Coalgebras and Modal Logic

Alexander Kurz[1]

CWI

P.O. Box 94079, 1090 GB Amsterdam

The Netherlands

e-mail: kurz@cwi.nl, url: `http://www.cwi.nl/~kurz`

[1]These lecture notes were written mainly during my stay as an ERCIM fellow at the Masaryk University, Brno, Czech Republic. I owe special thanks to Luboš Brim, Jiří Rosický, and the Faculty for Informatics.

Preface

These course notes present universal coalgebra as a general theory of systems.

By 'system' we understand some entity running in and communicating with an environment. We also assume that a system has a fixed interface and that the environment can perform only those observations/experiments/communications on the system allowed by the interface. By 'general theory' we understand a theory which allows to investigate in a *uniform* way as many different types of systems as possible. Of course, here is a trade off: the more diverse the types of systems we admit for study, the less results we can expect to obtain in a uniform way. It is one of the aims of this course to show that the notion of coalgebra is general enough to cover many types of systems and is specific enough to allow for quite a number of interesting results.

The term 'universal' coalgebra not only refers to the generality of the theory but also reflects that universal coalgebra dualises (to some extent) the well-established area of universal algebra. To explore this duality is another of the main topics of this course. In particular we will see what can be said about the duality of logics for algebras and coalgebras.

Concerning prerequisites, Chapters 1 and 3 should be easily accessible. Chapter 2 requires a bit of category theory which was introduced in the course in more detail than in the text. The Appendix recalls the necessary definitions but cannot replace an introduction to category theory. Section 2.5 contains additional material illustrating the techniques presented in Chapter 2. Chapter 4 needs Chapters 2.1 and 3, Chapter 5 builds on Chapter 2 (excluding Section 2.5).

The exercises in the text are essential and are not meant to be skipped. They should be easy to solve, often even obvious, but the solution needs ideas which are important in the following. The exercises in the separate sections contain additional material which did not fit in a one week's course. They may be more difficult.

I gave courses based on earlier versions of these notes at the Faculty of Informatics of the Masaryk University in Brno, Czech Republic, and at ESSLLI 2001 in Helsinki, Finland. I would like to thank the participants, for many fruitful and enjoyable discussions and for comments helping to improve these notes. I am also grateful to Dirk Pattinson for discussions on Chapter 4.3. Diagrams were produced with Paul Taylor's macro package.

Amsterdam, October 2001.

Contents

Chapter 1

Systems – An Introduction

The aim of this chapter is to show why and how coalgebras model systems. The emphasis is on a number of familiar examples, leaving a uniform treatment of them to Chapter 2.

Section 1.1 starts with an informal understanding of systems and processes and proposes a possible formalisation. Section 1.2, taking a particular type of systems as example, shows the ingredients of a general theory of systems. Central notions are behavioural equivalence, final system, bisimulation, and coinduction. Section 1.3 shows in detail how very much the same theory can be developed for a different type of systems and Section 1.4 presents even more examples which can be treated along the same lines.

1.1 Systems and Processes

Before we start to look for a mathematical model for systems, we should agree on an informal level on what we understand by 'system'. The following—hopefully—seems reasonable.

1. Systems are *reactive*, that is, unlike algorithms, they are not supposed to terminate and announce a result, but are supposed to run, possibly forever, and to communicate, while running, with their environment.

2. The possible communications between a system and the environment are described in an *interface*. An external observer can observe a system only through the interface.

3. The external observer's view is called the *black box view* of a system. The black box view is given by the complete observable *behaviour* of the system.

In general, given a system, we are interested rather in its behaviour than in the actual system itself. One challenge for a general theory of systems is (1) to allow for a rich class of interfaces (types of systems) and, at the same time, (2) to describe the relation of systems and their behaviours in a uniform way. The aim of this chapter is to give enough examples showing that the theory of coalgebras achieves (1). To see how (2) is solved is postponed until chapter 2. Another challenge would be to find a uniform logic to specify systems. This issue is discussed in Chapters 4 and 5.

In order to describe a system we think of it as a set of states X and a transition-function ξ describing for every state $x \in X$ the effect $\xi(x)$ of taking an *observable transition* in state x. That is, a **system** is a function

$$X \xrightarrow{\xi} \Sigma X$$

where we use the notation ΣX to indicate the set of possible outcomes of taking a transition. Σ is called the type or **signature**, X is called the **carrier** or set of states of the system, and ξ is called the **structure** or transition-function of the system.

When we start to run or observe a system (X, ξ) we assume that it is in a given initial state x_0. A system together with a state is called a **process**.[1] We denote processes by $((X, \xi), x_0)$ or shorter (X, ξ, x_0). If the system is clear from the context we also denote the process simply by its initial state.

Example 1.1.1 (Streams).

1. Consider a system for which 'taking a transition' just means to output an element $a \in A$ of some given set A. Such a system is given by a function

$$X \xrightarrow{\xi} A$$

   where $\Sigma X = A$. This system can only take one transition in its life-time.

2. A system that can output elements of A forever can be described by a function

$$X \xrightarrow{\xi} A \times X$$

   Suppose the system is in some state x_0 and takes a transition yielding $\xi(x_0) = (a, x_1)$. Then a next transition can be taken in x_1, and so on. Such a process (X, ξ, x_0) is called a **stream**.

Exercise 1.1.2. This exercise presents some more simple example of systems (the first two systems below may seem to be too simple to be interesting at all but they are useful to build up more complicated systems). You should try to convince yourself of the following:

1. Imagine a system that can do nothing but stop. Such a system can be modeled by a function

$$X \xrightarrow{\xi} 1,$$

   where $1 = \{*\}$ denotes some one-element set.

[1]The term 'process' is often used to denote an equivalence class of processes up to a notion of behavioural equivalence, see Section 1.2.8.

2. Imagine a system which, like a simple clock or metronome, just take transitions but produces no further output. Such a system can be modeled by a function

$$X \xrightarrow{\xi} X.$$

3. Consider the example of streams above. How can we model streams that are not necessarily infinite but also may terminate? A system that may output forever elements in A but also may stop is a function

$$X \xrightarrow{\xi} A \times X + 1,$$

where $+$ should be understood as exclusive or (formally it denotes disjoint union of sets).

1.2 Ingredients of a Theory of Systems

A theory of systems should describe the relation of systems and their behaviours in terms of a given interface. This section explains how this can be done by means of the example of streams. It is also shown how the notion of behaviour leads to final systems and, therefore, allows definitions and proofs by coinduction.

1.2.1 Interfaces

We said that a signature Σ for systems is an operation mapping a set (of states) to a set ΣX containing the possible effects of an observable transition. We have seen the following examples:

Σ	ΣX	Process
1	1	stop
A	A	outputs $a \in A$ once
Id	X	metronome (running forever)
$A \times -$	$A \times X$	stream over A
$A \times - + 1$	$A \times X + 1$	finite or infinite list over A

In each of this cases we expect from an interface to specify the "observable effect" of a transition. Thinking a bit about the examples we see that Σ itself provides us with an appropriate notion of interface.

1.2.2 The Black Box View of a Process

From the point of view of the environment the states of a system are not observable. For example, specifier and user of a system are not interested in the system itself but only in the complete observable *behaviour* of the system. In the following we explain this notion of behaviour.

Let us reconsider the example of streams

$$X \xrightarrow{\xi} A \times X$$

and think about the appropriate notion of behaviour.

We assume that the system is in a given state $x_0 \in X$. Starting from x_0 the system takes a transition $\xi(x_0) = (a_0, x_1)$ and continues with $\xi(x_1) = (a_1, x_2)$ and so on. That is, the system produces an infinite list

$$(x_0, (a_0, x_1), (a_1, x_2), \dots).$$

Assuming that the states x_i are not observable, the **behaviour of the process** $((X, \xi), x_0)$ is then given by[2]

$$Beh(x_0) \quad = \quad (a_0, a_1, a_2 \dots)$$

where we should in fact write $Beh_{(X,\xi)}(x_0)$ but usually drop the subscript.

1.2.3 The Black Box View of a System

We have seen that we can assign to every state of a system its behaviour. The *behaviour of a system* is simply the set of all these behaviours. A fundamental observation is now that

the behaviour of a system is itself a system.

To explain this, let (X, ξ) be a system and $Beh(X) = \{Beh(x) : x \in X\}$ the set of all behaviours of X. To conceive of $Beh(X)$ as a system we have to exhibit a transition-function $\beta : Beh(X) \to A \times Beh(X)$. β has to map an infinite list $l = (a_0, a_1, a_2, \dots)$ into $A \times Beh(X)$. There is an obvious candidate:

$$\begin{aligned} \beta : Beh(X) &\to \ A \times Beh(X) \\ (a_0, a_1, a_2, \dots) &\mapsto \langle a_0, (a_1, a_2, \dots)\rangle \end{aligned}$$

(We use $(\dots)$ to indicate lists and $\langle \cdot, \cdot \rangle$ to denote tuples of a cartesian product.)

Having seen that the behaviour of a system is a system we can ask what the behaviour of a behaviour is. Following our intuition that the behaviour gives us all we can know of a system, we expect the behaviour of a behaviour to be the behaviour itself. This is made precise in the following

[2] In case you need a precise definition (for example to solve the exercises) go ahead to Definition 1.2.4.

Exercise 1.2.1. Since $(Beh(X), \beta)$ is a system, we can consider for each $l \in Beh(X)$ the behaviour of l. Convince yourself that

1. the behaviour of some $l \in Beh(X)$ is l

and conclude that

2. the behaviour of the system $(Beh(X), \beta)$ is $(Beh(X), \beta)$.

As a consequence of the previous exercise, we know that x and $Beh(x)$ have the same behaviour. The next section shows that this is due to Beh being a *morphism of systems*.

1.2.4 Morphisms of Systems

In a general theory of systems we are interested not so much in particular systems but more in the relationships between different systems or in structural properties of collections of systems. The main tool to investigate the relationships between systems are structure preserving mappings between systems.

First, let us introduce a notation which is convenient to work with streams.

Notation 1.2.2. Given a stream (X, ξ, x) with $\xi(x) = (a, x')$ we write $head(x)$ for the first value a and $tail(x)$ for the remainder x'. As usual we dropped the subscripts of $head_{(X,\xi)}(x)$ and $tail_{(X,\xi)}(x)$.

Using this notation, we define the notion of a morphism for streams.

Definition 1.2.3. Let $X \xrightarrow{\xi} A \times X$ and $X' \xrightarrow{\xi'} A \times X'$ be two systems. A homomorphism, or morphism for short, is a function $f : X \to X'$ such that

$$head(f(x)) = head(x) \tag{1.1}$$

$$tail(f(x)) = f(tail(x)) \tag{1.2}$$

(In these equations, the occurrences of *head* and *tail* on the left-hand side refer to (X', ξ'), the occurrences on the right-hand side to (X, ξ).)

Another use of the notation above is that we can now give a precise definition of behaviour.

Definition 1.2.4 (Behaviour of streams). Given a system $X \to A \times X$ and $x_0 \in X$ define $Beh(x_0) = (head(tail^n(x_0)))_{n \in \mathbb{N}}$ where $tail^n$ is defined inductively via $tail^0(x) = x$, $tail^{n+1}(x) = tail(tail^n(x))$.

The following exercises are essential. First show that behaviours are invariant under morphisms.

Exercise 1.2.5 (Behaviours are invariant under morphisms).
Given a (stream)morphism $f : (X, \xi) \to (Y, \eta)$, show that the behaviour of $x \in X$ equals the behaviour of $f(x) \in Y$.

In particular, two states that can be identified by a morphism have the same behaviour. To show the converse, namely that any two states that have the same behaviour can be identified by some morphism, we show that $Beh : X \to Beh(X)$ is a morphism.

Exercise 1.2.6 ($Beh : X \to Beh(X)$ is a morphism). Let (X, ξ) be a system (of streams) and $(Beh(X), \beta)$ its behaviour.

1. Show that the mapping $Beh : X \to Beh(X)$ is a morphism $(X, \xi) \to (Beh(X), \beta)$.
2. Show that $Beh : X \to Beh(X)$ is moreover the unique such morphism (Hint: Use induction to reason about lists $l = ((a_i)_{i \in \mathbb{N}})$).

As a corollary to the three exercises we obtain the fundamental relationship between behaviours and morphisms:

Observation 1.2.7. Two states have the same behaviour iff these states are identified by some morphisms.

1.2.5 The Black Box View of the Class of all Systems

Much of the power of a general theory of systems comes from the observation that

all behaviours of all systems constitute themselves a system.

We explain this in the case of streams again.

Since for any process (X, ξ, x) its behaviour is an infinite list $(a_i)_{i \in \mathbb{N}}$, the set of all behaviours of all processes is $A^{\mathbb{N}} = \{f : \mathbb{N} \to A\} = \{(a_i)_{i \in \mathbb{N}}, a_i \in A\}$. Now, in the same way as the behaviour of a system, we can equip the set of all behaviours of all systems with a transition structure that makes it into a system:

$$\begin{aligned} \zeta : A^{\mathbb{N}} &\to \ A \times A^{\mathbb{N}} \\ (a_0, a_1, a_2, \dots) &\mapsto \langle a_0, (a_1, a_2, \dots) \rangle \end{aligned} \tag{1.3}$$

This concept of a *system of all behaviours* is important for the following reason: Intuitively, all we can know from a black box point of view about systems must be contained in this system of all behaviours. We therefore expect it to play a central role in the theory of systems and in fact it does. For the moment, we will content ourselves to characterise this system of all behaviours in a simple but most useful way:

Since we know from Exercise 1.2.6 that the mapping from a system to its behaviour is a morphism, we know that

- for any system there must be a morphism into the system of all behaviours (namely the one mapping each process to its behaviour).

Moreover, since morphisms preserve behaviours,

- for any system there can be at most one morphism into the system of all behaviours.

This argument shows that the system of all behaviours is a final system:

Definition 1.2.8 (Final system). A system (Z, ζ) is called *final* (or *terminal*) iff for all systems (X, ξ) there is a unique morphism $(X, \xi) \to (Z, \zeta)$.

That the system of all behaviours is characterised by finality is shown by

Proposition 1.2.9. *Any two final systems are isomorphic.*

Proof. Let (Z, ζ) and (Z', ζ') be two final systems. The existence part of finality gives us two morphisms $f : Z' \to Z$ and $f' : Z \to Z'$ and the uniqueness part shows that $f \circ f' = \mathrm{id}_Z$ and $f' \circ f = \mathrm{id}_{Z'}$, that is, f and f' are isomorphisms. $\square$

In the following exercise you are asked to make precise, in the case of streams, the above argument that the system of all behaviours is the final system.

Exercise 1.2.10. Show that $(A^{\mathbb{N}}, \zeta)$ as given by (1.3) is final.

1.2.6 Behavioural Equivalence

Notions of observational or behavioural equivalence (like e.g. bisimulation) play a central role in the theory of processes or state based dynamic systems. Once we have a notion of behaviour there is an obvious definition of behavioural equivalence:

Two processes/systems are behaviourally equivalent iff they have the same behaviour.

This can be made precise using the notion of the final system.

Definition 1.2.11. Let (X, ξ), (X', ξ') be two systems and *Beh*, *Beh'* the two corresponding unique morphisms into the final system.

1. Two processes (X, ξ, x), (X', ξ', x') are behaviourally equivalent iff $\mathit{Beh}(x) = \mathit{Beh}'(x')$.

2. Two systems (X, ξ), (X', ξ') are behaviourally equivalent iff $\mathit{Beh}(X) = \mathit{Beh}'(X')$.

This definition has the advantage that it agrees with our understanding of the final system as the system of all the behaviours. Recalling, however, that two processes are behaviourally equivalent iff they can be identified by a morphism (Observation 1.2.7), there is another obvious definition of behavioural equivalence.

Definition 1.2.12. Let (X, ξ), (X', ξ') be two systems.

1. Two processes (X, ξ, x), (X', ξ', x') are behaviourally equivalent iff there are morphisms

$$\begin{tikzcd}
(X,\xi) \arrow[dr, "f"'] & & (X',\xi') \arrow[dl, "f'"] \\
& \bullet &
\end{tikzcd}$$

such that $f(x) = f'(x')$.

2. Two systems (X, ξ), (X', ξ') are behaviourally equivalent iff there are surjective morphisms

$$\begin{tikzcd}
(X,\xi) \arrow[dr, two heads, "f"'] & & (X',\xi') \arrow[dl, two heads, "f'"] \\
& \bullet &
\end{tikzcd}$$

This definition is more elementary in the sense that it only relies on the notion of morphism and not on the existence of final systems.

Exercise 1.2.13. Show that both definitions are equivalent if the final system exists. [Hint: Use that (1) the disjoint union of two systems is a system and that (2) every morphism $g : (X, \xi) \to (Z, \zeta)$ 'factors through its image', ie that there is a surjective morphism e and an injective morphism m such that $g = (X, \xi) \overset{e}{\twoheadrightarrow} \mathrm{Im}(g) \overset{m}{\hookrightarrow} (Z, \zeta)$.]

1.2.7 Bisimulation

In the previous section, we have seen two definitions of behavioural equivalence. This section shows that behavioural equivalence agrees with what is known as bisimulation.

Definition 1.2.14. Let (X, ξ), (X', ξ') be two systems of streams and $R \subset X \times X'$. Then R is a **bisimulation** iff

$$\begin{aligned}
x \; R \; x' \;&\Rightarrow\; \mathit{head}(x) = \mathit{head}(x') \\
x \; R \; x' \;&\Rightarrow\; \mathit{tail}(x) \; R \; \mathit{tail}(x')
\end{aligned}$$

Two processes (X, ξ, x), (X', ξ', x') are **bisimilar** iff there is a bisimulation R such that $x \; R \; x'$.

Such a relation is called a bisimulation since $x \; R \; x'$ implies that a transition $x \mapsto \langle \mathit{head}(x), \mathit{tail}(x) \rangle$ can be simulated by a transition $x' \mapsto \langle \mathit{head}(x'), \mathit{tail}(x') \rangle$ and vice versa.

In order to prove that two processes are bisimilar we first have to choose an appropriate relation R (which is usually not difficult) and then to check that it is indeed a bisimulation (which is not difficult if we made a good choice of R in the first place). A first example for this strategy is given by " $\Rightarrow$ " of the proof below, we will see more examples in sections 1.2.8 and 1.3.3.

Proposition 1.2.15. *Two processes are behaviourally equivalent iff they are bisimilar.*

Proof. Let (X, ξ) and (X', ξ') be two systems of streams. Recall that $x \in X$ and $x' \in X'$ are behaviourally equivalent iff $head(tail^n(x)) = head(tail^n(x'))$ for all in $n \in \mathbb{N}$ (see Definition 1.2.4).
" $\Rightarrow$ ": Let (X, ξ, x), (X', ξ', x') be two behaviourally equivalent processes. Define $R = \{\langle tail^n(x), tail^n(x') \rangle, n \in \mathbb{N}\}$. To show that R is a bisimulation note that $y\ R\ y' \;\Rightarrow\; tail(y)\ R\ tail(y')$ is trivially satisfied by definition of R. Moreover, since x, x' have the same behaviour, ie $head(tail^n(x)) = head(tail^n(x'))$ for all $n \in \mathbb{N}$, it follows $y\ R\ y' \;\Rightarrow\; head(y) = head(y')$.
" $\Leftarrow$ ": Let R be a bisimulation for (X, ξ) and (X', ξ') and let $x\ R\ x'$. We have to show that $head(tail^n(x)) = head(tail^n(x'))$ for all $n \in \mathbb{N}$. But $tail^n(x)\ R\ tail^n(x')$ is easily shown by induction on $n \in \mathbb{N}$. □

Remark 1.2.16. To reason about behavioural equivalence the definitions of the previous section are convenient. But to establish that two given processes are behaviourally equivalent, the standard technique is to exhibit a bisimulation. (The reason is that to check whether a relation is a bisimulation we only need to consider single transitions and not complete behaviours. The inductive reasoning we save is hidden in the above Proposition, part " $\Leftarrow$ ".)

1.2.8 Coinduction

Usually, one is interested in processes only up to behavioural equivalence. It is therefore sensible to consider behavioural equivalence as equality on processes. From this point of view, we can consider the elements of the final system as the processes:

Observation 1.2.17. In the final system, two processes are behaviourally equivalent iff they are equal.

This allows for a substantial simplification: Instead of reasoning about processes up to behavioural equivalence we reason up to equality. For example, instead of defining an operation on processes by defining it on representatives of equivalence classes and then showing that the definition is invariant under the choice of representatives, we can use the principle of **definition by coinduction**. It goes as follows.

Since we know that for any system $X \xrightarrow{\xi} \Sigma X$ there is a *unique morphism* into the final system (Z, ζ), we can define a function $f : X \to Z$ just by giving an appropriate structure ξ:

$$\text{for all } X \xrightarrow{\xi} \Sigma X \text{ there is a unique morphism } (X, \xi) \xrightarrow{f} (Z, \zeta)$$

We say that a function $f : X \to Z$ is defined by coinduction if it arises in such a way from a $\xi : X \to \Sigma X$.

For example, let us define the operation merging two streams. That is, we are looking for a function

$$merge : A^{\mathbb{N}} \times A^{\mathbb{N}} \to A^{\mathbb{N}}$$

such that

$$\begin{align} head(merge(l_1, l_2)) &= head(l_1) \tag{1.4} \\ tail(merge(l_1, l_2)) &= merge(l_2, tail(l_1)) \tag{1.5} \end{align}$$

This looks more circular than like a definition, but defining (note that we let X above to be $A^{\mathbb{N}} \times A^{\mathbb{N}}$ now)

$$\begin{aligned} \xi : A^{\mathbb{N}} \times A^{\mathbb{N}} &\to A \times A^{\mathbb{N}} \times A^{\mathbb{N}} \\ \langle l_1, l_2 \rangle &\mapsto \langle head(l_1), \langle l_2, tail(l_1) \rangle \rangle, \end{aligned}$$

it is not difficult to see that *merge* is defined by coinduction:

Exercise 1.2.18. Let *merge* be an arbitrary function $A^{\mathbb{N}} \times A^{\mathbb{N}} \to A^{\mathbb{N}}$. Show that *merge* is a morphism $(A^{\mathbb{N}} \times A^{\mathbb{N}}, \xi) \to (A^{\mathbb{N}}, \zeta)$ iff it satisfies 1.4 and 1.5. [Hint: Show that 1.4 and 1.5 are instances of 1.1 and 1.2 of Definition 1.2.3.]

It follows that there is a unique function $merge : A^{\mathbb{N}} \times A^{\mathbb{N}} \to A^{\mathbb{N}}$ satisfying 1.4 and 1.5, that is, 1.4 and 1.5 are a valid definition. (Existence follows since there is a morphism $(A^{\mathbb{N}} \times A^{\mathbb{N}}, \xi) \to (A^{\mathbb{N}}, \zeta)$ and, moreover, this morphism satisfies 1.4 and 1.5. Uniqueness follows since every function satisfying 1.4 and 1.5 is a morphism and morphisms into the final system are unique.)

For another example do the following

Exercise 1.2.19. Find a function $\xi : A^{\mathbb{N}} \to A \times A^{\mathbb{N}}$ showing that

$$\begin{align} head(even(l)) &= head(l) \tag{1.6} \\ tail(even(l)) &= even(tail(tail(l))) \tag{1.7} \end{align}$$

is a coinductive definition.

Observation 1.2.17 stated that in the final system two processes are behaviourally equivalent iff they are equal. This statement is also called the **coinduction proof principle**: In order to show that two elements of a final system are equal it is enough to show that they are behaviourally equivalent or, in view of Proposition 1.2.15, that they are bisimilar.

For an example of an application of the coinduction proof principle, recall the functions *merge* and *even* and define $odd(x) = even(tail(x))$. We now want to show

$$merge(even(x), odd(x)) = x$$

It is not difficult to guess a bisimulation

$$R = \{\langle merge(even(x), odd(x)) \, , \, x \rangle \, , \ x \in A^{\mathbb{N}}\}.$$

We just have to check the two clauses of Definition 1.2.14. For the first calculate

$$\begin{aligned} head(merge(even(x), odd(x))) &= head(even(x)) \\ &= head(x) \end{aligned}$$

and for the second

$$\begin{aligned} tail(merge(even(x), odd(x))) &= merge(odd(x), tail(even(x))) \\ &= merge(odd(x), even(tail(tail(x)))) \\ &= merge(even(tail(x)), odd(tail(x))), \end{aligned}$$

which is related to $tail(x)$ by definition of R.

1.2.9 Summary and Exercise

We started with the idea that the black box view of a system $X \to \Sigma X$ is obtained by not allowing to observe the states of the system. This idea lead us—for the signature $\Sigma = (A \times -)$—to the following observations:

- We can assign to each process its behaviour.
- Two processes have the same behaviour iff the processes can be identified by some morphisms. (In particular: Behaviours are invariant under morphisms.)
- There is a final system (Z, ζ) and
  - (Z, ζ) contains all behaviours of all systems;
  - the unique morphism $(X, \xi) \to (Z, \zeta)$ assigns to each $x \in X$ its behaviour.
- The final system gives rise to the definition and proof principle of coinduction.

