aligned}$$

We now show that, given any Mealy machine $(S, \mathsf{out}, \mathsf{next})$, the map $t : s \mapsto t(s)$ is a morphism of Mealy machines. This is a matter of noting that, for $i \in I$ and $s \in S$,

$$\begin{aligned}
\mathsf{out}_M(t(s), i) &= t(s)(i) && \text{Def'n of } \mathsf{out}_M \\
&= \mathsf{out}(s, i) \cdot t(\mathsf{next}(s, i))(\epsilon) && \text{Def'n of } t(s)\text{; note } i \in I \\
&= \mathsf{out}(s, i). && \text{Def'n of } t
\end{aligned}$$

The second condition on homomorphisms of Mealy machines can be verified as follows (again $i \in I$ and $s \in S$):

$$\begin{aligned}
\mathsf{next}_M(t(s), i) &= \lambda j.\mathsf{tl}(t(s)(i \cdot j)) && \text{Def'n of } \mathsf{next}_M \\
&= \lambda j.\mathsf{tl}(\mathsf{out}(s, i) \cdot t(\mathsf{next}(s, i)(j))) && \text{Def'n of } t \\
&= \lambda j.t(\mathsf{next}(s, i))(j) && \mathsf{out}(s, i) \in O \\
&= t(\mathsf{next}(s, i))
\end{aligned}$$

The statement of the lemma is now obvious, since we have shown t to be a morphism of Mealy machines. □

1.3.3 Behavioural Equivalence on Kripke Models

In this section, we continue our discussion on the coalgebraic presentation of Kripke models, which we have begun in Section 1.1.5 and 1.2.4. The main goal of this section is to show, that two states of Kripke models are behaviourally equivalent (in the sense of Definition 1.2.10) if and only if they are bisimilar. In a nutshell, two states are bisimilar, if they can simulate (or match) each other's transitions. Bisimilarity is important both in modal logic [10] and in process calculi [41, 38]. The discovery of bisimilarity as an essentially coalgebraic notion (due to Aczel [1]) is commonly regarded as the starting point of the general theory of coalgebras. We discuss bisimilarity in the context of Kripke models. A characterisation of bisimilarity on labelled transition systems (cf Exercise 1.5.4) in terms of coalgebraic behavioural equivalence can be obtained along similar lines. We begin with introducing bisimilarity:

Definition 1.3.8. Suppose (S, R, V) and (S', R', V') are Kripke models over a set P of propositions. A *bisimulation* on (S, R, V) and (S', R', V') is a relation $B \subseteq S \times S'$ such that

(*i*) $p \in V(s) \iff p \in V(s')$ for all $(s, s') \in R$ and all $p \in P$.

(*ii*) Whenever $s_0 R s_1$ and all $(s_0, s'_0) \in B$, then there is $s'_1 \in S'$ such that $s'_0 R s'_1$ and $(s_1, s'_1) \in B$.

(*iii*) Whenever $s'_0 R' s'_1$ and all $(s_0, s'_0) \in B$, then there is $s_1 \in S$ such that $s_0 R s_1$ and $(s_1, s'_1) \in B$.

We call two states s and s' *bisimilar*, if there is a bisimulation $B \subseteq S \times S'$ with $(s, s') \in B$.

We note the following basic facts on bisimulations:

Lemma 1.3.9. *Suppose (S, R, V), (S', R', V') and (S'', R'', V'') are Kripke models.*

(i) $\Delta_S = \{(s, s) \mid s \in S\}$ is a bisimulation on (S, R, V) and itself.

(ii) If $B \subseteq S \times S'$ is a bisimulation on (S, R, V) and (S', R', V'), then $B^{\mathrm{op}} = \{(s', s) \mid (s, s') \in B\}$ is a bisimulation on (S', R', V') and (S, R, V).

(iii) If $B \subseteq S \times S'$ and $B' \subseteq S' \times S''$ are bisimulations, then $B' \circ B \subseteq S \times S''$ is a bisimulation on (S, R, V) and (S'', R'', V'') (here "$\circ$" denotes relational composition).

Proof. See Exercise 1.5.11. □

The crucial fact needed in the proof of the fact that behaviourally equivalent states of Kripke models are bisimilar is relies on the fact that the graph of a coalgebra morphism is a bisimulation. The formulation of this insight requires

Notation 1.3.10. Suppose $f : A \to B$ is a function. The *graph* of f, denoted by $G(f)$ is given by $G(f) = \{(a, f(a)) \mid a \in A\}$.

We obtain:

Lemma 1.3.11. *Suppose (S, R, V), (S', R', V') are Kripke models with associated coalgebraic representations (S, σ) and (S', σ') and $f : S \to S'$ is a coalgebra morphism. Then $G(f)$ is a bisimulation.*

Proof. Immediate from Lemma 1.2.13. □

The last lemma is the key step in the proof of the first half of the comparison between bisimulation and behavioural equivalence, which we not formulate.

Proposition 1.3.12. *Suppose (S, R, V), (S', R', V') are Kripke models and $(s, s') \in S \times S'$ are behaviourally equivalent (as states of the associated coalgebras). Then s, s' are bisimilar.*

Proof. If f and g are coalgebra morphisms identifying s and s', then $B = G(g)^{\mathrm{op}} \circ G(f)$ is a bisimulation with $(s, s') \in B$. □

The crucial observation for the proof of the converse is, that bisimulations carry a coalgebra structure.

Lemma 1.3.13. *Suppose $TX = \mathcal{P}(X) \times \mathcal{P}(P)$ and (S, σ), (S, σ') are T-coalgebras and $B \subseteq S \times S'$ is bisimulation on the associated Kripke models. If $\beta = \langle \beta_1, \beta_2 \rangle : B \to TB$ is defined by*

$$\beta_1(s, s') = \{(t, t') \in B. t \in \sigma_1(s), t' \in \sigma'_1(s')\}$$

(where $\sigma_i = \pi_i \circ \sigma_1$ and $\sigma'_i = \pi_i \circ \sigma$ for $i = 1, 2$) and

$$\beta_2(s, s') = \sigma_2(s)$$

then (B, β) is a T-coalgebra and $\pi_1 : (B, \beta) \to (S, \sigma)$ and $\pi_2 : (B, \beta) \to (S, \sigma)$ are coalgebra morphisms.

Proof. Clearly $(B, \beta) \in \mathsf{CoAlg}(T)$. We just show that π_1 is a homomorphism; one deals with π_2 analogously (note that $\pi_2 \circ \sigma(s) = \pi_2 \circ \sigma'(s')$ whenever $(s, s') \in B$). We have to show that

- $\sigma_1(s) \subseteq \mathcal{P}(\pi_1) \circ \beta_1(s, s')$
- $\sigma_1(s) \supseteq \mathcal{P}(\pi_1) \circ \beta_1(s, s')$
- $\sigma_2(s) = \beta_2(s, s')$

for all $(s, s') \in B$.

For the first statement, suppose $t \in \sigma_1(s)$ and $(s, s') \in B$. Since B is a bisimulation, we find $t' \in S'$ with $t' \in \sigma_1'(s')$ and $(t, t') \in B$. Hence $(t, t') \in \beta_1(s, s')$, and $t \in \mathcal{P}(\pi_1) \circ \beta_1(s, s')$. The second statement is similar: Suppose $t \in \mathcal{P}(\pi_1) \circ \beta_1(s)$. Then there exists $t' \in S'$ with $(t, t') \in \beta_1(s)$. By definition of β_1, we have $t \in \sigma_1(s)$. The last statement is immediate by definition of β_2. □

1.4 Notes

The aim of this chapter was to give an example oriented introduction to the world of coalgebras; other introductions which cover the material presented here include [22, 29, 35]. All of the material covered here and much more can be found in Rutten [53]. The notion of behavioural equivalence we are using in these notes is due to Kurz [32, 35]. Kripke models were introduced to give a rigorous semantics for modal logic; they have found an application in modern computer science as semantical base for model checking (automated verification) tools and techniques [16, 15]. The notion of p-morphism is due to Segerberg [54]. Automata were studies from a coalgebraic perspective first in [51].

1.5 Exercises

Exercise 1.5.1. Prove Lemma 1.1.3.

Exercise 1.5.2. Prove Lemma 1.1.7.

Exercise 1.5.3 (Moore machines). Find an operation T on sets such that the T-coalgebras are precisely the Moore machines. A Moore machine with input alphabet I and output alphabet O is tuple $(S, \mathsf{next}, \mathsf{out})$, where $\mathsf{next} : S \times I \to S$ is the state transition function and $\mathsf{out} : S \to O$ associates an output to every state. (Note that, in contrast to Mealy machines, the output only depends on the current state.)

Exercise 1.5.4. For which operation T on sets do we recover labelled transition systems as T-coalgebras? Given a set L (of labels), a labelled transition system is a tuple (S, R), where $R \subseteq S \times L \times S$. In a labelled transition system, a triple $(s, l, s') \in R$ corresponds to a labelled state transition $s \xrightarrow{l} s'$.

Exercise 1.5.5. Stacks are datatypes which come with an operation push, which pushes an element on top of the stack and pop, which returns the topmost element of the stack, and fails, it the stack is empty. Find an operation T on sets such that T-coalgebras are stacks. (Hint: Failure can be modelled using a one element set of errors.)

Exercise 1.5.6. This exercise shows, that the converse of Lemma 1.2.4 is in general false. Find two stream automata $(S, \mathsf{hd}, \mathsf{tl})$ and $(S', \mathsf{hd}', \mathsf{tl}')$ and a behaviour preserving function $f : S \to S'$ which is *not* a homomorphism of stream automata. (A function $f : S \to S'$ is behaviour preserving, if $\mathsf{hd} \circ \mathsf{tl}^n(s) = \mathsf{hd}' \circ \mathsf{tl}'^n(f(s))$ for all $s \in S$ and $n \in \mathbb{N}$.)

Exercise 1.5.7. Show that the extension of both $TX = D \times X$ and $T'X = (O \times X)^I$ to functions, given by Notation 1.2.3 and Notation 1.2.7, yields functors T and T'.

Exercise 1.5.8. Suppose T and S are functors on set. Define an action on functions for the operation $UX = TX + SX$ such that U becomes a functor on sets. Prove that the functor laws (Definition 1.2.9) are fulfilled.

Exercise 1.5.9. Give a description of homomorphism of Automata with Error conditions (Section 1.1.4) using the functoriality of the operation $TX = (O \times (S + E))^I$ established in Exercise 1.5.8.

Exercise 1.5.10. Consider the stream automaton (Z, ζ_1, ζ_2) from Lemma 1.3.1 and show, that for every stream automaton $(S, \mathsf{hd}, \mathsf{tl})$ there is precisely one homomorphism $(S, \mathsf{hd}, \mathsf{tl}) \to (Z, \zeta_1, \zeta_2)$.

Exercise 1.5.11. Prove Lemma 1.3.9.

Chapter 2

Universal Constructions

After having seen many examples of coalgebras, we are now concerned with the general theory. This chapter deals with universal constructions on coalgebras, proves that behavioural equivalence is an equivalence relation and establishes the existence of final coalgebras. In order to formulate (some of) the constructions, it is convenient to use the language of categories, which we introduce briefly in the first section.

2.1 Basic Concepts of Category Theory

In the sequel, we are going to deal with universal constructions, which are most easily formalised in the context of category theory. This section contains the basic definitions.

The motivation for the notion of "category" is, that it formalises a class of mathematical structures, along with structure preserving functions between them. This is done as follows:

Definition 2.1.1. A *category* $\mathbb{C}$ consists of

- (i) A bunch of objects $\mathsf{obj}(\mathbb{C})$
- (ii) A bunch of arrows $\mathbb{C}(A, B)$ for any two $A, B \in \mathsf{obj}(\mathbb{C})$
- (iii) an arrow $1_A \in \mathbb{C}(A, A)$ for all $A \in \mathsf{obj}(\mathbb{C})$
- (iv) An operation $\circ_{ABC} : \mathbb{C}(B, C) \times \mathbb{C}(A, B) \to \mathbb{C}(A, C)$ for any three $A, B, C \in \mathsf{obj}(\mathbb{C})$

(we write $f : A \to B$ if $f \in \mathbb{C}(A, B)$ and $g \circ f$ for $\circ(g, f)$), subject to the following conditions:

(*i*) If $f : A \to B$, $g : B \to C$ and $h : C \to D$, then $h \circ (g \circ f) = (h \circ g) \circ f$.

(*ii*) If $f : A \to B$ and $g : B \to C$, then $1_B \circ f = f$ and $g \circ 1_B = g$.

The arrows are often also called morphisms of the respective category. As already said, the main examples are mathematical structures, endowed with structure preserving maps:

Example 2.1.2. (*i*) The category Set of sets has sets as objects and $\mathsf{Set}(A, B)$ is the set of functions from A to B.

(*ii*) The category Grp of groups has groups as objects and group homomorphisms as arrows.

(*iii*) The category $\mathsf{CoAlg}(T)$ has T-coalgebras as objects; the arrows $f : (C, \gamma) \to (D, \delta)$ are the morphisms of coalgebras.

In these notes, we'll only use the category Set of sets and the category $\mathsf{CoAlg}(T)$ of T-coalgebras. However, many of the concepts we use here can also be fruitfully applied in other categories. Just as there are homomorphisms between groups (or coalgebras, for that matter), there's also a notion of homomorphism of categories. We have already met a special instance: endofunctors on Set.

Definition 2.1.3. Suppose $\mathbb{C}$ and $\mathbb{D}$ are categories. A *functor* $F : \mathbb{C} \to \mathbb{D}$ consists of

(*i*) A mapping $F : \mathsf{obj}(\mathbb{C}) \to \mathsf{obj}(\mathbb{D})$

(*ii*) a mapping $F(A, B) : \mathbb{C}(A, B) \to \mathbb{D}(FA, FB)$

subject to the following conditions (we write Ff for $F(A, B)(f)$):

(*i*) $F1_A = 1_{FA}$ for all $A \in \mathsf{obj}(\mathbb{C})$

(*ii*) $F(g \circ f) = (Fg) \circ (Ff)$ for all $f : A \to B$ and all $g : B \to C \in \mathbb{C}$.

Example 2.1.4. (*i*) Every functor on sets (cf. Definition 1.2.9) is a functor in the sense of the above definition.

(*ii*) The assignment $U(C, \gamma) = C$ and $Uf = f$ defines a functor $U : \mathsf{CoAlg}(T) \to \mathsf{Set}$; this functor is often called the "underlying" or "forgetful" functor.

We will often use the functor U to relate properties of coalgebras with properties of their carrier sets.

2.2 Some Categorical Constructions

This section discusses some categorical constructions on coalgebras. In the next section, we use the constructions presented here in order to prove that behavioural equivalence is transitive.

2.2.1 Coproducts

We begin with looking at coproducts, which we have already met in the first chapter, introduced as Notation 1.1.6. We now generalise the notion of coproduct to categories as follows:

Definition 2.2.1. Suppose $\mathbb{C}$ is a category and $A, B \in \mathbb{C}$. A *coproduct diagram* over A, Bin $\mathbb{C}$ is a triple $(A+B, \mathrm{in}_l : A \to A+B, \mathrm{in}_r : B \to A+B)$, if the following holds:

For all $C \in \mathbb{C}$ and all $f : A \to C$ and all $g : B \to C$, there exists a unique $[f, g] : A+B \to C$ such that $[f, g] \circ \mathrm{in}_l = f$ and $[f, g] \circ \mathrm{in}_r = g$. In this case, $A+B$ is also called the *coproduct* of A and B.

The latter property ("for all …there exists a unique …") is the so-called *universal property of coproducts.* It is often visualised diagrammatically as follows:

$$\begin{tikzcd}
A \arrow[r, "\mathrm{in}_l"] \arrow[dr, "\forall f"'] & A+B \arrow[d, "\exists![f,g]"] & B \arrow[l, "\mathrm{in}_r"'] \arrow[dl, "\forall g"] \\
& C &
\end{tikzcd}$$

We have used the rather suggestive notation $A+B$ for the coproduct and $[f, g]$ for the uniquely defined map. Often, coproducts (and also other constructions) can be explicitely defined in terms of their ingredients. Note that in particular categories, coproduct diagrams may or may not exist.

If they do exist, they provide us with the following proof principle:

Lemma 2.2.2. *Suppose $(A+B, \mathrm{in}_l, \mathrm{in}_r)$ is a coproduct diagram and $f, g : A+B \to C$ are arrows of some category $\mathbb{C}$. Then $f = g$ iff $f \circ \mathrm{in}_l = g \circ \mathrm{in}_l$ and $f \circ \mathrm{in}_r = g \circ \mathrm{in}_r$.*

Proof. Clearly $f = g$ implies $f \circ \mathrm{in}_l = g \circ \mathrm{in}_l$ and $f \circ \mathrm{in}_r = g \circ \mathrm{in}_r$. For the converse, note that both f and g make the diagram

$$\begin{tikzcd}
A \arrow[r, "\mathrm{in}_l"] \arrow[dr, "f\circ\mathrm{in}_l=g\circ\mathrm{in}_l"'] & A+B \arrow[d, "f{,}g"] & B \arrow[l, "\mathrm{in}_r"'] \arrow[dl, "f\circ\mathrm{in}_r=g\circ\mathrm{in}_r"] \\
& C &
\end{tikzcd}$$

commute, hence $f = g$ by the universal property of coproducts. $\square$

The above lemma illustrates how to do proofs with universal properties. We will often apply similar techniques without explicitely formulating the underlying proof principle.

We have already seen that coproducts exist in the category of sets: Lemma 1.1.7 states that $(A+B, \mathrm{in}_l, \mathrm{in}_r)$ is a coproduct diagram for all sets A, B (with $A + B, \mathrm{in}_r$ and in_l given as in Notation 1.1.6). Hence, in Set, coproduct is disjoint union. We now show, that coproducts of coalgebras exist, and that they're also disjoint unions.

Lemma 2.2.3. *Suppose (C, γ) and $(D, \delta) \in \mathsf{CoAlg}(T)$ and let $\overline{\gamma + \delta} = (\gamma + \delta) \circ [T\mathrm{in}_l, T\mathrm{in}_r]$. Then $((C + D, \overline{\gamma + \delta}), \mathrm{in}_l, \mathrm{in}_r)$ is a coproduct diagram in $\mathsf{CoAlg}(T)$.*

Proof. In order to qualify as a coproduct diagram, we first need to show that $((C + D, \overline{\gamma + \delta}), \mathrm{in}_l, \mathrm{in}_r)$ is a diagram, that is, in_l and in_r are morphisms of coalgebras. That is, we have to show that

$$\begin{tikzcd} C \arrow[r, "\mathrm{in}_l"] \arrow[d, "\gamma"'] & C + D \arrow[d, "\overline{\gamma+\delta}"] \\ TC \arrow[r, "T\mathrm{in}_l"'] & T(C + D), \end{tikzcd}$$

and the analogous diagram for the right injection, commute. This follows from calculating

$$\begin{aligned} \overline{\gamma + \delta} \circ \mathrm{in}_l &= [T\mathrm{in}_l, T\mathrm{in}_r] \circ [\mathrm{in}_l \circ \gamma, \mathrm{in}_r \circ \delta] \circ \mathrm{in}_l && \text{expanding } \overline{\gamma + \delta} \\ &= [T\mathrm{in}_l, T\mathrm{in}_r] \circ \mathrm{in}_l \circ \gamma && \text{Universal Property} \\ &= T\mathrm{in}_l \circ \gamma && \text{Universal Property, again} \end{aligned}$$

Now suppose $(E, \epsilon) \in \mathsf{CoAlg}(T)$ and $f : (C, \gamma) \to (E, \epsilon), g : (D, \delta) \to (E, \epsilon)$ are morphisms of coalgebras. We have to show that

- There is a coalgebras morphism $u : (C + D, \overline{\gamma + \delta}) \to (E, \epsilon)$, which satisfies $u \circ \mathrm{in}_l = f$ and $u \circ \mathrm{in}_r = g$, and
- u is unique wrt. this property.

We begin with uniqueness: Suppose $u : (C + D, \overline{\gamma + \delta})$ has the required properties. Since coalgebra morphisms are in particular functions, we can consider $u : C + D$ as a (set theoretic) function satisfying $u \circ \mathrm{in}_l = f$ and $u \circ \mathrm{in}_r = g$. Thus, by the 2.2.2, $u = [f, g]$ and u is the unique function with the property $u \circ \mathrm{in}_l = f$, $u \circ \mathrm{in}_r = g$. We thus have to show that $u = [f, g]$ is actually a morphism of coalgebras, i.e.

$$\begin{tikzcd}
C + D \arrow[r, "{[f,g]}"] \arrow[d, "\overline{\gamma+\delta}"'] & E \arrow[d, "\epsilon"] \\
T(C + D) \arrow[r, "{T[f,g]}"'] & TE
\end{tikzcd}$$

commutes. Using Lemma 2.2.2, it suffices to show that

$$T[f, g] \circ \overline{\gamma + \delta} \circ \mathrm{in}_l = \epsilon \circ [f, g] \circ \mathrm{in}_l$$

plus the corresponding analogous equation for the right injection in_r. The above equation follows from

$$\begin{aligned}
T[f, g] \circ \overline{\gamma + \delta} \circ \mathrm{in}_l &= T[f, g] \circ [T\mathrm{in}_l, T\mathrm{in}_r] \circ [\mathrm{in}_l \circ \gamma, \mathrm{in}_r \circ \delta] \circ \mathrm{in}_l && \text{Expanding } \gamma + \delta \\
&= T[f, g] \circ [T\mathrm{in}_l, T\mathrm{in}_r] \circ \mathrm{in}_l \circ \gamma && \text{Univ. Prop.} \\
&= T[f, g] \circ T\mathrm{in}_l \circ \gamma && \text{Univ. Prop., again} \\
&= T([f, g] \circ \mathrm{in}_l) \circ \gamma && T \text{ is functorial} \\
&= Tf \circ \gamma && \text{Univ. Prop., again} \\
&= \epsilon \circ f && f \in \mathsf{CoAlg}(T) \\
&= \epsilon \circ [f, g] \circ \mathrm{in}_l && \text{Univ. Prop., again}
\end{aligned}$$

as required. The case for in_r is analogous. □

Note that the coproduct in $\mathsf{CoAlg}(T)$ was constructed just as the coproduct in Set. To express this observation formally, we set the forgetful functor to relate $\mathsf{CoAlg}(T)$ with Set. In general, we use the following terminology:

Definition 2.2.4. Suppose $F : \mathbb{C} \to \mathbb{D}$ is a functor. We say that F *preserves coproducts*, if (FC, Ff, Fg) is a coproduct diagram (in $\mathbb{D}$), whenever (C, f, g) is a coproduct diagram in $\mathbb{C}$.

In the proof of Lemma 2.2.3, we have shown that $U : \mathsf{CoAlg}(T) \to \mathsf{Set}$ preserves coproducts. We note this as corollary:

Corollary 2.2.5. *The forgetful functor $U : \mathsf{CoAlg}(T) \to \mathsf{Set}$ preserves coproducts.*

2.2.2 Coequalisers

In the last section, we have given a categorical description of disjoint unions (as coproducts). Now, we discuss another such construction, namely quotienting. In category theory, this is formalised using the notion of coequalisers:

Definition 2.2.6. Suppose $f, g : A \to B$ are arrows of some category $\mathbb{C}$. A *coequaliser diagram* over f and g is a triple (C, c) where $C \in \mathbb{C}$ and $c : B \to C$, which satisfies the following property:

For all $D \in \mathbb{C}$ and all $h : B \to D$ there is a unique $u : C \to D$ such that $u \circ c = h$.

As with coproducts, there is the following diagrammatic visualisation:

$$\begin{tikzcd}
A \arrow[r, shift left, "f"] \arrow[r, shift right, "g"'] & B \arrow[r, "c"] \arrow[dr, "\forall h"'] & C \arrow[d, "\exists! u"] \\
 & & D
\end{tikzcd}$$

Coequalisers allow us to construct the quotient of a set by an equivalence relation:

Example 2.2.7. Suppose B is a set and $R \subseteq B \times B$ is an equivalence relation. We denote the equivalence class of $b \in B$ wrt. R by $[b]$; $c : B \to B/R$ is the canonical projection. Then $(B/R, c)$ is a coequaliser diagram over the projections π_1 and π_2, both mapping $R \to B$ (see Exercise 2.10.1).

In general however, the arrows of which we construct coequalisers are not the projections of an equivalence relation. The construction of coequalisers in the general case takes the quotient by the generated equivalence relation. This works as follows:

Definition 2.2.8. Suppose $R \subseteq B \times B$ is a relation. The *equivalence generated by R* is the least equivalence relation which contains R; this relation is denoted by R^*.

Note that generated equivalences always exist: one takes the intersection of all equivalence relations which contain R. Using this terminology, we can describe the construction of coequalisers in Set as follows:

Lemma 2.2.9. *Suppose $f, g : A \to B$ are functions and R^* is the equivalence generated by $R = \{(f(a), g(a)) \mid a \in A\}$; we denote the canonical projection by $c : B \to B/R$. Then $(B/R, c)$ is a coequaliser diagram.*

Proof. Suppose D is any set and $h : B \to D$ is any function satisfying $h \circ f = h \circ g$. We have to show that

- There's $u : B/R^* \to D$ with $u \circ c = h$
- u is unique wrt. this property.

Denoting the equivalence class of $b \in B$ wrt. R by $[b]$, we can define u by $u([b]) = h(b)$. We first show that u is well defined. To this end, consider $\mathsf{Ke}(h) = \{(b, b') \in B \times B \mid h(b) = h(b')\}$. Then $\mathsf{Ke}(h)$ is an equivalence relation and $\mathsf{Ke}(h) \supseteq R$ (since $(b, b') \in R \implies (b, b') = (f(a), g(a))$ for some $a \in A$ and $h(b) = h \circ f(a) = h \circ g(a) = h(b')$, hence $(b, b') \in S$), from which we conclude $\mathsf{Ke}(h) \supseteq R^*$ by definition of R^*. Now $[b] = [b']$ implies $(b, b') \in R^*$, which in turn implies $(b, b') \in \mathsf{Ke}(h)$, that is, $h(b) = h(b')$. The equation $u \circ c = h$ is immediate from the definition of u. This leaves uniqueness: Suppose $v : B/R^* \to D$ satisfies $v \circ c = h$. Then $v([b]) = v \circ c(b) = h(b) = u([b])$ by definition of u, hence $u = v$. □

The construction of coequalisers in Set was quite cumbersome. Fortunately, we don't have to repeat this construction in the category $\mathsf{CoAlg}(T)$: we just have to know about the universal property, which characterises coequalisers.

The next proposition transports the construction of coequalisers to $\mathsf{CoAlg}(T)$.