In case you want to get more familiarity with the notions of behaviour, morphism, and final system you can try the following

Exercise 1.2.20. We have seen the following signatures. (Recall that 1 denotes a one-element set. Also note, that we overloaded the notations 1 and A: Both denote a set, but also the corresponding constant operation mapping any set to 1, respectively A. Id denotes the identity operation.)

Σ	ΣX	Process
1	1	stop
A	A	outputs $a \in A$ once
Id	X	metronome (running forever)
$A \times -$	$A \times X$	stream over A
$A \times - + 1$	$A \times X + 1$	finite or infinite list over A

The example of streams has been discussed in detail. For (some or all of) the other cases do the following:

1. Choose an appropriate notion of morphism for each Σ.
2. What processes then, according to Definition 1.2.12, are behaviourally equivalent?
3. Does this notion of behavioural equivalence agree with what you would expect?
4. Describe the system of all behaviours. Check that it is the final system.

1.3 An Extended Example: Deterministic Automata

We first show how we can deal with inputs and then go through the the theory of systems presented in the previous section by means of the example of deterministic automata. In particular, we show that the languages accepted by deterministic automata constitute the final deterministic automata.

1.3.1 Systems with Input

We have seen systems that can output elements or stop. To model automata we need to be able to deal with input. To begin with, suppose we want to model a system

$$X \times I \to X$$

which only allows to input elements of I. The problem here is that we agreed in the beginning of this chapter to describe systems by functions of the kind

$$X \to \dots$$

and not of the kind[3]

$$\dots \to X.$$

[3]Functions of the kind $\dots \to X$ will appear again in section 2.3 as algebras.

Here, a little well-known trick called currying comes to help: Given

$$f : X \times I \to X,$$

$f(x, -)$ is a function $I \to X$ for each $x \in X$. It follows that $f(-, -)$ is function from X to the functions $I \to X$. In order to express this succinctly, we use the following

Notation 1.3.1. Given sets I, X denote by X^I the set of functions $I \to X$.

By the discussion above, we now write functions

$$X \times I \to X$$

as functions

$$X \to X^I$$

which are in the form we chose to express systems.

1.3.2 Moore and Mealy Automata

Deterministic automata are used in different ways. For example, to define the language of finite words accepted by an automata. In this case we should say that the behaviour of an automaton is its accepted language. This is pursued in the next subsection. Here we are interested in viewing automata as systems possibly running forever. Therefore, similar to the example of streams, we will describe the behaviour of an automata as the tree of all its infinite runs.

Suppose we are given the following data

input alphabet	I
output alphabet	O
set of states	X
initial state	x_0

then a process (X, ξ, x_0) given by

$$X \xrightarrow{\xi} O \times X^I$$

is a deterministic automaton (so-called **Moore automaton**): for each state $x \in X$, $\xi(x) = (o, \delta_x)$ where o is the output in state x and $\delta_x : I \to X$ is the function determining on input from I the next state.

What is the right notion of **morphism** for Moore automata? In order to give a definition, we decompose functions $X \xrightarrow{\xi} O \times X^I$ into two functions

$$\begin{aligned} out &: X \to O \\ next &: X \times I \to X \end{aligned} \tag{1.8}$$

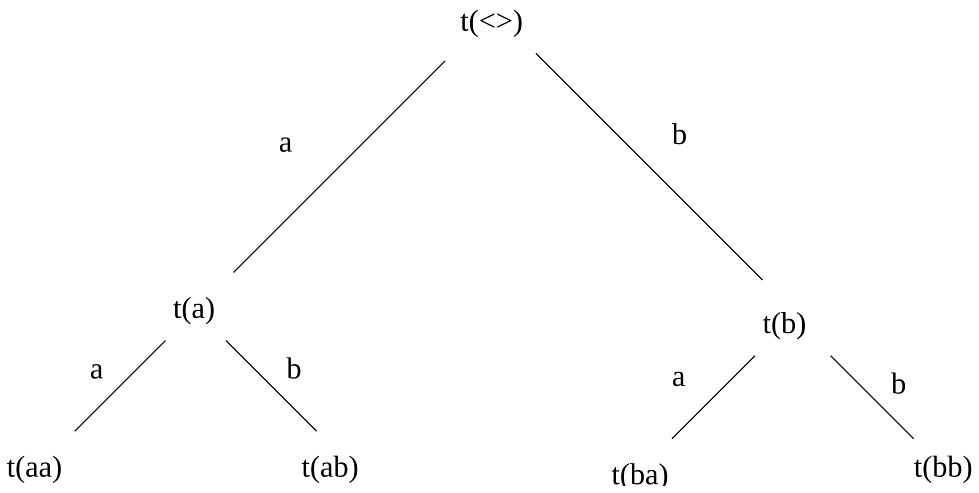


Figure 1.1: Part of a tree $t : \{a, b\}^* \to O$

and then say that $f : X \to X'$ is a morphism $(X, \xi) \to (X', \xi')$ iff

$$\begin{aligned} out'(x) &= out(x) \\ next'(f(x), i) &= f(next(x, i)) \end{aligned}$$

What is the **behaviour** of a Moore automata? It is an infinite tree where branches are sequences of inputs and nodes are labeled with outputs, or, more formally, a function $t : I^* \to O$ (I^* denoting the set of all finite words over I). Figure 1.1 shows part of such a tree in the case that $I = \{a, b\}$.

Exercise 1.3.2.

1. Give a formal definition of $Beh(x_0)$ similar to the one for streams in Definition 1.2.4. [Hint: Define an auxiliary function $next^* : X \times I^* \to X$ extending $next$ from letters to words.]

2. Show that behaviours are invariant under morphisms.

In order to describe the system of all behaviours, we have to equip the set Z of all trees $t : I^* \to O$ with a transition structure. For this note that the subtree of t obtained along an edge i is given by the tree $t' : I^* \to O$

$$t' = \lambda w.t(i \cdot w)$$

where $\cdot$ denotes concatenation of words and the λ-notation is used to indicate the argument of the function. Therefore, the **final system** (Z, ζ) is given by $Z = O^{I^*}$ and

$$\begin{aligned} out(t) &= t(\langle\rangle) \\ next(t) &= \lambda i.\lambda w.t(i \cdot w)) \end{aligned} \tag{1.9}$$

Exercise 1.3.3. Show that (Z, ζ) is indeed the final system.

Before we come to examples of definitions and proofs by coinduction in the next subsection, we first define an appropriate notion of bisimulation. Given (X, ξ), (X', ξ'), we call $R \subset X \times X'$ a **bisimulation** for Moore automata iff

$$\begin{aligned} x \; R \; x' \;&\Rightarrow\; out(x) = out'(x') \\ x \; R \; x' \;&\Rightarrow\; next(x)(i) \; R \; next'(x')(i) \quad \text{for all } i \in I \end{aligned}$$

Exercise 1.3.4. Show that two Moore automata are related by a bisimulation iff they are behaviourally equivalent (compare Proposition 1.2.15).

Exercise 1.3.5 (Mealy automata). In Mealy automata outputs depend not only on the current state but also on the input.

1. Modify the signature of a Moore automaton in such a way that outputs depend on the current state and on the input.
2. Define morphisms of Mealy automata.
3. Describe the behaviour of a Mealy automata [Hint: look for a tree $I^+ \to O$ (I^+ is $I^* - \{\langle\rangle\}$ where $\langle\rangle$ denotes the empty word).]
4. Give a transition function for the system of all behaviours of and show that this system is final.

1.3.3 The Final Automaton of all Languages

Consider a Moore automata with $O = 2$ a two element set:

$$X \xrightarrow{\xi} 2 \times X^I$$

Denoting the elements of 2 by *true* and *false*, we say that $x \in X$ is a final or accepting state iff

$$out(x) = true$$

We have thus obtained the usual notion of a deterministic automata except from the fact that we put no restriction on the set of states X or the set of inputs I to be finite.

How does our notion of behaviour relate to the language accepted by an automaton? We have seen that for $x_0 \in X$

$$Beh(x_0) : I^* \to 2.$$

is a function mapping finite words to *true* or *false*, hence a predicate on words. We can therefore, equivalently, think of $Beh(x_0)$ as the set of words

$$\mathcal{L}(x_0) = \{w \in I^* : Beh(x)(w) = true\}$$

which is nothing but the language accepted by (X, ξ, x_0).

It follows that the set of all languages is the final system of deterministic automata: The transition structure of the final system (1.9) becomes (let $L \subset I^*, i \in I$)

$$\begin{aligned} out(L) &= (\langle\rangle \in L) \\ next(L)(i) &= \{w : i \cdot w \in L\} \end{aligned}$$

Writing $L\downarrow$ for the proposition '$\langle\rangle \in L$' and L_i for $\{w : i \cdot w \in L\}$ the transition structure on the final system of all languages can be written succinctly as

$$\begin{array}{ll} L\downarrow & \\ L_i & \text{for all } i \in I \end{array} \tag{1.10}$$

(1.10) gives us a convenient notation for the use of coinduction. For example we can give **coinductive definitions** of union

$$\begin{array}{rcl} (L+K)\downarrow & \text{iff} & L\downarrow \text{ or } K\downarrow \\ (L+K)_i & = & L_i + K_i \end{array}$$

sequential composition

$$\begin{array}{rcl} (LK)\downarrow & \text{iff} & L\downarrow \text{ and } K\downarrow \\ (LK)_i & = & \begin{cases} L_iK & \text{if not } L\downarrow \\ L_iK + K_i & \text{if } L\downarrow \end{cases} \end{array}$$

and Kleene star

$$\begin{array}{rcl} L^*\downarrow & \text{iff} & true \\ (L^*)_i & = & L_iL^* \end{array}$$

Finally we illustrate the **coinduction proof principle**. First note that R is a bisimulation iff

$$\begin{array}{rcl} L\,R\,K & \Rightarrow & (L\downarrow \Leftrightarrow K\downarrow) \\ L\,R\,K & \Rightarrow & L_i\,R\,K_i \;\; \text{for all } i \in I \end{array}$$

For a first example we want to show

$$\{\langle\rangle\} + LL^* = L^*. \tag{1.11}$$

Immediately from the definitions, we obtain

$$(\{\langle\rangle\} + LL^*)\downarrow \quad \text{iff} \quad L^*\downarrow$$

and (using $L_iL^* + L_iL^* = L_iL^*$ in case $L\downarrow$)

$$(\{\langle\rangle\} + LL^*)_i \quad = \quad (L^*)_i$$

showing that both sides of the equation (1.11) are bisimilar and hence equal.

Exercise 1.3.6. Show that

1. $K + \emptyset = K$,
2. $K + K = K$.

For a second example try to show

$$K(L + M) = KL + KM.$$

You will see that we cannot proceed as in the first example but have to find a bisimulation. Try to find an appropriate bisimulation!

Exercise 1.3.7. Show that $\{(K(L + M) + N, KL + KM + N) : K, M, L, N \subset I^*\}$ is a bisimulation.

1.4 More Examples

You can skip this section or return later to it, but you should think briefly about morphisms for transition systems, see exercise 1.4.3.

1.4.1 Objects and Classes

In object-oriented programming procedures are called methods. Writing X for the state-space of an *object*, the type of a method m is of the form

$$m : X \times I \to E + O \times X,$$

meaning that for each state $x \in X$ and each input $i \in I$, $m(x, i)$—usually written as $x.m(i)$—either raises an exception in E or yields an output in O and a new state in X.

A *class* in object oriented programming is given as a set of methods

$$m_j : X \times I_j \to E_j + O_j \times X \qquad (1 \le j \le n)$$

which can be written as

$$m_j : X \to (E_j + O_j \times X)^{I_j} \qquad (1 \le j \le n).$$

Since functions $f_j : X \to Y_j$, $(1 \le j \le n)$, are nothing else than a single function $\langle f_1, \ldots f_n\rangle : X \to \prod_{1 \le j \le n} Y_j$, we define the *signature of the class* to be

$$\Sigma X = \prod_{1 \le j \le n} (E_j + O_j \times X)^{I_j}$$

and an implementation to be a system

$$X \xrightarrow{\langle m_1, \ldots m_n\rangle} \Sigma X.$$

The main reason why this view of classes as systems is attractive, is that it naturally takes into account that objects are encapsulated: The only way to access an object is via one of the methods. Therefore, there is for each class a notion of behavioural equivalence which expresses that two objects are equivalent iff they cannot be distinguished by applying the methods to them. This notion of behavioural equivalence coincides with the one given by the final system.

Exercise 1.4.1. Assume a class with one method $m : X \times I \to E + O \times X$. Describe the final system. [Hint: Similar to Equations (1.9), consider trees $I^* \to (E + O)$ but be careful proving the uniqueness part.]

1.4.2 Datatypes

Traditionally, datatypes are defined by constructors as *initial algebras*. Then further operations are defined by induction. For an example consider the following specification of stacks over elements in A.

spec STACK

constructors

$new : \ 1 \to \text{stack}$

$push : \text{A} \times \text{stack} \to \text{stack}$

operations

$\langle top, pop\rangle : \text{stack} \to \text{A} \times \text{stack} + 1$

axioms

$\langle top(new), pop(new)\rangle = *$

$\langle top(push(a, s)), pop(push(a, s))\rangle = \langle a, s\rangle$

new gives an empty stack and *push* then allows to construct new stacks from old. That stacks are to be considered elements of the initial algebra given by *new* and *push*—ie the **initial algebra semantics** of stacks—means that

- only data which can be constructed by *new* and *push* is considered to be a stack (the datatype contains *no junk*) and
- whenever two stacks are constructed in two different ways from *new* and *push*, then they are different (the datatype has *no confusion*).

Finally, *pop* and *top* are defined inductively from *new* and *push*.

There is also a different view on datatypes. We can consider stacks as interacting with an environment via *top* and *pop*. Then two stacks are behaviourally equivalent iff they cannot be distinguished by using only *top* and *pop*. Identifying behaviourally equivalent stacks, we can consider stacks as the elements of the *final system* given by *top* and *pop* and define *new* and *push* coinductively:

spec STACK

observers

$\langle top, pop\rangle : \text{stack} \to \text{A} \times \text{stack} + 1$

operations

$new : \ 1 \to \text{stack}$

$push : \text{A} \times \text{stack} \to \text{stack}$

axioms

$\langle top(new), pop(new)\rangle = *$

$\langle top(push(a,s)), pop(push(a,s))\rangle = \langle a, s\rangle$

That stacks are to be considered elements of the final system given by *top* and *pop*—ie the **final coalgebra semantics** for stacks—means that

- two stacks with the same behaviour are considered to be equal and
- any data being observable with *top* and *pop* is considered to be a stack.

Exercise 1.4.2. Think about how the two conditions are related which characterise, respectively, the initial algebra semantics ('no junk', 'no confusion') and the final coalgebra semantics. Can you describe this relationship in a formal way?

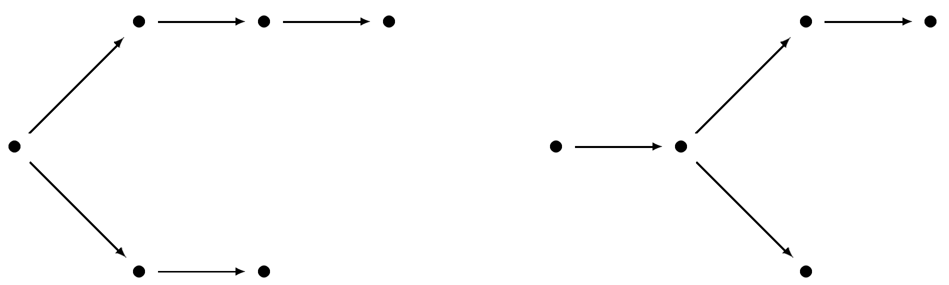
1.4.3 Transition Systems

Of course, all of the previous examples can be understood as special cases of transition systems. But usually transition systems allow for non-determinism, a feature that was not present so far.[4] As it is often the case, non-determinism can be modeled by the use of the powerset operation. Let us denote by $\mathcal{P}X$ the set of subsets of X. Then a system

$$\xi : X \to \mathcal{P}X$$

maps each $x \in X$ to a set $\xi(x) \subset X$. Interpreting $\xi(x)$ as the set of successors of x, we see that (X, ξ) is a transition system.

What are the observations that can be made of such a system? From our discussion of section 1.1 we expect an observer to be able to count transitions. In particular, an observer can detect termination (no transition possible).[5] But it is less clear whether an observer should be able to distinguish between different branching structures as eg in



There are different possibilities here. But is there some kind of canonical choice? In section 1.2.4 we made the observation that two processes are behaviourally equivalent iff they can be identified by some morphisms. This suggests that we could try to find the appropriate notion of behaviour by finding the appropriate notion of morphism. The following exercise shows that, again, there are several possibilities. In particular, not all reasonable notions of morphism lead to a useful notion of behaviour.

Exercise 1.4.3. Given a system $X \xrightarrow{\xi} \mathcal{P}X$, we think of it as a graph (X, R), $R \subset X \times X$ (let $x \; R \; y \;\; \Leftrightarrow \;\; y \in \xi(x)$). Accordingly, it is natural to consider graph morphisms $f : (X, R) \to (X', R')$ as system morphisms. Graph morphisms are functions $f : X \to X'$ such that

$$x \, R \, y \;\Rightarrow\; f(x) \; R \; f(y).$$

1. Using Definition 1.2.12, characterise behaviours of graphs. What is the final graph?

2. What goes wrong? Can you modify the notion of a graph morphism in such a way that Definition 1.2.12 becomes more interesting?

[4]Note that the operator $+$ does not introduce non-determinism: Systems like eg $X \to A \times X + 1$ or $X \to X + X$ are deterministic.

[5]There is a hidden assumption here, namely that distinguishability is symmetric. For more sophisticated notions of observations see eg Vickers [70].

the feature	how to model it	typical system	and process
output	$O \times -$	$X \to O \times X$	stream
input	$(-)^I$	$X \to (O \times X)^I$	deterministic automaton
exceptions, errors	$E + -$	$X \to (E + O \times X)^I$	object (one method)
multiple methods	$- \times -$	$X \stackrel{\langle m_1, m_2 \rangle}{\longrightarrow} \Sigma_1 X \times \Sigma_2 X$	object with two methods of types Σ_1, Σ_2
nondeterminism	$\mathcal{P}$	$X \to \mathcal{P}(Act \times X) \simeq$ $X \to \mathcal{P}(X)^{Act}$	as in Milner's CCS or in process algebra

Table 1.1: How to Model ...

In chapter 2 we will see that the notion of a coalgebra provides us automatically with an appropriate notion of morphism and behaviour for transition systems.

We finish this section by giving some more examples of transition systems. Imagine that we want the system to be able to output information. Then a system is given by a function

$$\langle \xi, v \rangle : X \to \mathcal{P}X \times C$$

where $\xi : X \to \mathcal{P}X$ is a system as before and $v : X \to C$ assigns a $c \in C$ to each state x.

In the previous example, we labeled the states via a function $v : X \to C$. We can also label the transitions with 'actions' $a \in A$

$$\xi : X \to \mathcal{P}(X \times A)$$

Processes (X, ξ, x) for the signature $\Sigma X = \mathcal{P}(X \times A)$ are processes in the sense of Milner's CCS or process algebra.

1.5 Summary of Examples

In the previous examples we have seen how to model certain features that systems may have by choosing the appropriate signature. Table 1.1 gives a summary.

The examples we have seen so far were motivated by research areas where systems can usefully be modeled by coalgebras. Table 1.2 gives an overview and pointers to the literature where more references may be found.

1.6 Exercises and Problem

Exercise 1.6.1 (Contexts). If we think of the environment as performing experiments on processes, the behaviour of a process x should be determined by knowing the outcome

the application area	some literature
automata theory	[27, 61]
(behavioural) differential equations	[63]
control theory	[62]
object oriented programs	[26]
(algebraic) specification	[51, 13, 36]
process algebra	[2, 67]
probabilistic transition systems	[14, 73]
modal logic	see Chapters 4, 5

Table 1.2: Application Areas for Coalgebras

of each experiment performed on x. In the case of streams $\Sigma X = A \times X$ this can be formalised as follows. An experiment is a 'term with a hole' $head(tail^n(-))$, often called a *context*. Performing the experiment on x consists of plugging x into the term and looking for the outcome. The set of possible outcomes is A.

1. Give a transition structure on the set of functions from experiments to outcomes and show that it is the final coalgebra.

2. Do the same for deterministic automata.

Problem 1.6.2. Try to find out for which signatures the final coalgebra can be described as the set of functions from experiments to outcomes (as in the exercise above).

Exercise 1.6.3 (Minimal realisation of automata). In Section 1.3 we have described deterministic automata as processes for the signature $\Sigma = O \times \mathrm{Id}^I$. In automata theory, one is often interested in finding the minimal automata realising a behaviour $t : I^* \to O$. Show that the description of automata as systems trivialises the existence of a minimal realisation: The minimal automaton realising the behaviour $t : I^* \to O$ is just the smallest subsystem of the final system generated by t.

The next exercise shows that for non-deterministic systems there is no essential difference between inputs and outputs (as long as the parameter A is kept fixed).

Exercise 1.6.4. Give a bijection between $\mathcal{P}(A \times X)$ and $(\mathcal{P}X)^A$. [6]

[6] It should also be checked that the bijection is natural in X, see the Appendix for a definition of natural transformations.

1.7 Notes

Although known and studied before, the current interest in coalgebras goes back to Peter Aczel's book [2] "Non-well founded set theory" where he gives a description of the final system for the signature $\mathcal{P}(A\times-)$ and a final semantics for processes in the sense of Milner's CCS. In particular, he recognised that the behavioural equivalence given by the final system is the bisimilarity as known from process theory. Then Aczel and Mendler [1] showed that final coalgebras exist under rather general circumstances. The research inspired by the view of coalgebras as systems was then continued in eg [57, 59, 58, 67]. The idea of universal coalgebra as a general theory of system is due to Rutten [60, 64]. The example of merging streams in Section 1.2.8 is taken from [25]. For a recent study of and further references on coinduction see Bartels [9].

The example of deterministic automata as coalgebras was studied in Rutten [61]. Classes in the sense of object-oriented programming as final coalgebras are due to Reichel [49] and Jacobs [28]. The idea of specifying datatypes only up to behavioural equivalence goes back to Reichel, see eg [48]. For recent work on behavioural specifications based on the duality of algebras and coalgebras see [34, 11].

For supplementary introductions to systems and coalgebras see Gumm [21], Jacobs and Rutten [25], Rutten [64].

Chapter 2

Coalgebra

We first show that by considering signatures as functors, we can deal in a uniform way with all the examples of the previous chapter. Second, we present some general ways to construct new coalgebras from old ones. Third, (equational specifications of) algebras are reviewed. Finally, duality of algebras and coalgebras is discussed.

2.1 Coalgebras

If we look back at the previous chapter, we see that the theory of systems we presented was almost uniform in all signatures. The only thing that had to be invented separately for each new signature was the notion of morphism. It is therefore a natural question to ask whether we can modify the definition of a signature in such a way that it includes in a natural way the right notion of morphism. This is indeed the case: We just have to require the signature to be a *functor*.

The other move we make is from an operation (or functor) on sets to a functor on *arbitrary categories*. The main reason for us to do this is that it is necessary to make precise the *duality* of modal and equational logic in chapter 5.[1]

Moreover, in building a theory it often pays to use only those assumptions which are really needed. Category theory allows us to formulate these assumptions in a succinct way. As a consequence, we obtain more general results, simpler (and reusable) proofs, and new insights in why certain results hold.

Finally, let us mention that the use of categories does not create too many new difficulties: Although it is important to note that notions like coproduct, quotient, embedding, and union of images work in all appropriate categories, one can (and should) think in terms of the corresponding notions familiar from sets.

Definition 2.1.1. Given a category $\mathcal{X}$, called the base category, and a functor $\Sigma : \mathcal{X} \to \mathcal{X}$, a Σ-**coalgebra** (X, ξ) is given by an arrow $\xi : X \to \Sigma X$ in $\mathcal{X}$. A morphism between two

[1] As will be explained later, algebras over Set are dual to coalgebras over $\mathsf{Set}^{\mathrm{op}}$. Hence, in order to use duality, we have to work in a setting allowing not only coalgebras over Set but also over $\mathsf{Set}^{\mathrm{op}}$. The cleanest way to do this is to set up the theory for coalgebras over arbitrary categories $\mathcal{X}$.

coalgebras $f : (X, \xi) \to (X', \xi')$ is an arrow f in $\mathcal{X}$ such that $\xi' \circ f = \Sigma f \circ \xi$:

$$\begin{tikzcd} X \arrow[r, "\xi"] \arrow[d, "f"'] & \Sigma X \arrow[d, "\Sigma f"] \\ X' \arrow[r, "\xi'"'] & \Sigma X' \end{tikzcd}$$

The category of coalgebras and morphisms is denoted by $\mathsf{Coalg}(\Sigma)$.