Proposition 2.2.10. *Let $f, g : (A, \alpha) \to (B, \beta) \in \mathsf{CoAlg}(T)$ and suppose (C, c) is a coequaliser diagram over f and g in Set. Then $((C, \gamma), c)$ is a coequaliser diagram in $\mathsf{CoAlg}(T)$, where γ is the unique morphism satisfying $\beta \circ Tc = \gamma \circ c$..*

Proof. We first show that our construction is well defined, that is, $(Tc \circ \beta) \circ f = (Tc \circ \beta) \circ g$; otherwise our definition of γ would be nonsensical. We have

$$\begin{aligned}
Tc \circ \beta \circ f &= Tc \circ Tf \circ \alpha && \text{since } f \in \mathsf{CoAlg}(T)\\
&= T(c \circ f) \circ \alpha && \text{Functoriality of } T\\
&= T(c \circ g) \circ \alpha && \text{Property of coequalisers}\\
&= Tc \circ Tg \circ \alpha && \text{as above}\\
&= Tc \circ \beta \circ g && \text{as above,}
\end{aligned}$$

hence we can speak of the unique γ for which

$$\gamma \circ c = Tc \circ \beta$$

Note that property (2.2.2) says that $c : (B, \beta) \to (C, \gamma)$ is a morphism of coalgebras. In order to unmask $((C, \gamma), c)$ as a coequaliser diagram, suppose $(D, \delta) \in \mathsf{CoAlg}(T)$ and $h : (B, \beta) \to (D, \delta)$ is a morphism. We need to show that

- There's $u : (C, \gamma) \to (D, \delta) \in \mathsf{CoAlg}(T)$ such that $u \circ c = h$
- u is the only morphism with that property.

Since we have assumed that (C, c) si a coequaliser diagram in Set, there's a unique $u : C \to D$ which satisfies $u \circ c = h$. Hence we have to show that u is a morphism of coalgebras, that is,

$$\begin{tikzcd}
C \arrow[r, "u"] \arrow[d, "\gamma"'] & D \arrow[d, "\delta"] \\
TC \arrow[r, "Tu"'] & TD
\end{tikzcd}$$

commutes. You are asked to do this yourself in Exercise 2.10.2. □

As in the case of coproducts, we have constructed coequalisers as in the category of sets, and then provided them with a transition structure. Using the obvious analogy of Definition 2.2.4, we note:

Corollary 2.2.11. *The forgetful functor $U : \mathsf{CoAlg}(T) \to \mathsf{Set}$ preserves coequalisers.*

2.2.3 Pushouts

We discuss one last categorical construction, pushouts. They allow us to form unions of systems, where certain parts are identified The categorical definition is the following:

Definition 2.2.12. Suppose $f : A \to B$ and $g : A \to C$ are arrows in some category $\mathbb{C}$. A *pushout diagram* over f, g is a triple (D, h, i) with $D \in \mathbb{C}$ and $h : B \to D$ and $i : C \to D$, if the following universal property is satisfied:

For all $X \in \mathbb{C}$ and all $x : B \to X$, $y : C \to X$ with $x \circ f = y \circ g$ there is a unique $u : D \to X$ such that $u \circ i = y$ and $u \circ h = x$.

Diagrammatically, the situation is as follows:

$$\begin{tikzcd}
A \arrow[r, "f"] \arrow[d, "g"'] & B \arrow[d, "h"] \arrow[ddr, bend left, "\forall x"] & \\
C \arrow[r, "i"'] \arrow[drr, bend right, "\forall y"'] & D \arrow[dr, "\exists! u"] & \\
& & X
\end{tikzcd}$$

One example of pushouts is the union of sets:

Example 2.2.13. Suppose B, C are sets and $A = B \cap C$ and $D = B \cup C$. Then (D, i_B, i_C), where $i_B : B \to D$ and $i_C : C \to D$ are the inclusions, is a pushout diagram. To see this, suppose X is another set, $x : B \to X$ and $y : C \to X$ satisfy $x \circ j_B = y \circ j_C$, where $j_B : A \to B$ and $j_C : A \to C$ are the inclusions. We define $u : D \to X$ by $u(d) = x(d)$ if $d \in B$ and $u(d) = y(d)$ if $d \in C$. Note that u is well defined, since for $d \in B \cap C$ we have $x(d) = y(d)$ by assumption on x and y. It is easy to see that u has the required property.

In the sequel, we're not only interested in pushouts of inclusion mappings. It turns out, that in the general case, pushouts can be constructed from coproducts and coequalisers:

Proposition 2.2.14. *Suppose $f : A \to B$ and $g : A \to C$ are arrows of some category $\mathbb{C}$. If $(B + C, \mathrm{in}_l, \mathrm{in}_r)$ is a coproduct diagram over B and C and (D, d) is a coequaliser diagram over $\mathrm{in}_l \circ f$ and $\mathrm{in}'_r \circ g$, then $(D, d \circ \mathrm{in}_l, d \circ \mathrm{in}_r)$ is a pushout diagram over f and g.*

Proof. Suppose $x : B \to X$ and $y : C \to X$ satisfy $x \circ f = g \circ y$. Consider the following diagram:

$$\begin{tikzcd}
A \arrow[r, "f"] \arrow[d, "g"'] & B \arrow[d, "\mathrm{in}_l"] \arrow[rrddd, bend left, "x"] & & \\
C \arrow[r, "\mathrm{in}_r"'] \arrow[rrrdd, bend right, "y"'] & B+C \arrow[dr, "d"] \arrow[rrdd, bend right, "{[x,y]}"'] & & \\
& & D \arrow[dr, "u"] & \\
& & & X
\end{tikzcd}$$

and the map $[x, y] : B + C \to X$. Since $[x, y] \circ \mathrm{in}_l \circ f = x \circ f = y \circ g = [x, y] \circ \mathrm{in}_r \circ y$ and (D, d) is a coequaliser diagram over $\mathrm{in}_l \circ f$ and $\mathrm{in}_r \circ g$, there is a unique $u : D \to X$ such that $v \circ d = [x, y]$.

To see that $(D, d \circ \mathrm{in}_l, d \circ \mathrm{in}_r)$ is a pushout diagram, we have to show that

- $u \circ d \circ \mathrm{in}_l = x$ and $u \circ d \circ \mathrm{in}_r = y$
- u is unique wrt. this property.

The equations for u follow from $u \circ d \circ \mathrm{in}_l = [x, y] \circ \mathrm{in}_l = x$; the equation for y is derived analogously.

Now suppose $v : D \to X$ satisfies $v \circ d \circ \mathrm{in}_l = x$ and $v \circ d \circ \mathrm{in}_r = y$. Then $v \circ d = [x, y]$ by the universal property of coproducts (cf. Lemma 2.2.2). But u is unique wrt. $u \circ d = [x, y]$, hence $u = v$. $\square$

Since both coproducts and coequalisers exist in our two categories Set and $\mathsf{CoAlg}(T)$ of interest, we conclude that we can construct pushouts in both. Since pushouts were constructed using coproducts and coequalisers, and both are preserved by the forgetful functor $U : \mathsf{CoAlg}(T) \to \mathsf{Set}$, have that U also preserves pushouts.

Corollary 2.2.15. *The forgetful functor $U : \mathsf{CoAlg}(T) \to \mathsf{Set}$ preserves pushouts.*

2.3 More on Behavioural Equivalence

In this section, we use the constructions presented before and establish an important property of behavioural equivalence: its transitivity. Our main theorem is the following:

Theorem 2.3.1 (Behavioural Equivalence is Transitive). *Suppose $(C,\gamma),(D,\delta)$ and $(E,\epsilon) \in \mathsf{CoAlg}(T)$ and $(c,d,e) \in C \times D \times E$. Then $c \approx d$ and $d \approx e$ implies $c \approx e$.*

In order to show that $c \approx e$, we have to construct two morphisms into some T-coalgebra, which identify c and e. This is where the constructions of the last chapter come in handy. Using pushouts, we can prove the above theorem as follows:

Proof. Since $c \approx d$, there is $(F,\phi) \in \mathsf{CoAlg}(T)$ and a pair of morphisms $f : (C,\gamma) \to (F,\phi)$ and $g : (D,\delta) \to (F,\phi)$ with $f(c) = g(d)$. Because $d \approx e$, we find $(G,\rho) \in \mathsf{CoAlg}(T)$, $h : (D,\delta) \to (G,\rho)$ and $i : (E,\epsilon) \to (G,\rho)$ with $h(d) = i(e)$. Consider the pushout (I,ι) of g and h in $\mathsf{CoAlg}(T)$:

$$\begin{tikzcd}
C \arrow[dr, "f"'] & & D \arrow[dl, "g"'] \arrow[dr, "h"] & & E \arrow[dl, "i"] \\
& F \arrow[dr, "j"'] & & G \arrow[dl, "k"] & \\
& & I & &
\end{tikzcd}$$

Since we have taken the pushout in $\mathsf{CoAlg}(T)$, all arrows in the above diagram are coalgebra morphisms, in particular, $j \circ f$ and $k \circ i$. Now $j \circ f(c) = j \circ g(d) = k \circ h(d) = i \circ k(e)$ by assumption on f, g, h and i and the fact that (I,ι) is the vertex of a pushout diagram. $\square$

We obtain the following immediate corollary:

Corollary 2.3.2 (Behavioural Equivalence is an Equivalence Relation). *Suppose $(C,\gamma) \in \mathsf{CoAlg}(T)$. Then $\approx_C = \{(c,c') \in C \times C \mid c \approx c'\}$ is an equivalence relation.*

We can also use pushouts to give a description of behavioural equivalence on a single system, which is simpler that the original definition:

Proposition 2.3.3. *Suppose $(C, \gamma) \in \mathsf{CoAlg}(T)$ and $c, c' \in C$. The following are equivalent:*

(i) $c \approx c'$

(ii) There are (E, ϵ) and $e : (C, \gamma) \to (E, \epsilon) \in \mathsf{CoAlg}(T)$ with $f(c) = f(c')$.

Note that the original definition is in terms of a pair of morphisms. The proof is easy:

Proof. Clearly the second statement implies the first. For the converse, suppose $c \approx d$, that is, there $(D, \delta), f : (C, \gamma) \to (D, \delta)$ and $g : (C, \gamma) \to (D, \delta)$ with $f(c) = g(c')$. Consider a pushout $((E, \epsilon), h, i)$ over f and g. We have a commuting diagram

$$\begin{tikzcd} C \arrow[r, "f"] \arrow[d, "g"'] & D \arrow[d, "h"] \\ D \arrow[r, "i"'] & E \end{tikzcd}$$

where all vertices are coalgebras (we have suppressed the structure maps) and all edges are morphisms. Thus putting $e = h \circ f = i \circ g$ does the job. □

Given an equivalence relation R on the carrier of some T-coalgebra (C, γ), it is natural to ask whether we can form the quotient $(C/R, \gamma/R)$, which involves the definition of a transition structure $\gamma/R : C/R \to T(C/R)$ on the quotient. This is of particular interest when R is behavioural equivalence on (C, γ), since then $(C/R, \gamma/R)$ is the same as the original system (C, γ), but with all observationally equivalent states identified. We think of quotienting by behavioural equivalence as abstracting away non-observable details. In order to define the transition structure γ/R on the quotient, we have to deal with images first.

2.4 Images and Factorisations

It is well known that we can factor every function $f : A \to B$ as $f = m \circ e$, where I is the image of f in B. In particular, every function factors as the composition of a surjection and an injection:

$$\begin{tikzcd} A \arrow[r, "f"] \arrow[d, "e"'] & B \\ I \arrow[ur, "m"'] & \end{tikzcd}$$

This factorisation is unique in the following sense:

Proposition 2.4.1. *Suppose $f : A \to B$ is a function which factors as $f = m \circ e$ and $f = m' \circ e'$, where*

- *$e : A \to I$ is surjective, $m : I \to B$ is injective, and*
- *$e' : A \to I'$ is surjective, $m' : I' \to B$ is injective.*

Then there is a unique bijection $i : I \to I'$ such that

$$\begin{tikzcd}
& I \arrow[dd, "i"'] \arrow[dr, "m"] & \\
A \arrow[ur, "e"] \arrow[dr, "e'"'] & & B \\
& I' \arrow[ur, "m'"'] &
\end{tikzcd}$$

commutes.

Proof. Exercise 2.10.5. □

We now lift this property to the category $\mathsf{CoAlg}(T)$ of T-coalgebras. Since every morphism of coalgebras is in particular a set theoretic function, clearly every morphism factors as a surjection followed by an injection. What's not clear is, that this is a factorisation in $\mathsf{CoAlg}(T)$, i.e. that both the injection and the surjection are actually morphisms of coalgebras. Our result is the following:

Theorem 2.4.2. *Suppose $f : (C, \gamma) \to (D, \delta) \in \mathsf{CoAlg}(T)$. Then there is $(I, \iota) \in \mathsf{CoAlg}(T)$ and a pair of morphisms $e : (C, \gamma) \to (I, \iota)$ and $m : (I, \iota) \to (D, \delta)$ such that*

(i) $f = m \circ e$

(ii) f is injective and g is surjective.

Proof. Let $I = f[C] = \{f(c) \mid c \in C\} \subseteq D$ denote the direct image of f. Clearly we have set theoretic functions $e : C \to I$ and $m : I \to D$ which satisfy $f = m \circ e$, e surjective and m injective (put $e(c) = f(c)$ and $m(d) = d$). We have to show that e and m lift to homomorphisms of coalgebras, that is, we need to find a transition structure $\iota : I \to TI$ such that the diagram

$$\begin{tikzcd}
C \arrow[r, "e"] \arrow[d, "\gamma"'] & I \arrow[r, "m"] \arrow[d, "\iota"'] & D \arrow[d, "\delta"'] \arrow[l, bend right, "\bar{m}"'] \\
TC \arrow[r, "Te"'] & TI \arrow[r, "Tm"'] & TD
\end{tikzcd}$$

commutes. We assume wlog that $I \neq \emptyset$, for if $I = \emptyset$, we have $C = \emptyset$ and taking the empty function for ι makes both squares commute trivially. So assume $I \neq \emptyset$.

Since m is injective and $I \neq \emptyset$, there is a left inverse $\bar{m} : D \to I$ of m, that is, $\bar{m}$ satisfies $\bar{m} \circ m = \mathrm{id}_I$. We put $\iota = T\bar{m} \circ \delta \circ m$. To see that this definition turns e and m into homomorphisms, we have to show

- $Te \circ \gamma = \iota \circ e$
- $Tm \circ \iota = \delta \circ m$.

For the first equation, we calculate

$$\begin{aligned}
\iota \circ e &= T\bar{m} \circ \delta \circ m \circ e && \text{Def'n of } \iota \\
&= T\bar{m} \circ Tm \circ Te \circ \gamma && f = m \circ e \text{ is a morphism plus functor laws} \\
&= Te \circ \gamma && \bar{m} \circ m = \mathrm{id}_I \text{ plus functor laws}
\end{aligned}$$

For the second equation, it suffices to show that $Tm \circ \iota \circ e = \delta \circ m \circ e$ since e is a surjection. We obtain

$$\begin{aligned}
Tm \circ \iota \circ e &= Tm \circ T\bar{m} \circ \delta \circ m \circ e && \text{Def'n of } \iota \\
&= Tm \circ T\bar{m} \circ Tm \circ Te \circ \gamma && f = m \circ e \text{ is a morphism, functor laws} \\
&= Tm \circ Te \circ \gamma && \bar{m} \circ m = \mathrm{id}_I\text{, functor laws} \\
&= \delta \circ m \circ e && f = m \circ e \text{ is a morphism, functor laws}
\end{aligned}$$

which finishes the proof. □

2.5 Quotients modulo Behavioural Equivalence

We now describe the quotient construction on coalgebras. We're not completely general here in that we only discuss quotients modulo behavioural equivalence. We can always form the quotient of the carrier of some system modulo behavioural equivalence, obtaining a set (of equivalence classes). The following proposition guarantees the existence of a unique transition structure on this set:

Proposition 2.5.1. *Suppose $(C, \gamma) \in \mathsf{CoAlg}(T)$ and let $\approx = \{(c, c') \in C \times C \mid c \approx c'\}$. Then there is a unique $\gamma/\approx : C/\approx \to T(C/\approx)$ such that the canonical projection $p : C \to C/\approx$ is a homomorphism of coalgebras.*

Proof. Consider the diagram

$$\begin{tikzcd}
C \arrow[r, "p"] \arrow[d, "\gamma"'] & C/{\approx} \arrow[d, "\gamma/{\approx}"] \\
TC \arrow[r, "Tp"'] & T(C/{\approx}).
\end{tikzcd}$$

First suppose that there are $\gamma_0, \gamma_1 : C/{\approx} \to T(C/{\approx})$ with $\gamma_1 \circ p = Tp \circ \gamma = \gamma_2 \circ p$. If $[c] \in C/{\approx}$, we have $\gamma_1([c]) = \gamma_1 \circ p(c) = \gamma_2 \circ p(c) = \gamma_2([c])$. Hence $\gamma/{\approx}$ is unique. It remains to show existence. The above considerations lead us to define

$$\gamma/{\approx}\ ([c]) = Tp \circ \gamma(c),$$

and we now need to show that this is in fact a definition, that is for $[c] = [c']$, we have $Tp \circ \gamma(c) = Tp \circ \gamma(c')$. So suppose $c, c' \in C$ with $[c] = [c']$, that is, $p(c) = p(c')$, that is, $c \approx c'$. By Proposition 2.3.3, there are (E, ϵ) and $e : (C, \gamma) \to (E, \epsilon) \in \mathsf{CoAlg}(T)$ with $e(c) = e(c')$. By Theorem 2.4.2, we can assume that e is a surjection (for if not, replace E by the image of e). Consider the diagram

$$\begin{tikzcd}
C \arrow[r, "p"'] \arrow[rr, bend left, "e"] \arrow[d, "\gamma"'] & C/{\approx} & E \arrow[l, "f"] \arrow[d, "\epsilon"] \\
TC \arrow[r, "Tp"] \arrow[rr, bend right, "Te"'] & T(C/{\approx}) & TE, \arrow[l, "Tf"']
\end{tikzcd}$$

where $p : C \to C/{\approx}$ is the canonical projection and $f : E \to C/{\approx}$ is defined by $f(e(c)) = p(c)$, where $c \in C$. Note that f is well defined, since $e(c_0) = e(c_1)$ implies $c_0 \approx c_1$. Clearly f is a surjection and $f \circ e = p$.

We are now in the position to tackle our goal, that is, to show that $Tp \circ \gamma(c) = Tp \circ \gamma(c')$. We do this by calculating

$$\begin{aligned}
Tp \circ \gamma(c) &= Tf \circ Te \circ \gamma(c) && f \circ e = p \text{ plus functor laws} \\
&= Tf \circ \epsilon \circ e(c) && e \text{ is a morphism} \\
&= Tf \circ \epsilon \circ e(c') && \text{by assumption on } e \\
&= Tp \circ \gamma(c). && \text{as above}
\end{aligned}$$

□

Note that in the above proof, we have used quite a number of concepts treated earlier. We have used pushouts (by appealing to Proposition 2.3.3) and images (which have allowed us to assume e surjective).

2.6 Simple and Extensive Systems

We now discuss two important classes of systems: simple systems and extensive systems. The importance of simple systems lies in the fact that they allow proofs by coinduction, while extensive systems have the property that they incorporate all the possible observable behaviour.

We begin with simple systems. As mentioned, simple systems allow proofs by coinduction, that is, they satisfy the coinductive proof principle:

Definition 2.6.1. Suppose $(C, \gamma) \in \mathsf{CoAlg}(T)$. We say that (C, γ) is *simple* or (C, γ) *satisfies the coinduction proof principle*, if $c = c'$ whenever $c \approx c'$.

That is, in order to show that two states are equal, it suffices to show that they are behaviourally equivalent (the next chapter discusses two methods for establishing that two states are behaviourally equivalent).

We collect some handy properties of simple systems:

Lemma 2.6.2. *Suppose* $(C, \gamma) \in \mathsf{CoAlg}(T)$ *is simple.*

(i) For all $(D, \delta) \in \mathsf{CoAlg}(T)$ *there is at most one morphism* $f : (C, \gamma) \to (D, \delta)$ *of coalgebras.*

(ii) If $(D, \delta) \in \mathsf{CoAlg}(T)$ *and* $f : (D, \delta) \to (C, \gamma)$ *is a morphism of coalgebras, then* f *is injective.*

Proof. Suppose (C, γ) is simple and $f, g : (D, \delta) \to (C, \gamma)$ are two morphisms. Let $c \in C$. Then $f(c) \approx g(c)$, hence $f(c) = g(c)$, thus $f = g$.

To see that any morphism is injective, suppose $f : (D, \delta) \to (C, \gamma) \in \mathsf{CoAlg}(T)$. If $d, d' \in D$ with $f(d) = f(d')$, we have $d \approx f(d) = f(d') \approx d'$, hence $d = d'$ by simplicity. □

Note that there are cases where no morphism between two T-coalgebras exists at all (see Exercise 2.10.4).

The other class of systems, which we briefly mention here are called extensive. Note that this terminology is somewhat non-standard, at least I didn't find it in the literature.

Definition 2.6.3. A system $(C, \gamma) \in \mathsf{CoAlg}(T)$ is called *extensive*, if for all other systems (D, δ) and all $d \in D$, there is $c \in C$ with $c \approx d$.

In other words, an extensive system realises every possible observable behaviour. Extensive systems have a somewhat complimentary property compared to simple systems:

Lemma 2.6.4. *Let $(C, \gamma) \in \mathsf{CoAlg}(T)$. If, for all $(D, \delta)_{\in}\mathsf{CoAlg}(T)$ there is a morphism $f : (D, \delta) \to (C, \gamma) \in \mathsf{CoAlg}(T)$, then (C, γ) is extensive.*

Proof. Because morphisms preserve observable behaviour. □

The notion of extensivity is not too important by itself, but it can be used to give a characterisation of final systems, which we are going to do in the next section.

2.7 Final Coalgebras

We now introduce a concept which will accompany us right until the end of these notes: final coalgebras. The import of final coalgebras lies in the fact that they provide an abstract model of all possible observations. We will first give the abstract definition and then the promised characterisation in terms of simplicity and extensivity.

Definition 2.7.1. Suppose $(C, \gamma) \in \mathsf{CoAlg}(T)$. We say that (C, γ) is *final*, if there is a unique morphism of coalgebras $(C, \gamma) \to (D, \delta)$ for all $(D, \delta) \in \mathsf{CoAlg}(T)$.

This already implies that a final system contains every possible observable behaviour. The promised characterisation of final coalgebras is now as follows:

Theorem 2.7.2. *Suppose $(C, \gamma) \in \mathsf{CoAlg}(T)$. The following are equivalent:*

(i) (C, γ) is final.

(ii) (C, γ) is simple and extensive.

Proof. First suppose that (C, γ) is final. Lemma 2.6.4 shows that (C, γ) is simple. If (D, δ) is arbitrary and $u : (C, \gamma) \to (D, \delta)$ is the (unique) morphism given by finality, note that $d \approx u(d)$, hence (C, γ) is extensive.

Now suppose (C, γ) is simple and extensive. Let $(D, \delta) \in \mathsf{CoAlg}(T)$ and consider the relation $R = \{(d, c) \in D \times C \mid d \approx c\}$. If (d, c) and $(d, c') \in R$, we have $c \approx d \approx c'$, hence $c \approx c'$ thus $c = c'$ since (C, γ) is simple. Also, for any $d \in D$, there is a $c \in C$ with $(d, c) \in R$ by extensivity. Thus R is functional, allowing us to consider $u : D \to C$, defined by

$$u(d) = \text{the unique } c \text{ such that } (d, c) \in R.$$

We first show that f is a morphism of coalgebras, that is, u makes the diagram

$$\begin{tikzcd} D \arrow[r, "u"] \arrow[d, "\delta"'] & C \arrow[d, "\gamma"] \\ TD \arrow[r, "Tu"'] & TC \end{tikzcd}$$

commute. If $C = \emptyset$, we have $D = \emptyset$ and the claim is trivial. So assume wlog that $C \neq \emptyset$. Let $d \in D$. Then $d \approx u(d)$ by definition, hence there is (E_0, ϵ_0) and $f : (D, \delta) \to (E_0, \epsilon_0)$ and $g : (C, \gamma) \to (E_0, \epsilon_0)$ with $f_0(c) = g_0(u(c))$. Now let $\approx_{E_0} = \{(e, e') \in E \times E \mid e \approx e'\}$. By Theorem 2.3.1, $\approx_{E_0}$ is an equivalence. Put $E = E_0/{\approx_E}$. By Proposition 2.5.1 there is a unique transition structure $\epsilon : E \to TE$ such that the canonical projection $E_0 \to E_0/{\approx_{E_0}}$ is a morphism of coalgebras. Now let $f = p \circ f_0$ and $g = p \circ g_0$. We claim that

$$\begin{tikzcd} D \arrow[rr, "u"] \arrow[dr, "f"'] & & C \arrow[dl, "g"] \\ & E & \end{tikzcd}$$

commutes. For an arbitrary $d_0 \in D$, we have $f(d_0) \approx d_0 \approx u(d_0) \approx g \circ u(d_0)$, thus $f(d_0) = g \circ u(d_0)$ since (E, ϵ) is simple.

Furthermore, g is an injection by Lemma 2.6.2. Since we have assumed $C \neq \emptyset$, g has a left inverse $\bar{g} : E \to C$ satisfying $\bar{g} \circ g = \mathrm{id}_C$. Now consider

$$\begin{tikzcd} D \arrow[r, "u"] \arrow[d, "\delta"'] & C \arrow[r, "g"] \arrow[d, "\gamma"] & E \arrow[d, "\epsilon"] \\ TD \arrow[r, "Tu"'] & TC \arrow[r, "Tg"'] & TE, \end{tikzcd}$$

the right hand side commutes since g is a morphism of coalgebras. In order to show that the left side commutes, too, it suffices to show that $Tg \circ \gamma \circ u = Tg \circ Tu \circ \delta$, since Tg is injective with left inverse $T\bar{g}$. Now:

$$\begin{aligned} Tg \circ \gamma \circ u &= \epsilon \circ g \circ u && g \text{ is a morphism} \\ &= \epsilon \circ f && f = g \circ u, \text{ see above} \\ &= Tf \circ \delta && f \text{ is a morphism} \\ &= T(g \circ u) \circ \delta && f = g \circ u, \text{ see above} \\ &= Tg \circ Tu \circ \delta && \text{functoriality of } T. \end{aligned}$$

It follows from Lemma 2.6.2, that u is the only morphism of coalgebras mapping $(D, \delta) \to (C, \gamma)$. □

2.8 Simple Properties of Final Systems

This section collects some simple properties of final systems. First, final systems are unique:

Lemma 2.8.1. *Suppose (Z, ζ) and (Z, ζ') are final. Then $(Z, \zeta) \cong (Z', \zeta')$.*

Proof. Finality of both systems provides us with two mutually inverse morphisms $f : (Z, \zeta) \to (Z', \zeta')$ and $g : (Z', \zeta') \to (Z, \zeta)$. $\square$

The second property, known as "Lambek's Lemma", can be used (among other applications) to show, that final systems don't always exist:

Theorem 2.8.2. *If (Z, ζ) is final, then ζ is a bijection.*

Proof. Consider

$$\begin{tikzcd}
Z \arrow[r, "\zeta"] \arrow[d, "\zeta"'] & TZ \arrow[r, "!"] \arrow[d, "T\zeta"'] & Z \arrow[d, "\zeta"] \\
TZ \arrow[r, "T\zeta"'] & FT^2Z \arrow[r, "T!"'] & TZ
\end{tikzcd}$$

where $! : (TZ, T\zeta) \to (Z, \zeta)$ is the unique T-morphism. By the universal property of (Z, ζ), $! \circ \zeta = \mathrm{id}_Z$.

Now look at

$$\begin{tikzcd}
TZ \arrow[r, "!"] \arrow[d, "T\zeta"'] & Z \arrow[d, "\zeta"] \\
T^2Z \arrow[r, "T!"] \arrow[d, "T!"'] & TZ \arrow[d, "\mathrm{id}_{TZ}"] \\
TZ \arrow[r, "\mathrm{id}_{TZ}"'] & TZ
\end{tikzcd}$$

For the left vertical arrow we have $T! \circ T\zeta = T(! \circ \zeta) = T\mathrm{id}_Z = \mathrm{id}_{TZ}$, hence $\zeta \circ ! = \zeta \circ ! \circ \mathrm{id}_{TZ} = \mathrm{id}_{TZ} \circ T! \circ T\zeta = \mathrm{id}_{TZ}$. $\square$

This can be used to show, that the covariant power set functor does not admit a final coalgebra:

Corollary 2.8.3. *There is no final $\mathcal{P}$-coalgebra.*

The next chapter addresses the question regarding when final coalgebras do exist. Moreover, we illustrate one particular area, where they come in handy: Proofs and definitions by coinduction.

2.9 Notes

The notions of products, equalisers, pullbacks, as well as their duals are instances of the general context of limit (resp. colimit) in a category. The treatment of limits in the general case is beyond the scope of these notes, but can be found in any textbook on category theory, e.g. [36, 11, 45, 8]. Our construction of pushouts is an instance of a general theorem: In any category, the existence of finite / arbitrary products and equalisers suffices to construct finite / arbitrary limits. We have not treated uniqueness of our categorical constructions: viewing them as limits, it can easily be seen that they are unique up to morphisms of the appropriate structure. We have only discussed the existence of colimits, i.e. coproducts, coequalisers and pushouts. The construction of limits (comprising products, equalisers and pullbacks) is far more involved; see [34, 61, 46, 21].

2.10 Exercises

Exercise 2.10.1. Show that $(B/R, c)$ is indeed a coequaliser diagram in Example 2.2.7.

Exercise 2.10.2. Complete the proof of Proposition 2.2.10 and show, that u, as constructed in the proof, is indeed a morphism of coalgebras. *Hint:* Use the universal property of coequalisers in a way similar to Lemma 2.2.2.

Exercise 2.10.3. In which sense are coproduct diagrams, coequaliser diagrams and pushout diagrams determined uniquely? *Hint:* Consider a "canonical" notion of morphism between different coproduct diagrams (resp. coequaliser diagrams, pushout diagrams).

Exercise 2.10.4. Give an example of two T-coalgebras, for some particular signature functor T, such there is no morphism from one coalgebra to the other.

Exercise 2.10.5. Prove Proposition 2.4.1.

Exercise 2.10.6. Give a description of the final Mealy machine in elementary terms.

Exercise 2.10.7. Show that a behaviour preserving function (in the sense of Exercise 1.5.6) is a morphism of stream automata, if its codomain is simple (Exercise 1.5.6 has asked you to construct a counterexample in the general case).

Chapter 3

Definitions and Proofs by Coinduction

This chapter is devoted to coinduction, both as a definition principle and as a proof principle. We illustrate the algebra / coalgebra duality by deriving two coinductive definition principles from their algebraic counterparts iteration and primitive recursion. Since coinductive definitions only work on final coalgebras, we start with a theorem which establishes that these guys actually do exist.

3.1 Final Coalgebras

If we want that final coalgebras exist, we have to put some restriction on the endofunctor T (see Corollary 2.8.3). In this section, we show that final coalgebras exist for so-called accessible functors. The existence proof makes use of yet another categorical concept, limits. We introduce (and explain) both concepts, before diving into the existence proof for final coalgebras.

3.1.1 Accessible Functors

Our proof regarding existence of final coalgebras will be based on the notion of accessibility. Informally, a functor is accessible, if the action of the functor on "large" sets can be determined from the action on small sets. The formal definition is the following (recall the notation $f[\cdot]$ for direct image):

Definition 3.1.1. Suppose κ is a regular cardinal. A functor $T : \mathsf{Set} \to \mathsf{Set}$ is *κ-accessible*, if the following holds:

For all sets X and all $x \in TX$ there exists a subset $Y \subseteq X$ with $\mathsf{card}(Y) < \kappa$ such that $x \in (Ti)[Y]$, where $i : Y \to X$ is the inclusion.

A functor is called *accessible*, if it is κ-accessible for some regular cardinal κ.

Remark 3.1.2. The definition we have given above is not the one which is commonly found in textbooks [11, 37, 36]. The standard definition of accessibility is via preservation of filtered colimits. If you know about filtered colimits, it's an easy exercise to see, that the definition via filtered colimits is (on the category of sets) equivalent to the one we have given here.

The class of accessible functors contains nearly all functors (with the exception of the powerset functor) which are of interest for us.

However, the bounded version of the powerset functor, which we introduce presently, functor is accessible:

Notation 3.1.3. Suppose κ is a cardinal number. The *bounded powerset functor* $\mathcal{P}_\kappa$ is defined by

$$\begin{aligned} \mathcal{P}_\kappa(X) &= \{\mathfrak{x} \subseteq X \mid \mathsf{card}(\mathfrak{x}) < \kappa\} && \text{on sets } X \\ \mathcal{P}_\kappa(f)(\mathfrak{x}) &= f[\mathfrak{x}] && \text{if } f : X \to Y \text{ is a function.} \end{aligned}$$

Using this notation, we can give some closure properties of the class of accessible functors:

Proposition 3.1.4. *The following functors are accessible:*

(i) the identity functor

(ii) all constant functors

(iii) the bounded powerset functor $\mathcal{P}_\kappa$

Furthermore, if T, S *are accessible, then so are* $T \circ S$, $T + S$, $T + S$ *and* T^A, *whenever* A *is a set.*

Proof. See Exercise 3.7.1. □

We leave it as another exercise to show, that the unbounded powerset functor $\mathcal{P}$ is *not* accessible.

3.1.2 Limits along Sequences

The second notion which we discuss before we can embark on the existence proof for final coalgebras is the notion of limits along (ordinal indexed) sequences. As with our discussion of coproducts, coequalisers and pushouts, limits along sequences are an instance of a more general notion, the discussion of which is beyond the scope of these notes. For sequences, however, the situation is a follows:

Definition 3.1.5. Suppose o is an ordinal and $\mathbb{C}$ is a category. A *o-sequence* in $\mathbb{C}$ consists of

- An object $C_\alpha \in \mathbb{C}$ for all $\alpha < o$
- An arrow $f^\alpha_\beta : C_\alpha \to C_\beta$ for all $\beta \leq \alpha < o$

such that the following conditions are met:

- $f^\alpha_\alpha = \mathrm{id}_{C_\alpha}$ for all $\alpha < o$
- $f^\beta_\gamma \circ f^\alpha_\beta = f^\alpha_\gamma$ for all $\gamma \leq \beta \leq \alpha$.

A *cone* over an o-sequence is a pair $(C, (p_\alpha)_{\alpha<o})$ consisting of a set C and a family of arrows $p_\alpha : C \to C_\alpha$ such that $f^\alpha_\beta \circ p_\alpha = p_\beta$ for all $\beta \leq \alpha < o$.

A *morphism of cones* $m : (C, (p_\alpha) \to (D, (q_\alpha))$ is a function $m : C \to D$ such that such that $q_\alpha = p_\alpha \circ f$ for all $\alpha < o$.

Finally, a cone $(C, (p_\alpha))$ is *limiting*, if for any other cone $(D, (q_\alpha)$ there is a unique morphism $u : C \to D$ of cones.

The use of limits is crucial for our construction of the final coalgebra. However, we don't (at least not yet) know whether limits exist. This is the next (easy) step which we take:

Lemma 3.1.6. *Every o-sequence has a limiting cone, which is unique up to morphisms of cones.*

Proof. Take

$$C = \{(c_\alpha)_{\alpha<o} \in \prod_{\alpha<o} C_\alpha \mid \forall\gamma \leq \beta < o. f^\beta_\gamma(c_\beta) = c_\gamma\}$$

and define the projections by $p_\beta((c_\alpha)_{\alpha<o}) = c_\beta$. It is now easy to verify the required properties. □

As with coproducts, pushouts and the like, we just do the underlying construction of limits once explicitely. When we use limits later on, we'll reason with the universal property.

We now dive into the construction of final coalgebras.

3.1.3 The Terminal Sequence

We begin with a discussion of the terminal sequence, which is best thought of as a sequence of approximants to the final coalgebra (that is, the final object in the category $\mathsf{CoAlg}(T)$). A more detailed account of the terminal sequence can be found in [62].

The *terminal sequence* associated with T is an ordinal indexed sequence of sets (Z_α) together with a family (p^α_β) of functions $p^\alpha_\beta : Z_\alpha \to Z_\beta$ for all ordinals $\beta \leq \alpha$ such that

- $Z_{\alpha+1} = TZ_\alpha$ and $p^{\alpha+1}_{\beta+1} = Tp^\alpha_\beta$ for all $\beta \leq \alpha$
- $p^\alpha_\alpha = \mathrm{id}_{Z_\alpha}$ and $p^\alpha_\gamma = p^\beta_\gamma \circ p^\alpha_\beta$ for $\gamma \leq \beta \leq \alpha$
- The cone $(Z_\alpha, (p^\alpha_\beta)_{\beta<\alpha})$ is limiting[1] whenever α is a limit ordinal.

Thinking of Z_α as the α-fold application of T to the limit 1 of the empty diagram, we sometimes write $Z_\alpha = T^\alpha 1$ in the sequel. With this notation, the terminal sequence of T is the continuation of the sequence

$$\begin{tikzcd} 1 & T1 \arrow[l, "!"'] & T^2 1 \arrow[l, "T!"'] & T^3 1 \arrow[l, "T^2!"'] & \cdots \end{tikzcd}$$

through the class of all ordinal numbers, where 0 is considered as a limit ordinal. Intuitively, $T^\alpha 1$ represents behaviours which can be exhibited in α steps. For example, if $TX = D \times X$ and $n \in \mathbb{N}$, then $T^n 1 \cong D^n$ contains all lists of length n.

Note that every coalgebra (C, γ) gives rise to a cone $(C, (\gamma_\alpha : C \to T^\alpha 1))$ over the terminal sequence as follows:

- If $\alpha = \beta + 1$ is a successor ordinal, let $\gamma_\alpha = T\gamma_\beta \circ \gamma : C \to T^\alpha 1$.
- If α is a limit ordinal, γ_α is the unique map for which $\gamma_\beta = p^\alpha_\beta \circ \gamma_\alpha$ for all $\beta < \alpha$.

Our plan is to construct a final coalgebra via iteration along the terminal sequence. The next lemma gives us an upper bound for the termination of this construction. Basically it says that we need not go further than the accessibility degree of the endofunctor under consideration, since further steps only cut down on the constructed object.

Lemma 3.1.7. *Suppose T is κ-accessible. Then $p^{\kappa+1}_\kappa$ is monic.*

[1] See [36] for the definition of limiting cones.

Proof. Suppose $p_\kappa^{\kappa+1}(x) = p_\kappa^{\kappa+1}(y)$ for $x, y \in Z_{\kappa+1}$. Then $p_\alpha^{\kappa+1}(x) = p_\alpha^\kappa \circ p_\kappa^{\kappa+1}(x) = p_\alpha^\kappa \circ p_\kappa^{\kappa+1}(y) = p_\alpha^{\kappa+1}(y)$ for all $\alpha \leq \kappa$. Furthermore, by definition of the terminal sequence, $Tp_\alpha^\kappa(x) = P_{\alpha+1}^{\kappa+1}(x) = p_{\alpha+1}^{\kappa+1}(y) = Tp_\alpha^\kappa(y)$ for all $\alpha < \kappa$. Since T is κ-accessible, there exists a subset $S \subseteq Z_\kappa$ with $\mathsf{card}(S) < \kappa$ and $x', y' \in S$ such that $x = Ti(x')$ and $y = Ti(y')$, where $i : S \to Z_\kappa$ is the inclusion. Hence $T(p_\alpha^\kappa \circ i)(x') = Tp_\alpha^\kappa(x) = Tp_\alpha^\kappa(y) = T(p_\alpha^\kappa \circ i)(y')$ for all $\alpha < \kappa$.

We claim that there exists some ordinal $\sigma < \kappa$ such that $p_\sigma^\kappa \circ i$ is monic. If the claim is established, we conclude $x' = y'$ (since any Set-endofunctor preserves monics) and $x = Ti(x') = Ti(y') = y$.

To prove the claim, consider the set $S' = \{s \subseteq S \mid \mathsf{card}(s) = 2\}$ of distinct elements of S. For all $s = \{s_1, s_2\} \in S'$, there exists an ordinal $\sigma_s < \kappa$ such that $p_{\sigma_s}^\kappa(s_1) \neq p_{\sigma_s}^\kappa(s_2)$ because $s_1 \neq s_2$. Since κ is regular, $\mathsf{card}(S') < \kappa$ and $\sigma = \cup\{\sigma_s \mid s \in S'\} < \kappa$. It is immediate to see that $p_\sigma^\kappa \circ i$ is monic.

□

Since $p_\kappa^{\kappa+1}$ is monic, there exists a left inverse e of $p_\kappa^{\kappa+1}$, that is, $e \circ p_\kappa^{\kappa+1} = \mathrm{id}_{TZ_\kappa}$. Note that e qualifies as coalgebra structure on Z_κ. The coalgebra thus obtained has the property that any morphism of cones $(C, (\gamma_\alpha)_{\alpha<\kappa}) \to (Z_\kappa, (p_\alpha)_{\alpha<\kappa})$ whose domain is induced by an object $(C, \gamma) \in \mathsf{CoAlg}(T)$, is actually a morphism of coalgebras:

Lemma 3.1.8. *Suppose T is κ-accessible and e is a left inverse of $p_\kappa^{\kappa+1}$. Then $\gamma_\kappa : (C, \gamma) \to (Z_\kappa, e)$ is a morphism of coalgebras.*

Proof. Note that $\gamma_\kappa : C \to Z_\kappa$ is the unique morphism for which $\gamma_\alpha = p_\alpha^\kappa \circ \gamma_\kappa$ for all $\alpha < \kappa$. Consider the diagram

$$\begin{tikzcd}
C \arrow[r, "\gamma_\kappa"] \arrow[d, "\gamma"'] & Z_\kappa \arrow[d, "e"] & \\
TC \arrow[r, "T\gamma_\kappa"'] & TZ_\kappa \arrow[r, "p_\kappa^{\kappa+1}"'] & Z_\kappa .
\end{tikzcd}$$

In order to see that $e \circ \gamma_\kappa = T\gamma_\kappa \circ \gamma$, it suffices to show that $p_\kappa^{\kappa+1} \circ e \circ \gamma_\kappa = p_\kappa^{\kappa+1} \circ T\gamma_\kappa \circ \gamma$, since $p_\kappa^{\kappa+1}$ is monic. Since Z_κ is the vertex of a limiting cone, the last equality can be established by showing that

$$p_\alpha^\kappa \circ p_\kappa^{\kappa+1} \circ e \circ \gamma_\kappa = p_\alpha^\kappa \circ p_\kappa^{\kappa+1} \circ T\gamma_\kappa \circ \gamma.$$

If $\alpha = \beta + 1$ is a successor ordinal, we obtain $p_\alpha^\kappa \circ p_\kappa^{\kappa+1} \circ e \circ \gamma_\kappa = p_\alpha^{\kappa+1} \circ e \circ p_\kappa^{\kappa+1} \circ \gamma_{\kappa+1} = p_\alpha^{\kappa+1} \circ \gamma_{\kappa+1} = \gamma_\alpha$ for the left side of the above equation. For the right side, we calculate $p_\alpha^\kappa \circ p_\kappa^{\kappa+1} \circ T\gamma_\kappa \circ \gamma = p_{\beta+1}^{\kappa+1} \circ T\gamma_\kappa \circ \gamma = T(p_\beta^\kappa \circ \gamma_\kappa) \circ \gamma = T\gamma_\beta \circ \gamma = \gamma_\alpha$ by definition of the sequence (γ_α). If α is a limit ordinal, the equation follows at once. □

Since every T-coalgebra (C, γ) induces a map $\gamma_\kappa : C \to Z_\kappa$, we have just shown that (Z_κ, e) is weakly terminal. We note this as:

Corollary 3.1.9. *Suppose T is κ-accessible and e is a left inverse of $p_\kappa^{\kappa+1}$. Then (Z_κ, e) is weakly terminal.*

The last corollary in particular implies the existence of a final coalgebra. We note this as

Theorem 3.1.10. *Suppose T is accessible. Then there exists a final T-coalgebra.*

Proof. In Corollary 3.1.9 we have seen that T admits a weakly final coalgebra (C, γ). Put $Z = Cmbeq_C$ (where $\approx_C = \{(c, c') \mid c \approx c\}$ as usual) and equip Z with the unique transition structure $\zeta : Z \to TZ$, which turns the projections $p : C \to Cmbeq_C = Z$ into a morphism. Clearly (Z, ζ) is simple and extensive, therefore final in $\mathsf{CoAlg}(T)$. $\square$

However, we can extract more information from our proof. We can extract a proof principle for showing that two states are behaviourally equivalent: this is the case if and only if they have the same projections into the terminal sequence.

In the theorem, (Z, ζ) denotes the final T-coalgebra; the unique morphism $(E, \epsilon) \to (Z, \zeta)$ is denoted by $!_\epsilon$.

Theorem 3.1.11 (Induction Along the Terminal Sequence). *Suppose T is κ-accessible and $(C, \gamma), (D, \delta) \in \mathsf{Set}_T$. The following are equivalent for $(c, d) \in C \times D$:*

(i) c and d are behaviourally equivalent

(ii) $!_\gamma(c) = !_\delta(d)$

(iii) For all $\alpha < \kappa$: $\gamma_\alpha(c) = \delta_\alpha(d)$.

Proof. Obviously $(i) \implies (ii) \implies (iii)$. So assume $\gamma_\alpha(c) = \delta_\alpha(d)$ for all $\alpha < \kappa$. Consider the cones $(C, (\gamma_\alpha)_{\alpha<\kappa})$ and $(D, (\delta_\alpha)_{\alpha<\kappa})$. By Lemma 3.1.8, the mediating morphisms $\gamma_\kappa : C \to Z_\kappa$ and $\delta_\kappa : D \to Z_\kappa$ are coalgebra morphisms with codomain (Z_κ, e) where e is a left inverse of $p_\kappa^{\kappa+1}$. Since $\gamma_\alpha(c) = \delta_\alpha(d)$ for all $\alpha < \kappa$, we have $\gamma_\kappa(c) = \gamma_\kappa(d)$. Hence c and d are behaviourally equivalent. $\square$

Intuitively, $\gamma_\alpha(c)$ represents the behaviour of c, which is observable in at most α transition steps. Thus $\gamma_\alpha(c) = \delta_\alpha(d)$ asserts that the α-step behaviour of c and d coincide. The theorem therefore allows us to conclude that c and d

are behaviourally equivalent if their α-step behaviours coincide for all α less than the accessibility degree of T. In particular, it gives us a method for proving equalities between states of a simple (in particular final) system: we can use the theorem to show that two states are behaviourally equivalent, and then conclude that they are equal using simplicity; this is sometimes called a proof by coinduction. We illustrate the theorem by means of some examples.

Example 3.1.12. Suppose L is some finite or infinite set (of labels).

(i) Suppose $TX = L \times X$. Then T is polynomial, hence ω-accessible. Note that the elements $T^n 1 \cong L^n$ of the terminal sequence associated to T are the sequences of labels, which have length n.

Given a T-coalgebra (C, γ), every state $c_0 \in C$ gives rise to an infinite sequence $c_0 \xrightarrow{l_1} c_1 \xrightarrow{l_2} c_2 \ldots$ by putting $c \xrightarrow{l} c'$ iff $\gamma(c) = (l, c')$. In this setup, we have $\gamma_n(c_0) = (l_1, \ldots, l_n)$, that is, the sequence of the first n labels given by c_0. Theorem 3.1.11 states that two states c and d of T-coalgebras are behaviourally equivalent iff they give rise to the same finite sequences of labels.

(ii) Suppose $TX = \mathcal{P}_\omega(L \times X)$. Because $\mathcal{P}_\omega$ is ω-accessible, T is ω-accessible, since ω-accessible functors are closed under composition.

A T-coalgebra (C, γ) is a finitely branching labelled transition system: put $c \xrightarrow{l} c'$ iff $(l, c') \in \gamma(c)$. Given two T-coalgebras (C, γ) and (D, δ), we define a relation $\sim_n$ on $C \times D$ by induction on n as follows: $\sim_0 = C \times D$ and $c \sim_{n+1} d$ iff

- $\forall c'. c \xrightarrow{l} c' \implies \exists d'. d \xrightarrow{l} d'$ and $c' \sim_n d'$;
- $\forall d'. d \xrightarrow{l} d' \implies \exists c'. c \xrightarrow{l} c'$ and $c' \sim_n d'$.

We obtain that $c \sim_n d$ if and only if $\gamma_n(c) = \delta_n(d)$. The relation $\sim_n$ was used to characterise bisimilarity for finitely branching labelled transition systems in [24]. Intuitively, $c \sim_n d$ if c and d are bisimilar for the first n transition steps. In this setting, Theorem 3.1.11 states that c and d are behaviourally equivalent iff $c \sim_n d$ for all $n \in \omega$.

(iii) Suppose κ is a regular cardinal such that $\kappa > \omega$ and consider $TX = \mathcal{P}_\kappa(X)$. Then T is κ-accessible.

Now take $C = \omega + 2$ and $\gamma(c) = \{c' \mid c' \in c\}$. One obtains $\gamma_\alpha(c) = \gamma_\alpha(c')$ iff $c \cap \alpha = c' \cap \alpha$, for $c, c' \in C$ and $\alpha < \kappa$. Hence $\gamma_\alpha(\omega) = \gamma_\alpha(\omega+1)$ for all $\alpha \leq \omega$ but $\gamma_{\omega+1}(\omega) \neq \gamma_{\omega+1}(\omega + 1)$. Hence ω and $\omega + 1$ are not behaviourally equivalent.

Writing $c \to c'$ for $c' \in \gamma(c)$, this can be explained by the fact that $\omega+1$ has a successor (namely ω) which allows for arbitrary long sequences

$\omega \to n_k \to \cdots \to n_0 = 0$, for $n_0 < n_1 < \ldots n_k < \omega$, whereas there is no successor of ω with this property.

Note that this also shows that induction up to the accessibility degree of T is necessary to establish behavioural equivalence of two points.

3.2 On Categorical Duality

In the last section, we have shown, how one can do proofs by coinduction: show that two states of a simple system are behaviourally equivalent and conclude that they are equal because of simplicity. As with induction, there's also a definition principle, which allows us to do proofs by coinduction. In order to get a better understanding of coinductive definitions, we present them in line with their inductive counterparts. The present chapter, on categorical duality, provides us with the means to establish the link between induction and coinduction on a formal level.

We begin with the description of opposite categories:

Definition 3.2.1. Suppose $\mathbb{C}$ is a category. The *opposite category* $\mathbb{C}^{\mathrm{op}}$ is given as follows:

- $\mathsf{obj}\mathbb{C}^{\mathrm{op}} = \mathsf{obj}(\mathbb{C})$
- 1_A (in $\mathbb{C}^{\mathrm{op}}$) $= 1_A$ (in $\mathbb{C}$)
- $\mathbb{C}^{\mathrm{op}}(A, B) = \mathbb{C}(B, A)$
- $g \circ_{\mathbb{C}^{\mathrm{op}}} f = f \circ_{\mathbb{C}} g$

where $\circ_{\mathbb{C}}$ denotes composition of arrows in $\mathbb{C}$ and $\circ_{\mathbb{C}^{\mathrm{op}}}$ is composition in $\mathbb{C}$.

It is easy to see that $\mathbb{C}^{\mathrm{op}}$ is a category, and we leave the proof to the reader. Note that the passage from a category $\mathbb{C}$ to its opposite $\mathbb{C}^{\mathrm{op}}$ is functorial.

In order to avoid the confusion which sometimes arises when working both with $\mathbb{C}$ and $\mathbb{C}^{\mathrm{op}}$ (where am I?), we make the following convention:

Notation 3.2.2. If $A \in \mathbb{C}$ is an object of $\mathbb{C}$, we write A^{op} for the corresponding object in $\mathbb{C}^{\mathrm{op}}$; if it's clear from the context whether we're in $\mathbb{C}$ or in $\mathbb{C}^{\mathrm{op}}$, we drop the superscript. Also, if $f : A \to B$ is an arrow in $\mathbb{C}$, we write f^{op} for the associated arrow in $\mathbb{C}^{\mathrm{op}}$.

Note that $f^{\mathrm{op}} : B^{\mathrm{op}} \to A^{\mathrm{op}}$ for an arrow $f : A \to B$ in $\mathbb{C}$. Any functor between two categories automatically extends to a functor on the associated opposites:

Lemma 3.2.3. *Suppose $F : \mathbb{C} \to \mathbb{D}$ is a functor. Then so is $F^{\mathrm{op}} : \mathbb{C}^{\mathrm{op}} \to \mathbb{D}^{\mathrm{op}}$, which is given by*

$$F^{\mathrm{op}}(C^{\mathrm{op}}) = F(C)^{\mathrm{op}} \quad \textit{and} \quad F^{\mathrm{op}}(f^{\mathrm{op}}) = F(f)^{\mathrm{op}}$$

for objects C^{op} and arrows f^{op} of $\mathbb{C}^{\mathrm{op}}$.