We will explain in more detail what in means for the signature to be a functor. Assume $\mathcal{X} = \mathsf{Set}$. Then we extend the signatures we have seen so far to functors as follows (let $C \in \mathsf{Set}$ and $f : X \to Y \in \mathsf{Set}$):

Σ	ΣX	Σf
C	C	$\mathrm{id}_C : C \to C$
Id	X	f
$(-)^C$	X^C	$f^C : X^C \to Y^C$ $g \mapsto f \circ g$

As we did earlier, we overloaded notation by denoting with C the set as well as the constant functor mapping any set to C. id_C denotes the identity map on C and X^C is function space.

As we have seen before, from these functors, we can build more interesting ones, using $\times$ and $+$, like eg $\Sigma X = (E + A \times X)^I$. To make this precise, we note that $\times$ and $+$ are functors as well. There action on functions $f_1 : X_1 \to Y_1$, $f_2 : X_2 \to Y_2$ is the following:

$- + -$	$f_1 + f_2 : X_1 + X_2 \to Y_1 + Y_2$ $x \in X_1 \mapsto f_1(x)$ $x \in X_2 \mapsto f_2(x)$
$- \times -$	$f_1 \times f_2 : X_1 \times X_2 \to Y_1 \times Y_2$ $\langle x_1, x_2 \rangle \mapsto \langle f_1(x_1), f_2(x_2) \rangle$

It is perhaps not worth looking at these definitions in detail. There are no reasonable alternatives anyway. But these definitions show that any expression build from constants, identity, exponentiation with a constant, $+$, and $\times$ gives rise to a functor (this is due to the fact that the composition of functors is a functor). To understand how these functors act on functions, do the following

Exercise 2.1.2. Check that for $\Sigma = A \times \mathrm{Id}$ and $\Sigma = O \times \mathrm{Id}^I$ the coalgebra morphisms of Definition 2.1.1 agree with the system morphisms of Definition 1.2.3 and of (1.8) in Chapter 1.3.2, respectively. Describe the morphisms for "classes" with signatures $(E + O \times \mathrm{Id})^I$, see Chapter 1.4.1.

Next, we show that the notion of coalgebra morphism also helps us to find the morphisms of transition systems, ie for signatures involving $\mathcal{P}$. Remember that the first idea that came to our mind in Exercise 1.4.3 did not give rise to a reasonable notion of behaviour. But, as for the signatures discussed above, there is one obvious way to extend $\mathcal{P}$ to functions, namely as direct image:

Σ	ΣX	Σf
$\mathcal{P}$	$\{W : W \subset X\}$	$\mathcal{P}f : \mathcal{P}X \to \mathcal{P}Y$ $W \mapsto f(W) = \{f(x) : x \in W\}$

We next characterise morphisms of $\mathcal{P}$-coalgebras. Recall that we can write a $\mathcal{P}$-coalgebra (X, ξ) as (X, R) with $R \subset X \times X$ and $x\ R\ y \;\Leftrightarrow\; y \in \xi(x)$.

Proposition 2.1.3. *Let (X, R) and (X', R') be two $\mathcal{P}$-coalgebras. A function $f : X \to X'$ is a $\mathcal{P}$-coalgebra morphism iff*

$$x\ R\ y \;\Rightarrow\; f(x)\ R'\ f(y) \tag{2.1}$$

$$f(x)\ R'\ y' \;\Rightarrow\; \exists y \in X\ .\ x\ R\ y\ \&\ f(y) = y' \tag{2.2}$$

Proof. The commuting square defining coalgebra morphisms translates into the condition that for all $x \in X$ it holds $\{y' : f(x)\ R'\ y'\} = \{f(y) : x\ R\ y\}$. "$\supset$" is (2.1) and "$\subset$" is (2.2). $\square$

Note that (2.1) says that f is a graph morphism. It expresses that $(X', R', f(x))$ simulates (X, R, x). (2.2) is the converse stating that (X, R, x) simulates $(X', R', f(x))$. The following is therefore no surprise but should be checked for once nevertheless.

Exercise 2.1.4. Let Σ be $\mathcal{P}$ or $\mathcal{P}(A \times -)$ and (X, ξ), (Y, η) two Σ-coalgebras. Then $x \in X$ and $y \in Y$ are behaviourally equivalent (Definition 1.2.12(1)) iff they are bisimilar in the usual sense of modal logic or process algebra (Definition 3.2.1).

To summarise, we have seen that the signatures discussed in Chapter 1 give rise to functors and that, therefore, the systems of Chapter 1 are coalgebras.[2] In particular, the general theory of systems outlined in Chapter 1 is now uniformly available to all categories of coalgebras over sets.

[2]This reflects a general experience: All interesting signatures seem to give rise to functors. But not every operation can be extended to a functor, see the exercises.

2.2 Basic Constructions on Coalgebras

Not all constructions we want to perform on systems are universal in the sense that they can be made in a uniform way for all signatures. For example, if we want to make systems communicate, we usually need to know the specific signature. On the other hand, some simple but important constructions are universal.

2.2.1 Coproducts

Intuitively, it is clear that we can form the disjoint union of two systems just by forming the disjoint union of the carriers and using each transition-function for each component. This seems so obvious that it might be worth to look at

Exercise 2.2.1. Is it possible to have a disjoint union for algebras? (Take any familiar example like perhaps monoids or groups. Or use stacks with operations *new* and *push* as in Chapter 1.4.2.)

The general construction of a coproduct of coalgebras is as easy as the informal description given above. Let $\Sigma : \mathcal{X} \to \mathcal{X}$ and suppose that for a family $(X_i, \xi_i)_{i \in I}$ of Σ-coalgebras the coproduct $\coprod_I X_i$ of the carriers exists. Then the coproduct $(\coprod_I X_i, \xi)$ of the coalgebras is given by

$$\begin{tikzcd}
\coprod\limits_I X_i \arrow[r, dotted, "\xi"] & \Sigma(\coprod\limits_I X_i) \\
X_i \arrow[u, hook, "in_i"] \arrow[r, "\xi_i"] & \Sigma X_i \arrow[u, "\Sigma in_i"']
\end{tikzcd}$$

where ξ exists due to the universal property of the coproduct in $\mathcal{X}$. That is, the coproduct of coalgebras is given and completely determined by the coproduct of the carriers.

Exercise 2.2.2. Apply the construction above to the signature $\mathcal{P}$.

2.2.2 Quotients and Subcoalgebras

This section deals with quotients and subcoalgebras. We first describe quotients and subcoalgebras for the case $\mathcal{X} = \mathsf{Set}$ and then introduce factorisation systems to deal with the general case.

We say that a coalgebra morphism $e : (X, \xi) \to (X', \xi')$ is a **quotient** iff e is surjective. We also call (X', ξ') a quotient or (homomorphic) image of (X, ξ). We say that $m : (X', \xi') \to (X, \xi)$ is a **subcoalgebra** or **embedding** iff m is injective. We also call (X', ξ') a subcoalgebra of (X, ξ).

The following proposition shows that the transition-structure on a quotient (X', ξ') is completely determined by the structure on (X, ξ).

Proposition 2.2.3. *Let $\Sigma : \mathsf{Set} \to \mathsf{Set}$, $(X, \xi), (X', \xi'), (X', \xi'')$ be Σ-coalgebras and $e : X \to X'$ surjective. If e is a morphism $(X, \xi) \to (X', \xi')$ as well as a morphism $(X, \xi) \to (X', \xi'')$ then $\xi' = \xi''$.*

Proof. Follows from e being epi. □

Similarly, the structure on a subcoalgebra (X', ξ') is completely determined by the structure on (X, ξ).

Proposition 2.2.4. *Let $\Sigma : \mathsf{Set} \to \mathsf{Set}$, $(X, \xi), (X', \xi'), (X', \xi'')$ be Σ-coalgebras and $m : X' \to X$ injective. If m is a morphism $(X', \xi') \to (X, \xi)$ as well as a morphism $(X', \xi'') \to (X, \xi)$ then $\xi' = \xi''$.*

Proof. If X' is not empty, then Σm is injective, hence mono, hence $\xi' = \xi''$. If X' is empty then $\xi' = \xi''$ because the only map with empty domain is the empty map. □

As mentioned already, to establish the duality of modal and equational logic, we cannot restrict our attention to the base category Set. Unfortunately, there is no categorical generalisation of the set-based notions surjective and injective which is appropriate in all settings. But it turns out that for the purpose of this lecture (and in many other circumstances as well) we only need the following properties of quotients and embeddings.

Definition 2.2.5 (Factorisation system). Let $\mathcal{C}$ be a category and E, M be classes of arrows in $\mathcal{C}$. We call arrows in E *quotients*, arrows in M *embeddings*, and (E, M) a *factorisation system* in $\mathcal{C}$ iff

1. $f \in E$ and $f \in M$ implies f iso.
2. E, M are closed under composition.
3. Every arrow f in $\mathcal{C}$ has a factorisation $f = m \circ e$ for some $m \in M$ and $e \in E$. We call m the *image* of f and e the *kernel*[3] of f.
4. Factorisations are given up to unique isomorphism, ie for all $e, e' \in E$ and all $m, m' \in M$ as in the diagram

$$\begin{tikzcd} \bullet \arrow[r, "e"] \arrow[d, "e'"'] & \bullet \arrow[d, "m"] \arrow[dl, dotted, "h"] \\ \bullet \arrow[r, "m'"'] & \bullet \end{tikzcd}$$

there is a unique isomorphism h making the triangles commute. □

[3]The kernel of a function $f : X \to Y$ is usually defined as $\mathrm{Ker}\, f = \{(x_1, x_2) : f(x_1) = f(x_2)\}$. Since $\mathrm{Ker}\, f$ describes the quotient part of the factorisation of f up to unique isomorphism, it seems justified to use 'kernel' for the quotient part of a factorisation in general.

The use of the letters E and M derives from the fact that in most applications of factorisation systems arrows in E are epi and arrows in M are mono. For example in Set is $(Epi, Mono) = (Surj, Inj)$ a factorisation system.

The most important property of factorisation systems is the following:

Proposition 2.2.6 (Unique diagonalisation). *Let (E, M) be a factorisation system in $\mathcal{C}$. For any commuting square*

$$\begin{tikzcd}
\bullet \arrow[r, "e"] \arrow[d, "l"'] & \bullet \arrow[d, "r"] \arrow[dl, dotted, "d"] \\
\bullet \arrow[r, "m"'] & \bullet
\end{tikzcd}$$

with $m \in M$ and $e \in E$, there is a unique diagonal d making the triangles commute.

Proof. (The proof is technical and can be skipped.) Factoring $l = m_l \circ e_l$ and $r = m_r \circ e_r$ gives a unique isomorphism h in the left-hand diagram

$$\begin{tikzcd}
\bullet \arrow[r, "e"] \arrow[d, "e_l"'] & \bullet \arrow[d, "e_r"] \\
\bullet \arrow[d, "m_l"'] & \bullet \arrow[l, "h"] \arrow[d, "m_r"] \\
\bullet \arrow[r, "m"'] & \bullet
\end{tikzcd}
\qquad
\begin{tikzcd}
\bullet \arrow[rr, "e"] \arrow[d, "e_l"'] & & \bullet \arrow[d, "e_r"] \arrow[dl, "e_d"'] \\
\bullet \arrow[d, "m_l"'] & \bullet \arrow[l, "h_2"'] \arrow[dl, "m_d"'] & \bullet \arrow[l, "h_1"] \arrow[d, "m_r"] \\
\bullet \arrow[rr, "m"'] & & \bullet
\end{tikzcd}$$

Then $m_l \circ h \circ e_r$ is a diagonal as required. To show uniqueness consider any diagonal d (ie $d \circ e = l$ and $m \circ d = r$). Factoring $d = m_d \circ e_d$ gives unique isos h_1, h_2 with $e_d = h_1 \circ e_r$ and $m_d = m_l \circ h_2$. Since $h_2 \circ h_1$ must be h, it follows $d = m_l \circ h \circ e_r$. $\square$

The unique diagonalisation property is often useful in order to arrows d. We will need it in Chapter 5. For a first example we show that over Set every coalgebra morphism factors as a surjective and an injective one and that this factorisation is calculated as the factorisation in Set:

Proposition 2.2.7. *Let $\Sigma : \mathsf{Set} \to \mathsf{Set}$, $f : (X, \xi) \to (Y, \eta)$ a Σ-morphism, and $X \xrightarrow{e} X' \xrightarrow{m} Y$ an $(Surj, Inj)$-factorisation of f in Set. Then there is a unique function $\xi' : X' \to \Sigma X'$ making e and m into coalgebra morphisms.*

Proof. Consider the diagram

$$\begin{tikzcd}
X \arrow[r, "e"] \arrow[d, "\xi"'] & X' \arrow[r, "m"] \arrow[d, dotted, "\xi'"'] & Y \arrow[d, "\eta"'] \\
\Sigma X \arrow[r, "\Sigma e"'] & \Sigma X' \arrow[r, "\Sigma m"'] & \Sigma Y
\end{tikzcd}$$

In case that X' is not empty, we know that Σm is injective (Exercise A.0.2) and we can use unique diagonalisation. The case X' empty works as usual (see the proof of Proposition 2.2.4). □

Note that already in this case the argument by unique diagonalisation is easier than a direct proof going back to the element-wise definition of the image X' of f.

As a corollary note that the surjective and injective morphisms form a factorisation system in $\mathsf{Coalg}(\Sigma)$:

Corollary 2.2.8. *Let $\Sigma : \mathsf{Set} \to \mathsf{Set}$ and denote by Surj and Inj the class of surjective and injective Σ-morphisms. $(Surj, Inj)$ is a factorisation system in $\mathsf{Coalg}(\Sigma)$.*

Proof. Existence of factorisations was shown in the proposition. To see that factorisations are unique up to iso let $(X,\xi), (Y_1,\nu_1), (Y_2,\nu_2), (Z,\zeta) \in \mathsf{Coalg}(\Sigma)$ and consider the left-hand diagram

$$\begin{tikzcd}
X \arrow[r, "e"] \arrow[d, "e'"'] & Y_1 \arrow[d, "m"] \arrow[dl, dotted, "h"] \\
Y_2 \arrow[r, "m'"'] & Z
\end{tikzcd}
\qquad
\begin{tikzcd}
\Sigma X \arrow[r, "\Sigma e"] & \Sigma Y_1 \arrow[r, "\Sigma h"] & \Sigma Y_2 \\
X \arrow[u, "\xi"] \arrow[r, "e"'] & Y_1 \arrow[u, "\nu_1"'] \arrow[r, "h"'] & Y_2 \arrow[u, "\nu_2"']
\end{tikzcd}$$

There is a unique iso h. To see that h is a coalgebra morphism we use the following standard argument. Consider the right-hand diagram above. We know that the left square and that the outer rectangle commute (since e and $e' = h \circ e$ are coalgebra morphisms). Since e is surjective (hence epi), the right-hand square also commutes. □

Finally, the following proposition summarises what we have shown about factorisation systems in case the base category is not Set (the proofs remain unchanged).

Proposition 2.2.9. *Let (E, M) be a factorisation system in $\mathcal{X}$ and $\Sigma : \mathcal{X} \to \mathcal{X}$. Assume that arrows in E are epi and that Σ preserves arrows in M.*[4] *Let E', M' be the coalgebra morphisms which are in E, M, respectively. Then (E', M') is a factorisation system in* $\mathsf{Coalg}(\Sigma)$.

[4] Σ *preserves arrows in* M iff $m \in M \Rightarrow \Sigma m \in M$.

2.2.3 Unions

The perhaps most important construction for us is 'union of subcoalgebras'. To explain the interest, suppose we have a system and a property φ on its states and we want to know the largest subcoalgebra whose states all satisfy φ. In case that the union of subcoalgebras exists we can take the union of all subcoalgebras satisfying φ.

We first deal with the case $\mathcal{X} = \mathsf{Set}$. Suppose we are given a family of coalgebra morphisms $(s_i)_{i\in I}$

$$(X_i, \xi_i) \xrightarrow{s_i} (X, \xi)$$

and we want to describe the union of the images of the s_i. It should be obvious that a **union of the images** of the s_i, if it exists, has to be given by a factorisation $s_i = m \circ e_i$

$$(X, \xi_i) \xrightarrow{e_i} (X', \xi') \xrightarrow{m} (X, \xi)$$

such that m is injective and the e_i are *collectively surjective* (ie for each $x' \in X'$ there is $i \in I$ and $x \in X_i$ such that $e'_i(x) = x'$). This determines the union of images up to unique isomorphism and we write

$$(X', \xi') = \bigcup\{\mathrm{Im}(s_i) : i \in I\}.$$

We show that coalgebras over sets have union of images.

Proposition 2.2.10. *Let* $\Sigma : \mathsf{Set} \to \mathsf{Set}$. *Then* $\mathsf{Coalg}(\Sigma)$ *has union of images.*

Proof. Let $(s_i : A_i \to A)_{i\in I}$ be a family of morphisms in $\mathsf{Coalg}(\Sigma)$. The idea is to construct unions as disjoint unions (coproducts) followed by a quotient. But we have to be a bit careful since $\coprod_I A_i$ may not exists since I is allowed to be a proper class. Let

$$A_i \xrightarrow{e_i} A'_i \xrightarrow{m_i} A$$

be a $(Surj, Inj)$-factorisation of each s_i. Since A has—up to isomorphism—only a *set* of subcoalgebras there is a sub*set* $J \subset I$ and a function $f : I \to J$ choosing for each A'_i, $i \in I$, an isomorphic subcoalgebra $A'_{f(i)}$. Moreover, there are morphisms $e'_i : A_i \to A'_{f(i)}$ such that $s_i = m_{f(i)} \circ e'_i$. Since J is a set the coproduct $in_j : A'_j \to \coprod_{j\in J} A'_j$ exists. Now consider

$$\begin{tikzcd}
 & \coprod\limits_J A'_j \arrow[r, "e"] \arrow[dr, dotted, "g"] & A' \arrow[d, "m"] \\
A_i \arrow[r, "e'_i"] & A'_{f(i)} \arrow[u, "in_{f(i)}"] \arrow[r, "m_{f(i)}"] & A
\end{tikzcd} \tag{2.3}$$

where g is given by the universal property of the coproduct and $m \circ e$ is the $(Surj, Inj)$-factorisation of g. It follows that the s_i have a factorisation $m \circ (e \circ in_{f(i)} \circ e'_i)$. Observing that $(e \circ in_{f(i)} \circ e'_i)_{i\in I}$ is collectively surjective finishes the proof. $\square$

For the general case, we note that in the proof of the proposition above we can replace $\mathsf{Coalg}(\Sigma)$ by any category $\mathcal{C}$ which has a factorisation system, coproducts, and for each $A \in \mathcal{C}$, up to isomorphism, only a set of embeddings.[5] For the purposes of this course, the following definition will be convenient:

Definition 2.2.11 (Union of images). We say that a category $\mathcal{C}$ with a factorisation system has *union of images* iff $\mathcal{C}$ has coproducts and for each $A \in \mathcal{C}$, up to isomorphism, only a set of embeddings $A' \to A$. Union of images are then given as quotients of coproducts as in (2.3).

A more general notion of union of images can be given using factorisation systems for sinks, see Adámek, Herrlich, Strecker [3], Chapter 15 and [36] for their application to coalgebras.

Unions of images have a diagonalisation property which is similar to the one for factorisation systems:

Proposition 2.2.12. *Let (E, M) be a factorisation system in a category $\mathcal{C}$ which has union of images. Let $n \circ e_i$ be the union of the images of a family s_i. Then if for $m \in M$, $f \in \mathcal{C}$, and a family t_i in $\mathcal{C}$, the square*

$$\begin{tikzcd} A_i \arrow[r, "e_i"] \arrow[d, "t_i"'] & B \arrow[d, "f"] \arrow[dl, dotted, "d"'] \\ C \arrow[r, "m"'] & D \end{tikzcd}$$

commutes for all i then there is a unique diagonal d making the triangles commute.

Proof. Redraw the square above with $e_i = e \circ in_{f(i)} \circ e'_i$ as in (2.3). Then use unique diagonalisation for factorisation systems (2 times) and the universal property of the coproduct. □

2.2.4 Final and Cofree Coalgebras

We have seen in Chapter 1.2 that final coalgebras play an important role because they classify processes up to behavioural equivalence. *Cofree coalgebras* do the same, but allow the environment additional observations called colourings. We first take the time to discuss colourings in some detail and then explain cofree coalgebras.

Given a coalgebra $X \xrightarrow{\xi} \Sigma X$ and a set 'of colours' C, a **colouring** of (X, ξ) in C is a function $X \xrightarrow{c} C$. c is simply a marking or labeling of the states. Its import is that

[5] The technical term for this last condition is that $\mathcal{C}$ is *wellpowered*.

we can use colourings c to make additional observations. Consider eg the $\mathcal{P}_\omega$-coalgebra (Y, η, y_0) given by

$$\begin{tikzcd} & y_0 \arrow[dl] \arrow[dr] & \\ y_1 & & y_2 \end{tikzcd}$$

Here, the two states y_1, y_2 are behaviourally equivalent for an external observer. But allowing colourings $c : Y \to C$, $C = \{c_1, c_2\}$, an external observer can distinguish y_1, y_2 by choosing a colouring with eg $c(y_1) = c_1$ and $c(y_2) = c_2$.

That is, allowing colourings increases the observational power of the environment. If we want to stay with our basic paradigm that two elements cannot be distinguished by an external observer iff these elements cannot be identified by some morphisms (see Observation 1.2.7), we need to require morphisms to respect colourings. This gives rise to a new category of coalgebras with colourings:

Definition 2.2.13 ($\mathsf{Coalg}(\Sigma, C)$). Let $\Sigma : \mathcal{X} \to \mathcal{X}$ and $C \in \mathcal{X}$. $\mathsf{Coalg}(\Sigma, C)$ is the category having objects

$$(X \xrightarrow{\xi} \Sigma X \ , \ c : X \to C)$$

where $X \xrightarrow{\xi} \Sigma X$ is a Σ-coalgebra and $c : X \to C$ an arrow in $\mathcal{X}$. We call these objects (Σ, C)-coalgebras and denote them by $((X, \xi), c)$ or (X, ξ, c). A (Σ, C)-morphism $f : (X, \xi, c) \to (X', \xi', c')$ is a Σ-morphism $(X, \xi) \to (X', \xi')$ such that

$$\begin{tikzcd} X \arrow[rr, "f"] \arrow[dr, "c"'] & & X' \arrow[dl, "c'"] \\ & C & \end{tikzcd}$$

commutes.

The last condition expresses that (Σ, C)-morphisms preserve colours.

We can now define cofree coalgebras

Definition 2.2.14 (Cofree Coalgebras). A (Σ, C)-coalgebra $(Z_C, \zeta_C, \varepsilon_C)$ is called the cofree Σ-coalgebra over C iff it is final in $\mathsf{Coalg}(\Sigma, C)$. We say that $\mathsf{Coalg}(\Sigma)$ has cofree coalgebras if cofree coalgebras exists for all $C \in \mathcal{X}$.

Usually, we leave ε_C implicit and call (Z_C, ζ_C) alone the cofree Σ-coalgebra over C. In the following exercise you are asked to unravel the definition of a cofree coalgebra.

Exercise 2.2.15. Show that (Z_C, ζ_C) is cofree over C iff for all Σ-coalgebras (X, ξ) and all colourings $c : X \to C$ there is a unique Σ-morphism $c^\sharp : (X, \xi) \to (Z_C, \zeta_C)$ such that

$$\begin{tikzcd} X \arrow[rr, dotted, "c^\sharp"] \arrow[dr, "c"'] & & Z \arrow[dl, "\varepsilon_C"] \\ & C & \end{tikzcd}$$

commutes.

The diagram above is not 'well-typed' in the sense that two arrows are colourings (from the base category) and another one is a coalgebra morphism. This can be corrected by introducing the following

Notation 2.2.16 (Forgetful functor). The forgetful functor is the operation $U : \mathsf{Coalg}(\Sigma) \to \mathcal{X}$ mapping a coalgebra to its carrier and a coalgebra morphism $f : (X, \xi) \to (X', \xi')$ to $f : X \to X'$.

Note that U is indeed a functor. Restating the exercise above using the new terminology yields a formulation of cofreeness which will be of use in Chapter 5.

Proposition 2.2.17. *Let $U : \mathsf{Coalg}(\Sigma) \to \mathcal{X}$ be the forgetful functor. $\mathsf{Coalg}(\Sigma)$ has cofree coalgebras iff for each $C \in \mathcal{X}$ there is a Σ-coalgebra FC and a colouring $\varepsilon_C : UFC \to C$ such that for any Σ-coalgebra A and any colouring $c : UA \to C$ there is a unique coalgebra morphism $c^\sharp : A \to FC$ such that the triangle*

$$\begin{tikzcd} & UFC \arrow[dl, "\varepsilon_C"'] \\ C & UA \arrow[l, "c"] \arrow[u, "Uc^\sharp"'] \end{tikzcd} \qquad \begin{tikzcd} FC \\ A \arrow[u, "c^\sharp"'] \end{tikzcd}$$

commutes.

We finish this section with two exercises which should make you familiar with the notions of (Σ, C)-coalgebra and cofree coalgebra.

Exercise 2.2.18. Give an isomorphism $\mathsf{Coalg}(\Sigma, C) \simeq \mathsf{Coalg}(\Sigma \times C)$ (for any base category with binary products).

Exercise 2.2.19. Consider three $\mathcal{P}_\omega$-coalgebras (X, ξ, x_0), (Y, η, y_0), (Z', ζ', z_0) given by

$$\begin{tikzcd} & x_0 \arrow[dl] \arrow[d] \arrow[dr] & \\ x_1 & x_2 & x_3 \end{tikzcd} \qquad \begin{tikzcd} & y_0 \arrow[dl] \arrow[dr] & \\ y_1 & & y_2 \end{tikzcd} \qquad \begin{tikzcd} z_0 \arrow[d] \\ z_1 \end{tikzcd}$$

1. Show that (Z', ζ') is a subcoalgebra of the final $\mathcal{P}_\omega$-coalgebra.

2. Show that (X, ξ), (Y, η), (Z', ζ') are behaviourally equivalent.

3. Say that two processes (X, ξ, x_0), (Y, η, y_0) are *C-behaviourally equivalent* iff

- for each colouring $c : X \to C$ there is a colouring $d : Y \to C$ such that (X, ξ, c, x_0) and (Y, η, d, y_0) are behaviourally equivalent and, vice versa,
- for each colouring $d : Y \to C$ there is a colouring $c : X \to C$ such that (X, ξ, c, x_0) and (Y, η, d, y_0) are behaviourally equivalent.

Show that

(a) (Y, η, y_0) and (Z', ζ', z_0) are not 2-behaviourally equivalent,

(b) (X, ξ, x_0) and (Y, η, y_0) are 2-behaviourally equivalent,

(c) (X, ξ, x_0) and (Y, η, y_0) are not 3-behaviourally equivalent,

where 2 and 3 denote sets of respective cardinality.

2.3 Algebras

This section reviews algebras as far as needed to understand the duality to coalgebras. We also briefly review the semantics of equations. Some basic familiarity with algebras and equational logic will be helpful, see eg Wechler [71] for an introduction.

Definition 2.3.1. Given a category $\mathcal{X}$, called the base category, and a functor $\Sigma : \mathcal{X} \to \mathcal{X}$, a Σ-**algebra** (Y, ν) is given by an arrow $\nu : \Sigma Y \to Y$ in $\mathcal{X}$. A morphism between two algebras $(Y, \nu) \to (Y', \nu')$ is an arrow f in $\mathcal{X}$ such that $\nu' \circ \Sigma f = f \circ \nu$:

$$\begin{tikzcd} \Sigma Y \arrow[r, "\nu"] \arrow[d, "\Sigma f"'] & Y \arrow[d, "f"] \\ \Sigma Y' \arrow[r, "\nu'"'] & Y' \end{tikzcd}$$

The category of Σ-algebras and morphisms is denoted by $\mathsf{Alg}(\Sigma)$. The forgetful functor $U : \mathsf{Alg}(\Sigma) \to \mathcal{X}$ maps algebras (Y, ν) to the carrier Y and morphisms $f : (Y, \nu) \to (Y', \nu')$ to the arrows $f : Y \to Y'$.

This notion of algebras for a functor includes algebras defined by operations in the usual sense. To give examples it is useful to have the following

Notation 2.3.2. A family of functions $(f_i : Y_i \to Y)_{1 \le i \le n}$ can equivalently be written as a single function

$$Y_1 + \ldots + Y_n \xrightarrow{[f_1, \ldots, f_n]} Y$$

where as before $+$ denotes disjoint unions of sets and $[f_1, \ldots, f_n]$ is the function which applies f_i to arguments from Y_i. (This equivalence is valid in any category with coproducts.)

Example 2.3.3. The following examples show that algebras in the standard sense are algebras for a functor.

1. Consider algebras given by a constant 0 and a unary operation s. The corresponding functor is $\Sigma Y = 1 + Y$ and a Σ-algebra is given by

$$1 + Y \xrightarrow{[0, s]} Y$$

2. Consider the following signature for algebras

$$\begin{aligned} &new : \to \text{stack} \\ &push : \text{A} \times \text{stack} \to \text{stack} \end{aligned}$$

The corresponding functor is $\Sigma Y = 1 + A \times Y$ and a Σ-algebra is given by

$$1 + A \times Y \xrightarrow{[new, push]} Y$$

3. Groups $(Y, e, (-)^{-1}, \cdot)$ are algebras for the functor $\Sigma Y = 1 + Y + Y \times Y$:

$$1 + Y + Y \times Y \xrightarrow{[\, e, (-)^{-1}, \cdot\,]} Y.$$

In the same way, any signature for algebras given by some collection of operation symbols gives rise to a functor.

A Σ-algebra (I, ι) is **initial** iff it is initial in $\mathsf{Alg}(\Sigma)$, ie iff for any Σ-algebra (Y, ν) there is a unique Σ-morphism

$$(I, \iota) \to (Y, \nu).$$

I consists precisely of all terms that can be formed from the operations in the signature. For example, the natural numbers are the initial algebra for the functor $\Sigma Y = 1 + Y$ (read 0 as zero and s as successor):

$$1 + \mathbb{N} \xrightarrow{[0, s]} \mathbb{N}$$

To say that $(\mathbb{N}, [0, s])$ is initial is equivalent to the principle of **induction**. To see that initiality gives rise to induction, recall that defining a function $f : \mathbb{N} \to Y$ by induction means to give a $y_0 \in Y$ such that $f(0) = y_0$ and a $t : Y \to Y$ such that $f(s(n)) = t(f(n))$, that is, to give

$$1 + Y \xrightarrow{[y_0, t]} Y$$

such that

$$\begin{tikzcd} 1+\mathbb{N} \arrow[r, "{[0,s]}"] \arrow[d, "\mathrm{id}_1+f"'] & \mathbb{N} \arrow[d, "f"] \\ 1+Y \arrow[r, "{[y_0,t]}"] & Y \end{tikzcd}$$

Exercise 2.3.4. Check that the diagram above commutes iff $f(0) = y_0$ and $f(s(n)) = t(f(n))$ for all $n \in \mathbb{N}$.

Exercise 2.3.5. Compare induction with coinduction (see Chapter 1.2.8).

Equations

As shown by the example of groups in Example 2.3.3 (or the example of stacks in Chapter 1.4.2), we often want algebras not to be given only by a signature but also by some equations as eg

$$\begin{aligned} e \cdot x &= x \\ x \cdot e &= x \\ (x \cdot y) \cdot z &= x \cdot (y \cdot z) \end{aligned}$$

where x, y, z are variables from some set X.

Although it should be clear what it means for an algebra (Y, ν) to satisfy a set of equations Φ, we want a precise definition. Since equations are formed from terms, we first need a description of terms in variables from X. This is provided by the notion of a free algebra over X.

Definition 2.3.6 (Free algebra). Let $\Sigma : \mathcal{X} \to \mathcal{X}$ and $X \in \mathcal{X}$. The free Σ-algebra over X is given by an algebra (A_X, α_X) and an arrow $\eta_X : X \to A_X$ such that for each algebra (Y, ν) and each $v : X \to Y$ there is a unique algebra morphism $v^\sharp : (A_X, \alpha_X) \to (Y, \nu)$ such that $v^\sharp \circ \eta_X = v$:

$$\begin{tikzcd} A_X \arrow[rr, dotted, "v^\sharp"] & & Y \\ & X \arrow[ul, "\eta_X"] \arrow[ur, "v"'] & \end{tikzcd}$$

We say that $\mathsf{Alg}(\Sigma)$ has free algebras iff for all $X \in \mathcal{X}$ there is a free algebra over X.

Remark. In the case $\mathcal{X} = \mathsf{Set}$ we read this as follows: For a set of variables X there is the *term algebra* (A_X, α_X) which has as a carrier A_X all terms formed from operations in Σ and variables in X. η_X is the inclusion of variables into terms. v is an *assignment of variables* to elements of (Y, ν). The condition above now expresses the familiar fact that any assignment of variables v defines a unique interpretation $v^\sharp$ of terms.

Exercise 2.3.7. Compare the definition of free algebras with the characterisation of cofree coalgebras in Exercise 2.2.15.

We can now say that, in the case $\mathcal{X} = \mathsf{Set}$, an equation $t = t'$ of terms t, t' in variables from X is an element of $(t, t') \in A_X \times A_X$. Satisfaction of equations is then defined as follows.

Definition 2.3.8 (Satisfaction of equations). Let $\Sigma : \mathsf{Set} \to \mathsf{Set}$ and (A_X, α_X) a free Σ-algebra over X. Let $(Y, \nu) \in \mathsf{Alg}(\Sigma)$ and $\Phi \subset A_X \times A_X$ (ie Φ is a set of equations in variables from X). For an equation $(t, t') \in \Phi$ and an assignment $v : X \to Y$ define

$$(Y, \nu), v \models (t, t') \quad \text{iff} \quad v^\sharp(t) = v^\sharp(t')$$

We write $(Y, \nu) \models \Phi$ and say that (Y, ν) satisfies Φ, or that Φ holds in (Y, ν), iff $(Y, \nu), v \models (t, t')$ for all $(t, t') \in \Phi$ and all assignments $v : X \to Y$.

How the concept of a set of equations can be generalised to arbitrary categories (with factorisation system) is shown in Chapter 5.3.

2.4 Duality

We briefly review categorical duality. A category $\mathcal{C}$ consists of a class of objects, also denoted by $\mathcal{C}$, and for all $A, B \in \mathcal{C}$ of a set of arrows (or morphisms) $\mathcal{C}(A, B)$. The *dual* (or opposite) category $\mathcal{C}^{\mathrm{op}}$ has the same objects and arrows $\mathcal{C}^{\mathrm{op}}(A, B) = \mathcal{C}(B, A)$. We write A^{op} and f^{op} for $A \in \mathcal{C}$ and $f \in \mathcal{C}(B, A)$ to indicate when we think of A as an object in $\mathcal{C}^{\mathrm{op}}$ and of f as an arrow in $\mathcal{C}^{\mathrm{op}}(A, B)$. Duality can now be formalised as follows: Let P be a property of objects or arrows in $\mathcal{C}$. We then say that

an object A (arrow f, respectively) in $\mathcal{C}$ has property co-P
iff A^{op} (f^{op}, respectively) has property P.

For example, an object A is co-initial in $\mathcal{C}$ (usually called terminal or final) iff A is initial in $\mathcal{C}^{\mathrm{op}}$; a morphism $f \in \mathcal{C}(A, B)$ is co-mono (usually called epi) iff f^{op} is mono; C is a co-product $A + B$ iff C^{op} is a product $A^{\mathrm{op}} \times B^{\mathrm{op}}$. Of particular importance for us is

Exercise 2.4.1. Show that (E, M) is a factorisation system in $\mathcal{C}$ iff (M, E) is a factorisation system in $\mathcal{C}^{\mathrm{op}}$.

The duality principle can also be extended to functors. The dual of a functor $F : \mathcal{C} \to \mathcal{D}$ is the functor $F^{\mathrm{op}} : \mathcal{C}^{\mathrm{op}} \to \mathcal{D}^{\mathrm{op}}$ which acts on objects and morphisms as F does. We can now state precisely that algebras are dual to coalgebras:

Proposition 2.4.2. *Let $\Sigma : \mathcal{X} \to \mathcal{X}$. $\mathsf{Alg}(\Sigma)^{op}$ is equivalent to $\mathsf{Coalg}(\Sigma^{op})$.*

Proof. The iso maps objects $(\Sigma X \xrightarrow{\xi} X)^{\mathrm{op}}$ to $X^{\mathrm{op}} \xrightarrow{\xi^{\mathrm{op}}} \Sigma^{\mathrm{op}} X^{\mathrm{op}}$ and is the identity on morphisms. □

Note that the base category $\mathcal{X}$ gets dualised as well. To emphasise this trivial but important point we state an evident corollary to the proposition:

Corollary 2.4.3. *Let $\Sigma : \mathcal{X} \to \mathcal{X}$. Then the forgetful functor $U : \mathsf{Alg}(\Sigma) \to \mathcal{X}$ is dual to the forgetful functor $U^{op} : \mathsf{Coalg}(\Sigma^{op}) \to \mathcal{X}^{op}$.*

The fact that the base category has to be dualised makes it difficult to exploit the duality of algebras and coalgebras. For example exercise 2.6.5 shows that $\mathsf{Set}^{\mathrm{op}}$ is isomorphic to the category of complete atomic boolean algebras. Therefore, coalgebras over set are isomorphic to algebras over complete atomic boolean algebras, a fact that seems not very helpful in the study of coalgebras.

Nevertheless, duality may still be exploited. The idea is to take care that **the proofs of our results do not depend on a specific base category**. There are two possibilities. First, it may be that the properties we are interested in can be formulated in a way such that they do not depend on the base category at all. Second, if some properties P of the base category are needed, we are careful to keep track of them. In this way, we are able to obtain results that hold, say, for all algebras over base categories satisfying P, and all coalgebras over base categories satisfying co-P. Even if we are only interested in results about algebras as well as coalgebras over sets, this approach is still useful. For example, Set is wellpowered and cowellpowered, complete and cocomplete and has a factorisation system.

$\mathsf{Coalg}(\Sigma)$	$\mathsf{Alg}(\Sigma^{\mathrm{op}})$
factorisation system (E, M)	factorisation system (M, E)
subcoalgebras	quotients
quotients	subalgebras
coproducts	products
union of images	intersection of kernels
final coalgebra	initial algebra
cofree coalgebra	free algebra
coinduction	induction
behaviour	reachable part
bisimulation	subalgebra (?)

Table 2.1: List of Dualities

Table 2.1 summarises dualities which are interesting for us. The notion of *intersection of kernels* is given by the dual of Definition 2.2.11 and will be illustrated in the next section. *Behaviour and reachable part* are dual notions: the behaviour of a coalgebra (X, ξ)

is given by the factorisation $(X, \xi) \xrightarrow{e} Beh(X, \xi) \xrightarrow{m} (Z, \zeta)$ of the unique morphism $(X, \xi) \to (Z, \zeta)$ into the final coalgebra (Z, ζ); the reachable part of an algebra (Y, ν) is given by the factorisation $(I, \iota) \xrightarrow{e} Reach(Y, \nu) \xrightarrow{m} (Y, \nu)$ of the unique morphism $(I, \iota) \to (Y, \nu)$ from the initial algebra. *Bisimulation and subalgebra* are dual notions if we consider the essence of a bisimulation being to define a quotient on a system.

To investigate whether this table can be extended to *logics* for (co)algebras is the purpose of Chapter 5.

2.5 Extended Example: Limits

This section illustrates the material we have seen so far, namely duality, union of images, and cofree coalgebras.

We have seen that that colimits of coalgebras are easy (ie calculated as in Set). What about limits? We will use our intuition on algebras and then duality to provide a solution.

The first step is to describe colimits of algebras. In order to think of something concrete, let us consider the coproduct of two monoids $(Y_1, e, \cdot)$ and $(Y_2, e, \cdot)$. Clearly, it can not be given by the coproduct (disjoint union) $Y_1 + Y_2$ because, roughly speaking, $Y_1 + Y_2$ contains not enough elements. For example, $Y_1 + Y_2$ contains no element which we could consider as the composition of some $y_1 \in Y_1$ and some $y_2 \in Y_2$. Surely, the free algebra over both carriers, $F(Y_1 + Y_2)$, contains enough elements (eg $y_1 \cdot y_2$). But $F(Y_1 + Y_2)$ is not the coproduct itself because it does not satisfy enough equations. For example, if $x_1 \cdot y_1 = z_1$ in $(Y_1, e, \cdot)$ then $x_1 \cdot y_1 \neq z_1$ in $F(Y_1 + Y_2)$ just because it is the free algebra (which means that it satisfies no equations but those enforced by the monoid axioms). So what we need to find is the appropriate equations which make the quotient of $F(Y_1 + Y_2)$ into the coproduct of $(Y_1, e, \cdot)$ and $(Y_2, e, \cdot)$. But this is easy. Consider two algebra morphisms as in

$$
\begin{tikzcd}
(Y_1, e, \cdot) \arrow[dr, "f"'] & F(Y_1 + Y_2) \arrow[d, "h_{f,g}"] & (Y_2, e, \cdot) \arrow[dl, "g"] \\
 & (Z, e, \cdot) &
\end{tikzcd}
$$

Since $F(Y_1 + Y_2)$ is free, there is a unique algebra morphism $h_{f,g}$ which agrees with f and g on Y_1 and Y_2, respectively. The kernel $\operatorname{Ker} h_{f,g} = \{(t, t') : h_{f,g}(t) = h_{f,g}(t')\}$ contains precisely the equations[6] (t, t') satisfied by $(Z, e, \cdot)$. Now, in order to find the equations satisfied by the coproduct we take the intersections of all these kernels: Let

$$\Phi = \bigcap \{\operatorname{Ker} h_{f,g} : \exists\, (Z, e, \cdot) \ \& \ f : (Y_1, e, \cdot) \to (Z, e, \cdot) \ \& \ g : (Y_2, e, \cdot) \to (Z, e, \cdot)\}$$

and define the coproduct of $(Y_1, e, \cdot)$ and $(Y_2, e, \cdot)$ as the quotient of $F(Y_1 + Y_2)$ wrt to the smallest congruence generated by Φ. Of course, it remains to be checked that this definition yields indeed the coproduct. If you are interested in the details, dualise the proof below.

[6] Recall that we write equations as (t, t') rather than $t = t'$.

The principle of duality now makes us guess that limits of coalgebras must be obtained as certain subcoalgebras of cofree coalgebras. This idea gives rise to the following theorem and proof.

Theorem 2.5.1. *Let* Σ *be a functor on* Set *such that* $\mathsf{Coalg}(\Sigma)$ *has cofree coalgebras. Then* $\mathsf{Coalg}(\Sigma)$ *has all limits and they are constructed as shown in the proof.*

Proof. Let $D : \mathcal{I} \to \mathsf{Coalg}(\Sigma)$ be a diagram in $\mathsf{Coalg}(\Sigma)$. Let $c_i : L \to UDi$ be a limit of UD in Set. Consider the cofree coalgebra FL over L with colouring $\varepsilon_L : UFL \to L$.

$$\begin{tikzcd}
UDi & L \arrow[l, "c_i"'] & UFL \arrow[l, "\varepsilon_L"'] \\
UA_j \arrow[u, "Uf_j^i"] \arrow[ur, "g_j"] \arrow[urr, "Ug_j^\sharp"] \arrow[rr, "Ue_j"'] & & UB \arrow[u, "Um"']
\end{tikzcd}$$

Let $f_j^i : A_j \to Di$ be a cone for the diagram D. Since L is a limit of UD, there is a unique $g_j : UA_j \to L$ such that $Uf_j^i = c_i \circ g_j$. Since FL is cofree, g_j lifts to a unique $g_j^\sharp : A \to FL$ such that $\varepsilon_L \circ Ug_j^\sharp = g_j$.

Let $\{f_j^i : A_j \to Di : j \in J\}$ be the class of cones for D. We have seen that every cone $f_j^i : A_j \to Di$ gives rise to a $g_j^\sharp : A_j \to FL$. The limit B of D is now the union of images $A_j \xrightarrow{e_j} B \xrightarrow{m} FL$ of the family $(g_j^\sharp)_{j \in J}$.

To find the limiting cone consider $l_i = c_i \circ \varepsilon_L \circ Um$. Since for all $i \in \mathcal{I}$, $l_i \circ Ue_j = Uf_j^i$ are morphisms in $\mathsf{Coalg}(\Sigma)$ and the family of morphisms $(e_j)_{j \in J}$ is collectively surjective, it follows that the $l_i : B \to Di$ are coalgebra morphisms. Furthermore, l_i is a cone for D because it is a cone for UD which in turn holds because Uf_j^i is a cone for UD (for all j) and the $(Ue_j)_{j \in J}$ are collectively surjective.

To complete the proof we have to show that every cone in $\mathsf{Coalg}(\Sigma)$ over D factors uniquely through $l_i : B \to Di$. The existence follows from the definition of B, uniqueness from m being mono. $\square$

Remark 2.5.2. The construction of the limit shown in the proof of the theorem can be used to obtain detailed information on limits. As an example consider the following coalgebra A for the (finite) powerset functor:

$$\begin{tikzcd}
 & s_0 \arrow[dl] \arrow[dr] & \\
s_1 & & s_2
\end{tikzcd}$$

(The carrier of A is $\{s_0, s_1, s_2\}$ and the transition relation is as depicted in the diagram.) You are invited to use the construction in the proof of the theorem to prove the following remarks on the product $A \times A$:

- The product $A \times A$ is not the largest bisimulation because the product has 'too many states' (the largest bisimulation on A has 5).
- $A \times A$ is finite (the construction in the proof of the theorem also allows to calculate the precise number of states in $A \times A$ though this requires a bit more work).
- Define A' by adding transitions from s_1 and s_2 to s_0 in the coalgebra A. Then $A' \times A'$ is infinite.

2.6 Exercises

Exercise 2.6.1. Some strange operations give rise to functors and some don't:

1. Define $AM : \mathsf{Set} \to \mathsf{Set}$ as $AM(X) = \{(x, y, z) \in X^3 : |\{x, y, z\}| \leq 2\}$. Show that AM can be extended to a functor.
2. Define $B : \mathsf{Set} \to \mathsf{Set}$ as $B(X) = \{(x, y, z) \in X^3 : |\{x, y, z\}| \geq 2\}$. Show that B can *not* be extended to a functor. [Hint: Do the next exercise first.]

Exercise 2.6.2. Show that, roughly speaking, operations have to be monotone in order to allow for an extension to functors.

1. A functor that maps some non-empty set to the empty set maps any non-empty set to the empty set.
2. Denote the cardinality of a set X by $|X|$. Suppose $|X| \neq 0$. Then $|X| \leq |Y| \Rightarrow |\Sigma X| \leq |\Sigma Y|$.

Hint: Use that functors map injective functions with non-empty domain to injective functions.

Exercise 2.6.3 (Largest fixed points as final coalgebras). Let $\mathcal{X}$ be the category of sets with inclusions as morphisms. Show that functors are monotone operators and final coalgebras are largest fixed points.

Exercise 2.6.4 (Lambek's lemma). From the previous exercise, we expect the structure of a final coalgebra to be an isomorphism. Trying to generalise the previous proof leads us to consider

$$\begin{tikzcd} Z \arrow[r, "\zeta"] \arrow[d, "\zeta"'] & \Sigma Z \arrow[r, "f"] \arrow[d, "\Sigma\zeta"'] & Z \arrow[d, "\zeta"'] \\ \Sigma Z \arrow[r, "\Sigma\zeta"] & \Sigma\Sigma Z \arrow[r, "\Sigma f"] & \Sigma Z \end{tikzcd}$$

Assuming that (Z, ζ) is final, show that ζ is iso.

Exercise 2.6.5 ($\mathsf{Set}^{\mathrm{op}}$). A boolean algebra A is *complete* iff it has infinite joins, that is, for all subsets $B \subset A$ there exists a least upper bound $\bigvee B$ in A. An element $a \in A$ is called an *atom* iff for all $b \in A$, $0 < b \leq a \;\Rightarrow\; a = b$. A boolean algebra is *atomic* iff every element is a (possibly infinite) join of atoms. Morphisms are boolean algebra morphisms which preserve infinite joins (hence meets). Denote the category of complete atomic boolean algebras by $CABA$. The following shows that $\mathsf{Set}^{\mathrm{op}}$ is equivalent to $CABA$.[7]

1. Show that $\varphi : \mathsf{Set}^{\mathrm{op}} \to CABA$, $(Y \xrightarrow{f^{\mathrm{op}}} X) \mapsto (\mathcal{P}Y \xrightarrow{f^{-1}} \mathcal{P}X)$ defines a faithful functor. [Here $\mathcal{P}Y, \mathcal{P}X$ denotes the powerset endowed with the structure of a complete atomic boolean algebra.]