Proof. Follows directly from the definitions. $\square$

We can now introduce the concept of algebras for an endofunctor in a categorical context and use duality to transfer results and intuitions from the world of algebras to coalgebras.

Definition 3.2.4. Let $\mathbb{C}$ be a category. A *T-algebra* is a pair (A, α) where $A \in \mathbb{C}$ is an object and $\alpha : TA \to A$ is an arrow.

If (A, α) and (B, β) are T-algebras, then a morphism $f : A \to B$ is a *morphism* of algebras, if $f \circ \alpha = Tf \circ \beta$.

Diagrammatically speaking, an algebra morphism makes the diagram

$$\begin{tikzcd}
TA \arrow[r, "Tf"] \arrow[d, "\alpha"'] & TB \arrow[d, "\beta"'] \\
A \arrow[r, "f"'] & B
\end{tikzcd}$$

commute. Extending the definition of coalgebras from the category of sets to other categories, we arrive at the following observations:

$$T\text{-algebras } (A, \alpha) \text{ in } \mathbb{C} \iff T^{\mathrm{op}}\text{- coalgebras } (A^{\mathrm{op}}, \alpha^{\mathrm{op}}) \text{ in } \mathbb{C}^{\mathrm{op}}.$$

$$\text{Algebra-morphisms } f : (A, \alpha) \to (B, \beta) \iff \text{coalgebra-morphisms}$$
$$f : (A^{\mathrm{op}}, \alpha^{\mathrm{op}}) \to (B^{\mathrm{op}}, \beta^{\mathrm{op}})$$

What this means is the following: If we prove a statement about algebras (or coalgebras) in an arbitrary category by purely categorical means, then the statement also holds for coalgebras (resp. algebras) in arbitrary categories.

Hence we can use categorical duality as tool to economise on proofs, and we shall do so when treating inductive definitions and their coinductive counterparts simultaneously, and then later transfer the results using categorical duality.

Note that nearly all of our constructions (products, coproducts, pushouts, etc.) were formulated in a purely categorical way, and therefore we have the dual notions readily available.

We have the following line-up of concepts and their duals:

Initial Object An object I is *initial*, if, for all $A \in \mathbb{C}$, there's a unique arrow $u : I \to A$.

Final Object F is a final (or terminal) object of $\mathbb{C}$, if F^{op} is an initial object of $\mathbb{C}^{\mathrm{op}}$.

Product Diagram A *product diagram* over $A, B \in \mathbb{C}$ is a triple $(P, p_1 : P \to A, p_2 : P \to B)$ such that:

For all $X \in \mathbb{C}$ and all $f_1 : X \to A$ and $f_2 : X \to B$ there's a unique $u : X \to P$ such that $u \circ p_i \circ u = q_i$ for $i = 1, 2$.

Coproduct Diagram A *coproduct diagram* (over $A, B \in \mathbb{C}$) is a triple (C, c_1, c_2) such that $(C^{\mathrm{op}}, c_1^{\mathrm{op}}, c_2^{\mathrm{op}})$ is a product diagram in $\mathbb{C}^{\mathrm{op}}$.

Equaliser Diagram An *equaliser diagram* (over a pair $f, g : A \to B$ of parallel arrows) is a tuple $(E, e : E \to E)$ such that:

For all $h : C \to A$ with $f \circ h = g \circ h$ there's a unique $u : C \to E$ with $e \circ u = h$.

Coequaliser Diagram A *coequaliser diagram* (over a pair $f, g : A \to B$ of parallel arrows) is a tuple $(C, c : B \to C)$ such that $(C^{\mathrm{op}}, c^{\mathrm{op}})$ is an equaliser diagram over f^{op} and g^{op} in $\mathbb{C}^{\mathrm{op}}$.

Pullback Diagram A *pullback diagram* over a pair of arrows $f_1 : B \to C$ and $f_2 : C \to D$ with common codomain is a triple $(A, p_1 : A \to B, p_2 : A \to C)$ such that:

For all $X \in \mathbb{C}$ and all $h_1 : X \to B$ and $h_2 : X \to C$ with $f_1 \circ h_1 = f_2 \circ h_2$ there's a unique $u : X \to A$ such that $p_i \circ u = h_i$ for $i = 1, 2$.

Pushout Diagram A *pushout diagram* over a pair of arrows $f_1 : A \to B$ and $f_2 : A \to C$ is a triple (D, p_1, p_2) such that $(D^{\mathrm{op}}, p_1^{\mathrm{op}}, p_2^{\mathrm{op}})$ is a pullback diagram over f_1^{op} and f_2^{op} in $\mathbb{C}^{\mathrm{op}}$.

T-Algebra A T-algebra is a pair (A, α) where $A \in \mathbb{C}$ and $\alpha : A \to TA)$.

T-coalgebra A T-coalgebra is a pair (C, γ) such that $(C^{\mathrm{op}}, \gamma^{\mathrm{op}})$ is an T^{op}-algebra in $\mathbb{C}^{\mathrm{op}}$.

Algebra morphism An *algebra morphism* between T-algebras (A, α) and (B, β) is an arrow $f : A \to B$ such that $f \circ \alpha = \beta \circ Tf$.

Coalgebra Morphism A *coalgebra morphism* between T-coalgebras (C, γ) and (D, δ) is an arrow $f : C \to D$ such that f^{op} is an algebra morphism between the T^{op}-algebras $(C^{\mathrm{op}}, \gamma^{\mathrm{op}})$ and $(D^{\mathrm{op}}, \delta^{\mathrm{op}})$.

3.3 Coinductive Definitions

In the following, we present two definition principles for functions on coalgebras: coiteration and primitive corecursion. Both are obtained by dualising the corresponding principles, as known from the natural numbers (or other algebraic structures).

In the following, we discuss algebras and coalgebras simultaneously. Dually to coalgebras, initial algebras are of special importance in the context of algebras:

Definition 3.3.1. Suppose $T : \mathbb{C} \to \mathbb{C}$ is an endofunctor and (A, α) is a T-algebra. Then (A, α) is *initial*, if, for every other T-algebra (B, β), there exists a unique homomorphism $u : (A, \alpha) \to (B, \beta)$.

Of course, an initial T-algebra is the same as a final T^{op}-coalgebra. In the sequel, will have to consider initial algebras / final coalgebras for different signatures. In order to avoid confusion, we introduce the following:

Notation 3.3.2. The initial T algebra (if it exists) is denoted by $(\mu X.TX, \mathsf{in})$, the final T-coalgebra (again subject to its existence) by $(\nu X.TX, \mathsf{out})$.

We begin with the discussion of the duality between iteration and coiteration.

3.3.1 Iteration and Coiteration

We begin with the more familiar concept of iteration: On the natural numbers, this amounts to saying that a pair of functions $f_0 : 1 \to A$ and $f_s : A \to A$ uniquely determine a function $f : \mathbb{N} \to A$ via $f(0) = f_0(0)$ and $f(n + 1) = f_s(f(n))$. In order to see the duality with coalgebras, we have to lift this principle to arbitrary initial algebras.

Theorem 3.3.3 (Iteration). *For all $f : TA \to A$ there exists a unique $g : \mu X.TX \to A$ such that*

$$\begin{tikzcd} T\mu X.TX \arrow[r, "Tg"] \arrow[d, "\mathsf{in}"'] & TA \arrow[d, "f"] \\ \mu X.TX \arrow[r, "g"'] & A \end{tikzcd}$$

commutes.

Proof. Because $\mu X.TX$ is the initial algebra. □

Example 3.3.4 (Iteration on $\mathbb{N}$). Suppose $TX = 1 + X$. Then $[0, s] : 1 + \mathbb{N} \to \mathbb{N}$ is initial.

Given $a_0 : 1 \to A$ and $a_s : A \to A$, then there is a unique $g : \mathbb{N} \to A$ making

$$\begin{tikzcd} 1 + \mathbb{N} \arrow[r, "1+g"] \arrow[d, "{[0,s]}"'] & A \arrow[d, "{[a_0,a_s]}"] \\ \mathbb{N} \arrow[r, "g"'] & A \end{tikzcd}$$

commute. That is, g is the unique solution to the equations

$$\begin{aligned} g \circ 0 &= a_0 \\ g \circ s &= a_s \circ g. \end{aligned}$$

Dually, we have

Theorem 3.3.5 (Coiteration). *For all $f : C \to TC$ there exists a unique $g : C \to \nu X.TX$ such that*

$$\begin{tikzcd} C \arrow[r, "g"] \arrow[d, "f"'] & \nu X.TX \arrow[d, "\mathsf{out}"] \\ TC \arrow[r, "Tg"'] & T\nu X.TX \end{tikzcd}$$

commutes.

There is no need for a proof: the theorem is just the dual version of Theorem 3.3.3 (although it is of course immediate that the statement just says that final coalgebras are final). Let's give an example of a definition by coiteration:

Example 3.3.6 (Coiteration on Streams). Let $TX = L \times X$ for some fixed set L and denote the set of (infinite) sequences over L by $L^{\mathbb{N}} = \{f \mid f : \mathbb{N} \to L\}$. Then $\langle \mathsf{hd}, \mathsf{tl} \rangle : L^{\mathbb{N}} \to L \times L^{\mathbb{N}}$ is final, where $\mathsf{hd}(f) = f(0)$ and $\mathsf{tl}(f) = \lambda n.f(n+1)$ (see for example [29], Section 6.3).

Given $f_h : C \to L$ and $f_t : C \to C$, there is a unique $g : C \to L^{\mathbb{N}}$ such that

$$\begin{tikzcd} C \arrow[r, "g"] \arrow[d, "\langle f_h{,}f_t \rangle"'] & L^{\mathbb{N}} \arrow[d, "\langle \mathsf{hd}{,}\mathsf{tl} \rangle"] \\ L \times C \arrow[r, "L\times g"'] & L \times L^{\mathbb{N}} \end{tikzcd}$$

commutes. That is, there's a unique solution g to the set of equations

$$\begin{aligned} \mathsf{hd} \circ g &= f_h \\ \mathsf{tl} \circ g &= g \circ f_t. \end{aligned}$$

Let's set up a coiterative arrow for merging of streams:

Example 3.3.7 (Merging of Streams). We're looking for a function $\mathsf{merge} : L^{\mathbb{N}} \times L^{\mathbb{N}} \to L^{\mathbb{N}}$ satisfying the equations

$$\begin{aligned} \mathsf{hd} \circ \mathsf{merge}(s_1, s_2) &= \mathsf{hd}(s_1) \\ \mathsf{tl} \circ \mathsf{merge} &= \mathsf{merge}(s_2, \mathsf{tl}(s_1)) \end{aligned}$$

Define $f_h : L^{\mathbb{N}} \times L^{\mathbb{N}} \to L$ by $f_h(s_1, s_2) = \mathsf{hd} \circ \pi_1$ and $f_t = \langle \pi_2, \mathsf{tl} \circ \pi_1 \rangle$.

By the previous example, there's a unique $\mathsf{merge} : L^{\mathbb{N}} \times L^{\mathbb{N}} \to L^{\mathbb{N}}$ such that

$$\begin{aligned} \mathsf{hd} \circ \mathsf{merge} &= f_h = \mathsf{hd} \circ \pi_1 \\ \mathsf{tl} \circ \mathsf{merge} &= \mathsf{merge} \circ f_t = \mathsf{merge} \circ \langle \pi_2, \mathsf{tl} \circ \pi_1 \rangle. \end{aligned}$$

Throwing in the arguments we arrive at $\mathsf{hd} \circ \mathsf{merge}(s_1, s_2) = \mathsf{hd} \circ \pi_1(s_1, s_2) = \mathsf{hd}(s_1)$ and $\mathsf{tl} \circ \mathsf{merge}(s_1, s_2) = \mathsf{merge} \circ \langle \pi_2, \mathsf{tl} \circ \pi_1 \rangle(s_1, s_2) = \mathsf{merge}(s_2, \mathsf{tl}(s_1))$, as desired.

3.3.2 Primitive (Co)Recursion

Primitive corecursion is for coalgebras what primitive recursion is for algebras. In a nutshell, primitive recursion on algebras works as follows: Given two functions $f : P \times 1 \to A$ and $g : P \times \mathbb{N} \times \mathbb{N} \to A$ there's a unique function $h : P \times \mathbb{N} \to A$ such that

$$\begin{aligned} h(p, 0) &= f(p, 0) \\ h(p, n+1) &= g(p, n, h(p, n)). \end{aligned}$$

The set P is just a set of parameters and by no means special; for simplicity, we just treat the case without parameters. As with coiteration, we first generalise primitive recursion to arbitrary algebras, give a categorical proof and then apply duality.

Theorem 3.3.8 (Primitive Recursion). *For all $f : T((\mu X.TX) \times A) \to A$ there exists a unique $g : \mu X.TX \to A$ such that*

$$\begin{tikzcd} T\mu X.TX \arrow[r, "T\langle \mathrm{id}, g \rangle"] \arrow[d, "\mathsf{in}"'] & T(\mu X.TX \times A) \arrow[d, "f"] \\ \mu X.TX \arrow[r, "g"'] & A \end{tikzcd} \tag{3.1}$$

commutes.

Proof. Consider the diagram

$$\begin{tikzcd} \mu X.TX \arrow[r, "Th"] \arrow[d, "\mathsf{in}"'] & T(\mu X.TX \times A) \arrow[d, "\langle \mathsf{in} \circ T\pi_1, f \rangle"] \\ \mu X.TX \arrow[r, "h"'] & \mu X.TX \times A. \end{tikzcd} \tag{3.2}$$

By the iteration theorem, h is unique wrt. making the above diagram commute.

Claim: $\pi_1 \circ h = \mathrm{id}$.

This follows from commutativity of the composite diagram

$$\begin{tikzcd}
\mu X.TX \arrow[r, "Th"] \arrow[d, "\mathsf{in}"'] & T(\mu X.TX \times A) \arrow[r, "T\pi_1"] \arrow[d, "\langle \mathsf{in} \circ T\pi_1, f\rangle"] & T\mu X.TX \arrow[d, "\mathsf{in}"] \\
\mu X.TX \arrow[r, "h"'] & \mu X.TX \times A \arrow[r, "\pi_1"'] & \mu X.TX,
\end{tikzcd}$$

where the right square commutes by functoriality of T. More precisely, $\pi_1 \circ h$ makes the diagram

$$\begin{tikzcd}
T\mu X.TX \arrow[r, "\pi_1 \circ h"] \arrow[d, "\mathsf{in}"'] & T\mu X.TX \arrow[d, "\mathsf{in}"] \\
\mu X.TX \arrow[r, "T\pi_1 \circ h"'] & \mu X.TX
\end{tikzcd}$$

commute, and so does the identity. By the universal property of initial algebras, we have $\pi_1 \circ h = \mathrm{id}$.

We show that $g = \pi_2 \circ h$ has the required property, that is, it makes (3.1) commute. With an eye to (3.2), we calculate

$$\begin{aligned}
f \circ T\langle \mathrm{id}, g\rangle &= \pi_2 \circ \langle \mathsf{in} \circ T\pi_1, f\rangle \circ Th && \text{since } f = \pi_2 \circ \langle \mathsf{in} \circ T\pi_1, f\rangle \\
&= \pi_2 \circ h \circ \mathsf{in} && \text{(3.2) commutes} \\
&= \pi_2 \circ \langle \mathrm{id}, g\rangle \circ \mathsf{in} \\
&= g \circ \mathsf{in}.
\end{aligned}$$

It remains to show that g is unique wrt. making (3.1) commute. So assume $g : \mu X.TX \to A$ has this property. It suffices to show that $h = \langle \mathrm{id}, g\rangle$ makes (3.2) commute. Let

$$\begin{aligned}
LHS &= \langle \mathsf{in} \circ T\pi_1, f\rangle \circ Th \\
RHS &= h \circ \mathsf{in}
\end{aligned}$$

By the universal property of cartesian products, it suffices to show that

$$\pi_i \circ LHS = \pi_i \circ RHS$$

for $i = 1, 2$.

For $i = 1$, we calculate

$$\begin{aligned}
\pi_1 \circ LHS &= \mathsf{in} \circ T\pi_1 \circ Th \\
&= \mathsf{in} \circ T(\pi_1 \circ \langle \mathrm{id}, g\rangle) && \text{since } h = \langle \mathrm{id}, g\rangle \\
&= \mathsf{in} \\
&= \pi_1 \circ \langle \mathrm{id}, g\rangle \circ \mathsf{in} \\
&= \pi_1 \circ RHS.
\end{aligned}$$

The case $i = 2$ is even easier, and we obtain

$$\begin{aligned}
\pi_2 \circ LHS &= f \circ T\langle \mathrm{id}, g\rangle && \text{since } h = \langle \mathrm{id}, g\rangle \\
&= g \circ \mathsf{in} && \text{(3.1) commutes} \\
&= \pi_2 \circ \langle \mathrm{id}, g\rangle \circ \mathsf{in} \\
&= \pi_2 \circ RHS,
\end{aligned}$$

so we're done. □

The next example shows, that our generalisation was correct with respect to our motivating example.

Example 3.3.9 (Primitive Recursion on $\mathbb{N}$). Suppose $TX = 1+X$ with initial algebra $[0,s] : 1+\mathbb{N} \to \mathbb{N}$. Given $a_0 : 1 \to A$ and $a_s : \mathbb{N} \times A \to A$, there's a unique $g : \mathbb{N} \to A$ such that

$$\begin{tikzcd}
1+\mathbb{N} \arrow[r, "1+\langle \mathrm{id},g\rangle"] \arrow[d, "{[0,s]}"'] & 1+\mathbb{N}\times A \arrow[d, "{[a_0,a_s]}"] \\
\mathbb{N} \arrow[r, "g"'] & A
\end{tikzcd}$$

commutes. In other words, g solves the equations

$$\begin{aligned}
g \circ 0 &= a_0 \\
g \circ s &= a_s \circ \langle \mathrm{id}, g\rangle
\end{aligned}$$

If you prefer explicit arguments, then the second equation reads $g(n+1) = f(n, g(n))$.

Reversing the directions of the arrows, we obtain a definition principle for coalgebras. Note that coproducts (denoted by $+$) are the dual of products (denoted by $\times$).

Theorem 3.3.10 (Primitive Corecursion). *For all $f : C \to T(\nu X.TX + C)$ there exists a unique $g : C \to \nu X.TX$ such that*

$$\begin{tikzcd}
C \arrow[r, "g"] \arrow[d, "f"'] & \nu X.TX \arrow[d, "\mathsf{out}"] \\
T(\nu X.TX + C) \arrow[r, "{T[\mathrm{id},g]}"'] & T(\nu X.TX)
\end{tikzcd}$$

commutes.

Example 3.3.11 (Primitive Corecursion on Streams). Let $TX = L \times X$ as above, and recall finality of $\langle \mathsf{hd}, \mathsf{tl} \rangle : L^{\mathbb{N}} \to L \times L^{\mathbb{N}}$.

Given $f_h : C \to L$ and $f_t : C \to L^{\mathbb{N}} + C$, there's a unique $g : C \to L^{\mathbb{N}}$ such that

$$\begin{tikzcd}
C \arrow[r, "g"] \arrow[d, "\langle f_h, f_t \rangle"'] & L^{\mathbb{N}} \arrow[d, "\langle \mathsf{hd}, \mathsf{tl} \rangle"] \\
L \times (L^{\mathbb{N}} + C) \arrow[r, "L \times [\mathrm{id}, g]"'] & L \times L^{\mathbb{N}}
\end{tikzcd}$$

commutes.

Equationally speaking, g satisfies

$$\begin{aligned}
\mathsf{hd} \circ g &= f_h \\
\mathsf{tl} \circ g &= [\mathrm{id}, g] \circ f_t
\end{aligned}$$

Appealing to the intuition that coproducts are disjoint unions (and leaving the coproduct injections implicit), we can rewrite the second equation to

$$\mathsf{tl} \circ g(s) = \begin{cases} f_t(s) & \text{if } f_t(s) \in L^{\mathbb{N}} \\ g \circ f_t(s) & \text{if } f_t(s) \in C \end{cases} .$$

We define insertion into sorted streams coinductively:

Example 3.3.12 (Sorted Insertion). Let $TX = \mathbb{R} \times X$. We're looking for a function $\mathsf{insert} : \mathbb{R}^{\mathbb{N}} \times \mathbb{R} \to \mathbb{R}^{\mathbb{N}}$ which satisfies

$$\mathsf{hd} \circ \mathsf{insert}(s, r) = \begin{cases} r & \text{if } r < \mathsf{hd}(s) \\ \mathsf{hd}(s) & \text{if } \mathsf{hd}(s) \leq r \end{cases} \tag{3.3}$$

$$\mathsf{tl} \circ \mathsf{insert}(s, r) = \begin{cases} s & \text{if } r < \mathsf{hd}(s) \\ \mathsf{insert}(\mathsf{tl}(s), r) & \text{if } \mathsf{hd}(s) \leq r \end{cases} \tag{3.4}$$

In order to obtain insert, we define f_h as to be expected, that is,

$$f_h(s, r) = \begin{cases} r & \text{if } r < \mathsf{hd}(s) \\ hd(s) & \text{if } \mathsf{hd}(s) \leq r \end{cases}$$

and we let

$$f_t(s, r) = \begin{cases} s \in \mathbb{R}^{\mathbb{N}} & \text{if } r \leq \mathsf{hd}(r) \\ (\mathsf{tl}(s), r) \in \mathbb{R}^{\mathbb{N}} \times \mathbb{R} & \text{if } \mathsf{hd}(s) \leq r \end{cases}$$

By the previous example, there's a unique arrow $\mathsf{insert} : \mathbb{R}^{\mathbb{N}} \times \mathbb{R} \to \mathbb{R}^{\mathbb{N}}$ satisfying

$$\begin{aligned}
\mathsf{hd} \circ g &= f_h \\
\mathsf{tl} \circ g &= [\mathrm{id}, g] \circ f_t .
\end{aligned}$$

So the first equation for insert is clearly satisfied. Regarding the second, we calculate

$$\begin{aligned}
\mathsf{tl} \circ \mathsf{insert}(s, r) &= [\mathrm{id}, \mathsf{insert}] \circ f_t(s, r) \\
&= [\mathrm{id}, \mathsf{insert}](\begin{cases} s & \text{if } r < \mathsf{hd}(s) \\ (\mathsf{tl}(s), r) & \text{if } \mathsf{hd}(s) \leq r \end{cases}) \\
&= \begin{cases} s & \text{if } r < \mathsf{hd}(s) \\ \mathsf{insert}(\mathsf{tl}(s), r) & \text{if } \mathsf{hd}(s) \leq r. \end{cases}
\end{aligned}$$

So merge has the required property.

3.4 Proofs by Terminal Sequence Induction

This section describes one class of applications of induction along the terminal sequence: We show how to use Theorem 3.1.11 to establish properties of co-recursively defined functions.

In the examples, we fix the endofunctor $TX = \mathbb{R} \times X$, where $\mathbb{R}$ denotes the set of reals. We have seen in Section 1.1.5 that every T-coalgebra defines a set of infinite streams of real numbers. In particular, $\mathsf{CoAlg}(T)$ has a final object (Z, ζ), where $Z = \mathbb{R}^\omega$ is the set of all infinite sequences of real numbers and $\zeta((r_n)_{n\in\omega}) = (r_0, (r_{n+1})_{n\in\omega})$ computes both head and remainder of a stream. In the sequel, we write $\zeta = \langle \mathsf{hd}, \mathsf{tl} \rangle$ to denote the first and second component of ζ.

One readily establishes that T is ω-accessible; hence we can prove properties of states of T-coalgebras by induction up to ω: If $z_0, z_1 \in Z$, we have $z_0 = z_1$ iff $\zeta_n(z_0) = \zeta_n(z_1)$ for all $n \in \omega$, where $\zeta_n : Z \to T^n 1$ is defined as in Section 3.1.3. Note that the base case ($n = 0$) is always trivial.

The first example, taken from Bartels [9], defines componentwise addition $\oplus$ of streams of reals using coiteration. We then use induction along the terminal sequence to show that $\oplus$ is a commutative operator.

Example 3.4.1. There is a unique function $\oplus : Z \times Z \to Z$ such that

$$\begin{aligned}
\mathsf{hd}(x \oplus y) &= \mathsf{hd}(x) + \mathsf{hd}(y) \\
\mathsf{tl}(x \oplus y) &= \mathsf{tl}(x) \oplus \mathsf{tl}(y)
\end{aligned}$$

for all $x, y \in Z$.

It is intuitively obvious, and can be – using an explicit representation of hd, tl and $\oplus$ – easily established that $\oplus$ is commutative, i.e. $x \oplus y = y \oplus x$ for all $x, y \in Z$. In order to demonstrate the power of Theorem 3.1.11, we give an alternative proof, which, as a by-product, omits the need for explicit representations.

Example 3.4.2. $x \oplus y = y \oplus x$ for all $x, y \in Z$.

Proof. Define $f : Z \times Z \to \mathbb{R} \times Z \times Z$ by $f(x, y) = (\mathsf{hd}(x) + \mathsf{hd}(y), \mathsf{tl}(x), \mathsf{tl}(y))$. Then $\oplus : Z \times Z \to Z$ is the unique function satisfying $\zeta \circ \oplus = f \circ T\oplus$. In order to show the commutativity of $\oplus$, it suffices to establish that $\zeta_n(x \oplus y) = \zeta_n(y \oplus x)$ for all $x, y \in Z$. If $n = 0$, there is nothing to show. If $n = m + 1$ this is just a matter of calculation:

$$\begin{aligned}
\zeta_n(x \oplus y) &= T\zeta_m \circ \zeta(x \oplus y) \\
&= T\zeta_m \circ T \oplus \circ f(x, y) \\
&= (\mathsf{hd}(x) + \mathsf{hd}(y), \zeta_m(x \oplus y)) \\
&= (\mathsf{hd}(y) + \mathsf{hd}(x), \zeta_m(y \oplus x)) \qquad \text{(by induction hypothesis)} \\
&= \zeta_n(y \oplus x)
\end{aligned}$$

which finishes the proof. □

We now give a second example for a proof by coinduction: picking up Example 3.3.12 again, we show that, for sorted insertion, the order in which we insert two elements into a stream is irrelevant.

Example 3.4.3. $\mathsf{insert}(x, \mathsf{insert}(y, s)) = \mathsf{insert}(y, \mathsf{insert}(x, s))$ for all $x, y \in \mathbb{R}$ and all $s \in Z$.

Proof. We use Theorem 3.1.11 and show that

$$\zeta_n(\mathsf{insert}(x, \mathsf{insert}(y, s))) = \zeta_n(\mathsf{insert}(y, \mathsf{insert}(x, s))) \tag{3.5}$$

for all $n < \omega$, $x, y \in \mathbb{R}$ and $s \in Z$. Again, there is nothing to show for $n = 0$. If $n = m + 1$, this is equivalent to

$$T\zeta_m \circ T[\mathsf{insert}, \mathrm{id}] \circ f(x, \mathsf{insert}(y, s)) = T\zeta_m \circ T[\mathsf{insert}, \mathrm{id}] \circ f(y, \mathsf{insert}(x, s)) \tag{3.6}$$

which in turn holds if the validity of both equations

$$f_1(x, \mathsf{insert}(y, s)) = f_1(y, \mathsf{insert}(x, s)) \tag{3.7}$$

and

$$\zeta_m \circ [\mathsf{insert}, \mathrm{id}] \circ f_2(x, \mathsf{insert}(y, s) = \zeta_m \circ [\mathsf{insert}, \mathrm{id}] \circ f_2(y, \mathsf{insert}(x, s) \tag{3.8}$$

can be established.

It is immediate that (3.7) holds, since Equation (3.3) implies that both sides evaluate to $\min(x, y, \mathsf{hd}(s))$. Abbreviating the left and right hand side of (3.8) by LHS and RHS, respectively, we obtain

$$\text{LHS} = \zeta_m \circ \begin{cases} \mathsf{insert}(x, \mathsf{tl} \circ \mathsf{insert}(y, s)) & \text{if } x > \mathsf{hd} \circ \mathsf{insert}(y, s) \\ \mathsf{insert}(y, s) & \text{if } x \leq \mathsf{hd} \circ \mathsf{insert}(y, s) \end{cases}$$

and

$$\text{RHS} = \zeta_m \circ \begin{cases} \mathsf{insert}(y, \mathsf{tl} \circ \mathsf{insert}(x, s)) & \text{if } y > \mathsf{hd} \circ \mathsf{insert}(x, s) \\ \mathsf{insert}(x, s) & \text{if } y \leq \mathsf{hd} \circ \mathsf{insert}(x, s) \end{cases}$$

We distinguish the different cases accordingly.

Case 1: $x > \mathsf{hd} \circ \mathsf{insert}(y, s)$ and $y > \mathsf{hd} \circ \mathsf{insert}(x, s)$. Then LHS $= \zeta_m \circ \mathsf{insert}(x, \mathsf{tl} \circ \mathsf{insert}(y, s)) = \zeta_m \circ \mathsf{insert}(x, \mathsf{insert}(y, \mathsf{tl}(s)) = \zeta_m \circ \mathsf{insert}(y, \mathsf{insert}(x, \mathsf{tl}(s)) = \zeta_m \circ \mathsf{insert}(y, \mathsf{tl} \circ \mathsf{insert}(x, s)) =$ RHS using the induction hypothesis (for $\mathsf{tl}(s)$) and Equation (3.3).

Case 2: $x > \mathsf{hd} \circ \mathsf{insert}(y, s)$ and $y \leq \mathsf{hd} \circ \mathsf{insert}(x, s)$. Then LHS $= \zeta_m \circ \mathsf{insert}(x, \mathsf{tl} \circ \mathsf{insert}(y, s)) = \zeta_m \circ \mathsf{insert}(x, s)$, again using (3.3).

Case 3: $x \leq \mathsf{hd} \circ \mathsf{insert}(y, s)$ and $y > \mathsf{hd} \circ \mathsf{insert}(x, s)$. The claim follows as with case 2.

Case 4: $x \leq \mathsf{hd} \circ \mathsf{insert}(y, s)$ and $y \leq \mathsf{hd} \circ \mathsf{insert}(x, s)$. Since $\mathsf{hd} \circ \mathsf{insert}(x, s) = \min(x, \mathsf{hd}(s))$, we conclude $x \leq \min(y, \mathsf{hd}(s)) \leq y \leq \min(x, \mathsf{hd}(s)) \leq x$, hence $x = y$ and the claim is obvious. □

3.5 Bisimulation Proofs

Another alternative for establishing that a pair of states is behaviourally equivalent is via bisimulations. Sloppily speaking, a bisimulation between two coalgebras is a relation, which is transition closed. The formal definition is as follows:

Definition 3.5.1. Suppose $(C, \gamma), (D, \delta) \in \mathsf{CoAlg}(T)$. A relation $B \subseteq C \times D$ is a *bisimulation*, if there exists $\beta : B \to TB$ such that $\pi_1 : (B, \beta) \to (C, \gamma)$ and $\pi_2 : (B, \beta) \to (D, \delta)$ are morphisms of coalgebras.

We call $(c, d) \in C \times D$ *bisimilar*, if there is a bisimulation $B \subseteq C \times D$ with $(c, d) \in B$.

The latter requirement can be expressed diagrammatically by saying that

$$\begin{tikzcd}
C \arrow[d, "\gamma"'] & B \arrow[l, "\pi_1"'] \arrow[r, "\pi_2"] \arrow[d, "\beta"'] & D \arrow[d, "\delta"'] \\
TC & TD \arrow[l, "T\pi_1"] \arrow[r, "T\pi_2"'] & TD
\end{tikzcd}$$

commutes. Clearly, bisimilarity implies behavioural equivalence:

Proposition 3.5.2. *Suppose* $(C, \gamma), (D, \delta) \in \mathsf{CoAlg}(T)$ *and* $(c, d) \in C \times D$ *are bisimilar. Then* c *and* d *are behaviourally equivalent.*

Proof. Follows at once, since behavioural equivalence is an equivalence relation. □

For the functor $TX = \mathbb{R} \times X$, we have the following characterisation of bisimulations:

Example 3.5.3. Suppose $(C, \langle \mathsf{hd}_C, \mathsf{tl}_C \rangle)$ and $(D, \langle \mathsf{hd}_D, \mathsf{tl}_D \rangle)$ are T-coalgebras for $TX = \mathbb{R} \times X$. Then $B \subseteq C \times D$ is a bisimulation if and only if

$$\mathsf{hd}_C(c) = \mathsf{hd}_D(d) \text{ and } (\mathsf{tl}_C(c), \mathsf{tl}_D(d)) \in B$$

for all $(c, d) \in B$.

We can use bisimulations to show that componentwise addition of streams is commutative as follows:

Example 3.5.4. We reconsider pointwise addition, already discussed in Example 3.4.1 and show that $x \oplus y = y \oplus x$ for all $x, y \in Z$. If $(x, y) \in Z \times Z$ are arbitrary, we have to exhibit a bisimulation B with $(x, y) \in B$. We put

$$B = \{(\mathsf{tl}^n(x) \oplus \mathsf{tl}^n(y), \mathsf{tl}^n(y) \oplus \mathsf{tl}^n(x)) \mid n \geq 0\}$$

It's obvious how to define the transition relation, which witnesses that B is a bisimulation:

$$\beta(\mathsf{tl}^n(x) \oplus \mathsf{tl}^n(y), \mathsf{tl}^n(y) \oplus \mathsf{tl}^n(x)) = (\mathsf{hd} \circ \mathsf{tl}^n(x) + \mathsf{hd} \circ \mathsf{tl}^n(y), (\mathsf{tl}^{n+1}(x) \oplus \mathsf{tl}^{n+1}(y), \mathsf{tl}^{n+1}(y) \oplus \mathsf{tl}^{n+1}(x)))$$

An easy calculation shows, that β satisfies the requirements of Definition 3.5.1.

Note that, in contrast to the bisimulation proof principle presented in [9], induction along the terminal sequence does not require the definition of a bisimulation, which – at least in some cases – requires some ingenuity in itself.

3.6 Notes

A more detailed account of the terminal sequence can be found in

Lemma 3.1.7 appears in [62] for the case of accessible functors on locally presentable categories, as well as in [3]; our proof is more elementary and tailored towards the set-theoretic case. [62].

The idea behind the categorical formulation of primitive (Co)recursion is (to my knowledge) due to [18] and has been treated in [59, 58] and, in a much more abstract setting, in [9].

One can show, that bisimilarity implies behavioural equivalence if the signature functor preserves weak pullbacks (the standing assumption in Rutten [53]. An example, where behavioural equivalence is the more appropriate notion of equivalence can be found in [32].

3.7 Exercise

Exercise 3.7.1. Prove Proposition 3.1.4.

Exercise 3.7.2. Prove Proposition 3.2.3.

Exercise 3.7.3. Give a concrete description of pullbacks in the category of sets.

Exercise 3.7.4. Give a precise formulation and a proof of the following statement: Pullbacks over a terminal object are products.

Exercise 3.7.5. Show that $(\mathbb{N}, [0, s])$ where $0 : 1 \to \mathbb{N}$ maps $0 \in 1$ to $0 \in \mathbb{N}$ and $s : \mathbb{N} \to \mathbb{N}$ is the successor function, is an initial algebra for $TX = 1 + X$ in the category of sets.

Exercise 3.7.6. Let $TX = 1 + A \times X$. Show that the set A^*, which contains all finite sequences of elements of A, has the structure of an initial T-algebra. (In functional programming, A^* is the set of lists over the type A.)

Exercise 3.7.7. Use iteration to define the length function $A^* \to A$ by iteration using that A^* is an initial algebra (see Exercise 3.7.6).

Exercise 3.7.8. Use primitive recursion to define the predecessor function $p : \mathbb{N} \to \mathbb{N}$, which is defined by $p(n) = \begin{cases} 0 & n = 0 \\ n - 1 & \text{o/w} \end{cases}$.

Exercise 3.7.9. Use primitive corecursion to define prefixing $p : A^\omega \times A \to A$, for infinite streams, where $p((a_0, a_1, a_2, \dots), a) = (a, a_0, a_1, a_2, \dots)$.

Chapter 4

Modal Logics for Coalgebras

4.1 Coalgebraic Logic

This section presents a very general logic, which can be used to specify properties of a large class of coalgebras. The basic idea behind the logic is, that one uses functor application to speak about successor states.

Since coalgebraic logic replaces modalities with functor application, we have to provide a suitable semantical handle to interpret formulas, which arise in this way. This is accomplished by extending the underlying endofunctor to relations in the following way:

Notation 4.1.1. If $T : \mathsf{Set} \to \mathsf{Set}$ is an endofunctor and $R \subseteq A \times B$ is a relation, let $\hat{T}(R) = \{(T\pi_1(r), T\pi_2(r)) \mid r \in R\}$. For sets A, we let $\hat{T}A = TA$.

It has first been noted by Carboni, Kelly and Wood [12] that this defines an endofunctor $\hat{T}$ on the category Rel of sets and relations (i.e. a relator) iff T preserves *weak pullbacks*, that is, ordinary pullbacks, but without the uniqueness requirement. In the sequel, we overload the symbol $\circ$ to denote relational as well as functional composition: If $R \subseteq A \times B$ and $S \subseteq B \times C$, we put $S \circ R = \{(a, c) \mid \exists b \in B.(a, b) \in R, (b, c) \in S\}$. Note that relational composition is written in the same order as function composition.

Proposition 4.1.2. *The functor $\hat{T}$ preserves weak pullbacks if and only if $\hat{T}(S \circ R) = \hat{T}(S) \circ \hat{T}(R)$ for all composable relations R, S.*

The (easy) proof is left as an exercise.

The extension of endofunctors to relations is an integral part of Moss' definition of the semantics of coalgebraic logic, which is given in terms of initial algebras. The following theorem, the proof of which can be found in [2], establishes the existence of initial algebras:

Theorem 4.1.3. *Suppose T is κ-accessible. Then there exists an initial T-algebra.*

Definition 4.1.4. Suppose T is κ-accessible and put $L = \mathcal{P}_\kappa + T$. The *language* $\mathcal{L}(T)$ of coalgebraic logic associated to T is the carrier of the initial L-algebra $(\mathcal{L}, \iota)$.

If $(C, \gamma) \in \mathsf{CoAlg}(T)$, put $d : \mathcal{P}_\kappa\mathcal{P}(C) \to \mathcal{P}(C), d(\mathfrak{x}) = \cap\mathfrak{x}$ and $e : T\mathcal{P}(C) \to \mathcal{P}(C)$, $e(x) = \{c \in C \mid (\gamma(c), t) \in \hat{T}(\epsilon_C)\}$, where $\epsilon_C \subseteq C \times \mathcal{P}(C)$ is the membership relation.

The *semantics* $[\![\cdot]\!]_{(C,\gamma)} : \mathcal{L}(T) \to \mathcal{P}(C)$ of $\mathcal{L}$ with respect to (C, γ) is the unique function with $[d, e] \circ [\![\cdot]\!]_{(C,\gamma)} = T[\![\cdot]\!]_{(C,\gamma)} \circ \iota$. If $c \in [\![\phi]\!]_{(C,\gamma)}$, we also write $c \models_{(C,\gamma)} \phi$; we drop the subscript (C, γ) whenever there is no danger of confusion.

In the above definition, the auxiliary function d is used to interpret conjunctions, and e takes care of the modalities. Note that the initial L-algebra $(\mathcal{L}, \iota)$ always exists since $\mathcal{L}$ is κ-accessible, see Proposition 3.1.4. If $\mathrm{in}_1 : \mathcal{P}_\kappa\mathcal{L} \to \mathcal{P}_\kappa + T\mathcal{L}$ and $\mathrm{in}_2 : T\mathcal{L} \to \mathcal{P}_\kappa\mathcal{L} + T\mathcal{L}$ denote the coproduct injections, we write $\bigwedge = \iota \circ \mathrm{in}_1$ and $\nabla = \iota \circ \mathrm{in}_2$. The language of coalgebraic logic can thus be described as the least set such that

$$\begin{aligned} \Phi \subseteq \mathcal{L}(T), \mathsf{card}(\Phi) < \kappa &\implies \bigwedge \Phi \in \mathcal{L}(T) \\ \phi \in T\mathcal{L}(T) &\implies \nabla\phi \in \mathcal{L}(T). \end{aligned}$$

If $(C, \gamma) \in \mathsf{CoAlg}(T)$, we then obtain

$$\begin{gathered} c \models \bigwedge \Phi \text{ iff } c \models \phi \text{ for all } \phi \in \Phi \\ c \models \nabla\phi \text{ iff } (\gamma(c), \phi) \in \hat{T}(\models) \end{gathered}$$

for subsets $\Phi \subseteq \mathcal{L}$ of cardinality less than κ and $\phi \in T\mathcal{L}$.

Note $\mathcal{L}$ contains $\mathtt{tt} = \bigwedge \emptyset$ and is closed under conjunctions of size $< \kappa$. We just give a brief example of the nature of coalgebraic logic:

Example 4.1.5. Let $TX = L \times X$, where L is a set of labels. As already mentioned, $\mathtt{tt} \in \mathcal{L}$ and obviously $[\![\mathtt{tt}]\!] = C$ for all $(C, \gamma) \in \mathsf{CoAlg}(T)$. If $l \in \mathcal{L}$, we have $(l, \mathtt{tt}) \in T\mathcal{L}$, hence $\nabla(l, \mathtt{tt}) \in \mathcal{L}$. Unravelling the definitions, one obtains $c \models \nabla(l, \mathtt{tt})$ if $\pi_1 \circ \gamma(c) = l$. In the same manner, one has $\nabla(m, \nabla(l, \mathtt{tt})) \in \mathcal{L}$ for $m \in L$ with $c \models \nabla(m, \nabla(l, \mathtt{tt}))$ iff the stream associated to c (cf. Section 1.1.2) begins with (m, l).

The previous example suggests, that we can use induction along the terminal sequence in order to establish an expressiveness theorem for coalgebraic logic: We use functor application to construct formulas for successor ordinals and conjunctions for limit ordinals. Before doing so, we establish a technical lemma and show that $\mathcal{L}(T)$ is *adequate*, i.e. does not distinguish behaviourally equivalent points. In the following, $\mathrm{G}(f)$ denotes the graph of a function.

Lemma 4.1.6. *$G(Tf) = \hat{T}(\mathrm{G}f)$, whenever f is a function.*

The proof is an easy calculation, and therefore omitted. This puts us into the position to tackle adequacy.

Theorem 4.1.7. *Suppose T is κ-accessible and preserves weak pullbacks. Then $[\![\phi]\!]_{(C,\gamma)} = f^{-1}([\![\phi]\!]_{(D,\delta)})$ for all $f : (C,\gamma) \to (D,\delta) \in \mathsf{CoAlg}(T)$ and all $\phi \in \mathcal{L}(T)$.*

Proof. Suppose $[d,e] : \mathcal{P}_\kappa\mathcal{P}C + T\mathcal{P}C \to \mathcal{P}(C)$ and $[d',e'] : \mathcal{P}_\kappa\mathcal{P}D + T\mathcal{P}C \to \mathcal{P}(D)$ are given as in Definition 4.1.4.

We prove that $f^{-1} \circ e' = e \circ T(f^{-1})$. Since $f^{-1} \circ d' = d \circ L(f^{-1})$, this suffices to show that

$$\begin{tikzcd}
\mathcal{P}_\kappa\mathcal{L} + T\mathcal{L} \arrow[r, "L[\![\cdot]\!]_{(D,\delta)}"] \arrow[d, "\iota"] & \mathcal{P}_\kappa\mathcal{P}D + T\mathcal{P}D \arrow[r, "L(f^{-1})"] \arrow[d, "{[d',e']}"] & \mathcal{P}_\kappa\mathcal{P}(C) + T\mathcal{P}C \arrow[d, "{[d,e]}"] \\
\mathcal{L} \arrow[r, "{[\![\cdot]\!]_{(D,\delta)}}"'] & \mathcal{P}(D) \arrow[r, "f^{-1}"'] & \mathcal{P}(C)
\end{tikzcd}$$

commutes, which establishes the claim.

In order to see that $f^{-1} \circ e' = e \circ T(f^{-1})$ suppose $t \in T\mathcal{P}D$ and $c \in C$. Then

$$\begin{aligned}
c \in f^{-1} \circ e'(t) &\text{ iff } (\delta \circ f(c), t) \in \hat{T}(\epsilon_D) \\
&\text{ iff } (Tf \circ \gamma(c), t) \in \hat{T}(\epsilon_D) && \text{(since } f \in \mathsf{CoAlg}(T)\text{)} \\
&\text{ iff } (\gamma(c), t) \in \hat{T}(\epsilon_D) \circ \hat{T}(\mathrm{G}f) && \text{(by Lemma 4.1.6)} \\
&\text{ iff } (\gamma(c), t) \in \hat{T}(\mathrm{G}f^{-1}) \circ \hat{T}(\epsilon_C) && \text{(by Proposition 4.1.2)} \\
&\text{ iff } (\gamma(c), T(f^{-1})(t)) \in \hat{T}(\epsilon_C) && \text{(by Lemma 4.1.6)} \\
&\text{ iff } c \in e \circ T(f^{-1})(t),
\end{aligned}$$

where $\epsilon_C \subseteq C \times \mathcal{P}(C)$ and $\epsilon_D \subseteq D \times \mathcal{P}(D)$ denote the respective membership relations. □

We obtain adequacy as a corollary:

Corollary 4.1.8. *Let $(C,\gamma), (D,\delta) \in \mathsf{CoAlg}(T)$. Then*

$$c \models_{(C,\gamma)} \phi \quad \textit{iff} \quad d \models_{(D,\delta)} \phi$$

whenever $(c,d) \in C \times D$ are behaviourally equivalent.

Proof. Suppose $f : (C,\gamma) \to (E,\epsilon)$ and $g : (D,\delta) \to (E,\epsilon) \in \mathsf{CoAlg}(T)$ with $f(c) = g(d)$. Then $c \models_{(C,\gamma)} \phi$ iff $f(c) \models_{(E,\epsilon)} \phi$ iff $g(d) \models_{(E,\epsilon)} \phi$ iff $d \models_{(D,\delta)} \phi$. □

We now come to the main theorem in the present section: The proof of expressiveness of coalgebraic modal logic using induction along the terminal sequence.

Theorem 4.1.9. *Suppose T is κ-accessible and preserves weak pullbacks, let $(C,\gamma),(D,\delta) \in \mathsf{CoAlg}(T)$. Then*

$$c \models_{(C,\gamma)} \phi \textit{ iff } d \models_{(D,\delta)} \phi$$

implies that $(c,d) \in C \times D$ are behaviourally equivalent.

Proof. Since T is accessible, there is a final object $(Z,\zeta) \in \mathsf{CoAlg}(T)$. We show that there exists a formula $\phi_z^\alpha \in \mathcal{L}$ with $[\![\phi_z^\alpha]\!]_{(Z,\zeta)} = \zeta_\alpha^{-1}(\{z\})$ for all $\alpha < \kappa$ and all $z \in T^\alpha 1$. The result then follows from Theorem 3.1.11.

If α is a limit ordinal, let $\phi_z^\alpha = \bigwedge\{\phi^\beta_{p^\alpha_\beta(z)} \mid \beta < \alpha\}$, where $p^\alpha_\beta : T^\alpha 1 \to T^\beta 1$ are the connecting morphisms of the terminal sequence as defined in Section 3.1.3.

Now suppose $\alpha = \beta + 1$ is a successor ordinal. By induction hypothesis, there is a map $f : T^\beta 1 \to \mathcal{L}$ with the property that $[\![f(z)]\!]_{(Z,\zeta)} = \zeta_\beta^{-1}(\{z\})$ for all $z \in T^\beta 1$. Given an element $z \in T^\alpha 1 = TT^\beta 1$, let $\phi_z^\alpha = \nabla(Tf(z))$. An easy calculation shows, that ϕ_z^α has the required property. □

4.2 Coalgebraic Modal Logic

This section introduces the framework of coalgebraic modal logic, which is an extension of multimodal logic, interpreted over coalgebras. Compared with Moss' approach (discussed in the previous section), coalgebraic modal logic can still be used for a large class of endofunctors, but has the advantage of a standard (multimodal) language. This is achieved through expressing the passage from a set X to TX by means of *predicate liftings*, instead of functor application. The present section introduces the framework of coalgebraic modal logic and shows its adequacy; the following section establishes an expressiveness result similar to Theorem 4.1.9 of the previous section.

We begin by introducing predicate liftings. This concept is at the heart of coalgebraic modal logic in that predicate liftings provide the semantical structure needed to interpret modal operators on arbitrary coalgebras.

Definition 4.2.1. A *predicate lifting* λ for T is a collection of order preserving maps $\lambda(X) : \mathcal{P}(X) \to \mathcal{P}(TX)$, where X ranges over the class of sets, such that

$$\lambda(X) \circ f^{-1} = (Tf)^{-1} \circ \lambda(Y)$$

for all functions $f : X \to Y$.

That is, predicate liftings are order preserving natural transformations, (see [36]) which lift predicates (subsets) $\mathfrak{x} \subseteq X$ on a set X to predicates $\lambda(X)(\mathfrak{x}) \subseteq TX$.

We illustrate the concept of predicate liftings by showing that they generalise the interpretation of the $\Box$-operator from Kripke models (see eg. [10, 19]) to coalgebras of arbitrary signature functors.

Example 4.2.2. Suppose $TX = \mathcal{P}(X) \times \mathcal{P}(A)$ as in Section 1.1.5. Consider the operation $\lambda(C) : \mathcal{P}(C) \to \mathcal{P}(TC)$ defined by

$$\lambda(C)(\mathfrak{c}) = \{(\mathfrak{a}, \mathfrak{c}') \in TC \mid \mathfrak{c}' \subseteq \mathfrak{c}\}.$$

An easy calculation shows, that this defines a predicate lifting λ. Now consider a T-coalgebra (C, γ) and a subset $\mathfrak{c} \subseteq C$, which we think of as the denotation of a modal formula ϕ. Then

$$\gamma^{-1} \circ \lambda(C)(\mathfrak{c}) = \{c \in C \mid \pi_1 \circ \gamma(c) \in \mathfrak{c}\}$$

(where $\pi_1 : \mathcal{P}(C) \times \mathcal{P}(A) \to \mathcal{P}(C)$ denotes first projection) corresponds to the interpretation of the modal formula $\Box\phi$ under the correspondence outlined in Section 1.1.5.

The definition of $\lambda(C)$ given in the last example can be rewritten (using the first projection $\pi_1 : TC \to \mathcal{P}(C)$) as $\lambda(C)(\mathfrak{c}) = \{t \in TC \mid \pi_1(t) \subseteq \mathfrak{c}\}$, and the naturality of λ follows immediately from the naturality of π_1. Replacing π_1 by an arbitrary natural transformation, we obtain a construction principle for predicate liftings:

Proposition 4.2.3. *Suppose $\mu : T \to \mathcal{P}$ is a natural transformation. Then the operation $\lambda(C) : \mathcal{P}(C) \to \mathcal{P}(TC)$, given by*

$$\lambda(C)(\mathfrak{c}) = \{c \in TC \mid \mu(C)(c) \subseteq \mathfrak{c}\}$$

defines a predicate lifting λ for T.

Proof. Let $f : C \to D$. We have to show that $\lambda(C) \circ f^{-1} = (Tf)^{-1} \circ \lambda(D)$, given that μ is natural, ie. $\mathcal{P}(f) \circ \mu(C) = \mu(D) \circ Tf$. For subsets $\mathfrak{d} \subseteq D$, this follows from calculating

$$\begin{aligned}\lambda(C) \circ f^{-1}(\mathfrak{d}) &= \{c \in TC \mid \mu(C)(c) \subseteq f^{-1}(\mathfrak{d})\} \\ &= \{c \in TC \mid \mathcal{P}(f) \circ \mu(C)(c) \subseteq \mathfrak{d}\} \\ &= \{c \in TC \mid \mu(D) \circ Tf(c) \subseteq \mathfrak{d}\} \\ &= (Tf)^{-1} \circ \lambda(D)(\mathfrak{d})\end{aligned}$$

which establishes the claim. □

Continuing our running example, we now show that predicate liftings can also be used to interpret atomic propositions of Kripke models.

Example 4.2.4. Again, let $TX = \mathcal{P}(X) \times \mathcal{P}(A)$. For some fixed $a \in A$, consider the (constant) operation $\lambda_a(C) : \mathcal{P}(C) \to \mathcal{P}(TC)$, given by

$$\lambda_a(C)(\mathfrak{c}) = \{(\mathfrak{c}', \mathfrak{a}) \in TC \mid a \in \mathfrak{a}\}.$$

Given an arbitrary subset $\mathfrak{c} \subseteq C$, we obtain

$$\gamma^{-1} \circ \lambda_a(C)(\mathfrak{c}) = \{c \in C \mid a \in \pi_2 \circ \gamma(c)\},$$

that is, the set of worlds satisfying proposition a under the correspondence outlined in Section 1.1.5.