2. Writing $\mathsf{at}(A)$ for the set of atoms of $A \in CABA$, show that $f : A \to \mathcal{P}(\mathsf{at}(A))$, $a \mapsto \{b \in \mathsf{at}(A) : b \leq a\}$ is a $CABA$-isomorphism.

3. To see that φ is full consider a $CABA$-morphism $f : \mathcal{P}X \to \mathcal{P}Y$ and let $g : Y \to X$ be such that $g(y)$ is the $x \in X$ with $y \in f(\{x\})$. Show that g is well-defined and $\varphi(g) = f$.

2.7 Notes

The founding paper for the area of universal coalgebra is Rutten [64]. Rutten's approach is based on sets as a base category whereas our goal was to treat (co)algebras over Set and $\mathsf{Set}^{\mathrm{op}}$ simultaneously. More on this approach can be found in [36]. For an overview of results which are specific to coalgebras over Set see Gumm [21]. The duality of behaviours and reachable parts has been studied by Arbib and Manes [4] and recently in the context of algebraic specifications in [11]. The construction of limits in Section 2.5 is from [36],

[7] Using the following characterisation of equivalence: Categories $\mathcal{A}$ and $\mathcal{B}$ are equivalent iff there is a full and faithful functor $\varphi : \mathcal{A} \to \mathcal{B}$ such that every $B \in \mathcal{B}$ is isomorphic to some $\varphi(A)$.

different proofs were given by Worrell [72], Power and Watanabe [47], and Gumm and Schröder [19].

A detailed textbook presentation of factorisation systems and their applications to algebra can be found in Adámek, Herrlich, Strecker [3].

Chapter 3

Modal Logic

The purpose of this chapter is to introduce modal logic as far as needed in this course. For more information see e.g. Blackburn, de Rijke, Venema [12] or Goldblatt [16].

3.1 Kripke Semantics

3.1.1 Introduction

Modal logic originated with the study of logics comprising modalities as eg 'necessarily'. In the beginning of the 20th century a piece of syntax was invented, nowadays mostly written $\Box$, in order to write formulas

$$\Box\varphi$$

having the intended meaning that φ holds necessarily. A question at that time was to describe axiomatically the valid formulas involving necessity. Different proposals were discussed generally including the following two axiom schemes and two rules.

(taut)	all propositional tautologies
(dist)	$\Box(\varphi \to \psi) \to \Box\varphi \to \Box\psi$
(mp)	from $\varphi, \varphi \to \psi$ derive ψ
(nec)	from φ derive $\Box\varphi$

The interpretation is: propositional tautologies are valid; if necessarily $\varphi \to \psi$ and necessarily φ then necessarily ψ; modus ponens is clear; if φ is valid, then necessarily φ is valid. The modal logic consisting of these axioms and rules is today usually denoted by **K**.

In general, one also proposed additional axiom schemes as eg

(refl)	$\Box\varphi \to \varphi$
(trans)	$\Box\varphi \to \Box\Box\varphi$

The interpretation is: if φ holds necessarily, then it holds indeed; if φ holds necessarily then it is necessary that it holds necessarily.

For a long time it was difficult to judge the value of such axiomatisation because there was no appropriate semantics of modal logic. This changed in the 1950s with the advent of possible worlds or Kripke semantics. The idea is to use graphs (X, R), $R \subset X \times X$, as models for modal logic and to think of X as a set of possible worlds and of R as an alternative relation. We then say that a formula holds necessarily in the world x iff it holds in all possible alternatives:

$$(X, R, x) \models \Box\varphi \quad \text{iff} \quad (X, R, y) \models \varphi \quad \text{for all } y \text{ with } xRy$$

A formula holds in (X, R) iff it holds in all worlds $x \in X$ and a formula is *valid* iff it holds in all (X, R).

Exercise 3.1.1. If you are not familiar with Kripke semantics, then show that (dist) is valid. Also show that (nec) is correct: if φ is valid, then also $\Box\varphi$. Show that $\varphi \to \Box\varphi$ is not valid.

3.1.2 Frames and Models

We presented modal logic avoiding any discussion of propositional logic. But there is an issue: namely whether we should interpret the atomic propositions $p \in \mathsf{Prop}$ of propositional logic as variables or as constants. This distinction gives rise to the notions of Kripke frame and Kripke model.

But first let us be precise about the language of modal logic.

Definition 3.1.2 (Modal language). Given a set of atomic propositions Prop, the set of all modal formulas $\mathcal{ML}$, sometimes written $\mathcal{ML}(\mathsf{Prop})$, is defined inductively by

$$\begin{array}{lcl} p \in \mathsf{Prop} & \Rightarrow & p \in \mathcal{ML} \\ \bot \in \mathcal{ML} & & \\ \varphi, \psi \in \mathcal{ML} & \Rightarrow & \varphi \to \psi \in \mathcal{ML} \\ \varphi \in \mathcal{ML} & \Rightarrow & \Box\varphi \in \mathcal{ML} \end{array}$$

$\bot$ is falsum. The other boolean operators $\top, \neg, \wedge, \vee$ can be defined from $\bot, \to$. The modal operator $\Diamond$ is defined as $\neg\Box\neg$.

If we understand atomic propositions as constants we need to extend graphs by interpretations of atomic propositions:

Definition 3.1.3. A **Kripke model** (X, R, V) consists of a set X , a relation $R \subset X \times X$ and a valuation $V : X \to \mathcal{P}\mathsf{Prop}$.

Elements of X are called states, (possible) worlds, or points. R is called the accessibility relation or alternative relation. Elements of Prop are called atomic propositions or propositional variables.

The idea is that V assigns to $x \in X$ the set of atomic propositions holding in x; the semantics of propositional connectives is as usual and the semantics of $\Box$ is as we have seen it. To summarise:

Definition 3.1.4 (Semantics of modal logic). For a Kripke model (X, R, V) and $x \in X$ define:

$$\begin{array}{lll} (X,R,V,x) \models p & \text{iff} & p \in V(x) \\ (X,R,V,x) \not\models \bot & & \\ (X,R,V,x) \models \varphi \to \psi & \text{iff} & (X,R,V,x) \models \varphi \;\Rightarrow\; (X,R,V,x) \models \psi \\ (X,R,V,x) \models \Box\varphi & \text{iff} & xRy \;\Rightarrow\; (X,R,V,y) \models \varphi \;\text{ for all } y \in X. \end{array}$$

φ holds in a model (X, R, V), written $(X, R, V) \models \varphi$, iff $(X, R, V, x) \models \varphi$ for all $x \in X$. And φ is valid, written $\models \varphi$, iff φ holds in all models.

Notation: We write $x \models \varphi$ instead of $(X, R, V, x) \models \varphi$ when (X, R, V) is clear from the context.

We can also take another perspective on atomic propositions. Studying eg the logic of necessity, one is interested in the formulas valid under all possible interpretations of atomic propositions. We then think of atomic propositions as propositional variables:

Definition 3.1.5. A **Kripke frame** (X, R) consists of a set X, and a relation $R \subset X \times X$. Models (X, R, V), $V : X \to \mathcal{P}\mathsf{Prop}$, are said to be *based on* (X, R) and (X, R) is called the *underlying frame* of the model.

A frame (X, R) satisfies a formula φ, or φ holds in (X, R), iff all models based on the frame satisfy φ:

$$(X, R) \models \varphi \quad \text{iff} \quad (X, R, V) \models \varphi \text{ for all } V : X \to \mathcal{P}\mathsf{Prop}$$

Note that φ holds in all models iff it holds in all frames.

One difference between models and frames is that the theory of a frame is always closed under substitution, see Exercise 3.4.1. For frames, it is therefore enough to consider axioms as eg $\Box p \to p$ for some $p \in \mathsf{Prop}$; for models, however, we would employ an axiom scheme $\Box\varphi \to \varphi$ corresponding to the set of axioms $\{\Box\varphi \to \varphi : \varphi \in \mathcal{ML}\}$. A more essential difference between models and frames is the topic of the next

Exercise 3.1.6. Let $p \in \mathsf{Prop}$.

1. Show that $\Box p \to p$ holds in all reflexive frames (X, R) (ie $\forall x \in X \,.\, xRx$).

2. Give an example of a non-reflexive model satisfying $\Box\varphi \to \varphi$ for all $\varphi \in \mathcal{ML}$. Is there a non-reflexive frame satisfying $\Box p \to p$?

From the point of view of logic, frames seem to be the interesting structures: When we ask what formulas are valid under all interpretations of propositional variables, it is natural to consider frames as the semantic structures for modal logic.

On the other hand, from the computer science point of view, models seem to be the natural structures. Consider a program or algorithm as given by a set of states X and a relation R, R giving for each state its successors. But program states are not just 'naked' elements, they carry additional information, typically the contents of the memory. This information can be thought of as being encoded by the valuation $V : X \to \mathcal{P}\mathsf{Prop}$. That is, thinking of modal logic as a specification language for transition systems (algorithms, programs), models are the natural semantic structures of modal logic.

But even then, the underlying frames (X, R) are of some interest. Often, we are not interested in arbitrary models (X, R, V) but want to restrict our attention to programs with special properties, eg deterministic ones. Being deterministic then, is a property of R, and hence a property of the class of underlying frames. For another typical example, think of Kripke models (X, R, V) as runs of programs. In this case we may want to require the underlying frames to have an initial state, to be reflexive, transitive, and perhaps linear.

The next section further develops the exercise above and discusses how modal logic can be used to describe certain frame classes.

3.1.3 Definability

We say that a class of frames $\mathcal{B}$ is defined by a class of formulas Φ iff $\mathcal{B} = \{(X, R) : (X, R) \models \Phi\}$. There are then two questions relating to definability:

- Given a class of formulas, can we characterise the defined class of frames?
- Given a class of frames, are there formulas defining it?

To illustrate the first question, suppose that someone proposes formulas (refl) and (trans) as axioms for necessity. Understanding then the defined class of frames would make it easier to judge the proposed axiomatisation (for example, as will be shown below, whether we accept (trans) depends on whether we think of the alternative relation as being transitive). To illustrate the second question, recall from the discussion at the end of the previous section that we might be interested in defining eg the class of deterministic frames or the class of reflexive, transitive, linear frames.

There exist only partial answers to these questions but many important cases are well-known. Table 3.1 gives a typical list of examples.

To check that a frame satisfying the first-order property also satisfies the modal formula is usually straight forward. If you are not familiar with this, you should do some of the correspondences in Table 3.1 as exercises. The converse direction is usually more difficult to establish. An easy but typical example is the case of (trans):

We show that only transitive frames satisfy $\Box p \to \Box\Box p$. Suppose (X, R) is not transitive, that is, there are $x, y, z \in X$ such that $xRy \wedge yRz \wedge \neg xRz$. We have to find a valuation V such that $(X, R, V, x) \not\models \Box p \to \Box\Box p$. Choose as extension of p the *smallest* set such that $(X, R, V, x) \models \Box p$ (ie let $p \in V(w) \;\Leftrightarrow\; xRw$). Now, $\neg xRz$ guarantees that $p \notin V(z)$ and it follows from $xRy \wedge yRz$ that $(X, R, V, x) \not\models \Box\Box p$.

	Name	Axiom
1	(refl)	$\Box p \to p$
2	(trans)	$\Box p \to \Box\Box p$
3	(ser)	$\Diamond\top$
4	(det)	$\Diamond p \to \Box p$
5	(fun)	$\Diamond p \leftrightarrow \Box p$
6	(dir)	$\Diamond\Box p \to \Box\Diamond p$

	Name	Conditions on R
1	reflexive	$\forall x(xRx)$
2	transitive	$\forall x \forall y \forall z(xRy \wedge yRz \to xRz)$
3	serial	$\forall x \exists y(xRy)$
4	deterministic	$\forall x \forall y \forall z(xRy \wedge xRz \to y = z)$
5	functional	$\forall x \exists! y(xRy)$
6	directed	$\forall x \forall y \forall z(xRy \wedge xRz \to \exists x'(yRx' \wedge zRx'))$

Table 3.1: Modal Formulas and First-Order Correspondences

3.1.4 Multimodal Logics

We have seen Kripke semantics for modal logics with one modality. But the basic ideas of modal logic and possible world semantics can be varied in many ways. We will discuss here only modal logics with more than one modality.

A multimodal logic has modalities $\Box_a$ for all $a \in A$ for some set A. That is, the last clause of Definition 3.1.2 is replaced by

$$\varphi \in \mathcal{ML},\, a \in A \quad \Rightarrow \quad \Box_a\varphi \in \mathcal{ML}$$

One should now write $\mathcal{ML}(\mathsf{Prop}, A)$ but if no confusion can arise we continue to use $\mathcal{ML}$.

A frame $(X, (R_a)_{a \in A})$ for a multimodal logic has a relation R_a for each modality $\Box_a$. A model $(X, (R_a)_{a \in A}, V)$ has additionally a valuation of atomic propositions.

Example 3.1.7 (Hennessy-Milner logic). Consider a multimodal logic without atomic propositions and with modalities $\Box_a$, $a \in A$, where we think of A as a set of actions and of $\Box_a\varphi$ as 'φ holds after a'. A Kripke model is then a transition system $(X, (R_a)_{a\in A})$ (remember that there are no atomic propositions and hence no valuation). It is customary to write $x \xrightarrow{a} y$ for $x\ R_a\ y$ and $[a]$ for $\Box_a$.

Example 3.1.8 (Multi-agent systems). Consider a multimodal logic with modalities $\Box_a$, $a \in A$, where we think of A as a set of agents and of $\Box_a\varphi$ as 'agent a knows φ'. Atomic propositions describe the facts agents can know. A Kripke model $(X, (R_a)_{a\in A}, V)$ can be understood as follows. X is a set of possible worlds and V describes the facts holding in each world. xR_ay means that agent a considers y as an alternative world for x. $x \models \Box_a\varphi$ means that φ holds in all worlds which are considered as alternative worlds by agent a, ie a knows φ.

Example 3.1.9 (Temporal logic). Consider a multimodal logic with two modalities $\circ, \Box$ where we think of $\circ\varphi$ as 'in the next state holds φ' and of $\Box\varphi$ as 'now and always in the future holds φ'. A particularly interesting Kripke frame for this logic is $(\mathbb{N}, S, \leq)$ where $m\ S\ n$ iff $n = m + 1$. Models based on this frame can be considered as runs of programs and the modal logic defined by this frame, linear temporal logic, plays an important role in the verification of programs, see eg [33, 43, 15, 66].

3.2 Bisimulation

Having seen Kripke frames and models, it is natural to ask what would be an appropriate notion of morphism for these structures. But instead of defining morphisms right away, we look first at relations between models. In particular, given two (multimodal) Kripke models $(X, (R_a)_{a\in A}, V)$, $(X', (R'_a)_{a\in A}, V')$, we are interested in describing relations $B \subset X \times X'$ such that

$$x\ B\ x' \ \Rightarrow\ (x \models \varphi\ \Leftrightarrow\ x' \models \varphi).$$

A careful analysis of the definition of $x \models \varphi$ leads to the following notion of bisimulation.

Definition 3.2.1 (Bisimulation). Given two Kripke models $(X, (R_a)_{a\in A}, V)$, $(X', (R'_a)_{a\in A}, V')$ we call $B \subset X \times X'$ a bisimulation between the models iff $x\ B\ x'$ implies that

$$\begin{aligned}
&V(x) = V(x') \\
&x \xrightarrow{a} y \ \Rightarrow\ \exists y' .\ x' \xrightarrow{a} y'\ \&\ y\ B\ y' \\
&x' \xrightarrow{a}{}' y' \ \Rightarrow\ \exists y .\ x \xrightarrow{a} y\ \&\ y\ B\ y'
\end{aligned}$$

(writing $\xrightarrow{a}$ for R_a and R'_a). x, x' are called *bisimilar* iff there is a bisimulation relating them. Bisimulations for frames can be obtained as a special case by ignoring the clause concerning the valuations V, V'.

Examples of (non-)bisimilarity can be found in the exercises. For us, the following is essential and an exercise that should not be missed.

Exercise 3.2.2. Show by induction on the structure of formulas that given two models $(X, (R_a)_{a\in A}, V)$, $(X', (R'_a)_{a\in A}, V')$ then for all $x \in X$, $x' \in X'$ it holds: x, x' bisimilar implies that $x \models \varphi \;\Leftrightarrow\; x' \models \varphi$ for all modal formulas φ.

We now define morphisms as functional bisimulations.

Definition 3.2.3. Given two Kripke models/frames $(X, \dots)$, $(X', \dots)$ a morphism $f : (X, \dots) \to (X', \dots)$ is a function $f : X \to X'$ such that its graph $\{(x, f(x)) : x \in X\}$ is a bisimulation.

These morphisms are usually called p-morphisms or bounded morphisms. The following observation—which should by now be no surprise—justifies to call them simply morphisms.

Proposition 3.2.4. *The morphisms of Kripke models/frames are precisely the coalgebra morphisms.*

Proof. (Monomodal) Kripke frames are $\mathcal{P}$-coalgebras and their morphisms were shown to be functional bisimulations in Proposition 2.1.3 (check this). Kripke models are $(\mathcal{P} \times \mathcal{P}\mathsf{Prop})$-coalgebras; multimodal Kripke frames are $\mathcal{P}(A \times -)$-coalgebras and multimodal Kripke models $(\mathcal{P}(A \times -) \times \mathcal{P}\mathsf{Prop})$-coalgebras. These cases are only slight variations. $\square$

Another way to phrase the relationship between coalgebras and Kripke models/frames is the following:

Proposition 3.2.5. *Let $(X, \dots)$, $(X', \dots)$ be two Kripke models/frames. Then $x \in X$, $x' \in X'$ are bisimilar (in the sense of modal logic) iff they are behaviourally equivalent (in the coalgebraic sense).*

The relationship between modal formulas and morphisms is summarised by the following two classical results. We need some standard terminology: a formula φ is preserved under quotients if $A \to A'$ surjective and $A \models \varphi$ implies $A' \models \varphi$; φ is preserved under submodels/subframes if $A' \to A$ injective and $A \models \varphi$ implies $A' \models \varphi$; φ is preserved under disjoint unions (or coproducts) if $A_i \models \varphi$ for all $i \in I$ implies $\coprod_I A_i \models \varphi$; φ is preserved under domains of quotients if $A' \to A$ surjective and $A \models \varphi$ implies $A' \models \varphi$.

Proposition 3.2.6. *Wrt Kripke models, modal formulas are preserved under quotients, submodels, disjoint unions, and domains of quotients.*

Proposition 3.2.7. *Wrt Kripke frames, modal formulas are preserved under quotients, subframes, and disjoint unions.*

The proof of this propositions is an easy corollary to Exercise 3.2.2.

3.3 The Logic of Bisimulation

The aim of this section is to substantiate the claim that modal logic is the logic of bisimulation. We have seen in Exercise 3.2.2 that for two models (X, R, V), (X', R', V'), and $x \in X$, $x' \in X'$

$$x, x' \text{ bisimilar} \quad \Rightarrow \quad \forall \varphi \in \mathcal{ML} : x \models \varphi \;\Leftrightarrow\; x' \models \varphi,$$

that is, bisimilarity implies modal equivalence. Unfortunately, the converse does not hold. Figure 3.1 shows an example where x has for each $n \in \mathbb{N}$ a branch of length n, and x' has additionally an infinite branch. That x and x' are not bisimilar is not difficult to see:

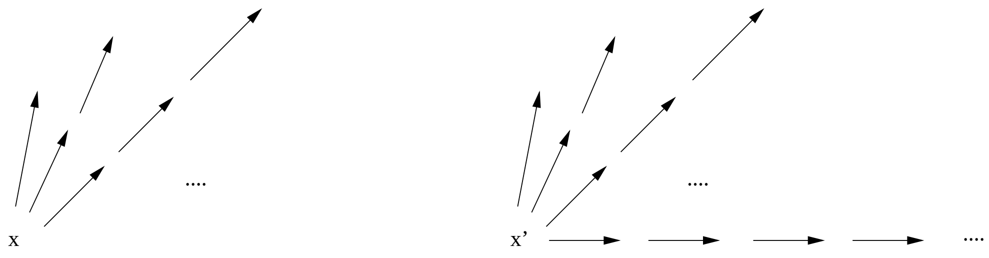


Figure 3.1: Modally equivalent but not bisimilar models

Exercise 3.3.1. Consider the models in Figure 3.1 (assume that all states satisfy the same atomic propositions). Show that x, x' are not bisimilar.

To show that x and x' are modally equivalent is not difficult either but requires a bit more work, see Exercise 3.4.5.

The example suggests (at least after having done Exercise 3.4.5) that the failure of modal logic to characterise states up to bisimilarity is related to the facts that

- a single modal formula can not express enough about an infinite branch, and that
- a transition system may have infinite branching.

And indeed, adjusting either of the two points above results in a perfect match of bisimilarity and modal expressiveness. This is the contents of the following two theorems.

The first idea is to increase expressiveness of modal logic using infinitary modal logic $\mathcal{ML}_\infty$. $\mathcal{ML}_\infty$ is defined as $\mathcal{ML}$ with the additional clause

$$\Phi \subset \mathcal{ML}_\infty \quad \Rightarrow \quad \bigwedge \Phi \in \mathcal{ML}_\infty$$

and stipulating $\quad x \models \bigwedge \Phi \;\Leftrightarrow\; \forall \varphi \in \Phi : x \models \varphi.$

Theorem 3.3.2. *For each model (X, R, V) and each $x \in X$ there is a formula $\varphi_x \in \mathcal{ML}_\infty$ such that for all models (X', R', V') and all $x' \in X'$*

$$x' \models \varphi_x \quad \textit{iff} \quad x, x' \textit{ bisimilar.}$$

The other idea is to restrict attention to models with finite branching.

Theorem 3.3.3 (Hennessy and Milner). *Let K be the class of image-finite Kripke models, ie for all (X, R, V) and all $x \in X$ the set $\{y : x\ R\ y\}$ is finite. Then for all (X, R, V), (X', R', V') in K and all $x \in X$, $x' \in X'$*

$$\forall \varphi \in \mathcal{ML} : x \models \varphi \;\Leftrightarrow\; x' \models \varphi \;\Rightarrow\; x, x' \textit{ bisimilar.}$$

From the point of view of classical first-order logic, however, the most satisfactory explanation of the relationship of modal logic to bisimulation is the following characterisation of modal logic as the bisimulation invariant fragment of first-order logic: A first-order formula is invariant under bisimulations iff it is equivalent to a modal formula.

To make this precise we note that a Kripke model (X, R, V) can also be understood as a first-order model with one binary relation R and one unary predicate P for each atomic proposition $p \in \mathsf{Prop}$. Let us call $\mathcal{FL}$ the corresponding first-order language (containing one relation symbol and for each atomic proposition a unary predicate symbol). The definition of $(X, R, V, x) \models \varphi$ in Section 3.1.2 can now be read as a translation $(-)^* : \mathcal{ML} \to \mathcal{FL}$ of modal formulas in first-order formulas with one free variable x:

$$\begin{aligned}
&p^* = P(x)\\
&\bot^* = \bot\\
&(\varphi \to \psi)^* = \varphi^* \to \psi^*\\
&(\Box\varphi)^* = \forall y : xRy \to \varphi^*[y/x]
\end{aligned}$$

where y is a variable not occurring free in φ^* (and $[y/x]$ denotes substitution of y for x).

Theorem 3.3.4 (van Benthem). *A first-order formula $\psi \in \mathcal{FL}$ is invariant under bisimulation iff it is logically equivalent to a translation φ^* of a modal formula $\varphi \in \mathcal{ML}$.*

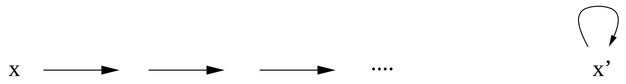
3.4 Exercises

Exercise 3.4.1. Show that the theory of a frame is closed under substitution. That is, for $\varphi, \psi \in \mathcal{ML}$ and $p \in \mathsf{Prop}$ it holds that $(X, R) \models \varphi \;\Rightarrow\; (X, R) \models \varphi[\psi/p]$ (where $[\psi/p]$ denotes substitution of ψ for p).

Exercise 3.4.2 (Examples of bisimilarities). Assume a monomodal language. Show that in the models given below the states x and x' are bisimilar.

1. The relational structure of the models is depicted below. For the valuations assume that $V(y) = V(z) = V'(y')$ and $V(x) = V'(x')$.

2. For the following models assume that all states have the same valuation.



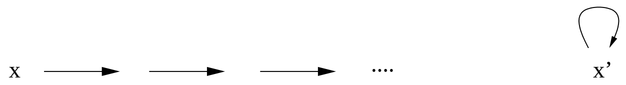
Exercise 3.4.3 (A non-example of bisimilarity). Assume a multimodal language with three modalities $A = \{a, b, c\}$ and no atomic propositions. Consider the two models below.

1. Show that x, x' are not bisimilar.

2. Give modal formulas that distinguish x and x'.

Note that both models show the same behaviour $\{ab, ac\}$ if only traces are considered.