Again, there is a more general principle underlying the construction of the (constant) lifting of the last example. In the following, we write $1 = \{0\}$ and, if X is a set, $!_X : X \to 1$ for the uniquely defined surjection.

Proposition 4.2.5. *Suppose $\mathfrak{a} \subseteq T1$. Then the operation $\lambda(C) : \mathcal{P}(C) \to \mathcal{P}(TC)$, given by*

$$\lambda(C)(\mathfrak{c}) = \{c \in TC \mid (T!_C)(c) \in \mathfrak{a}\}$$

defines a predicate lifting λ for T.

Proof. Note that $\lambda(C)(\mathfrak{c}) = (T!_C)^{-1}(\mathfrak{a})$, where $!_C : C \to 1$ is the unique morphism. Given $f : C \to D$, we have to show that $\lambda(C) \circ f^{-1} = (Tf)^{-1} \circ \lambda(D)$. But this is immediate, since

$$(Tf)^{-1} \circ \lambda(D)(\mathfrak{d}) = (Tf)^{-1} \circ (T!_D)^{-1}(\mathfrak{a}) = (T!_C)^{-1}(\mathfrak{a}) = \lambda(C)(f^{-1}(\mathfrak{d}))$$

for all subsets $\mathfrak{d} \subseteq D$, since the value of $\lambda(C)(\cdot)$ is independent of its argument. □

The following example shows, how Proposition 4.2.3 and Proposition 4.2.5 can be used to construct predicate liftings, which make assertions about deterministic automata.

Example 4.2.6. Suppose $TX = (X \times O)^I$, where I and O are sets. We have demonstrated in Section 1.1.3 that T-coalgebras are deterministic Mealy automata with input set I, producing elements of O as outputs.

Given an input $i \in I$, the natural transformation $\rho : T \to \mathcal{P}$, defined by $\rho(C)(f) = \{\pi_1(f(i))\}$ for $f \in (C \times O)^I = TC$, gives rise to a predicate lifting λ_i by Proposition 4.2.3. Intuitively, λ_i allows us to formulate properties about the successor state after consuming input $i \in I$.

If $(i, o) \in I \times O$, then the subset $\{f \in (1 \times O)^I \mid \pi_2(f(i)) = o\}$ gives rise to a lifting $\mu_{(i,o)}$ by Proposition 4.2.5. The lifting $\mu_{(i,o)}$ can be used to assert that the current state is such that processing of input i yields output o.

In classical modal logic, one often defines the operator $\Diamond$ by putting $\Diamond\phi = \neg\Box\neg\phi$. We conclude this section by showing that this can already be accomplished on the level of predicate liftings.

Proposition 4.2.7. *Suppose λ is a predicate lifting for T. Then the operation $\neg\lambda\neg(C) : \mathcal{P}(C) \to \mathcal{P}(TC)$, defined by*

$$\neg\lambda\neg(C)(\mathfrak{c}) = TC \setminus \lambda(C)(C \setminus \mathfrak{c})$$

is a predicate lifting.

Proof. Because negation preserves inverse images. □

For the remainder of this exposition Λ denotes a set of predicate liftings. Furthermore, we denote the extension of Λ by liftings of the form $\neg\lambda\neg$ by $\overline{\Lambda}$. That is, we put

$$\overline{\Lambda} = \Lambda \cup \{\neg\lambda\neg \mid \lambda \in \Lambda\}.$$

Since predicate liftings can be used to interpret both modalities and atomic propositions we are lead to study propositional logic, enriched with predicate lifting operators, as a logic for coalgebras.

Since the expressiveness and definability results require infinitary logics in the general case, the definition is parametric in a cardinal number κ. Note that atomic propositions also arise through predicate liftings (Example 4.2.4), hence we do not need to include atomic propositions in the definition.

Definition 4.2.8. Suppose κ is a cardinal number. The *language* $\mathcal{L}^\kappa(\Lambda)$ associated with Λ is the least set with grammar

$$\phi ::= \bigwedge \Phi \mid \neg\phi \mid [\lambda]\phi \qquad (\Phi \subseteq \mathcal{L}^\kappa(\Lambda) \text{ with } \mathsf{card}(\Phi) < \kappa \text{ and } \lambda \in \Lambda)$$

Given $(C, \gamma) \in \mathsf{CoAlg}(T)$, the semantics $[\![\phi]\!]_{(C,\gamma)} \subseteq C$ is given inductively by the clauses

- $[\![\bigwedge \Phi]\!]_{(C,\gamma)} = \bigcap_{\phi \in \Phi} [\![\phi]\!]_{(C,\gamma)}$
- $[\![\neg\phi]\!]_{(C,\gamma)} = C \setminus [\![\phi]\!]_{(C,\gamma)}$, and
- $[\![[\lambda]\phi]\!]_{(C,\gamma)} = \gamma^{-1} \circ \lambda(C)([\![\phi]\!]_{(C,\gamma)})$.

Note that $\mathcal{L}^\kappa(\Lambda)$ contains the formula $\mathtt{tt} = \bigwedge \emptyset$ (with $[\![\mathtt{tt}]\!]_{(C,\gamma)} = C$) and that $\mathcal{L}^\kappa(\Lambda)$ is finitary if $\kappa = \omega$. If we want to emphasise that a formula $\phi \in \mathcal{L}^\kappa(\Lambda)$ holds at a specific state $c \in C$ of a coalgebra (C, γ), we write $c \models_\gamma \phi$ for $c \in [\![\phi]\!]_{(C,\gamma)}$.

Given syntax and semantics of coalgebraic modal logic, we now begin the study of the relationship between logical and behavioural equivalence: As with coalgebraic logic (Theorem 4.1.8), we conclude invariance under behavioural equivalence from the fact that the interpretation of formulas is stable under coalgebra morphisms.

Theorem 4.2.9. *If $f : (C,\gamma) \to (D,\delta) \in \mathsf{CoAlg}(T)$, then*

$$[\![\phi]\!]_{(C,\gamma)} = f^{-1}([\![\phi]\!]_{(D,\delta)})$$

for all $\phi \in \mathcal{L}^\kappa(\Lambda)$.

Proof. We proceed by induction on the structure of ϕ. For conjunctions and negations, the claim is evident. So suppose $\phi \in \mathcal{L}^\kappa(\Lambda)$ and $[\![\phi]\!]_{(C,\gamma)} = f^{-1}([\![\phi]\!]_{(D,\delta)})$. Naturality of λ allows us to calculate

$$\begin{aligned} f^{-1}([\![[\lambda]\phi]\!]_{(D,\delta)}) &= (\delta \circ f)^{-1} \circ \lambda(D)([\![\phi]\!]_{(D,\delta)}) \\ &= (Tf \circ \gamma)^{-1} \circ \lambda(D)([\![\phi]\!]_{(D,\delta)}) \\ &= \gamma^{-1} \circ \lambda(C) \circ f^{-1}([\![\phi]\!]_{(D,\delta)}) \\ &= [\![[\lambda]\phi]\!]_{(C,\gamma)}, \end{aligned}$$

establishing the claim. □

The preceding lemma allows us to prove the invariance of coalgebraic modal logic under behavioural equivalence as in the case of coalgebraic logic.

Corollary 4.2.10. *Let $(C,\gamma), (D,\delta) \in \mathsf{CoAlg}(T)$ and $\phi \in \mathcal{L}^\kappa(\Lambda)$. Then*

$$c \models_{(C,\gamma)} \phi \quad \textit{iff} \quad d \models_{(D,\delta)} \phi$$

whenever $(c,d) \in C \times D$ are behaviourally equivalent.

Proof. As in the proof of Corollary 4.1.8. □

The preceding theorem states, that behavioural equivalence implies logical equivalence. The remainder of this paper is concerned with conditions on Λ that also ensure the converse.

4.3 Expressivity of Coalgebraic Modal Logic

While behaviourally equivalent states always have the same theory, the converse is not necessarily true (consider for example the logic given by the empty set of predicate liftings). Logics, for which the converse of Corollary 4.2.10 holds, are called expressive.

This section introduces *separation*, a condition on sets of predicate liftings, and establishes the promised characterisation of behavioural equivalence in logical terms.

The basic idea behind separation is the possibility of distinguishing individual points of TX by means of lifted subsets of X. This is formalised in the following definition.

Definition 4.3.1 (Separation). (i) Suppose C is a set and $\mathcal{C} \subseteq \mathcal{P}(C)$ is a system of subsets of C. We call $\mathcal{C}$ *separating*, if the map $s : C \to \mathcal{P}(\mathcal{C})$, $s(c) = \{\mathfrak{c} \in \mathcal{C} \mid c \in \mathfrak{c}\}$, is monic.

(ii) A set Λ of predicate liftings for T is called *separating*, if, for all sets C, the set $\{\lambda(C)(\mathfrak{c}) \mid \lambda \in \Lambda, \mathfrak{c} \subseteq C\}$ is a separating set of subsets of TC.

In a separating system of subsets, the system contains enough information to distinguish the individual elements of the underlying set. The intuition behind a separating set of predicate liftings is that elements of TC can be distinguished by means of the subsets $\lambda(C)(\mathfrak{c})$ obtained by applying the liftings. Many sets of predicate liftings are indeed separating, notably the predicate liftings giving rise to the interpretation of modalities and atoms in (standard) modal logic.

Example 4.3.2. Suppose $TX = \mathcal{P}(X) \times \mathcal{P}(A)$ as in Section 1.1.5 and consider

$$\Lambda = \{\lambda\} \cup \{\lambda_a \mid a \in A\},$$

where λ and the λ_as are given as in Example 4.2.2 and Example 4.2.4, respectively. We show that Λ is separating. Fix some set C and let $\mathcal{S} = \{\mu(C)(\mathfrak{c}) \mid \mu \in \Lambda \text{ and } \mathfrak{c} \subseteq C\}$. We establish that $s : TC \to \mathcal{P}(\mathcal{S})$, given by $s(c) = \{\mathfrak{s} \in \mathcal{S} \mid c \in \mathfrak{s}\}$ is injective. Note that by definition of s, we have $\mu(C)(\mathfrak{c}) \in s(c)$ iff $c \in \mu(C)(\mathfrak{c})$, for all $c \in C$, $\mathfrak{c} \subseteq C$ and $\mu \in \Lambda$.

So suppose $s(\mathfrak{c}_0, \mathfrak{a}_0) = s(\mathfrak{c}_1, \mathfrak{a}_1)$. Then $(\mathfrak{c}_0, \mathfrak{a}_0) \in \lambda(C)(\mathfrak{c}_0)$, hence $\lambda(C)(\mathfrak{c}_0) \in s(\mathfrak{c}_0, \mathfrak{a}_0) = s(\mathfrak{c}_1, \mathfrak{a}_1)$. So $(\mathfrak{c}_1, \mathfrak{a}_1) \in \lambda(C)(\mathfrak{c}_0)$, that is, $\mathfrak{c}_1 \subseteq \mathfrak{c}_0$ by definition of λ. Now assume $a \in \mathfrak{a}_1$. Then $(\mathfrak{c}_1, \mathfrak{a}_1) \in \lambda_a(C)(C)$, thus $\lambda_a(C)(C) \in s(\mathfrak{c}_1, \mathfrak{a}_1) = s(\mathfrak{c}_0, \mathfrak{a}_0)$. Therefore $(\mathfrak{c}_0, \mathfrak{a}_0) \in \lambda_a(C)(C)$, showing $a \in \mathfrak{a}_0$ and, since a was arbitrary, $\mathfrak{a}_1 \subseteq \mathfrak{a}_0$. We conclude $(\mathfrak{c}_0, \mathfrak{a}_0) = (\mathfrak{c}_1, \mathfrak{a}_1)$ by symmetry.

We now show that, given a separating set of predicate liftings, every singleton set $\{x\}$, for $x \in TX$, arises as the intersection of lifted subsets of X. In order to obtain this representation we have to use both liftings $\lambda \in \Lambda$ and liftings of the form $\neg\lambda\neg$, as introduced in Proposition 4.2.7. Recall the notation $\overline{\Lambda} = \Lambda \cup \{\neg\lambda\neg \mid \lambda \in \Lambda\}$ introduced in Section 4.2. Furthermore, if A is a set and $a \in A$, we denote the lattice of supersets of $\{a\}$ by $a\Uparrow = \{\mathfrak{a} \subseteq A \mid a \in \mathfrak{a}\}$.

Lemma 4.3.3. *Suppose Λ is separating and X is a set. Then*

$$\bigcap\{\lambda(X)(\mathfrak{x}) \mid \lambda \in \overline{\Lambda} \text{ and } \mathfrak{x} \in \lambda(X)^{-1}(x\Uparrow)\} = \{x\}$$

for all $x \in TX$.

Proof. Fix $x \in TX$ and let LHS $= \bigcap\{\lambda(X)(\mathfrak{x}) \mid \lambda \in \overline{\Lambda}$ and $\mathfrak{x} \in \lambda(X)^{-1}(x\Uparrow)\}$.

If λ is a predicate lifting, then $\mathfrak{x} \in \lambda(X)^{-1}(x\Uparrow)$ iff $x \in \lambda(X)(\mathfrak{x})$, therefore $\{x\} \subseteq$ LHS.

In order to see that LHS $\subseteq \{x\}$, consider the assignment $m(y) = \bigcup_{\lambda\in\Lambda}\{\lambda(X)(\mathfrak{x}) \mid \mathfrak{x} \in \lambda(X)^{-1}(y\Uparrow)\}$. Since Λ is separating, m is monic and it suffices to show that $m(x) = m(y)$ for all $y \in$ LHS. This is equivalent to

$$x \in \lambda(X)(\mathfrak{x}) \quad \text{iff} \quad y \in \lambda(X)(\mathfrak{x})$$

for all $y \in$ LHS, $\lambda \in \Lambda$ and $\mathfrak{x} \subseteq X$.

First suppose that $x \in \lambda(X)(\mathfrak{x})$ for some $\lambda \in \Lambda$ and some $\mathfrak{x} \subseteq X$. Since $y \in LHS$, clearly $y \in \lambda(X)(\mathfrak{x})$. Conversely, if $x \notin \lambda(X)(\mathfrak{x})$, we have $x \in \neg\lambda\neg(X \setminus \mathfrak{x})$, hence $y \in \neg\lambda\neg(X \setminus \mathfrak{x})$, which amounts to $y \notin \lambda(X)(\mathfrak{x})$. □

This lemma provides a first handle for isolating a single point $x \in TX$. However, we have to consider the liftings of sets whose cardinality is not bounded above (amounting to disjunctions of unbounded cardinality on the logical side). We can do better if T is κ-accessible:

Lemma 4.3.4. *Suppose T is κ-accessible, X is a set and $x \in TX$. Then there exists $\mathfrak{x}_0(x) \subseteq X$ with $\mathsf{card}(\mathfrak{x}_0(x)) < \kappa$ such that*

$$x \in \lambda(X)(\mathfrak{x}) \quad \textit{iff} \quad x \in \lambda(X)(\mathfrak{x} \cap \mathfrak{x}_0(x))$$

for all $\mathfrak{x} \subseteq X$ and all predicate liftings λ for T.

Proof. Since T is κ-accessible, there exists a subset $\mathfrak{x}_0 = \mathfrak{x}_0(x) \subseteq X$ with $\mathsf{card}(\mathfrak{x}_0) < \kappa$ such that $x = (Ti)(x_0)$ for some $x_0 \in \mathfrak{x}_0$, where $i : \mathfrak{x}_0 \to X$ denotes the inclusion.

Since predicate liftings preserve order by definition, we have $x \in \lambda(X)(\mathfrak{x})$ whenever $x \in \lambda(X)(\mathfrak{x} \cap \mathfrak{x}_0)$. For the other implication, suppose that $x \in \lambda(X)(\mathfrak{x})$ for some $\mathfrak{x} \subseteq X$ and consider the diagram

$$\begin{tikzcd}
\mathcal{P}(X) \arrow[r, "\lambda(X)"] \arrow[d, "i^{-1}"'] & \mathcal{P}(TX) \arrow[d, "(Ti)^{-1}"] \\
\mathcal{P}(\mathfrak{x}_0) \arrow[r, "\lambda(\mathfrak{x}_0)"'] & \mathcal{P}(T\mathfrak{x}_0)
\end{tikzcd}$$

which commutes by the naturality of λ. We obtain

$$\begin{aligned}
(Ti)^{-1}(\lambda(X)(\mathfrak{x} \cap \mathfrak{x}_0)) &= \lambda(\mathfrak{x}_0) \circ i^{-1}(\mathfrak{x} \cap \mathfrak{x}_0) \\
&= \lambda(\mathfrak{x}_0) \circ i^{-1}(\mathfrak{x}) \\
&= (Ti)^{-1}(\lambda(X)(\mathfrak{x})).
\end{aligned}$$

Since $x = (Ti)(x_0) \in \lambda(X)(\mathfrak{x})$, we have $x_0 \in (Ti)^{-1}(\lambda(X)(\mathfrak{x})) = (Ti)^{-1}(\lambda(X)(\mathfrak{x} \cap \mathfrak{x}_0))$, hence $x = (Ti)(x_0) \in \lambda(X)(\mathfrak{x} \cap \mathfrak{x}_0)$. □

Combining the last two lemmas allows us to isolate single points $x \in TX$ by liftings of subsets $\mathfrak{x} \subseteq X$, which are of cardinality less than κ. This is the content of the following corollary, which immediately follows from the fact that predicate liftings preserve order.

Corollary 4.3.5. *Suppose T is κ-accessible, Λ is separating and X is a set. Then*

$$\bigcap \{\lambda(X)(\mathfrak{x}) \mid \lambda \in \overline{\Lambda} \text{ and } \mathfrak{x} \in \lambda(X)^{-1}(t\uparrow), \mathfrak{x} \subseteq \mathfrak{x}_0(x)\} = \{x\}$$

for $x \in TX$ and $\mathfrak{x}_0(x)$ as in Lemma 4.3.4.

The next lemma transfers the preceding result to a logical setting. We show that the logics induced by a separating set of predicate liftings can distinguish distinct elements $z_0, z_1 \in T^\alpha 1$, for α less than the accessibility degree of T. As in Theorem 3.1.11, (Z, ζ) is the final T-coalgebra.

Lemma 4.3.6. *Suppose T is κ-accessible and Λ is separating. Then there exists a cardinal σ such that for all $\alpha < \kappa$ and all $z \in Z_\alpha$, there is a formula $\phi^\alpha_z \in \mathcal{L}^\sigma(\Lambda)$ with $[\![\phi^\alpha_z]\!]_{(Z,\zeta)} = \zeta_\alpha^{-1}(\{z\})$.*

Proof. Choose σ such that $\sigma > \mathsf{card}(\Lambda)$ and $\sigma > 2^\alpha$ for all $\alpha < \kappa$.

We define ϕ^α_z by transfinite induction. If α is a limit ordinal, let $\phi^\alpha_z = \bigwedge_{\beta<\alpha} \phi^\beta_{p^\alpha_\beta(z)}$, where $p^\alpha_\beta : Z_\alpha \to Z_\beta$ is the connecting morphism of the terminal sequence. We obtain $[\![\phi_z^\alpha]\!]_{(Z,\zeta)} = \zeta_\alpha^{-1}(\{z\})$ using the fact that $(Z_\alpha, (p_\beta^\alpha)_{\beta<\alpha})$ is a limiting cone.

Now suppose $\alpha = \beta + 1$ is a successor ordinal. By Lemma 4.3.4 there exists a subset $\mathfrak{z}_0 = \mathfrak{z}_0(z) \subseteq Z_\beta$ with $\mathsf{card}(\mathfrak{z}_0) < \kappa$ such that $z \in \lambda(Z_\beta)(\mathfrak{z})$ iff $z \in \lambda(Z_\beta)(\mathfrak{z} \cap \mathfrak{z}_0)$ for all $\mathfrak{z} \subseteq Z_\beta$ and all $\lambda \in \Lambda$.

We abbreviate $\phi_{\mathfrak{z}}^\beta = \bigvee_{z\in\mathfrak{z}} \phi_z^\beta$ for $\mathfrak{z} \subseteq Z_\beta$ and let $\phi_z^\alpha = \phi_p \wedge \phi_n$ where

$$\phi_p = \bigwedge_{\lambda\in\Lambda} \bigwedge\{[\lambda]\phi_{\mathfrak{z}}^\beta \mid \mathfrak{z} \subseteq \mathfrak{z}_0 \text{ and } \mathfrak{z} \in \lambda(Z_\beta)^{-1}(z\uparrow)\}$$

and

$$\phi_n = \bigwedge_{\lambda\in\Lambda} \bigwedge\{\neg[\lambda]\neg\phi_{\mathfrak{z}}^\beta \mid \mathfrak{z} \subseteq \mathfrak{z}_0 \text{ and } \mathfrak{z} \in (\neg\lambda\neg)(Z_\beta)^{-1}(z\uparrow)\}.$$

Corollary 4.3.5 and the naturality of predicate liftings ensure that $[\![\phi_z^\alpha]\!]_{(Z,\zeta)} = \zeta_\alpha^{-1}(\{z\})$.

□

The main theorem is now easy:

Theorem 4.3.7. *Suppose T is accessible and Λ is separating. Then there exists a cardinal σ such that, for $(C,\gamma), (D,\delta) \in \mathsf{CoAlg}(T)$,*

$$c \models_{(C,\gamma)} \phi \textit{ iff } d \models_{(D,\delta)} \phi$$

for all $\phi \in \mathcal{L}^\sigma(\Lambda)$ implies that $(c,d) \in C \times D$ are behaviourally equivalent.

Proof. Suppose that T is κ-accessible and let σ be given as in the previous Lemma. Denote the final T-coalgebra by (Z,ζ) and the logical theory of $z \in Z$ by $\mathrm{Th}(z) = \{\phi \in \mathcal{L}^\sigma(\Lambda) \mid c \models_\zeta \phi\}$. By Theorem 3.1.11 it suffices to show that $\zeta_\alpha(z_0) = \zeta_\alpha(z_1)$ for all $\alpha < \kappa$, if $z_0, z_1 \in Z$ with $\mathrm{Th}(z_0) = \mathrm{Th}(z_1)$.

Fix some ordinal $\alpha < \kappa$ and consider the formula $\phi = \phi^\alpha_{\zeta_\alpha(z_0)}$ as defined in the last lemma. Then $\phi \in \mathrm{Th}(z_0) = \mathrm{Th}(z_1)$. That is, $z_1 \in [\![\phi]\!]_{(Z,\zeta)} = \zeta_\alpha^{-1}(\{\zeta_\alpha(z_0)\})$ by construction. But this is clearly equivalent to $\zeta_\alpha(z_0) = \zeta_\alpha(z_1)$, which is what we had to show. □

The preceding theorem does not give an upper bound for the size of conjunctions needed to obtain an expressive logic. If we extract the bound from Lemma 4.3.6, we see that we need a cardinal which is larger than 2^α for every α which is less than the accessibility degree of T. Hence we obtain the following corollary for ω-accessible endofunctors:

Corollary 4.3.8. *Suppose T is ω-accessible, Λ is separating and finite. Then $\mathcal{L}^\omega(\Lambda)$ is expressive.*

If T is κ-accessible and $\mathsf{card}(\Lambda) < \kappa$ we do not in general obtain that $\mathcal{L}^\kappa(\Lambda)$ is expressive (unless κ is inaccessible). This can however be established if the predicate liftings under consideration preserve intersections:

Definition 4.3.9. We say that Λ is *intersection preserving*, if

$$\lambda(X)(\bigcap \mathfrak{X}) = \bigcap\{\lambda(X)(\mathfrak{x}) \mid \mathfrak{x} \in \mathfrak{X}\}$$

for all $\lambda \in \Lambda$, whenever X is a set and $\mathfrak{X} \subseteq \mathcal{P}(X)$.

This property is present in many examples. In particular, predicate liftings constructed via Proposition 4.2.3 and 4.2.5 have this property. Assuming the preservation of intersections, we have:

Lemma 4.3.10. *Suppose T is κ-accessible and Λ is separating, intersection-preserving and $\mathsf{card}(\Lambda) < \kappa$. Then, for all $\alpha < \kappa$ and all $z \in T^\alpha 1$, there exists a formula $\phi^\alpha_z \in \mathcal{L}^\kappa(\Lambda)$ with $[\![\phi^\alpha_z]\!]_{(Z,\zeta)} = \zeta_\alpha^{-1}(\{z\})$.*

Proof. We use the same notation as in the proof of Lemma 4.3.6 and treat the case of limit ordinals in the same way.

For successor ordinals $\alpha = \beta+1$, we put $\phi^\alpha_z = \phi_p \wedge \phi_n$, changing the definition of ϕ_p and ϕ_n to

$$\phi_p = \bigwedge_{\lambda\in\Lambda} [\lambda]\phi^\beta_{\mathfrak{z}_\lambda}$$

where $\mathfrak{z}_\lambda = \bigcap\{\mathfrak{z} \subseteq \mathfrak{z}_0 \mid \mathfrak{z} \in \lambda(Z_\beta)^{-1}(z\upharpoonright)\}$, and

$$\phi_n = \bigwedge_{\lambda\in\Lambda} \bigwedge\{\neg[\lambda]\neg\phi^\beta_{z'} \mid z' \in \mathfrak{z}_0 \text{ and } \{z'\} \in (\neg\lambda\neg)(Z_\beta)^{-1}(z\upharpoonright)\}$$

One then applies the preservation of intersections to obtain $[\![\phi^\alpha_z]\!]_{(Z,\zeta)} = \zeta_\alpha^{-1}(\{z\})$. □

For intersection preserving sets of predicate liftings, we thus obtain κ as an upper bound for the size of the conjunctions and disjunctions. This suffices to obtain an expressive logic for κ-accessible endofunctors:

Theorem 4.3.11. *Suppose T is κ-accessible, Λ is separating and intersection preserving. Then, for $(C,\gamma),(D,\delta) \in \mathsf{CoAlg}(T)$,*

$$c \models_{(C,\gamma)} \phi \textit{ iff } d \models_{(D,\delta)} \phi$$

for all $\phi \in \mathcal{L}(\Lambda)$ implies that $(c,d) \in C \times D$ are behaviourally equivalent.

Proof. Analogous to the proof of Theorem 4.3.7, but using Lemma 4.3.10 in place of Lemma 4.3.6. □

We conclude the section with some examples illustrating the expressiveness results.

Example 4.3.12.

(*i*) Let $TX = \mathcal{P}_\omega(L \times X)$. We have argued in Example 3.1.12 that a T-coalgebra (C, γ) is a finitely branching labelled transition system.

Consider, for $l \in L$, the natural transformation $\mu_l(C) : \mathcal{P}_\omega(L \times C) \to \mathcal{P}(C)$ given by $\mu_l(C) = i \circ \mathcal{P}_\omega(\pi_2)$, where $i : \mathcal{P}_\omega(C) \to \mathcal{P}(C)$ is the inclusion and $\pi_2 : L \times C \to C$ is the projection. By Proposition 4.2.3, every μ_l gives rise to a predicate lifting λ_l. A calculation similar to the one in Example 4.3.2 shows, that the set $\Lambda = \{\lambda_l \mid l \in L\}$ thus obtained is separating.

Given a T-coalgebra (C, γ), we put $c \xrightarrow{l}$ if $(l, c') \in \gamma(c)$. Under this correspondence, we have $c \models [\lambda_l]\phi$ iff $\forall c'.c \xrightarrow{l} c' \implies c' \models \phi$ for $c \in C$ and $\phi \in \mathcal{L}^\omega(\Lambda)$. Corollary 4.3.8 then re-proves the characterisation result by Hennessy and Milner [24] in the coalgebraic framework.

(*ii*) Suppose $TX = \mathcal{P}_\kappa(X) \times \mathcal{P}(A)$ for some set A (of atomic propositions) with $\mathsf{card}(A) < \kappa$. Consider the predicate lifting λ and, for $a \in A$, the liftings λ_a, as described in Example 4.2.2 and Example 4.2.4.

If $\Lambda = \{\lambda\} \cup \{\lambda_a \mid a \in A\}$, then Λ is intersection-preserving and, by Theorem 4.3.11, the logical equivalence induced by $\mathcal{L}^\kappa(\Lambda)$ coincides with behavioural equivalence. This amounts to saying that, in Kripke models with branching degree less than κ, modal logic with conjunctions of size less than κ characterises behavioural equivalence.

If $\kappa > \omega$, one can use an argument similar to that used in Example 3.1.12 to show that the equivalence induced $\mathcal{L}^\omega(\Lambda)$ is weaker than behavioural equivalence.

(*iii*) Suppose I and O are finite sets and $TX = (X \times O)^I$. We have shown in Example 1.1.3 that T-coalgebras are Mealy Machines. In Example 4.2.6 we have introduced the set $\Lambda = \{\lambda_i \mid i \in I\} \cup \{\mu_{(i,o)} \mid (i, o) \in I \times O\}$ of predicate liftings for T. It is easy to see that Λ is separating and intersection preserving. Hence $\mathcal{L}^\omega(\Lambda)$ characterises states of input output automata up to behavioural equivalence.

4.4 Notes

In Moss' original paper [39], the language of coalgebraic logic is a proper class of formulas and was obtained through an initial algebra construction, requiring that the endofunctor under consideration can be continously extended from sets to classes. We re-phrase coalgebraic logic in terms of the more standard notion of accessibility; this encompasses Moss' original setting, which can be recovered when considering κ-accessible functors, where κ is an inaccessible cardinal; our formulation has the advantage of being more fine grained, that is, it gives more precise bounds for the cardinality of conjunctions and disjunctions needed to obtain expressivity.

The extension of functors to relators was first discovered in [12]; see also [52] and [6] for applications of relators in a coalgebraic setting.

Predicate liftings were first used by Hermida and Jacobs [28] in the context of (co-)induction principles and later by Jacobs [27] in conjunction with modal logic. There, as well as in the related paper [48], predicate liftings are syntactically defined entities, and naturality, which we take as our defining property, is derived.

In contrast to the approaches presented in [27, 33, 48, 49], the framework of coalgebraic modal logic does not rely on the syntactic analysis of the signature functor and can be instantiated for functors whose accessibility degree is $> \omega$.

We have not treated multi-sorted logics in the present paper. The results extend smoothly to the multi-sorted case, that is, to coalgebras for functors $T : \mathsf{Set}^n \to \mathsf{Set}^n$.

The material on coalgebraic logic and coalgebraic modal logic is taken from [44]; a preliminary version of the expressivity results has appeared as [42].

4.5 exercises

Exercise 4.5.1. Prove Proposition 4.1.2.

Exercise 4.5.2. Prove Lemma 4.1.6.

Exercise 4.5.3. Consider $TX = 1 + A \times X$. Think of T-algebras as stream automata, which have the additional possibility of termination. Give examples of predicate liftings and formulas corresponding to the following statements:

(a) The automaton terminates in the next step, and (b) the automaton does not terminate and produces the element $a \in A$.

Exercise 4.5.4. For $TX = A \times X \times X$, we can think of T-coalgebras as infinite binary trees, the nodes of which are labelled with elements $a \in A$. Give predicate liftings which can be used to express that a formula holds in the left (resp. right) subtree.

Exercise 4.5.5. Extend the predicate liftings constructed in Exercise 4.5.3 and 4.5.4 such that the extended sets of liftings are separating.

Chapter 5

Proof Systems For Coalgebraic Logics

After having discussed logics for expressing properties of systems, we now focus on proof systems, which allow us to derive properties from one another. Of course, we only want to derive valid consequences, and therefore begin with making the notion of consequence precise. an assumption, if it is a valid consequence (of the assumption).

5.1 Properties of the Consequence Relation

In the context of modal logic, one distinguishes between two different such relations: If ϕ and ψ are modal formulas, one calls ψ a *global consequence* of ϕ, if

$$(\forall c \in C.\, c \models_\gamma \phi) \implies (\forall c \in C.\, c \models_\gamma \psi)$$

for all $(C, \gamma) \in \mathsf{CoAlg}(T)$. That is, the class of models, which globally satisfy ψ is a subclass of the models of ψ. We will be concerned with local consequence:

Definition 5.1.1 (Local Consequence). Let $\phi, \psi \in \mathcal{L}(\Lambda)$. We say that ψ is a *local consequence* of ϕ, if

$$\forall c \in C.\, (c \models_\gamma \phi \implies c \models_\gamma \psi)$$

for all $(C, \gamma) \in \mathsf{CoAlg}(T)$. If ψ is a local consequence of ϕ, we write $\phi \models \psi$.

The nature of the local consequence relation allows to prove

Lemma 5.1.2. *Let $\phi, \psi, \rho \in \mathcal{L}(\Lambda)$. Then $\phi \wedge \psi \models \rho$ iff $\phi \models \psi \to \rho$.*

The proof is straightforward, and therefore omitted.

Remark 5.1.3. The previous lemma opens two ways of finding appropriate proof systems for coalgebraic modal logic. We can either axiomatise the set of tautologies of the logic or the local consequence relation. We work with local consequence, since this notion allows us to express directly the fact that predicate liftings preserve order by means of a logical rule.

We now start analysing the inductive definition that gives rise to $\mathcal{L}(\Lambda)$. The road map is as follows: We consider a set L, which we think of as a set of formulas (and assume to be closed under propositional connectives), together with a function $d : L \to \mathcal{P}(C)$, which assigns to every formula $\phi \in L$ the set of states $d(\phi) \subseteq C$ which satisfy ϕ; note that we just require C to be a set.

To the map $d : L \to \mathcal{P}(C)$ we associate a function $\mathsf{Lift}(d) : \mathsf{Lift}(L) \to \mathcal{P}(TC)$ by inductively extending the assignment $[\lambda]\phi \mapsto \lambda(C)(d(\phi))$ to the whole of $\mathsf{Lift}(L)$. Starting with $(\mathcal{L}^0, 1, d^0)$ (where $\mathcal{L}^0$ is the set of propositional formulas over the empty set of atoms, 1 is any one element set and $d^0 : \mathcal{L}^0 \to \mathcal{P}(1)$ is the unique mapping), we obtain a sequence of objects $(\mathcal{L}^n, T^n1, d^n) = \mathsf{Lift}^n(\mathcal{L}^0, 1, d^0)$ by repeatedly applying Lift to $(\mathcal{L}^0, 1, d^0)$.

This construction allows us to view $\mathcal{L}$ as the stratification $\mathcal{L} = \bigcup_{n\in\omega} \mathcal{L}^n$, where $\mathcal{L}^n$ contains all $\phi \in \mathcal{L}$ with $\mathrm{rank}(\phi) \leq n$. Furthermore, we show how to reconstruct the semantics $[\![\phi]\!]_\gamma$ of $\phi \in \mathcal{L}^n$ with respect to an arbitrary model (C, γ) from $d^n(\phi)$. This enables us to reduce local consequence $\phi \models \psi$ to set-theoretic inclusion $d^n(\phi) \subseteq d^n(\psi)$. We begin with the definition of Lift on sets, which we think of as sets of formulas.

Definition 5.1.4. Suppose L is a set (of formulas). We denote the set of propositional formulas with the elements of L as atoms by $\mathsf{Prop}(L)$. Furthermore, let

$$\begin{aligned} \mathsf{Up}(L) &= \{[\lambda]\phi \mid \lambda \in \Lambda, \phi \in L\} \\ \mathsf{Lift}(L) &= \mathsf{Prop} \circ \mathsf{Up}(L) \end{aligned}$$

and $\mathcal{L}^0 = \mathsf{Prop}(\emptyset)$, $\mathcal{L}^{n+1} = \mathsf{Lift}(\mathcal{L}^n)$.

Note that an application of Lift to a set of modal formulas with rank $\leq n$ produces modal formulas with rank $\leq n + 1$, where the rank of a formula is given inductively by $\mathrm{rank}(\mathrm{ff}) = 0$, $\mathrm{rank}(\phi \to \psi) = \max\{\mathrm{rank}(\phi), \mathrm{rank}(\psi)\}$ and $\mathrm{rank}([\lambda]\phi) = 1 + \mathrm{rank}(\phi)$. We obtain the whole of $\mathcal{L}$ by iteration:

Lemma 5.1.5. *$\mathcal{L}^n = \{\phi \in \mathcal{L} \mid \mathrm{rank}(\phi) \leq n\}$. In particular, $\mathcal{L} = \bigcup_{n\in\omega} \mathcal{L}^n$.*

Proof. By induction on n, one shows that $\phi \in \mathcal{L}^n$ has rank at most n. For the other inclusion, use induction on the structure of $\phi \in \mathcal{L}$. □

Alternatively, one can show that Lift is monotone and characterise $\mathcal{L}$ as the least fixed point of Lift. Either way we obtain an alternative inductive definition of the language $\mathcal{L}$ of coalgebraic modal logic. The next step is to synchronise the (inductive) definition of the semantics of $\mathcal{L}$ with the definition obtained as the least fixed point of Lift. This is taken care of by

Definition 5.1.6. Suppose L is a set (of formulas) and C is a set. Given a (denotation) function $d : L \to \mathcal{P}(C)$ we denote the extension[1] of d to $\mathsf{Prop}(L)$ by $\mathsf{Prop}(d) : \mathsf{Prop}(L) \to \mathcal{P}(C)$. We also define functions $\mathsf{Up}(d) : \mathsf{Up}(L) \to \mathcal{P}(TC)$ and $\mathsf{Lift}(d) : \mathsf{Lift}(L) \to \mathcal{P}(TC)$ by

$$\begin{aligned} \mathsf{Up}(d)([\lambda]\phi) &= \lambda(C)(d(\phi)) \qquad\qquad \text{and} \\ \mathsf{Lift}(d)(\phi) &= \mathsf{Prop}(\mathsf{Up}(d))(\phi), \end{aligned}$$

respectively. We denote the function $\mathcal{L}^0 \to \mathcal{P}(1)$ given by $\phi \mapsto 1$ iff ϕ is a tautology (and $\phi \mapsto \emptyset$ otherwise) by d^0. Finally, let $d^{n+1} = \mathsf{Lift}(d^n) : \mathcal{L}^{n+1} \to \mathcal{P}(T^{n+1}1)$ and write $\phi \models_n \psi$ if $d^n(\phi) \subseteq d^n(\psi)$ (and $\phi, \psi \in \mathcal{L}^n$).

We sometimes call d^n the n-step denotation function, since it interprets formulas, which incorporate information about at most n transition steps. The next theorem shows, that – for formulas of rank $\leq n$ – the semantic consequence relation can be reconstructed from the n-step denotations:

Theorem 5.1.7. *Suppose $\phi, \psi \in \mathcal{L}^n$. Then $\phi \models \psi$ iff $\phi \models_n \psi$.*

Note that Theorem 5.1.7 allows us to replace semantical consequence (which is quantified over all models) by set-theoretic containment. For the proof, we introduce some auxiliary notation. If $f : C \to TC$ is a function, we inductively define a sequence of mappings $(f_n)_{n\in\omega}$ by

$$\begin{aligned} f_0 &= !_C : C \to T^0 1 = 1 \\ f_{n+1} &= Tf_n \circ f : C \to T^{n+1}1, \end{aligned}$$

where $!_C : C \to 1$ is the uniquely defined surjection. Given the definition of f_n, the next lemma establishes a relation between the n-step denotation $d^n(\phi)$ and the semantics $[\![\phi]\!]$ for formulas $\phi \in \mathcal{L}$.

Lemma 5.1.8. *Suppose $\phi \in \mathcal{L}^n$ and $(C, \gamma) \in \mathsf{CoAlg}(T)$. Then $[\![\phi]\!]_\gamma = \gamma_n^{-1} \circ d^n(\phi)$.*

Proof. By induction on n using the naturality of predicate liftings. □

[1]This extension is defined inductively by $\mathsf{Prop}(d)(\phi) = d(\phi)$ for $\phi \in L$, $\mathsf{Prop}(d)(\mathrm{ff}) = \emptyset$ and $\mathsf{Prop}(d)(\phi \to \psi) = (C \setminus \mathsf{Prop}(d)(\phi)) \cup \mathsf{Prop}(d)(\psi)$.

We continue by noting that, since T is assumed to be non-trivial, the canonical map $e^0 : T1 \to 1$ is a surjection, and has a right inverse $f^0 : 1 \to T1$ with $e^0 \circ f^0 = \mathrm{id}_1$. Letting $e^n = T^n e^0$ and $f^n = T^n f^0$, we obtain $e^n \circ f^n = \mathrm{id}_{T^n 1}$ by the functoriality of T. Note that $f^n : T^n 1 \to T^{n+1}1$ qualifies as a coalgebra structure. We need one little technical lemma before we are ready to embark on the proof of Theorem 5.1.7.

Lemma 5.1.9. *For all $k \leq n$: $f^n_k = T^k(!_{T^{n-k}1})$. In particular, $f^n_n = \mathrm{id}_{T^n 1}$.*

Proof. Fix an arbitrary $n \in \omega$ and proceed by induction on k. For $k = 0$ we have $f^n_0 = !_{T^n 1} = T^0(!_{T^n 1})$. To get from k to $k+1$ assume that the equation is valid for k (and that $k+1 \leq n$). Unravelling the definitions, we obtain $f^n_{k+1} = T(f^n_k) \circ f^n = T(T^k(!_{T^{n-k}1}) \circ T^{k+1} f^{n-k-1}) = T^{k+1}(!_{T^{n-k}1} \circ f^{n-k-1}) = T^{k+1}(!_{T^{n-(k+1)}1})$, as claimed. □

We are now ready for the

Proof of Theorem 5.1.7. Let $\phi, \psi \in \mathcal{L}^n$. First assume that $d^n(\phi) \subseteq d^n(\psi)$. If $(C, \gamma) \in \mathsf{CoAlg}(T)$ is any T-coalgebra, we obtain $[\![\phi]\!]_\gamma = \gamma_n^{-1} \circ d^n(\phi) \subseteq \gamma_n^{-1} \circ d^n(\psi) = [\![\psi]\!]_\gamma$ by two applications of Lemma 5.1.8 (and the fact that inverse images preserve inclusion). Hence $\phi \models \psi$. Now assume $\phi \models \psi$. Taking $(C, \gamma) = (T^n 1, f^n)$, we have $[\![\phi]\!]_{f^n} \subseteq [\![\psi]\!]_{f^n}$ by the definition of the semantical consequence relation $\models$. Since $f^n_n = \mathrm{id}_{T^n 1}$ (which was established in Lemma 5.1.9), we obtain $d^n(\phi) = (f^n_n)^{-1} \circ d^n(\phi) = [\![\phi]\!]_{f^n} \subseteq [\![\psi]\!]_{f^n} = (f^n_n)^{-1} \circ d^n(\psi) = d^n(\psi)$, again by applying Lemma 5.1.8 twice. □

This theorem characterises local consequence in terms of the so-called terminal sequence $(T^n 1)_{n \in \omega}$ of the underlying endofunctor T. The terminal sequence (iterated through the class of all ordinal numbers) is frequently used to construct final coalgebras (see [7, 4, 62]). Since we are working with finitary logic, there is no need to iterate the construction further than ω.

Using Theorem 5.1.7, we can determine validity $\phi \models \psi$ by just looking at one model: it suffices to determine the rank of ϕ and ψ and check, whether the n-step denotation $d^n(\phi)$ of ϕ is a subset of the n-step denotation $d^n(\psi)$ of ψ. This fact will be exploited twice in the sequel. In the next section, we construct a logical consequence relation $\vdash \subseteq \mathcal{L} \times \mathcal{L}$ which can be seen to arise as the union of relations $\vdash_n \subseteq \mathcal{L}^n \times \mathcal{L}^n$. Theorem 5.1.7 then allows us to prove local soundness and completeness results by induction on n. The second place where Theorem 5.1.7 will be important is the decidability of $\vdash$: if T is finite, that is, the approximants $T^n 1$ are finite sets, the problem $d^n(\phi) \subseteq d^n(\psi)$ is decidable for $\phi, \psi \in \mathcal{L}^n$.

5.2 Proof Systems for Coalgebraic Modal Logic

This section introduces proof systems for coalgebraic modal logic and establishes soundness and completeness. The results are obtained inductively by representing the entailment relation $\vdash \subseteq \mathcal{L} \times \mathcal{L}$ by a union of relations $\vdash_n \subseteq \mathcal{L}^n \times \mathcal{L}^n$. Throughout this section we fix a set Λ of predicate liftings for T. As in the preceding section we abbreviate $\mathcal{L}(\Lambda)$ by $\mathcal{L}$. Furthermore, we fix a denumerable set $\mathfrak{X} = \{x_1, x_2, \dots\}$ (of formulas) and a set $\mathsf{Ax} \subseteq \mathsf{Lift} \circ \mathsf{Prop}(\mathfrak{X}) \times \mathsf{Lift} \circ \mathsf{Prop}(\mathfrak{X})$ (of axiom schemes). The role of Ax is to encode information about the structure of T, i.e. it provides us with additional information which is needed to obtain a complete axiomatisation of $\models$. If $\phi, \psi \in \mathsf{Lift} \circ \mathsf{Prop}(\mathfrak{X})$, we write $\phi \vdash \psi \in \mathsf{Ax}$, instead of $(\phi, \psi) \in \mathsf{Ax}$. We illustrate the role of Ax by means of a small example.

Example 5.2.1. In the case of (standard) modal logic, we need axioms to express distributivity of $\Box$ over conjunctions. That is, If $TX = \mathcal{P}(X) \times \mathcal{P}(A)$ (as in Section 1.1.5) and $\Box$ denotes the lifting λ in Example 4.2.2, an axiomatisation of local consequence needs the axiom

$$\Box x_1 \wedge \Box x_2 \vdash \Box(x_1 \wedge x_2),$$

formalising that $\Box$ distributes over conjunctions. This can be accommodated in the above definition of axiom: the expressions $x_1 \wedge x_2$, x_1 and x_2 are elements of $\mathsf{Prop}(\mathfrak{X})$, hence both $\Box(x_1 \wedge x_2)$ and $\Box x_1 \wedge \Box x_2$ are elements of $\mathsf{Lift} \circ \mathsf{Prop}(\mathfrak{X})$. Thus $\Box x_1 \wedge \Box x_2 \vdash \Box(x_1 \wedge x_2)$ is a possible axiom.

Note that axiom schemes are required to be of a rather special form, that is, they are not allowed to contain nested modal operators. Assuming axioms of this form, substitution instances with formulas of rank $\leq n$ have rank $\leq n+1$. This enables us to define n-step consequence relations $\vdash_n \subseteq \mathcal{L}^n \times \mathcal{L}^n$ such that the union of all $\vdash_n$'s equals the logical consequence relation. The synchronism of the construction of $\mathcal{L}$ with both the n-step consequence relations $\vdash_n$ and the n-step denotation functions d_n will then allow us to use induction on n to prove soundness and completeness results for coalgebraic modal logic.

Since we do not restrict our attention to a specific endofunctor T, we do not consider a concrete set of axioms that we work with. Instead, we state what we understand by the term "axiom scheme" (that is, an element of $\mathsf{Lift} \circ \mathsf{Prop}(\mathfrak{X}) \times \mathsf{Lift} \circ \mathsf{Prop}(\mathfrak{X})$) and investigate conditions on (sets of) axioms schemes, which ensure soundness and completeness of the logic arising through the set Λ of liftings.

An example of the general theory, where we instantiate the theorems with the case of Kripke models (Section 1.1.5), is given at the end of this section. We now introduce the logical consequence relation of coalgebraic modal logic.

Definition 5.2.2. We define $\vdash \subseteq \mathcal{L} \times \mathcal{L}$ to be the least relation which

- is closed under propositional entailment
- is closed under the rule

$$(\text{op})\frac{\phi \vdash \psi}{[\lambda]\phi \vdash [\lambda]\psi} \quad (\lambda \in \Lambda)$$

- contains all substitution instances of axioms $\phi \vdash \psi \in \mathsf{Ax}$.

The relation $\vdash$ will be the object of study for the remainder of this section. As in the previous section, we show that $\vdash$ arises as the union of relations $\vdash_n \subseteq \mathcal{L}^n \times \mathcal{L}^n$. By exploiting Lemma 5.1.7, we show soundness and completeness by arguing that, for $\phi, \psi \in \mathcal{L}^n$, we have $\phi \vdash \psi$ iff $\phi \vdash_n \psi$ iff $\phi \models_n \psi$ iff $\phi \models \psi$. We begin by introducing the operators used in the definition of the approximating relations $\vdash_n$.

In order to facilitate our presentation, we assume that every relation comes with its carrier set, i.e. we consider relations as pairs $(E, \vdash_E)$ where E is a set and $\vdash_E \subseteq E \times E$. We write $\vec{x} = (x_0, \dots, x_n)$ and $\vec{\alpha} = (\alpha_0, \dots, \alpha_n)$ for finite sequences of variables (or formulas) and denote the substitution of x_i by α_i in a formula ϕ by $\phi[\vec{x}/\vec{\alpha}]$, where we implicitly assume that both sequences are of equal length.

Definition 5.2.3. Suppose E is a set (of formulas) equipped with a relation $\vdash_E \subseteq E \times E$. We denote the least relation on $\mathsf{Prop}(E)$, which contains $\vdash_E$ and is closed under the rules and axioms of propositional logic by $(\mathsf{Prop}(E), \vdash_{\mathsf{Prop}(E)})$. Furthermore, let

$$\begin{aligned}
\mathsf{Up}(E, \vdash) &= (\mathsf{Up}(E), \{[\lambda]\phi \vdash [\lambda]\psi \mid \lambda \in \Lambda, \phi \vdash_E \psi\} \\
\mathsf{Ax}(E, \vdash) &= (\mathsf{Lift}(E), \{\phi[\vec{\alpha}/\vec{x}] \vdash \psi[\vec{\alpha}/\vec{x}] \mid \phi(\vec{x}) \vdash \psi(\vec{x}) \in \mathsf{Ax}, \vec{\alpha} \in \vec{E}\} \\
\mathsf{Lift}(E, \vdash) &= \mathsf{Prop}(\mathsf{Ax}(E, \vdash_E) \cup \mathsf{Up}(E, \vdash_E))
\end{aligned}$$

where $(E, \vdash_E) \cup (F, \vdash_F) = (E \cup F, \vdash_E \cup \vdash_F)$. By abuse of notation, we write $(\mathsf{Lift}(E), \vdash_{\mathsf{Lift}(E)})$ for $\mathsf{Lift}(E, \vdash)$.

Finally, we define $(\mathcal{L}^0, \vdash_0) = \mathsf{Prop}(\emptyset, \emptyset)$ and $(\mathcal{L}^{n+1}, \vdash_{n+1}) = \mathsf{Lift}(\mathcal{L}^n, \vdash_n)$.

We proceed as in the previous section and show that Lift allows us to construct the entailment relation $\vdash \subseteq \mathcal{L} \times \mathcal{L}$ of coalgebraic modal logic. Note that Lift "lifts" an (entailment) relation $\vdash_E \subseteq E \times E$ to the set $\mathsf{Lift}(E)$ of formulas containing one more modality than the formulas in E.

As with the n-step denotations, there is a close relationship between formulas $\phi \in \mathcal{L}^n$ of rank $\leq n$ and the entailment relation $\vdash_n \subseteq \mathcal{L}^n \times \mathcal{L}^n$, providing a syntactic counterpart to Theorem 5.1.7:

Lemma 5.2.4. *Suppose $\phi, \psi \in \mathcal{L}^n$. Then $\phi \vdash \psi$ iff $\phi \vdash_n \psi$.*

Proof. We use induction on n. For $n = 0$, the claim holds trivially. For $n > 0$, one uses induction on the derivation of $\phi \vdash_n \psi$. If $\phi \vdash_n \psi$ was derived by propositional reasoning, the claim follows, since $\vdash_n$ is closed under propositional reasoning. If (op) was used, we have that $\phi = [\lambda]\phi_0$, $\psi = [\lambda]\psi_0$ for $\phi_0, \psi_0 \in \mathcal{L}^{n-1}$ (otherwise $\phi, \psi \notin \mathcal{L}^n$) with $\phi_0 \vdash_{n-1} \psi_0$ and hence $\phi \vdash_n \psi$ by induction hypothesis. If $\phi \vdash \psi$ is a substitution instance of an axiom, we have $\phi \vdash_n \psi$ since $\vdash_n$ is closed under substitution instances of axioms. Note that, if $\phi = \sigma(\phi_0, \ldots, \phi_k)$ with σ the right (or left) side of a substitution instance of an axiom, we can always assume that every $\phi_j \in \mathcal{L}^{n-1}$ if $\phi \in \mathcal{L}^n$ by definition of axioms. $\square$

So far, the only restriction on axiom schemes was their syntactic form, which allowed us to construct the entailment relation of coalgebraic modal logic as union of all n-step consequence relations. This does not exclude axiom schemes which are not sound: for example, $\mathsf{tt} \vdash \mathsf{ff}$ (where tt denotes logical truth) qualifies as axiom scheme. We now restrict ourselves to admissible axioms, which guarantees soundness of coalgebraic modal logic.

Definition 5.2.5. We call an axiom scheme $\phi \vdash \psi \in \mathsf{Ax}$ *admissible*, if $\mathsf{Lift}(d)(\phi) \subseteq \mathsf{Lift}(d)(\psi)$ for all functions $d : \mathfrak{X} \to \mathcal{P}(C)$.