Exercise 3.4.4 (Bisimilarity of frames). For frames bisimilarity does not imply modal equivalence. First note that x, x' in the the following two *frames* are bisimilar.



Now, show

1. $x \models \varphi \;\Rightarrow\; x' \models \varphi$

2. $x' \models \varphi \not\Rightarrow x \models \varphi$

Exercise 3.4.5 (Modal equivalence does not imply bisimilarity). Denote by (X, R, V) and (X', R', V') the two models of Figure 3.1. The aim is to show that x and x' are modally equivalent. We need two definitions.

The depth of a modal formula counts the number of nested boxes, ie $depth(\bot) = depth(p) = 0$, $depth(\varphi \to \psi) = \max(depth(\varphi), depth(\psi))$, $depth(\Box\varphi) = depth(\varphi) + 1$.

Denote by $Cut(x, n)$ the model which is obtained from (X, R, V) by deleting all states which are not reachable from x in n or fewer than n steps. For example $Cut(x, 0)$ consists just of x. Similarly define $Cut'(x', n)$.

1. Show that $depth(\varphi) \leq n$ implies that $(Cut(x, n), x) \models \varphi \;\Leftrightarrow\; (X, R, V, x) \models \varphi$ and that $(Cut'(x', n), x') \models \varphi \;\Leftrightarrow\; (X', R', V', x') \models \varphi$.

2. Conclude that for all modal formulas $(X, R, V, x) \models \varphi \;\Leftrightarrow\; (X', R', V', x') \models \varphi$.

3.5 Notes

For background on modal logic the reader is referred to Chapter 2 and 3 of Blackburn, de Rijke, Venema [12]. We just note that bisimulation goes back, in its functional form, to Segerberg [65], and in its relational form to van Benthem [68]; Theorem 3.3.2 can be found in Barwise and Moss [10], Theorem 3.3.3 is due to Hennessy and Milner [23], and Theorem 3.3.4 to van Benthem [68, 69].

Chapter 4

Modal Logics for Coalgebras

The investigation of modal logics for coalgebras is still a young area of research. Since it is not in a definite shape yet, we will content ourselves to describe a few approaches.

We put emphasis on the theory of coalgebras as a general theory of systems. Not in the sense, of course, that it solves all problems concerning systems. But general in the sense that it offers tools that apply uniformly to a large class of systems. An obvious question from this perspective is, whether we can deal with logics for coalgebras in a uniform way. This question is of interest from a computer science point of view because coalgebras are systems and logics are specification languages.

This chapter presents some approaches which represent the tension between achieving uniformness and remaining close to the specific signature. The ideal description of a logic for coalgebras would work uniformly for all signatures and, at the same time, would reflect for each signature our intuition of coalgebras as transition systems.

The following three sections present three approaches that may be compared wrt this dilemma. The first solves the problem of being uniform but its syntax is neither familiar nor practical to work with. The second only works for specific signatures but its syntax is familiar modal logic. The third tries to get as much as possible from both approaches.

The order of the presentation reflects the historical development.

4.1 Coalgebraic Logic

Coalgebraic logic is an ingenious invention by Larry Moss. To appreciate it, before continuing to read on, try to think of a syntax and semantics of a logic for coalgebras working in a uniform way for all signatures $\Sigma : \mathsf{Set} \to \mathsf{Set}$. Moreover, formulas of the logic should be invariant under behavioural equivalence and the logic should be reasonably expressive. *Reasonably expressive* can be made precise by requiring that admitting infinite conjunctions, the logic should be able to characterise processes (elements of coalgebras) up to behavioural equivalence (compare with Theorem 3.3.2).

The aim is to find a language $\mathcal{L}_\Sigma$ and for each Σ-coalgebra (X, ξ) a relation $\models_\Sigma \subset X \times \mathcal{L}_\Sigma$ satisfying the above requirements. The starting point is that signatures are functors on

Set and may hence also be applied to sets of formulas $\mathcal{L}_\Sigma$ and relations $\models_\Sigma$.

We allow $\mathcal{L}_\Sigma$ and $\models_\Sigma$ to be proper classes. The category of classes is denoted by SET. Functors Σ on Set are extended to functors on SET via $\Sigma K = \bigcup\{\Sigma X : X \subset K, X \text{ a set}\}$ for classes K. Moreover, Σ is assumed to weakly preserve pullbacks.

Definition 4.1.1 (coalgebraic logic, syntax). $\mathcal{L}_\Sigma$ is defined to be the least class satisfying:

$$\begin{array}{lcl} \Phi \subset \mathcal{L}_\Sigma, \Phi \text{ a set} & \Longrightarrow & \bigwedge \Phi \in \mathcal{L}_\Sigma \\ \varphi \in \Sigma(\mathcal{L}_\Sigma) & \Longrightarrow & \varphi \in \mathcal{L}_\Sigma \end{array}$$

(That is, $\mathcal{L}_\Sigma$ is the initial algebra wrt the functor $\mathcal{P} + \Sigma$.) Due to the first clause $\bigwedge \emptyset$, denoted by *true*, is in $\mathcal{L}_\Sigma$ and $\mathcal{L}_\Sigma$ is a proper class. The last clause uses the fact that Σ is a functor on SET and can also be applied to classes of formulas.

Example 4.1.2. Let $\Sigma X = A \times X$ and $a_i \in A$. Then *true*, (a_0, true), $(a_0, (a_1, \mathit{true}))$, $\bigwedge\{(a_0, \ldots, a_n, \mathit{true}) : n \in \mathbb{N}\}$ are examples of formulas.

Exercise 4.1.3. Give examples of formulas for $\Sigma = O \times \mathrm{Id}^I$ (ie deterministic automata, see Chapter 1.3) and $\Sigma = \mathcal{P}$.

The semantics of coalgebraic logic goes as follows.

Definition 4.1.4 (coalgebraic logic, semantics). Given a coalgebra (X, ξ) define $\models_\Sigma \subset X \times \mathcal{L}_\Sigma$ as the least relation such that (let $x \in X$):

$$\begin{array}{lcl} x \models_\Sigma \varphi \text{ for all } \varphi \in \Phi, \Phi \subset \mathcal{L}_\Sigma, \Phi \text{ a set} & \Rightarrow & x \models_\Sigma \bigwedge \Phi \\ \text{there is } w \in \Sigma(\models_\Sigma) \text{ s.t. } \Sigma\pi_1(w) = \xi(x),\ \Sigma\pi_2(w) = \varphi & \Rightarrow & x \models_\Sigma \varphi \end{array}$$

where π_1, π_2 denote the projections from the product $X \times \mathcal{L}_\Sigma$ to its components.

The following exercise explains how this definition works in the example of streams. In particular, it shows that formulas are invariant under behavioural equivalence.

Exercise 4.1.5. Let $X \xrightarrow{\xi} A \times X$ and $x \in X$. Show that

$$x \models_{A\times\mathrm{Id}} (a_0, \ldots a_n, \mathit{true}) \quad \text{iff} \quad \mathit{head}(\mathit{tail}^i(x)) = a_i \ \text{ for all } \ 0 \leq i \leq n$$

where $\mathit{head} : X \to A$ and $\mathit{tail} : X \to X$ denote the components of ξ.

The following theorem summarises the main properties of coalgebraic logic. For proofs we refer to the original paper [44].

Theorem 4.1.6. *Let $\Sigma : \mathsf{Set} \to \mathsf{Set}$ be a functor weakly preserving pullbacks. Then*

1. *formulas of $\mathcal{L}_\Sigma$ are invariant under behavioural equivalence and*

2. *for each coalgebra (X, ξ) and each $x \in X$ there is a formula $\varphi_x \in \mathcal{L}_\Sigma$ such that for all coalgebras (X', ξ') and all $x' \in X'$*

$$x' \models_\Sigma \varphi_x \quad \textit{iff} \quad x, x' \textit{ behaviourally equivalent.}$$

This theorem shows that coalgebraic logic reflects precisely the notion of behavioural equivalence. Moreover, Exercise 4.4.1 shows that, eg in the case of $\Sigma = \mathcal{P}$, every formula of modal logic is equivalent to a formula of coalgebraic logic (with negation). We can therefore—neglecting syntactical 'details'—consider coalgebraic logic as modal logic. But what happened, then, to the modalities of modal logic? To explain this, let us take a closer look at coalgebraic logic in the case $\Sigma = \mathcal{P}$. Recall that formulas are either of the form $\bigwedge \Phi$ or—due to the second clause of Definition 4.1.1—of the form Φ for $\Phi \in \mathcal{P}\mathcal{L}_\mathcal{P}$. We take a look at the formulas of the second kind.[1]

Proposition 4.1.7. *Let $\Phi \in \mathcal{P}\mathcal{L}_\mathcal{P}$, $X \xrightarrow{\xi} \mathcal{P}X$, and $x \in X$. Then*

$$x \models_\mathcal{P} \Phi \;\Leftrightarrow\; \begin{cases} \forall y \in \xi(x) : \exists \varphi \in \Phi : y \models_\mathcal{P} \varphi \quad \text{and} \\ \forall \varphi \in \Phi : \exists y \in \xi(x) : y \models_\mathcal{P} \varphi \end{cases}$$

Using the modal operators $\Box, \Diamond$, this can be rewritten as

$$x \models \Box \bigvee \Phi \wedge \bigwedge \Diamond \Phi$$

where $\Diamond\Phi = \{\Diamond\varphi : \varphi \in \Phi\}$. This shows that the modalities are still there, but in 'some strange way'. Whether it is possible to extract the familiar modalities from a functor Σ will be addressed in Section 4.3.

4.2 Logics Designed for Specific Signatures

As we have seen, coalgebraic logic solves the problem of describing a logic which depends in a uniform way on the given signature. But in some other respects it is far away from what we are used to call modal logic. It is therefore natural to give up a bit of generality and see what can be achieved in concrete examples. The basic idea is simply to find a translation of coalgebras into transition systems and then to use standard modal logic as a logic for coalgebras.

Let us consider as an example the signatures of classes, see Chapter 1.4.1. Signatures are then functors

$$\Sigma X = \prod_{1 \le m \le n} (E_m + O_m \times X)^{I_m}. \tag{4.1}$$

[1] We write Φ instead of φ because, due to $\Sigma = \mathcal{P}$, the formula Φ is indeed a set of formulas. Unfortunately, this may cause confusion because $x \models \Phi$ could be understood as "$x \models \varphi$ for all $\varphi \in \Phi$" or as $x \models \Phi$ according to the second clause of Definition 4.1.4. It is the latter understanding of $x \models \Phi$ which is discussed in the following.

where for each method m the sets I_m, O_m, E_m denote the sets of inputs, outputs, exceptions, respectively.

It is now straight forward to translate coalgebras $X \to \Sigma X$ into Kripke models $(X, (R_a)_{a\in A}, V)$. To each method and each input corresponds a relation, ie $A = \{(m, i) : 1 \leq m \leq n, i \in I_m\}$. Atomic propositions are used to describe the outcomes of the methods, that is we put $\mathsf{Prop} = \{(m, i, o) : 1 \leq m \leq n, i \in I_m, o \in O_m\} \cup \{(m, i, e) : 1 \leq m \leq n, i \in I_m, e \in E_m\}$ and interpret the proposition (m, i, o) as 'method m applied to input i yields output o' and (m, i, e) as 'method m applied to input i yields exception e'.[2]

To discuss this translation in a bit more detail, let us denote by $K(\Sigma)$ the class of all Kripke models $(X, (R_a)_{a\in A}, X \xrightarrow{V} \mathcal{P}\mathsf{Prop})$ with A and Prop as described above. The translation of coalgebras to models will be denoted by

$$ck : \mathsf{Coalg}(\Sigma) \to K(\Sigma).$$

Of course, we are not interested in all of $K(\Sigma)$ but only in those models in $K(\Sigma)$ which are Σ-coalgebras. That is, we are interested in the image of ck. The idea is now

- to describe the image of ck, up to bisimulation, by modal formulas, and
- to add these formulas as axioms to a standard modal logic for $K(\Sigma)$.

which will result in a modal logic for the image of ck and hence for $\mathsf{Coalg}(\Sigma)$. Moreover, this logic will in general inherit good properties from the modal logic for $K(\Sigma)$ such as completeness and decidability (model checking). You can work this out as Exercise 4.4.2.

Of course, as described so far, this approach suffers from the fact that, although straight forward and easy, it only works for the signatures describing classes as in (4.1). It can be generalised, though, to all signatures which are defined inductively via

$$\Sigma ::= \mathrm{Id} \mid A \mid \Sigma \times \Sigma \mid \Sigma + \Sigma \mid \Sigma^A \mid \mathcal{P}\Sigma.$$

This line of research was pursued by Rößiger [56, 53, 54, 55].

4.3 Modalities from Functors

We have seen that coalgebraic logic solves the problem of a uniform approach to modal logics for coalgebras but that it has no modalities. And we have seen that for specific functors we can give a modal logic with modalities by translating them to Kripke models and then using standard modalities from modal logic. This section presents an approach with allows to extract the modalities from the functors Σ in a uniform way.

[2] A more detailed description of the translation of coalgebras into Kripke models can be found in Exercise 4.4.2, see properties (1)–(3).

4.3.1 Modalities Induced by Natural Transformations $\Sigma \to \mathcal{P}$

The beautiful insight, due to Pattinson [45], is that modalities (at least those studied so far for coalgebras) correspond to natural transformations $\Sigma \to \mathcal{P}$ (or to natural relations on Σ, see [45]). We first take a look at a few examples of natural transformations.

Exercise 4.3.1.

1. Show that $\mu : (E + O \times -) \to \mathcal{P}$ defined by

$$\begin{aligned} \mu_X : E + O \times X &\to \mathcal{P}X \\ e &\mapsto \emptyset \\ (o, x) &\mapsto \{x\} \end{aligned}$$

   is a natural transformation (where $e \in E$, $o \in O$, $x \in X$).

2. Show that, for each $i \in I$, $\mu[i] : O \times (-)^I \to \mathcal{P}$ defined by

$$\begin{aligned} \mu[i]_X : O \times X^I &\to \mathcal{P}X \\ (o, f) &\mapsto \{f(i)\} \end{aligned}$$

   is a natural transformation.

3. Let $\Sigma X = \prod_{1 \le m \le n} (E_m + O_m \times X)^{I_m}$. Show that for all $1 \le m \le n$ and $i_m \in I_m$

$$\mu[m, i] : \Sigma \to \mathcal{P}$$

   defined by

$$\begin{aligned} \mu[m, i]_X : \Sigma X &\to \mathcal{P}X \\ \langle f_1, \dots f_n \rangle &\mapsto \begin{cases} \{x\} \ \text{ if } \ f_m(i) = (o, x) \ \text{ for some } o \in O_m \\ \emptyset \ \text{ otherwise} \end{cases} \end{aligned}$$

   is a natural transformation (where $1 \le m \le n$ and $i \in I_m$).

Remark 4.3.2. Note how the natural transformations μ of the exercise above extract from ΣX the successors. In detail, $\xi : X \to \Sigma X$ being a coalgebra for the signatures (1) to (3), respectively:

1. $\mu_X(\xi(x_0))$ is the empty set if x_0 has no successor and otherwise the singleton containing the unique successor of x_0.

2. $\mu[i]_X(\xi(x_0))$ gives the successor of x_0 obtained on input i.

3. $\mu[m,i]_X(\xi(x_0))$ gives the successor of x_0 obtained by applying method m with input i to state x_0.

Also note that, in case (3), $\mu[m,i]_X \circ \xi$ gives precisely the relation $R_{(m,i)}$ which was discussed in the previous section as the relation corresponding in a Kripke model to a Σ-coalgebra (X,ξ) (see also Exercise 4.4.2).

That only singletons $\{x\}$ appear on the right hand side is of course due to the systems being deterministic.

Exercise 4.3.3. Show that $\mu[a] : \mathcal{P}(A \times -) \to \mathcal{P}$ defined for all $a \in A$ by

$$\begin{aligned} \mu[a]_X : \mathcal{P}(A \times X) &\to \mathcal{P}X \\ Q &\mapsto \{x : (a,x) \in Q\} \end{aligned}$$

is a natural transformation.

We have seen that for a system $\xi : X \to \Sigma X$ and a natural transformation $\mu : \Sigma \to \mathcal{P}$, the μ-successors of $x \in X$ are given by $\mu_X(\xi(x))$. The semantics of modal operators, one for each natural transformation μ, can now be defined as usual for all $x \in X$

$$x \models \Box_\mu \varphi \quad \text{iff} \quad y \models \varphi \text{ for all } y \in \mu_X(\xi(x)) \tag{4.2}$$

or, equivalently, but of use below

$$[\![\Box_\mu \varphi]\!] = \xi^{-1}(\mu_X^-([\![\varphi]\!])) \tag{4.3}$$

where $[\![\varphi]\!]$, called the extension of φ, is $\{x \in X : x \models \varphi\}$ and μ^- is defined via[3]

$$\begin{aligned} \mu_X^- : 2^X &\to 2^{\Sigma X} \\ P &\mapsto \{s \in \Sigma X : \mu_X(s) \subset P\}. \end{aligned} \tag{4.4}$$

[*Comment on notation:* It would be more precise to write $(X,\xi,x) \models \varphi$ instead of $x \models \varphi$ (cf Definition 3.1.4) and $[\![\varphi]\!]_{(X,\xi)}$ instead of $[\![\varphi]\!]$ but we suppress (X,ξ) when convenient.]

What is the point of requiring the transformations μ to be natural? It is precisely this requirement, what makes in the definition above $\Box_\mu$ into a *modal* operator. This is the contents of the next proposition.

Proposition 4.3.4. *Let $\Sigma : \mathsf{Set} \to \mathsf{Set}$ and $\mu : \Sigma \to \mathcal{P}$ a natural transformation. If φ is invariant under behavioural equivalence then so is $\Box_\mu \varphi$.*

[3] Recall that 2^X is the set of functions $X \to 2$ for some fixed two-element set 2, ie $2^X \simeq \mathcal{P}X$ is the set of subsets of X. Here, we prefer the notation 2^X since $2^{(-)}$ is the contravariant functor mapping $f : X \to Y$ to the inverse image $f^{-1} = 2^f : 2^Y \to 2^X$.

Proof. Consider a coalgebra (X,ξ) and let (Z,ζ) be its behaviour. First note that any subset $U \subset X$ of a coalgebra (X,ξ) is invariant under behavioural equivalence iff there is $V \subset Z$ such that $U = !^{-1}(V)$ where ! is the unique morphism into (Z,ζ). Now consider

$$\begin{tikzcd}
2^X & 2^{\Sigma X} \arrow[l, "\xi^{-1}"'] & 2^X \arrow[l, "\mu^-_X"'] \\
2^Z \arrow[u, "!^{-1}"] & 2^{\Sigma Z} \arrow[l, "\zeta^{-1}"] \arrow[u, "(\Sigma !)^{-1}"'] & 2^Z \arrow[l, "\mu^-_Z"] \arrow[u, "!^{-1}"']
\end{tikzcd}$$

The left-hand square commutes since ! is a morphism and $2^{(-)} = (-)^{-1}$ is a functor. The right-hand square commutes since μ^- is natural (Exercise 4.4.3). Now, consider $!(\llbracket\varphi\rrbracket)$ as an element of the lower right corner 2^Z. Since φ is invariant, ie $!^{-1}(!(\llbracket\varphi\rrbracket)) = \llbracket\varphi\rrbracket$, going up and left gives us $\llbracket\Box_\mu\varphi\rrbracket$ (see (4.3)). And going left and up shows that $\llbracket\Box_\mu\varphi\rrbracket = !^{-1}(V)$ for some $V \subset Z$. $\Box$

Having seen how to obtain modalities from natural transformations $\Sigma \to \mathcal{P}$ we use the same idea for atomic propositions. Given Σ, we call atomic propositions any set Prop together with a natural transformation $\nu : \Sigma \to \mathcal{P}\mathsf{Prop}$.[4] We then define as usual (compare with first clause of Definition 3.1.4) for any $x \in X$:

$$x \models p \quad \text{iff} \quad p \in (\nu_X \circ \xi)(x) \tag{4.5}$$

We have seen that modal operators and atomic propositions arise from natural transformations. Suppose we have chosen a set $\{\mu[a] : a \in A\}$ of natural transformations $\Sigma \to \mathcal{P}$ and a natural transformation $\nu : \Sigma \to \mathcal{P}\mathsf{Prop}$ of atomic propositions. Is there a condition telling us that we chose enough $\mu[a]$ and atomic propositions in order to get a reasonably[5] expressive modal logic? A sufficient condition is: To be able to embed Σ-coalgebras into Kripke models, ie into $\prod_{a\in A}\mathcal{P} \times \mathcal{P}\mathsf{Prop}$ coalgebras.[6] Stated more formally, the condition becomes: The natural transformation induced by the $\mu[a]$, $a \in A$, and ν

$$\Sigma \longrightarrow \prod_{a\in A}\mathcal{P} \times \mathcal{P}\mathsf{Prop}$$

is injective. The proof that this condition implies that the corresponding logic is reasonable expressive is the main result of [45].

[4] The notation $\mathcal{P}\mathsf{Prop}$ corresponds to the atomic propositions part of the signature for Kripke models $\Sigma = \mathcal{P} \times \mathcal{P}\mathsf{Prop}$ and is appropriate since we consider Prop to be a given constant. In case we would like to compare Kripke models with different sets of atomic propositions, we should rather use signatures $\Sigma = \mathcal{P} \times 2^{\mathsf{Prop}}$ and natural transformations $\Sigma \to 2^{\mathsf{Prop}}$, see [35].

[5] Again, 'reasonably' expressive can be made precise by requiring that if infinitary conjunctions are allowed then formulas should characterise elements of coalgebras up to behavioural equivalence.

[6] Recall that $\prod_{a\in A}\mathcal{P} \simeq \mathcal{P}^A \simeq \mathcal{P}(A \times -)$.

To summarise, we have seen how natural transformations can be used to extract modalities from functors. Comparing with the previous section, the next challenge would be to ask whether it is possible to find axioms in a uniform way which characterise the image of the embedding of coalgebras into Kripke frames. Another open question is whether the approach can be extended to cover all (weakly pullback preserving) functors Σ.

4.3.2 Modalities Induced by Predicate Liftings

Going back to the explanation of modal operators as induced by natural transformations $\mu : \Sigma \to \mathcal{P}$, we see that for the proof of Proposition 4.3.4 only the naturality of $\mu^- : 2^{\mathrm{Id}} \to 2^{\Sigma}$ is required. It is therefore an obvious idea to consider modalities induced by natural transformations $2^{\mathrm{Id}} \to 2^{\Sigma}$. Natural transformations $2^{\mathrm{Id}} \to 2^{\Sigma}$ can be considered as predicate liftings in the sense that $2^X \to 2^{\Sigma X}$ is a lifting of Σ from an operation on carrier sets X to an operation on predicates $P \in 2^X$ on X.

Having said that we want to consider modal operators induced by natural transformations $2^{\mathrm{Id}} \to 2^{\Sigma}$, what about atomic propositions? There are slight variations possible (see Exercise 4.4.4) and we propose the following. Atomic propositions $p \in \mathsf{Prop}$ are induced by natural transformations

$$\hat{p} : \Sigma \to 2$$

where $2 = \{\mathit{true}, \mathit{false}\}$. We then define

$$x \models p \quad \text{iff} \quad (\hat{p}_X \circ \xi)(x) = \mathit{true} \tag{4.6}$$

which is equivalent to

$$[\![p]\!] = \xi^{-1}(\hat{p}_X). \tag{4.7}$$

[Note that $\hat{p}_X \in 2^{\Sigma X}$ and recall $\xi^{-1} : 2^{\Sigma X} \to 2^X$.]

To summarise, given a set $\mathcal{M}$ of natural transformations $2^{\mathrm{Id}} \to 2^{\Sigma}$ and a set Prop of natural transformations $\Sigma \to 2$, we obtain a language $\mathcal{L}(\mathcal{M}, \mathsf{Prop})$ as laid out in

Definition 4.3.5 (Syntax and semantics of $\mathcal{L}(\mathcal{M}, \mathsf{Prop})$). Suppose $\Sigma : \mathsf{Set} \to \mathsf{Set}$, $\mathcal{M}$ a set of natural transformations $2^{\mathrm{Id}} \to 2^{\Sigma}$, and Prop a set of natural transformations $\Sigma \to 2$. Then the language $\mathcal{L}(\mathcal{M}, \mathsf{Prop})$ is the least set containing $\perp$ and p for all $p \in \mathsf{Prop}$ and closed under implication $\to$ and containing the formula $\Box_\mu \varphi$ for every $\mu \in \mathcal{M}$ and every $\varphi \in \mathcal{L}(\mathcal{M}, \mathsf{Prop})$.