That is, admissibility of axiom schemes means that interpreting every variable $x \in \mathfrak{X}$ as a subset of some set C, the (interpretation of the) left side is a subset of (the interpretation of) the right hand side.

Example 5.2.6. Consider the signature functor $TX = \mathcal{P}(X) \times \mathcal{P}(A)$ from Section 1.1.5 along with the set $\Lambda = \{\lambda\} \cup \{\lambda_a \mid a \in A\}$ of predicate liftings from Example 4.2.2. The axioms

$$\mathsf{tt} \vdash [\lambda]\mathsf{tt} \qquad [\lambda]\phi \wedge [\lambda]\psi \vdash [\lambda](\phi \wedge \psi) \qquad [\lambda_a]\phi \vdash [\lambda_a]\psi$$

where a ranges over the elements of A, are admissible. Note that the last axiom expresses that the liftings λ_a are constant.

Assuming admissibility of axiom schemes, soundness of coalgebraic modal logic is immediate: We argue that $\phi \vdash_n \psi$ implies that $\phi \models_n \psi$ and use Theorem 5.1.7 and Lemma 5.2.4. Since all $\vdash_n$'s are preorders and we construct $\vdash_{n+1}$ from $\vdash_n$ by applying Lift, it is handy to consider the lifting of arbitrary preorders first.

Theorem 5.2.7 (Soundness). *Suppose Ax is a set of admissible axioms.*

(i) If $(E, \vdash_E)$ is a preorder and $d : E \to \mathcal{P}(C)$ preserves order, then so does $\mathsf{Lift}(d) : \mathsf{Lift}(E) \to \mathcal{P}(TC)$.

(ii) For all $\phi, \psi \in \mathcal{L}$, we have $\phi \models \psi$ whenever $\phi \vdash \psi$.

Proof. (*i*) Suppose $(E, \vdash)$ is a preorder and $d : E \to \mathcal{P}(C)$ preserves order. It follows by induction on the judgement $\phi \vdash_{\mathsf{Lift}(E)} \psi$ that $\mathsf{Lift}(d)$ preserves order: The case of axioms holds by assumption, propositional entailment is sound, and applications of the rule (op) are sound since predicate liftings preserve order.

(*ii*) We first show that $\phi \vdash_n \psi \implies \phi \models_n \psi$ for all $n \in \omega$ and all $\phi, \psi \in \mathcal{L}^n$. For $n = 0$, the claim follows from the soundness of propositional reasoning (recall that $\mathcal{L}^0$ is the set of propositional formulas over the empty set of atoms). Now suppose that $\phi, \psi \in \mathcal{L}^{n+1}$ with $\phi \vdash_{n+1} \psi$. By (*i*), we have that $d^{n+1} = \mathsf{Lift}(d^n)$ preserves order, and the claim follows.

Now suppose that $\phi \vdash \psi$. Since $\mathcal{L} = \bigcup_{n \in \omega} \mathcal{L}^n$, there exists $n \in \omega$ such that $\phi, \psi \in \mathcal{L}^n$. By Lemma 5.2.4 we have $\phi \vdash_n \psi$ and by the above we conclude that $\phi \models_n \psi$. Soundness follows by Theorem 5.1.7.

□

Hence the axioms schemes presented in Example 5.2.6 only allow to derive valid judgements. In category-theoretic terms, admissibility allows us to consider Lift as an endofunctor on the category of preorders (and order-preserving functions).

Having dealt with soundness, we now investigate conditions, under which we also obtain a completeness theorem. The line of reasoning is the same as in the proof of soundness: We isolate a property (reflexivity) of a set of axiom schemes, show that – for reflexive sets of axiom schemes – $\mathsf{Lift}(d)$ is order-reflecting whenever d is, and conclude that $\phi \vdash_n \psi$, whenever $\phi \models_n \psi$. We start with the definition of reflexivity:

Definition 5.2.8. We say that Ax is *reflexive*, if $\bigwedge \Phi \vdash_{\mathsf{Lift}(E)} \bigvee \Psi$ whenever $(E, \vdash_E)$ is a preorder reflected by a map $d : E \to \mathcal{P}(C)$ and $\Phi, \Psi \subseteq \mathsf{Up}(E)$ are finite with $\mathsf{Lift}(d)(\bigwedge \Phi) \subseteq \mathsf{Lift}(d)(\bigvee \Psi)$.

Reflexivity of Ax clearly holds in all cases where $\mathsf{Lift}(d)$ is order-reflecting whenever d reflects order. As we shall see later, reflexivity is actually equivalent to the fact that $\mathsf{Lift}(d)$ reflects order whenever d does. However, reflexivity is much easier to check, since it does not involve closure under the operations of propositional logic. We illustrate the concept of reflexivity by showing that the axiom schemes presented in Example 5.2.6 are also reflexive.

Example 5.2.9. The set of axiom schemes in Example 5.2.6

$$\mathsf{tt}\vdash[\lambda]\mathsf{tt} \qquad [\lambda]\phi\wedge[\lambda]\psi\vdash[\lambda](\phi\wedge\psi) \qquad [\lambda_a]\phi\vdash[\lambda_a]\psi$$

is reflexive. To see this, we show that

$$\mathsf{Lift}(d)(\bigwedge\Phi)\subseteq\mathsf{Lift}(d)(\bigvee\Psi)\implies\bigwedge\Phi\vdash_{\mathsf{Lift}(E)}\bigvee\Psi$$

for all finite $\Phi,\Psi\subseteq\mathsf{Up}(E)$, whenever $(E,\vdash_E)$ is a preorder, which is reflected by $d:E\to\mathcal{P}(C)$. So suppose

$$\bigwedge\Phi=\bigwedge_{i\in I}[\lambda]\phi_i\wedge\bigwedge_{j\in J}[\lambda_{a_j}]\phi_j \qquad \text{and}$$
$$\bigvee\Psi=\bigvee_{k\in K}[\lambda]\psi_k\vee\bigvee_{l\in L}[\lambda_{a_l}]\psi_l,$$

where $\phi_i,\phi_j,\psi_k,\psi_l\in E$ and the a_j, $a_k\in A$. Consider $(\mathfrak{c}_0,\mathfrak{a}_0)=(\bigcap_{i\in I}d(\phi_i),\{a_j\mid j\in J\})\in\mathsf{Lift}(d)(\Phi)$. We have $(\mathfrak{c}_0,\mathfrak{a}_0)\in\mathsf{Lift}(d)(\bigvee\Psi)$ and consider two cases.

Case 1: $\exists k\in K.\mathfrak{c}_0\subseteq d(\psi_k)$. Hence $\bigwedge_{i\in I}\psi_i\vdash_E\psi_k$. Using (op) and distributivity of conjunctions over $[\lambda]$, we obtain $\bigwedge_{i\in I}[\lambda]\psi_i\vdash_{\mathsf{Lift}(E)}[\lambda]\psi_k$, and $\bigwedge\Phi\vdash_{\mathsf{Lift}(E)}\bigvee\Psi$ by propositional entailment.

Case 2: $\exists l\in L.a_l\in\mathfrak{a}_0$. By the definition of $\mathfrak{a}_0$, there exists $j\in J$ such that $a_j=a_l$. By axiom $[\lambda_a]\sigma\vdash[\lambda_a]\tau$, we obtain $[\lambda_{a_j}]\phi_j\vdash_{\mathsf{Lift}(E)}[\lambda_{a_l}]\psi_l$ and $\bigwedge\Phi\vdash_{\mathsf{Lift}(E)}\bigvee\Psi$ follows by propositional entailment.

We now show that reflexivity is in fact sufficient to obtain a complete axiomatisation of local consequence.

Theorem 5.2.10 (Completeness). *Suppose* Ax *is a reflexive set of axioms.*

(i) If $(E,\vdash_E)$ is a preorder and $d:E\to\mathcal{P}(C)$ reflects order, then so does $\mathsf{Lift}(d):\mathsf{Lift}(E)\to\mathcal{P}(TC)$.

(ii) For all $\phi,\psi\in\mathcal{L}$, we have $\phi\vdash\psi$ whenever $\phi\models\psi$.

Proof. (i) By passing from arbitrary formulas to their conjunctive and disjunctive normal form. Negative literals are treated using the equivalences $\phi\wedge\neg\psi\vdash\rho$ iff $\phi\vdash\psi\vee\rho$ and $\rho\vdash\phi\vee\neg\psi$ iff $\rho\wedge\psi\vdash\phi$.

(ii) Using (i), we obtain that $\phi\vdash_n\psi$ whenever $\phi\models_n\psi$, for all $n\in\omega$. The claim follows as in the proof of Theorem 5.2.7.

□

This shows that reflexivity is actually equivalent to the fact that $\mathsf{Lift}(d)$ reflects order whenever d does. Turning back to Example 5.2.9, the preceding theorem shows that the axiom schemes given in the example constitute a complete axiomatisation of local consequence in the case of Kripke models. In essence, the completeness proof was done by induction on the rank of formulas. In particular, no canonical model construction has been used.

5.3 Finite Models and Decidability

This section shows how to use the tools developed in the previous sections to show that – under additional assumptions on the endofunctor under consideration – coalgebraic modal logic has the finite model property, and local consequence is decidable. We proceed as in (standard) modal logic and first establish the finite model property. Decidability then follows from the finite model property, if we can show, that finite models can be effectively constructed. For the whole section, we fix a set Ax of axiom schemes which we assume to be admissible and reflexive (in which case coalgebraic modal logic is sound and complete, cf. Theorem 5.2.7 and 5.2.10). The finite model property states that every formula ϕ, which is satisfiable (i.e. there exists $(C, \gamma) \in \mathsf{CoAlg}(T)$ with $[\![\phi]\!]_\gamma \neq \emptyset$), is satisfiable in a *finite* model (that is, for some (C, γ) with finite carrier C, we have $[\![\phi]\!]_\gamma \neq \emptyset$). We show that every formula, which is satisfiable, is satisfiable in a model with carrier $T^n 1$, where n is the rank of the formula. Hence the finite model property follows, if $T^n 1$ is a finite set. This motivates the following:

Definition 5.3.1. We call T *finite*, if TX is finite for finite sets X.

Now consider, as in Lemma 5.1.9, a right inverse $f^0 : T1 \to 1$ of the unique surjection $e^0 : T1 \to 1$ and let $f^n = T^n f$.

Proposition 5.3.2. *Suppose $\phi \in \mathcal{L}^n$ is satisfiable. Then ϕ is satisfiable in $(T^n 1, f^n)$.*

Proof. Since ϕ is satisfiable, there exists a model (C, γ) with $[\![\phi]\!]_\gamma \neq \emptyset$. By Lemma 5.1.8, we have $d^n(\phi) \neq \emptyset$ and $[\![\phi]\!]_{f^n} = f_n^{n-1} \circ d^n(\phi)$. Since $f_n^n = \mathrm{id}_{T^n 1}$ by Lemma 5.1.9, we have $[\![\phi]\!]_{f^n} \neq \emptyset$. □

Assuming that T is finite, we obtain the finite model property for coalgebraic modal logic:

Theorem 5.3.3 (Finite Model Property). *Suppose T is finite. Then every satisfiable formula is satisfiable in a finite model.*

Proof. Suppose $\phi \in \mathcal{L}$ is satisfiable. By the previous proposition (using the same notation), ϕ is satisfiable in $(T^n 1, f^n)$, where $n = \mathrm{rank}(\phi)$. Finiteness of $T^n 1$ follows by induction using finiteness of T. □

For propositional modal logic, we obtain

Example 5.3.4. Let $TX = \mathcal{P}(X) \times \mathcal{P}(A)$ for some *finite* set A of atomic propositions and consider the liftings introduced in Example 4.2.2. We have that T is finite, hence $\mathcal{L}$ has the finite model property. If A were infinite, we could (in this example) still establish the finite model property by arguing that a formula ϕ only contains finitely many liftings λ_a for $a \in A$, and can hence be interpreted over coalgebras for $T'X = \mathcal{P}(X) \times \mathcal{P}(A')$, where $A' = \{a \in A \mid \lambda_a \text{ occurs in } \phi\}$.

From the finite model property, one usually concludes decidability by showing that finite models can be effectively constructed. In the context of coalgebras for arbitrary endofunctors, effectivity has to be explicitely required: We call T *effective*, if TX (for sets X) and Tf (for functions f) can be effectively computed from X and f, respectively. We call a predicate lifting λ *effective*, if $\lambda(X)(\mathfrak{x})$ can be effectively obtained from X and $\mathfrak{x}$.

We leave it to the reader to formulate a precise definition of effectiveness in terms of natural number codings. For effective and finite functors T we obtain decidability of local consequence from the finite model property:

Theorem 5.3.5. *Suppose T is finite and effective and all $\lambda \in \Lambda$ are effective. Then the problem $\phi \vdash \psi$, where ϕ and ψ range over formulas in $\mathcal{L}$, is decidable.*

Proof. Clearly $\phi \not\vdash \psi$ iff $\phi \wedge \neg\psi$ is satisfiable. By Proposition 5.3.2, $\phi \wedge \neg\psi$ is satisfiable in $(T^n 1, f^n)$, where n is the rank of $\phi \wedge \neg\psi$ (which can be efficiently computed from ϕ and ψ). Since T is finite, the set $T^n 1$ is finite, and can be effectively obtained by assumption. Since all $\lambda \in \Lambda$ are effective, we can compute the semantics $[\![\phi \wedge \neg\psi]\!]_{f^n} \subseteq T^n 1$. Since $T^n 1$ is finite, we can decide, whether $[\![\phi \wedge \neg\psi]\!]_{f^n}$ is empty, that is, whether $\phi \wedge \neg\psi$ is satisfiable in $T^n 1$. □

Going back to Example 5.3.4, we thus find that local consequence is decidable for standard modal logic.

5.4 Conclusions and Related Work

To our knowledge, the use of induction along the terminal sequence is a novel approach to soundness, completeness and decidability proofs in modal logic. Although we have instantiated the presented framework only to Kripke models we remark that the framework can be instantiated with arbitrary signature functors, obtaining logics for a large class of state based systems (see [53] for examples).

Several approaches, including those taken in [27, 33, 49] use a canonical model construction in order to obtain results akin to the ones presented. The approach taken there applies to an inductively defined class of signature functors and adapts the canonical model construction accordingly. Our approach is different in that (a) it does not restrict us to an inductively defined class of signature functors and (b) it uses induction rather than canonical models as the main proof principle.

The only other approach to generalising modal logic to coalgebras of arbitrary (not syntactically defined) signature functors the author is aware of, is the paper of Moss [39]. Given an endofunctor, the syntax of his coalgebraic logic is obtained via an initial algebra construction. Consequently, Moss's approach applies to a large class of endofunctors T, but at the expense of an abstract syntax, which in particular lacks the notion of modal operators. Also, the paper of Moss does not contain a complete axiomatisation. Finally, we remark that, although we have just studied the one sorted case, the theory generalises in a straight forward way to multi-sorted modal logic, that is, to coalgebras for endofunctors $T : \mathsf{Set}^n \to \mathsf{Set}^n$.

The material in this section is taken from [43].

5.5 Exercises

Exercise 5.5.1. Give a detailed proof of Theorem 5.2.10.

Exercise 5.5.2. Give a formal proof of the argument used in Example 5.3.4 and show, that local consequence for propositional modal logic is decidable, even if the set of atomic propositions is infinite.

Exercise 5.5.3. Find axioms for the signature functors discussed in Exercise 4.5.3 and Exercise 4.5.4, which make the associated logics sound and complete.

Bibliography

[1] Peter Aczel. *Non-Well-Founded Sets.* Center for the Study of Language and Information, Stanford University, 1988.

[2] J. Adámek. Free algebras and automata realizations in the language of categories. *Comment. Math. Univ. Carolinae*, 15:589–602, 1974.

[3] J. Adámek. From Varieties of Algebras to Covarieties of Coalgebras. In A. Corradini, M. Lenisa, and U. Montanari, editors, *Coalgebraic Methods in Computer Science (CMCS'01)*, volume 44.1 of *Electr. Notes in Theoret. Comp. Sci.*, 2001.

[4] J. Adámek and V. Koubek. On the greatest fixed point of a set functor. *Theor. Comp. Sci.*, 150:57–75, 1995.

[5] S. Awodey and J. Hughes. The coalgebraic dual of Birkhoff's variety theorem. Technical Report CMU-PHIL-109, Carnegie Mellon University, November 2000.

[6] A. Baltag. A Logic for Coalgebraic Simulation. In H. Reichel, editor, *Coalgebraic Methods in Computer Science (CMCS'2000)*, volume 33 of *Electr. Notes in Theoret. Comp. Sci.*, 2000.

[7] M. Barr. Terminal coalgebras in well-fonded set theory. *Theor. Comp. Sci.*, 114:229–315, 1993. Korrigendum in Theor. Comp. Sci., 124:189-192, 1993.

[8] M. Barr and C. Wells. *Category Theory for Computing Science.* Prentice Hall, 1989.

[9] F. Bartels. Generalised Coinduction. In A. Corradini, M. Lenisa, and U. Montanari, editors, *Coalgebraic Methods in Computer Science (CMCS'01)*, volume 44.1 of *Electr. Notes in Theoret. Comp. Sci.*, 2001.

[10] P. Blackburn, M. de Rijke, and Y. Venema. *Modal Logic.* Cambridge University Press, 2001.

[11] F. Borceux. *Handbook of Categorical Algebra.* Cambridge University Press, 1994. 3 Volumes.

[12] A. Carboni, G. Kelly, and R. Wood. A 2-categorical approach to change of base and geometric morphisms I. Technical Report 90-1, Dept. of Pure Mathematics, Univ. of Sydney, 1990.

[13] A. Chagrov and M. Zakharyaschev. *Modal Logic*, volume 35 of *Oxford Logic Guides.* Oxford Science Publications, 1997.

[14] Corina Cîrstea. An algebra-coalgebra framework for system specification. In Horst Reichel, editor, *Coalgebraic Methods in Computer Science (CMCS'00)*, volume 33 of *Electronic Notes in Theoretical Computer Science*, pages 81–112, 2000.

[15] E. Clarke, O. Gumbeg, and D. Peled. *Model Checking.* MIT Press, 1999.

[16] E.M. Clarke and H. Schlingloff. Model checking. In A. Robinson and A. Voronkov, editors, *Handbook of Automated Reasoning*, pages 1367–1522. Elsevier, 2000.

[17] A. Corradini. A Complete Calculus for Equational Deduction in Coalgebraic Specification. In F. Parisi-Presicce, editor, *Recent Trends in Algebraic Development Techniques, WADT 97*, volume 1376 of *Lect. Notes in Comp. Sci.* Springer, 1998.

[18] H. Geuvers. Inductive and coinductive types with iteration and recursion. In B. Nordström, K. Petterson, and G. Plotkin, editors, *Proc. of the 1992 Workshop on Types for Proofs and Programs*, 1992.

[19] R. Goldblatt. *Logics of Time and Computation*, volume 7 of *CSLI Lecture Notes.* Center for the Study of Language and Information, Stanford University, 1992. Second Edition.

[20] R. Goldblatt. *Mathematics of Modality*, volume 43 of *CSLI Lecture Notes.* Center for the Study of Language and Information, Stanford University, 1993.

[21] H. P. Gumm and T. Schröder. Products of coalgebras. *Algebra Universalis*, 13:163–185, 2001.

[22] H. Peter Gumm. Elements of the general theory of coalgebras. LUATCS'99, 1999.

[23] H. Peter Gumm. Equational and implicational classes of coalgebras. *Theoretical Computer Science*, 260:57–69, 2001.

[24] M. Hennessy and R. Milner. On Observing Nondeterminism and Concurrency. In J. W. de Bakker and J. van Leeuwen, editors, *Automata, Languages and Programming, 7th Colloquium*, volume 85 of *Lecture Notes in Computer Science*, pages 299–309. Springer-Verlag, 1980.

[25] R. Motwani J. Hopcroft and J. Ullman. *Introduction to Automata Theory, Languages, and Computation.* Addison Wesley, 2001.

[26] B. Jacobs. Objets and Classes, coalgebraically. In B. Freitag, C.B. Jones, and C. Lengauer, editors, *Object-Orientation with Parallelism and Persistence.* Kluwer, 1996.

[27] B. Jacobs. Many-sorted coalgebraic modal logic: a model-theoretic study. *Theoret. Informatics and Applications*, 35(1):31–59, 2001.

[28] B. Jacobs and C. Hermida. Structural Induction and Coinduction in a Fibrational Setting. *Information and Computation*, 145:107–152, 1998.

[29] B. Jacobs and J. Rutten. A Tutorial on (Co)Algebras and (Co)Induction. *EATCS Bulletin*, 62:222–259, 1997.

[30] P. Johnstone, J. Power, T. Tujishita, H. Watanabe, and J. Worrell. On the structure of categories of coalgebras. *Theoretical Computer Science*, 260:87–117, 2001.

[31] A. Kurz. A Co-Variety-Theorem for Modal Logic. In *Proceedings of Advances in Modal Logic 2, Uppsala, 1998.* Center for the Study of Language and Information, Stanford University, 2000.

[32] A. Kurz. *Logics for Coalgebras and Applications to Computer Science.* PhD thesis, Universität München, April 2000.

[33] A. Kurz. Specifying Coalgebras with Modal Logic. *Theor. Comp. Sci.*, 260(1–2):119–138, 2001.

[34] A. Kurz and D. Pattinson. Notes on Coalgebras, Co-Fibrations and Concurrency. In H. Reichel, editor, *Coalgebraic Methods in Computer Science (CMCS'2000)*, volume 33 of *Electr. Notes in Theoret. Comp. Sci.*, 2000.

[35] Alexander Kurz. *Coalgebras and Modal Logic.* 2001. Course Notes for ESSLLI 2001.

[36] S. MacLane. *Categories for the Working Mathematician.* Springer, 1971.

[37] M. Makkai and R. Parè. *Accessible Categories: The Foundations of Categorical Model Theory.* Number 104 in Contemporary Mathematics. American Mathematical Society, 1989.

[38] R. Milner. *Communication and Concurrency.* International series in computer science. Prentice Hall, 1989.

[39] L. Moss. Coalgebraic Logic. *Annals of Pure and Applied Logic*, 96:277–317, 1999.

[40] T. Mossakowski, M. Roggenbach, and L. Schröder. Cocasl at work - modelling process algebra. In H.-P Gumm, editor, *Coalgebraic Methods in Computer Science (CMCS 2003)*, Electronic Notes in Theoretical Computer Science, pages 81–112, 2003. To appear.

[41] D. Park. Concurrency and Automata on Infinite Sequences. In P. Deussen, editor, *5th GI Conference*, volume 104 of *Lect. Notes in Comp. Sci.* Springer, 1981.

[42] D. Pattinson. Semantical Principles in the Modal Logic of Coalgebras. In H. Reichel and A. Ferreira, editors, *Proc. 18th Symposium on Theoretical Aspects of Computer Science (STACS 2001)*, volume 2010 of *Lect. Notes in Comp. Sci.* Springer, 2001.

[43] D. Pattinson. Coalgebraic Modal Logic: Soundness, Completeness and Decidability. *Theor. Comp. Sci.*, 2003. To appear.

[44] Dirk Pattinson. Expressive logics for coalgebras via terminal sequence induction. Technical report, Institut für Informatik, LMU München, 2002.

[45] Benjamin C. Pierce. *Basic Category Theory for Computer Scientists.* MIT Press, 1991.

[46] J. Power and H. Watanabe. An axiomatics for categories of coalgebras. In B. Jacobs, L. Moss, H. Reichel, and J. Rutten, editors, *Coalgebraic Methods in Computer Science (CMCS'98)*, volume 11 of *Electr. Notes in Theoret. Comp. Sci.*, 1998.

[47] H. Reichel. An Approach to Object Semantics based on Terminal Coalgebras. *Math. Struct. in Comp. Sci.*, 5:129–152, 1995.

[48] M. Rößiger. Coalgebras and Modal Logic. In H. Reichel, editor, *Coalgebraic Methods in Computer Science (CMCS'2000)*, volume 33 of *Electr. Notes in Theoret. Comp. Sci.*, 2000.

[49] M. Rößiger. From Modal Logic to Terminal Coalgebras. *Theor. Comp. Sci.*, 260:209–228, 2001.

[50] J. Rothe, B. Jacobs, and H. Tews. The Coalgebraic Class Specification Language CCSL. *Journal of Universal Computer Science*, 7, 2001.

[51] J. Rutten. Automata and coinduction (an exercise in coalgebra). In D. Sangiorgi and R. de Simone, editors, *Proc. CONCUR 1998*, volume 1466 of *Lect. Notes in Comp. Sci.* Springer, 1998.

[52] J. Rutten. Relators and Metric Bisimulations. In B. Jacobs, L. Moss, H. Reichel, and J. Rutten, editors, *Coalgebraic Methods in Computer Science (CMCS'98)*, volume 11 of *Electr. Notes in Theoret. Comp. Sci.*, 1999.

[53] J. Rutten. Universal Coalgebra: A theory of systems. *Theor. Comp. Sci.*, 249(1):3 – 80, 2000.

[54] Krister Segerberg. An essay in classical modal logic. *Filosofiska Studier 13*, 1971.

[55] T. Sudkamp. *Languages and Machines.* Addison Wesley, 1998.

[56] D. Turi and G. Plotkin. Towards a mathematical operational semantics. In *Proc. 12*$^{\text{th}}$ *LICS Conf.*, pages 280–291. IEEE, Computer Society Press, 1999.

[57] D. Turi and J. Rutten. On the foundations of final coalgebra semantics: non-well-founded sets, partial orders, metric spaces. *Mathematical Structures in Computer Science*, 8(5):481–540, 1998.

[58] T. Uustalu and V. Vene. Primitive (Co)Recursion and Course of Value (Co)Iteration, Categorically. *INFORMATICA (IMI, Lithuania)*, 10:5–26, 1999.

[59] V. Vene and T. Uustalu. Functional Programming with Apomorphisms. In *9th Nordic Workshop on Programming Theory*, 1997.

[60] Wolfgang Wechler. *Universal Algebra for Computer Scientists*, volume 25 of *EATCS Monographs on Theoretical Computer Science.* Springer, 1992.

[61] J. Worrell. Toposes of Coalgebras and Hidden Algebra. In B. Jacobs, , L. Moss, H. Reichel, and J. Rutten, editors, *Coalgebraic Methods in Computer Science (CMCS'98)*, volume 11 of *Electr. Notes in Theoret. Comp. Sci.*, 1998.

[62] J. Worrell. Terminal Sequences for Accessible Endofunctors. In B. Jacobs and J. Rutten, editors, *Coalgebraic Methods in Computer Science (CMCS'99)*, volume 19 of *Electr. Notes in Theoret. Comp. Sci.*, 1999.