Given a coalgebra (X, ξ), the semantics for boolean operators is as usual and for atomic propositions and modal operators as follows

$$[\![p]\!] = \xi^{-1}(p_X),$$
$$[\![\Box_\mu \varphi]\!] = \xi^{-1}(\mu_X([\![\varphi]\!]))$$

where as above $[\![\varphi]\!] = \{x \in X : x \models \varphi\}$.

Remark 4.3.6. This definition is from [35], with the difference that there atomic propositions $p : \Sigma \to 2$ are subsumed under predicate liftings $2^{\mathrm{Id}} \to 2^{\Sigma}$ (see Exercise 4.4.4) which yields a more concise definition but does not respect the traditional way of presenting modal logic.

We want to close this section with some background on predicate liftings and a comparison of natural transformations $2^{\mathrm{Id}} \to 2^{\Sigma}$ with the original notion of predicate lifting. Predicate liftings were introduced by Hermida and Jacobs [24] and formulated in a general setting called categorical logic which we try to sketch briefly in the following.[7] Consider a category $\mathcal{X}$ of 'types' and a category $\mathcal{E}$ of 'predicates'. Predicates and types are linked by a functor $p : \mathcal{E} \to \mathcal{X}$ which provides for each $X \in \mathcal{X}$ the category $p^{-1}(X)$ of predicates on X. Endowing p with appropriate structure, one can give an account of formulas (= predicates = objects in $\mathcal{E}$) and structures (= types = objects in $\mathcal{X}$) in a categorical framework, allowing a unified model theoretic treatment of a great variety of logics and type theories. In the following, only the simple (although paradigmatic) example below will be of interest.

Example 4.3.7 ($p : \mathsf{SubSet} \to \mathsf{Set}$). Define SubSet to be the following category: objects are pairs of sets (P, X) with $P \subset X$, morphisms $f : (P, X) \to (Q, Y)$ are functions $f : X \to Y$ with $f(P) \subset Q$ (equivalently $P \subset f^{-1}(Q)$). An object (P, X) is said to be a predicate P on X. Morphisms are—up to renaming via a function f—inclusions: the identity id_X is a morphism $\mathrm{id}_X : (P, X) \to (P', X)$ iff $P \subset P'$ which we read as 'P implies P''. Define the functor $p : \mathsf{SubSet} \to \mathsf{Set}$ as second projection on objects and on morphisms as mapping $f : (P, X) \to (Q, Y)$ to $f : X \to Y$. $\square$

The general definition of a predicate lifting involves the notion of a fibration (eg $p : \mathsf{SubSet} \to \mathsf{Set}$ is a fibration) and a fibred functor. It would lead us too far to define these notions here and we restrict attention to the special case of the example above.

Definition 4.3.8 (Predicate lifting). A predicate lifting of $\Sigma : \mathsf{Set} \to \mathsf{Set}$ is a functor $\bar{\Sigma} : \mathsf{SubSet} \to \mathsf{SubSet}$ such that the diagram

$$\begin{tikzcd} \mathsf{SubSet} \arrow[r, "\bar{\Sigma}"] \arrow[d, "p"'] & \mathsf{SubSet} \arrow[d, "p"] \\ \mathsf{Set} \arrow[r, "\Sigma"'] & \mathsf{Set} \end{tikzcd}$$

commutes. This means, in particular, that $\bar{\Sigma}$ maps a predicate P on X to a predicate $\bar{\Sigma}P$ on ΣX. Moreover, for any morphism $f : X \to Y$ in Set and any predicate Q on Y, it holds

$$\bar{\Sigma}(f^{-1}(Q)) = (\Sigma f)^{-1}(\bar{\Sigma}Q) \tag{4.8}$$

(recall that $f^{-1}(Q)$ is a predicate on X and $\bar{\Sigma}Q$ is a predicate on ΣY).

[7] For introductions to categorical logic see eg Lambek and Scott [41], Pitts [46] and Jacobs [29]. The latter also contains an introduction to fibrations.

This definition spells out what it means for $\bar{\Sigma}$ to be a fibred functor over Σ in case of the fibration $p : \mathsf{SubSet} \to \mathsf{Set}$.[8] The diagram illustrates why $\bar{\Sigma}$ is called a lifting of Σ. The next proposition relates natural transformations $2^{\mathrm{Id}} \to 2^{\Sigma}$ and predicate liftings for the fibration $p : \mathsf{SubSet} \to \mathsf{Set}$. We first need a definition.

We call a natural transformation $2^{\mathrm{Id}} \to 2^{\Sigma}$ *monotone* if $P \subset Q \subset X$ implies $\mu_X(P) \subset \mu_X(Q)$. The natural transformations $\mu^{-} : 2^{\mathrm{Id}} \to 2^{\Sigma}$ arising from natural transformations $\mu : \Sigma \to \mathcal{P}$ are monotone. For an example of a non-monotone natural transformation consider $\Sigma = \mathrm{Id}$ and $\neg : 2^{\mathrm{Id}} \to 2^{\mathrm{Id}}$ defined as complementation $\neg_X(P) = X - P$.

Proposition 4.3.9. *Let $\Sigma : \mathsf{Set} \to \mathsf{Set}$ and consider $p : \mathsf{SubSet} \to \mathsf{Set}$. There is a bijection between monotone natural transformations $2^{Id} \to 2^{\Sigma}$ and predicate liftings of Σ.*

Proof. First note that naturality of a transformation $\mu : 2^{\mathrm{Id}} \to 2^{\Sigma}$ means that $\mu_X(f^{-1}(Q)) = (\Sigma f)^{-1}(\mu_Y(Q))$ for all $f : X \to Y$ and $Q \subset Y$; compare this with (4.8).
Given $\mu : 2^{\mathrm{Id}} \to 2^{\Sigma}$, define $\bar{\Sigma}(P, X) = (\mu_X(P), \Sigma X)$. For a morphism $f : (P, X) \to (Q, Y)$ define $\bar{\Sigma} f = \Sigma f : (\mu_X(P), \Sigma X) \to (\mu_Y(Q), \Sigma Y)$ which is indeed a morphism in SubSet since μ is natural and monotone. Now check that the conditions of Definition 4.3.8 are satisfied.
Conversely, given $\bar{\Sigma}$, we write $(\bar{\Sigma} P, \Sigma X)$ for $\bar{\Sigma}(P, X)$. Define $\mu_X(P) = \bar{\Sigma} P$. μ is monotone since $\bar{\Sigma}$ is a functor and natural due to (4.8). □

Let us remark that the notion of a predicate lifting is more general than that of a monotone natural transformation $2^{\mathrm{Id}} \to 2^{\Sigma}$ since it can be stated for arbitrary fibrations.

4.4 Exercises

Exercise 4.4.1. Suppose we added negation to coalgebraic logic $\mathcal{CL}$. Show that in case $\Sigma = \mathcal{P}$

$$(\Box\varphi)^{\dagger} = \{\varphi^{\dagger}\} \vee \{\}$$

gives a translation $(-)^{\dagger} : \mathcal{ML} \to \mathcal{CL}$ from modal logic into coalgebraic logic which preserves and reflects satisfaction of formulas. [Hint: Use Proposition 4.1.7].

Exercise 4.4.2. Consider a signature Σ and the translation $ck : \mathsf{Coalg}(\Sigma) \to K(\Sigma)$ as in Section 4.2. The aim is to describe the image of ck in $K(\Sigma)$ by modal formulas.

Since methods are functions, a model $(X, (R_a)_{a \in A}, X \xrightarrow{V} \mathcal{P}\mathsf{Prop})$ in the image of ck has the following properties (recall A and Prop from Section 4.2 and let $x \in X$):

1. $(m, i, e) \in V(x) \;\Rightarrow\; x$ has no $R_{(m,i)}$-successor,
2. $(m, i, o) \in V(x) \;\Rightarrow\; x$ has precisely one $R_{(m,i)}$-successor,

[8] Readers familiar with fibrations will note that condition (4.8) expresses preservation of cartesian liftings.

3. $V(x)$ contains precisely n proposition, one for each method.

Writing the boxes of the modal logic as $[m,i]$ and the diamonds as $\langle m,i\rangle$, consider the following axiom schemes (assume E_m and O_m finite):

$$
\begin{array}{lll}
\text{(Ax1)} & (m,i,e) \to \neg\langle m,i\rangle\top & \\
\text{(Ax2a)} & (m,i,o) \to \langle m,i\rangle\top & \\
\text{(Ax2b)} & \langle m,i\rangle\varphi \to [m,i]\varphi & \\
\text{(Ax3a)} & (m,i,d) \to \neg(m,i,d') & \text{for all } d \neq d', d, d' \in E_m + O_m \\
\text{(Ax3b)} & \bigvee_{d\in E_m+O_m}(m,i,d) &
\end{array}
$$

and call Φ all modal formulas (in the language given by A and Prop) which are instances of one of the schemes.

Now show the following (point (2) requires knowledge about completeness in modal logic which was not presented in Chapter 3):

1. If a model $K(\Sigma)$ satisfies Φ then it is bisimilar (behaviourally equivalent) to a model in the image of ck. [Hint: Compare Table 3.1. The only axiom that requires a bit of work is (Ax2b) because it does not define determinism on *models* (but on frames).]

2. Φ provides a complete axiomatisation of the image of ck and hence of $\mathsf{Coalg}(\Sigma)$.

3. Modal equivalence implies bisimilarity. [Hint: Models are image-finite]

4. The canonical model is the final coalgebra. [Hint: Use (3).]

Exercise 4.4.3 (μ^-). Consider $\mu^- : 2^{\mathrm{Id}} \to 2^{\Sigma}$ as defined in (4.4) in Section 4.3.

1. Show (4.2) $\Leftrightarrow$ (4.3).

2. Show that $\mu : \Sigma \to \mathcal{P}$ natural implies that $\mu^- : 2^{\mathrm{Id}} \to 2^{\Sigma}$ is natural.

Exercise 4.4.4 (Atomic Propositions). The aim of this easy exercise is to compare different formalisations of atomic propositions as natural transformations.

1. Let $p : \Sigma \to 2$ be a natural transformation. Use

$$
\begin{tikzcd}
\Sigma X \arrow[r, "p_X"] \arrow[d, "\Sigma!"'] & 2 \arrow[d, "\mathrm{id}"] \\
\Sigma 1 \arrow[r, "p_1"] & 2
\end{tikzcd}
$$

to show that natural transformations $\Sigma \to 2$ are in bijection with subsets of $\Sigma 1$. This can be interpreted as follows: *The atomic propositions are precisely those observations obtained by abstracting away from the state space X.* Replacing X with the one-element set 1, the remaining observations are the same for all coalgebras, that is ‘atomic’.

2. Show that natural transformations $\nu : \Sigma \to \mathcal{P}\mathsf{Prop}$ are in bijection to families of natural transformations $(\hat{p} : \Sigma \to 2)_{p\in\mathsf{Prop}}$. [Hint: Use the bijections $\mathcal{P}\mathsf{Prop} \simeq 2^{\mathsf{Prop}} \simeq \prod_{\mathsf{Prop}} 2$.]

3. Show that a natural transformation $\Sigma \to 2$ can be considered as natural transformation $1 \to 2^{\Sigma}$ which in turn is a special case of a natural transformation $2^{\mathrm{Id}} \to 2^{\Sigma}$.

Exercise 4.4.5 (Atomic Propositions). Show that as for modal operators it is the naturality condition that guarantees that the evaluation of atomic propositions is invariant under behavioural equivalence.

4.5 Notes

The relationship of coalgebras and modal logic goes back to Barwise and Moss [10] where both topics appear together. Coalgebraic logic is due to Moss [44]. The modal logic for the signatures as classes is from [40]. The more general and difficult case of modal logics for inductively defined signatures was developed by Rößiger [56, 53, 54, 55], but see also Jacobs [32]; Jacobs [30] shows how temporal operators can be treated. The approach of obtaining modalities from functors via natural transformations is due to Pattinson [45].

Chapter 5

Duality of Modal and Equational Logic

We have seen so far that the theory of coalgebras provides us in a uniform way with a notion of behavioural equivalence for a large number of different types of systems. And we argued that modal logics are natural logics for coalgebras because they respect this notion of behavioural equivalence.

Being convinced that (variants of) modal logics are the natural logics for coalgebras, we want to answer in this chapter the question whether it is possible to make precise the intuition that

modal logic is to coalgebras what
equational logic is to algebras.

We will argue for a positive answer by showing that, up to logical equivalence, ie from a semantic point of view,

modal logic is dual to equational logic.

5.1 Preliminaries

(The basic notions needed in this chapter are recalled.)

For a functor Σ on a category $\mathcal{X}$, we denote the category of Σ-coalgebras by $\mathsf{Coalg}(\Sigma)$. We assume that $\mathsf{Coalg}(\Sigma)$ has cofree coalgebras. In the case of $\mathcal{X} = \mathsf{Set}$, if we allow coalgebras to have classes as carriers, we know by Aczel and Mendler's theorem that cofree coalgebras exist for all functors Σ.

We also assume that for each class of Σ-morphisms $s_i : A_i \to A$, $i \in I$, there exist the union of images (Chapter 2.2.3)

$$A_i \xrightarrow{e_i} \bigcup\{\mathrm{Im}(s_i) : i \in I\} \xrightarrow{m} A$$

(Σ, C)-coalgebras (Definition 2.2.13) are pairs consisting of a Σ-coalgebra and a valuation (colouring)

$$(A\ ,\ UA \xrightarrow{c} C).$$

A cofree coalgebra FC over colours C comes together with a colouring $\varepsilon_C : UFC \to C$. Recall that

$$(FC\ ,\ UFC \xrightarrow{\varepsilon_C} C)$$

is the final (Σ, C)-coalgebra. We also assume that $\mathcal{X}$ has a final object 1 which implies that $F1$ is the final Σ-coalgebra.

In case $\Sigma = \mathcal{P}$ and $C = \mathcal{P}\mathsf{Prop}$ Kripke frames are Σ-coalgebras and Kripke models are (Σ, C)-coalgebras. Recall that (Σ, C)-coalgebras can also be considered as $\Sigma \times C$-coalgebras.

5.2 Modal Formulas as Subcoalgebras

This section concentrates on the case $\mathcal{X} = \mathsf{Set}$. We can summarise the essence of the relation of coalgebras and modal logic studied in Chapters 3 and 4 as follows.

Assume a signature Σ over sets and a class of formulas $\mathcal{L}$. We write A for Σ-coalgebras, v for colourings $UA \to C$, a for elements of A. We call $\varphi \in \mathcal{L}$ **modal formulas for Σ-coalgebras in colours from C** iff there is a relation $\models$ of type $A, v, a \models \varphi$ such that

$$\text{formulas are invariant under } (\Sigma, C)\text{-behavioural equivalence.} \tag{5.1}$$

This means that for Σ-coalgebras A, A' and colourings $c : UA \to C$, $c' : UA' \to C$ and a morphism $f : A \to A'$ respecting the colourings (ie $c' = Uf \circ c$) it holds that

$$A, c, a \models \varphi \ \Leftrightarrow\ A', c', f(a) \models \varphi$$

for all $a \in UA$.

$\models$ gives rise to a satisfaction relation for (Σ, C)-coalgebras and for Σ-coalgebras via

$$A, v \models \varphi \quad \Longleftrightarrow \quad A, v, a \models \varphi \quad \forall a \in UA \tag{5.2}$$
$$A \models \varphi \quad \Longleftrightarrow \quad A, v \models \varphi \quad \forall v : UA \to C \tag{5.3}$$

We will now show that for any logic satisfying (5.1)–(5.3), we can characterise formulas—up to logical equivalence—as subcoalgebras of cofree coalgebras. First note that (5.1) and (5.2) imply that modal formulas are preserved under quotients, domains of morphisms, and unions of (Σ, C)-coalgebras.[1]

[1]This is a classical result in modal logic, see Proposition 3.2.6. (Domains of morphisms corresponds to domains of quotients and submodels; disjoint unions are a special case of unions; conversely, unions can be obtained as disjoint unions and quotients.)

Lemma 5.2.1 (Preservation modal formulas). *Assume a modal formula φ in colours from C.*

1. *If there is a (Σ, C)-morphism $(A', v') \to (A, v)$ then $A, v \models \varphi \;\Rightarrow\; A', v' \models \varphi$.*
2. *If there is a surjective (Σ, C)-morphism $(A, v) \to (A', v')$ then $A, v \models \varphi \;\Rightarrow\; A', v' \models \varphi$.*
3. *Assume a (Σ, C)-coalgebra (B, w), and a family of (Σ, C)-subcoalgebras $(A_i, v_i) \overset{i}{\hookrightarrow} (B, w)$, $i \in I$. Let (A', v') be the union of all (A_i, v_i), $i \in I$. Then $A_i, v_i \models \varphi$ for all $i \in I$ implies that $A', v' \models \varphi$.*

Proof. We show (3), (1) and (2) are similar. Assume $a' \in UA'$. Since (A', v') is the union of the (A_i, v_i), there is $j \in I$ such that $a' \in UA_j$. It follows now from $A_j, v_j, a' \models \varphi$ and (5.1) that $A', v', a' \models \varphi$. □

We first treat the case of formulas without propositional variables. In that case, formulas correspond to subcoalgebras of the final coalgebra.

Let $F1$ be the final Σ-coalgebra and φ a formula. Define $F1|\varphi$ to be the largest subcoalgebra of $F1$ satisfying φ, ie the union of all subcoalgebras satisfying φ (see Chapter 2.2.3). Let us denote by m_φ the inclusion $F1|\varphi \hookrightarrow F1$. Now, satisfaction can be characterised as follows.

Proposition 5.2.2. *$A \models \varphi$ iff the (unique) morphism $f : A \to F1$ factors through m_φ*

$$\begin{tikzcd} F1 & F1|\varphi \arrow[l, "m_\varphi"'] \\ & A \arrow[ul, "f"] \arrow[u, dotted] \end{tikzcd}$$

Proof. 'if': First note that it follows from Lemma 5.2.1.2 and 5.2.1.3 that $F1|\varphi \models \varphi$. Now suppose that f factors. Then Lemma 5.2.1.1 implies $A \models \varphi$. 'only if': $A \models \varphi$ implies that the image of f satisfies φ. Therefore f factors through m_φ by definition of $F1|\varphi$. □

The case of formulas with colours (propositional variables) is a bit more complicated but similar: Recall that the cofree Σ-coalgebra FC is the final (Σ, C)-coalgebra.

Definition 5.2.3 (Subcoalgebra corresponding to a formula). Let Σ be a signature on sets and φ a modal formula for Σ-coalgebras in colours C. Define $FC|\varphi$ to be the largest (Σ, C)-subcoalgebra satisfying φ, ie

$$FC|\varphi = \bigcup \{\mathrm{Im}(v^\sharp) : A, v \models \varphi\}$$

Denote by m_φ the inclusion $FC|\varphi \hookrightarrow FC$.

Proposition 5.2.4. *$A \models \varphi$ iff all morphisms $f : A \to FC$ factor through m_φ*

$$\begin{tikzcd} FC & FC|\varphi \arrow[l, "m_\varphi"'] \\ & A \arrow[ul, "f"] \arrow[u, dotted, "g"'] \end{tikzcd}$$

Proof. 'if': First note that we have $FC|\varphi, \varepsilon_C \circ m_\varphi \models \varphi$ by Lemma 5.2.1.2 and 5.2.1.3. Now let $v : UA \to C$. Since FC is cofree there is $v^\sharp : A \to FC$ with $\varepsilon_C \circ Uv^\sharp = v$. Since $v^\sharp$ factors, it follows $A, v \models \varphi$ by Lemma 5.2.1.1.

'only if': Let $f : A \to FC$. Note that f induces a colouring $v = \varepsilon_C \circ f$ on A and that $v^\sharp = f$. Now, $A \models \varphi$ implies $A, v \models \varphi$. Therefore $f = v^\sharp$ factors through m_φ by definition of $FC|\varphi$. □

The proposition shows that satisfaction of modal formulas can be characterised algebraically by *projectivity*:

Definition 5.2.5 (Projective). We say that A is projective wrt $B_1 \xrightarrow{m} B_0$ iff all $f : A \to B_0$ factor through m, ie iff for all $f : A \to B_0$ there is $f' : A \to B_1$ such that

$$\begin{tikzcd} B_0 & B_1 \arrow[l, "m"'] \\ & A \arrow[ul, "f"] \arrow[u, dotted, "f'"'] \end{tikzcd}$$

commutes.

Remark 5.2.6. In Definition 5.2.3 we defined the subcoalgebra corresponding to a formula. There are two variations possible.

1. One can work with the largest sub*set* $[\![\varphi]\!]^{FC} = \{x \in UFC : FC, \varepsilon_C, x \models \varphi\}$ satisfying φ. The analogue to Proposition 5.2.4 then goes as follows. $A \models \varphi$ iff for all morphisms $f : A \to FC$

$$\begin{tikzcd} UFC & {[\![\varphi]\!]^{FC}} \arrow[l, hook'] \\ & UA \arrow[ul, "Uf"] \arrow[u, dotted] \end{tikzcd}$$

   Uf factors through $[\![\varphi]\!]^{FC} \hookrightarrow UFC$.[2]

[2] Another variation would be to use $[\![\varphi]\!]^{FC} = \{x \in UFC : FC, x \models \varphi\}$ as in [38].

2. Whereas Definition 5.2.3 puts $FC|\varphi = \bigcup\{\mathrm{Im}(f) : A, \varepsilon_C \circ f \models \varphi \text{ and } f : A \to FC\}$, one can also work with the largest *invariant* subcoalgebra defined as

$$\overline{FC|\varphi} = \bigcup\{\mathrm{Im}(f) : A \models \varphi \text{ and } f : A \to FC\}.$$

Proposition 5.2.4 also holds when we substitute $\overline{FC|\varphi}$ for $FC|\varphi$. Intuitively, whereas $(FC|\varphi, \varepsilon_C \circ Um_\varphi)$ is the largest Kripke sub*model* of (FC, ε_C) satisfying φ, $\overline{FC|\varphi}$ is the largest Kripke sub*frame* of FC satisfying φ.

A subcoalgebra $m : A' \to A$ is called *invariant* iff A' is projective wrt m, or in a more logical notation, iff $A' \models m$. That $\overline{FC|\varphi}$ is invariant means that $\overline{FC|\varphi} \models \varphi$. Note that, in general, $FC|\varphi \models \varphi$ does *not* hold.

We conclude that invariant subcoalgebras of coalgebras cofree over C and modal formulas in colours C are, up to logical equivalence of formulas, in a one-to-one correspondence.

The three levels of subsets, submodels, and subframes correspond to the three levels of satisfaction relations $A, v, a \models \varphi$, $A, v \models \varphi$, and $A \models \varphi$. Logical operators as conjunction (intersection) or modal operators (see Chapter 4.3) are treated on the level of subsets. The semantics via projectivity is more naturally formulated on the level of models or frames. □

To summarise this section, we have seen that to each modal formula φ corresponds a subcoalgebra m_φ with cofree codomain such that satisfaction of φ is projectivity wrt m_φ. If the converse holds, namely that every subcoalgebra of a cofree coalgebra with cofree codomain corresponds to a formula, we say that the modal logic is **expressive**.

The import of the correspondence of formulas and subcoalgebras is that

- it allows to treat all logics for coalgebras we have seen or mentioned in the previous chapters in a uniform way,
- it is abstract and hence easy to work with technically,[3]
- it precisely dualises the satisfaction of equations for algebras, see below.

5.3 Equations as Quotients

As noted already, to give an account on the duality of equational logic and modal logic, we cannot restrict our attention to algebras over Set. Can we give an account on equations which does not rely on $\mathcal{X} = \mathsf{Set}$? Yes, and it goes as follows.

For each set of equations $\Phi \subset UFX \times UFX$ we can form the quotient

$$FX \xrightarrow{e_\Phi} FX/\Phi$$

[3] See for example the proof of theorem 5.5.2.

of FX wrt the smallest congruence relation generated by Φ. Now, satisfaction can be characterised as follows

Proposition 5.3.1. *$A \models \Phi$ iff all $f : FX \to A$ factor through e_Φ:*

$$\begin{tikzcd} FX \arrow[r, "e_\Phi"] \arrow[dr, "f"'] & FX/\Phi \arrow[d, dotted, "f'"] \\ & A \end{tikzcd}$$

Proof. First show that

$$f : FX \to A \text{ factors through } e_\Phi \quad \text{iff} \quad \forall t, t' \in UFX : e_\Phi(t) = e_\Phi(t') \ \Rightarrow\ f(t) = f(t') \qquad (5.4)$$

For " $\Rightarrow$ " of the Proposition consider $f : FX \to A$. Note that courtesy of $F \dashv U$, there is $v : X \to UA$ such that $v^\sharp = f$. To profit from (5.4) assume $e_\Phi(t) = e_\Phi(t')$, that is, (t, t') are in the in the smallest congruence generated by Φ. It follows from our assumption $A \models \Phi$ that $A, v \models \Phi$ and hence $v^\sharp(t) = v^\sharp(t')$.[4] That is, $f(t) = f(t')$ and by (5.4) f factors through e_Φ.
For " $\Leftarrow$ " of the Proposition, let $(t, t') \in \Phi$ and $v : X \to UA$. Since $v^\sharp$ factors through e_Φ, it follows from (5.4) that $v^\sharp(t) = v^\sharp(t')$, hence $A, v \models (t, t')$. ☐

Similarly, for each quotient $e : FX \to B$ with free domain FX we find a set of equations $\Phi_e = \{(t, t') : e(t) = e(t')\}$ such that

Proposition 5.3.2. *$A \models \Phi_e$ iff all $f : FX \to A$ factor through e.*

That is, in the case of $\mathcal{X} = \mathsf{Set}$, the two propositions above show that we can—from a semantic point of view—replace equations by quotients and describe satisfaction by injectivity:

Definition 5.3.3 (Injective). We say that A is injective wrt $B_0 \xrightarrow{e} B_1$ iff all $f : B_0 \to A$ factor through e, ie iff for all $f : B_0 \to A$ there is $f' : B_1 \to A$ such that

$$\begin{tikzcd} B_0 \arrow[r, "e"] \arrow[dr, "f"'] & B_1 \arrow[d, dotted, "f'"] \\ & A \end{tikzcd}$$

commutes.

To summarise, we have seen that to each set of equations Φ corresponds a quotient e_Φ with free domain such that satisfaction of Φ is injectivity wrt e_Φ. And, conversely, to each quotient with free domain corresponds a set of equations.

[4] Note that (t, t') is not necessarily in Φ. But since (t, t') is in the smallest congruence generated by Φ, ie (t, t') is in the intersection of all kernels of *all morphisms* whose kernels contain Φ, and since $v^\sharp$ is a morphism it also identifies t and t'.

5.4 Duality of Modal and Equational Logic

To summarise the two previous sections we can now extend Table 2.1 on page 46 as follows.

$\Sigma : \mathcal{X} \to \mathcal{X}$	$\Sigma^{\mathrm{op}} : \mathcal{X}^{\mathrm{op}} \to \mathcal{X}^{\mathrm{op}}$
$\mathsf{Coalg}(\Sigma)$	$\mathsf{Alg}(\Sigma^{\mathrm{op}})$
factorisation system (E, M) subcoalgebras $m \in M$	factorisation system (M, E) quotients $e \in E$
(formulas are) subcoalgebras of cofree coalgebras	(sets of equations are) quotients of free algebras
(modal rules are) subcoalgebras	(implications are) quotients
(satisfaction is)	
projectivity	injectivity

The duality of modal rules and implications is explained in the Exercises.

5.5 A (Co)Variety Theorem

This section gives as an application of the duality of modal and equational logic a proof of the following two theorems:

Theorem 5.5.1 (Variety theorem, HSP theorem). *Let* $\Sigma : \mathsf{Set} \to \mathsf{Set}$ *be a functor such that* $\mathsf{Alg}(\Sigma)$ *has free algebras. Then a class* $\mathcal{B}$ *of* Σ*-algebras is equationally definable iff* $\mathcal{B}$ *is closed under quotients, embeddings, and products.*

Theorem 5.5.2 (Covariety theorem). *Let* $\Sigma : \mathsf{Set} \to \mathsf{Set}$ *be a functor such that* $\mathsf{Coalg}(\Sigma)$ *has cofree coalgebras. Then a class* $\mathcal{B}$ *of* Σ*-coalgebras is definable by an expressive modal logic for* Σ*-coalgebras iff* $\mathcal{B}$ *is closed under embeddings, quotients, and coproducts.*

Note that these theorems are not each other's dual because Set has not been dualised. But our proof for second theorem presented below dualises to a proof of the first theorem if we are careful to keep track of the properties used in the proof. But before discussing this in more detail, let us see the proof.

Proof of the covariety theorem. "only if" is the easy direction which is an immediate corollary of Lemma 5.2.1. Nevertheless, we will take the time to show how the preservation properties can also be deduced using the properties of factorisation systems and projectivity.

We show that φ is invariant under embeddings, quotients, and coproducts. Denote by m_φ the subcoalgebra corresponding to φ (see Definition 5.2.3).

Let $e : A \to A'$ be a *quotient*. We show that A projective wrt m_φ implies A' projective wrt m_φ. Consider

$$\begin{tikzcd}
\bullet & \bullet \arrow[l, "m_\varphi"'] \\
A' \arrow[u, "f"] \arrow[ur, dotted] & A \arrow[l, "e"'] \arrow[u, "g"']
\end{tikzcd}$$

A projective wrt m_φ implies that for all f as in the diagram there is g making the square commute. Now, the dotted morphism exists due to unique diagonalisation (see 2.2.5) and shows that A' projective wrt m_φ.

For *coproducts* consider

$$\begin{tikzcd}
\bullet & \bullet \arrow[l, "m_\varphi"'] \\
\coprod A_i \arrow[u, "f"] \arrow[ur, dotted] & A_i \arrow[l, "in_i"'] \arrow[u, "g_i"']
\end{tikzcd}$$

A_i projective wrt m_φ implies that for all f as in the diagram there are g_i making the squares commute for all i. Now, the dotted morphism exist due to unique diagonalisation (see Proposition 2.2.12) and shows that $\coprod A_i$ is projective wrt m_φ.

For an *embedding* $m : A' \to A$ consider

$$\begin{tikzcd}
FC & \bullet \arrow[l, "m_\varphi"'] \\
A \arrow[u, "g"] & A' \arrow[l, "m"'] \arrow[ul, "f" description]
\end{tikzcd}$$

For all f as in the diagram, there is $g : A \to FC$ such that the triangle commutes (check that this is due to FC being cofree). Since A is projective wrt m_φ, g factors through m_φ, hence f factors as well.

"if": The main point is to find the defining modal formulas. Consider the collection of morphisms $(s_i : B_i \to FC)_{i \in I}$ which consists of all morphisms with codomain FC and the domain in $\mathcal{B}$. Let $m_C : F'C \to FC$ be the union of the images of the s_i (see Chapter 2.2.3).

<table>
<tr><td>Co-Variety Theorem</td><td>Variety Theorem</td></tr>
<tr><td colspan="2">$U : \mathcal{C} \to \mathcal{X}$</td></tr>
<tr><td>$\mathcal{C}$ has cofree objects FC</td><td>$\mathcal{C}$ has free objects FX</td></tr>
<tr><td colspan="2">$\mathcal{C}$ has a factorisation system such that:</td></tr>
<tr><td>$\mathcal{C}$ has union of images</td><td>$\mathcal{C}$ has intersection of kernels</td></tr>
<tr><td>$m : A' \to A$ embedding $\Rightarrow$
all $f : A' \to FC$ factor through m:
$$\begin{tikzcd} FC & \\ A \arrow[u, dotted] & A' \arrow[ul, "f"'] \arrow[l, "m"'] \end{tikzcd}$$</td><td>$e : A \to A'$ quotient $\Rightarrow$
all $f : FX \to A'$ factor through e:
$$\begin{tikzcd} FX \arrow[d, dotted] \arrow[dr, "f"] & \\ A \arrow[r, "e"] & A' \end{tikzcd}$$</td></tr>
<tr><td>$\forall A \,.\, \exists$ embedding $A \to FUA$</td><td>$\forall A \,.\, \exists$ quotient $FUA \to A$</td></tr>
</table>

Table 5.1: Properties used in the proof of the (co)variety theorem

Since we assume an *expressive* modal logic there is a formula φ_C corresponding to m_C (ie $A \models \varphi_C$ iff A projective wrt m_C). Let $\Phi = \{\varphi_C : C \in \mathsf{Set}\}$. We show that $\mathcal{B}$ is defined by Φ, ie

$$\mathcal{B} = \{A \in \mathsf{Coalg}(\Sigma) : A \models \Phi\}.$$

'$\subset$': By definition of the m_C, all $B \in \mathcal{B}$ are projective wrt to the m_C and hence satisfy the φ_C.
'$\supset$': Suppose $A \models \Phi$. In particular, $A \models \varphi_{UA}$, ie A is projective wrt $m_{UA} : F'UA \to FUA$. Note that there is an embedding $A \to FUA$. This embedding factors through m_{UA}, hence A is a subcoalgebra of $F'UA$. Since $\mathcal{B}$ is closed under coproducts and quotients we have $F'UA \in \mathcal{B}$ (see the proof of Proposition 2.2.10) and since $\mathcal{B}$ is closed under embeddings it follows $A \in \mathcal{B}$. $\square$

Let us now analyse the duality of the two theorems. Table 5.1 lists the properties we used to prove the covariety theorem. That is, in fact, we proved

Proposition 5.5.3. *Let $U : \mathcal{C} \to \mathcal{X}$ be a functor satisfying the properties of the left column of Table 5.1. Then a class $\mathcal{B} \subset \mathcal{C}$ is projective wrt to a class of embeddings with cofree codomains iff $\mathcal{B}$ is closed under embeddings, quotients, and coproducts.*

Since $U : \mathcal{C} \to \mathcal{X}$ satisfies the properties of the left column of Table 5.1 iff U^{op} satisfies the properties of the right column, the same proof shows the dual theorem

Proposition 5.5.4. *Let $U : \mathcal{C} \to \mathcal{X}$ be a functor satisfying the properties of the right column of Table 5.1. Then a class $\mathcal{B} \subset \mathcal{C}$ is injective wrt to a class of quotients with free domains iff $\mathcal{B}$ is closed under quotients, embeddings, and products.*

These two propositions are each other duals and the (co)variety theorems are their corollaries obtained by instantiating $\mathcal{C}$ with (co)algebras and using the correspondence of subcoalgebras/formulas and, respectively, of quotients/equations.

5.6 Exercises

Exercise 5.6.1 (Implications as quotients). An *implication* $\bigwedge \Phi \to (t, t')$ in variables from X consists of a set $\Phi \subset UFX \times UFX$ and a pair $(t, t') \in UFX \times UFX$. Define for an assignment $v : X \to UA$

$$A, v \models \bigwedge \Phi \to (t, t') \quad \text{iff} \quad A, v \models \Phi \;\Rightarrow\; A, v \models (t, t')$$

and $A \models \bigwedge \Phi \to (t, t')$ iff $A, v \models \bigwedge \Phi \to (t, t')$ for all assignments v.

To find the quotient corresponding to an implication $i = \bigwedge \Phi \to (t, t')$ consider, similarly to Section 5.3, the quotient e_i as in

$$\begin{tikzcd}
FX/\Phi \arrow[r, "e_i"] & FX/\,\Phi \cup \{(t, t')\} \\
FX \arrow[u, dotted, "e_\Phi"] \arrow[ur, dotted, "e_{(\Phi \cup \{(t,t')\})}"'] &
\end{tikzcd}$$

Show the following

1. $A \models i$ iff A is injective wrt e_i.

2. For each quotient e there is a class of implications Φ such that $A \models \Phi$ iff A is injective wrt e.

Exercise 5.6.2 (Modal rules as subcoalgebras). A *modal rule* φ/ψ in colours from C consists of two formulas φ, ψ in colours from C. For a colouring $v : UA \to C$, let

$$A, v \models \varphi/\psi \quad \text{iff} \quad A, v \models \varphi \;\Rightarrow\; A, v \models \psi$$

and $A \models \varphi/\psi$ iff $A, v \models \varphi/\psi$ for all colourings v.

For each modal rule φ/ψ find an embedding (subcoalgebra) m such that $A \models \varphi/\psi$ iff A is projective wrt m. [Hint: Dualise the diagram of the previous exercise.]

The aim of the next three exercises is to show that the (co)variety theorems can be formulated without referring to the base category. The idea is to replace free objects by 'projective objects' which are defined without reference to the forgetful functors.

Exercise 5.6.3 (Projective objects, enough projectives). In a category with a factorisation system, an object B is called *projective* iff for all quotients $e : A \to A'$ and all arrows $f : B \to A'$ there is an arrow $g : B \to A$ such that $f = e \circ g$:

$$\begin{tikzcd} B \arrow[d, dotted, "g"'] \arrow[dr, "f"] & \\ A \arrow[r, "e"'] & A' \end{tikzcd}$$

The category is said to have *enough projectives* iff for each object A there is a projective object B and a quotient $B \to A$.

For a category $\mathsf{Alg}(\Sigma)$ over Set which has free algebras FX for each set X show the following.

1. Free algebras are projective.
2. There is a quotient $FUA \to A$ for all algebras A.
3. $\mathsf{Alg}(\Sigma)$ has enough projectives.
4. B is projective iff it is a retract of a free algebra.

Exercise 5.6.4. The aim is to show that, concerning satisfaction of equations, there is no need not to distinguish between quotients with free domain and quotients with projective domain. Let $U : \mathsf{Alg}(\Sigma) \to \mathsf{Set}$ have free algebras and consider a quotient $e : B \to A$ with projective domain B. Let e' be the quotient given by the following pushout

$$\begin{tikzcd} B \arrow[r, "e"] \arrow[d, "m"'] & A \arrow[d, dotted] \\ FUB \arrow[r, dotted, "e'"'] & \bullet \arrow[ul, phantom, "\ulcorner", very near start] \end{tikzcd}$$

where FUB is the free algebra over the carrier of B and m exists because B is a retract of FUB (see (4) of the previous exercise).

Show that for any $C \in \mathsf{Alg}(\Sigma)$ it holds C injective wrt e iff C injective wrt e.

Exercise 5.6.5 (variety theorem 'without a base category'). Show the following variation of the variety theorem (or its dual): Let $\mathcal{C}$ be a category with factorisation system, intersection of kernels, and enough projectives. Then a class $\mathcal{B} \subset \mathcal{C}$ is injective wrt to a class of quotients with projective domains iff $\mathcal{B}$ is closed under quotients, embeddings, and products. [Hint: Note that 'having enough projectives' is just the last two conditions in the right column of Table 5.1.]

5.7 Notes

This chapter is based on [38] where, to the author's knowledge, the idea of the duality of modal and equational logic was expressed for the first time. That the duality can be made precise by understanding modal formulas as subcoalgebras and equations as quotients was shown in [37, 36]. The concept of equations as quotients and satisfaction as injectivity as well as categorical proofs of the variety and similar theorems are due to Banaschewski and Herrlich [7]. A textbook presentation is given in Adámek, Herrlich, Strecker [3], Chapter 16. Our proof of the covariety theorem was obtained by dualising a corresponding proof in [7]. The idea to derive a dual of the variety theorem by dualising injectivity to projectivity was discovered independently also by Roşu [50] and Awodey and Hughes [6]. The duality of implications and modal rules is treated in [39].

The duality of Kripke frames and modal algebras (see eg [12]) differs from the duality of algebras and coalgebras. More generally, the duality of Kripke frames and modal algebras is not a categorical one since the embedding of a Kripke frame into its ultrafilter extension is not a coalgebra morphism. In case of deterministic signatures, however, the duality is indeed categorical as shown by Jacobs [31].

A covariety theorem appears already in Rutten [64] but there is no discussion of what an appropriate logic for coalgebras could be. Gumm and Schröder [20] and Roşu [52] treat the case without colourings (ie $C = 1$) and Gumm [22] presents a co-variety theorem where 'coequations' φ are points in the carrier of cofree coalgebras and $A, v \models \varphi$ iff $\varphi \notin \mathrm{Im}(v^\sharp)$. Goldblatt [18, 17] restricts attention to specific signatures (polynomial functors) and proves definability results for finitary logics for coalgebras.

Appendix A

Category Theory

We collect the definitions of category theory needed in the course and not appearing in the text. Natural transformations are only used in Chapter 4.3 and adjunctions, as far as needed, are explained in the text. For introductory texts on category theory the reader is referred to one of [42, 3, 5, 8].

A **category** $\mathcal{A}$ consists of a class of objects, also denoted by $\mathcal{A}$, and, for all objects A, B, of a set of arrows, also called morphisms, $\mathcal{A}(A, B)$. We write $f : A \to B$ for $f \in \mathcal{A}(A, B)$ and call A the *domain*, B the *codomain* of f. Moreover, for all $\mathcal{A}(A, B)$, $\mathcal{A}(B, C)$ there is an operation

$$\begin{aligned} \circ_{A,B,C} : \mathcal{A}(A, B) \ \times \ \mathcal{A}(B, C) \ &\to \mathcal{A}(A, C) \\ (g : A \to B, f : B \to C) &\mapsto f \circ_{A,B,C} g : A \to C \end{aligned}$$

We drop the subscript and read $f \circ g$ as 'f after g'. There is also for each $A \in \mathcal{A}$ an 'identity' $\mathrm{id}_A : A \to A$. All this data has to satisfy

$$\begin{aligned} f \circ (g \circ h) &= (f \circ g) \circ h \\ \mathrm{id} \circ f &= f \\ f \circ \mathrm{id} &= f \end{aligned}$$

We dropped the subscripts which means that these equations have to be satisfied for all instantiations matching the required typing for $\circ$.

Set is the category of sets and functions. A **discrete** category is a category which has only identities as arrows.

$m : B \to C$ is **mono** iff $m \circ f = m \circ f' \ \Rightarrow \ f = f'$ for all $A \in \mathcal{A}$ and all $f, f' : A \to B$. $e : A \to B$ is **epi** iff $f \circ e = f' \circ e \ \Rightarrow \ f = f'$ for all $C \in \mathcal{A}$ and all $f, f' : B \to C$. In case that for two arrows $m : A \to B$ and $e : B \to A$ it holds that $e \circ m = \mathrm{id}_A$ then e is **split epi**, m **split mono**, and A a **retract** of B. (Show that split epis are epi and split monos are mono.) An arrow is **iso** iff it is split mono and split epi. If there is an iso $i : A \to B$ then A and B are called isomorphic, written $A \simeq B$.

Exercise A.0.1 (Monos, epis, isos in Set*).* A function is injective iff it is mono. A function with non-empty domain is injective iff it is split mono. A function is surjective iff it is epi iff it is split epi. A function is iso iff it is mono and epi.

A **functor** $H : \mathcal{A} \to \mathcal{X}$ from a category $\mathcal{A}$ to a category $\mathcal{X}$ consists of an operation H_0 on objects of $\mathcal{A}$ and an operation H_1 on arrows of $\mathcal{A}$, mapping each $f \in \mathcal{A}(A, B)$ to $H_1(f) \in \mathcal{X}(H_0(A), H_0(B))$ such that

$$H(\mathrm{id}_A) = \mathrm{id}_{H(A)}$$
$$H(f \circ g) = H(f) \circ (g)$$

where, following common usage, subscripts of H have been dropped. Examples for functors can be found in Chapter 2.1.

Exercise A.0.2. A functor $H : \mathsf{Set} \to \mathsf{Set}$ maps surjective functions to surjective functions and injective functions with non-empty domain to injective functions.

For the **dual** of a category and a functor see Chapter 2.4.

A **natural transformation** $\tau : G \to H$ from a functor $G : \mathcal{A} \to \mathcal{X}$ to a functor $H : \mathcal{A} \to \mathcal{X}$ consists of arrows $\tau_A : GA \to HA$ for each $A \in \mathcal{A}$ such that for each $f : A \to B$ in $\mathcal{A}$ the following diagram

$$\begin{tikzcd}
GA \arrow[r, "\tau_A"] \arrow[d, "Gf"'] & HA \arrow[d, "Hf"] \\
GB \arrow[r, "\tau_B"'] & HB
\end{tikzcd}$$

commutes.

A **diagram** is a functor $D : \mathcal{I} \to \mathcal{A}$. The name diagram indicates that we think of D as indexing objects in $\mathcal{A}$. We therefore denote objects in $\mathcal{I}$ by i, j. A **cone** $(A, (c_i : A \to Di)_{i \in \mathcal{I}})$ over a diagram D consists of an object $A \in \mathcal{A}$ and arrows $(c_i : A \to Di)_{i \in \mathcal{I}}$ such that for all $f : i \to j$ in $\mathcal{I}$

$$\begin{tikzcd}
& A \arrow[dl, "c_i"'] \arrow[dr, "c_j"] & \\
Di \arrow[rr, "Df"'] & & Dj
\end{tikzcd}$$

commutes.

A **limit** of the diagram D is a cone $(A, (c_i : A \to D_i)_{i \in \mathcal{I}})$ satisfying the following *universal property*: for any cone $(A', (c_i' : A' \to Di)_{i \in \mathcal{I}})$ over D there is a unique 'mediating' arrow $h : A' \to A$ such that for all $i \in \mathcal{I}$

$$\begin{tikzcd}
& A' \arrow[dl, "c_i'"'] \arrow[d, dotted, "h"] \\
Di & A \arrow[l, "c_i"]
\end{tikzcd}$$

commutes. $(A, (c_i : Di \to A)_{i \in \mathcal{I}})$ is a **colimit** of D iff $(A^{\mathrm{op}}, (c_i^{\mathrm{op}})_{i \in \mathcal{I}})$ is a limit of D^{op}. A **weak** (co)limit is defined like a (co)limit but the mediating arrow need not be unique.

Exercise A.0.3. Show that two different limits of the same diagram are isomorphic. Moreover, they are canonically isomorphic, that is, the isomorphisms are uniquely determined.

A **final** object, also called **terminal** object, is the limit of an empty diagram (ie $\mathcal{I}$ is empty). An **initial** object is, dually, the colimit of an empty diagram.

Exercise A.0.4. Show that in Set the initial object is the empty set and a terminal object is a set containing precisely one element.

A **product** of $D1$ and $D2$ is the limit $(D1 \times D2, \pi_1 : D1 \times D2 \to D1, \pi_2 : D1 \times D2 \to D2)$ indicated below

$$\begin{tikzcd}
& A' \arrow[ddl, "c_1'"'] \arrow[ddr, "c_2'"] \arrow[d, dotted, "h"] & \\
& D1 \times D2 \arrow[dl, "\pi_1"] \arrow[dr, "\pi_2"'] & \\
D1 & & D2
\end{tikzcd}$$

($\mathcal{I}$ is here the discrete category with two objects $\{1, 2\}$.)

Exercise A.0.5. Show that in Set the product is isomorphic to the cartesian product (with π_i being the projection to the i-th component).

Exercise A.0.6. Generalise the definition of the binary product to a definition of an infinite product.

A **pullback** (dual notion: **pushout**) is a limit of a diagram [1]

$$\begin{tikzcd} D1 \arrow[dr, "Df"'] & & D2 \arrow[dl, "Dg"] \\ & D0 & \end{tikzcd}$$

that is, a cone $(A, c_i : A \to Di)$ such that for all (A', c_1', c_2') with $Df \circ c_1' = Dg \circ c_2'$ there is a unique $h : A' \to A$ such that

$$\begin{tikzcd} & A' \arrow[ddl, "c_1'"'] \arrow[ddr, "c_2'"] \arrow[d, dotted, "h"] & \\ & A \arrow[dl, "c_1"] \arrow[dr, "c_2"'] & \\ D1 \arrow[dr, "Df"'] & & D2 \arrow[dl, "Dg"] \\ & D0 & \end{tikzcd}$$

commutes.

Exercise A.0.7. Show that in Set pullbacks exist and are given by the subset of $D1 \times D2$ 'satisfying the constraint $Df = Dg$', that is, by $\{(d_1, d_2) \in D1 \times D2 : Df(d_1) = Dg(d_2)\}$.

A functor $H : \mathcal{A} \to \mathcal{A}$ **weakly preserves pullbacks** iff it maps pullbacks (A, c_1, c_2) of a diagram D to weak pullbacks (HA, Hc_1, Hc_2) of the diagram HD. H **preserves weak pullbacks** iff it maps weak pullbacks (A, c_1, c_2) to weak pullbacks (HA, Hc_1, Hc_2). If $\mathcal{A}$ has pullbacks then H *weakly preserves pullbacks iff* H *preserves weak pullbacks.*

A **coproduct** $(D1 + D2, in_1 : D_1 \to D1 + D2, in_2 : D_2 \to D1 + D2)$ of $D1$ and $D2$ is

[1] $\mathcal{I}$ is here the category with objects $\{0, 1, 2\}$ and arrows (besides identities) $f : 1 \to 0$ and $g : 2 \to 0$.

the colimit given by

$$\begin{tikzcd}
 & A' & \\
 & D1 + D2 \arrow[u, dotted, "h"] & \\
D1 \arrow[ur, "in_1"'] \arrow[uur, "c_1"] & & D2 \arrow[ul, "in_2"] \arrow[uul, "c_2"']
\end{tikzcd}$$

Exercise A.0.8. Show that in Set the coproduct is isomorphic to the disjoint union.

Given a family of objects $(A_i)_{i \in I}$ the coproduct of the A_i is denoted by $\coprod_I A_i$. [2]

Exercise A.0.9. Let $\mathcal{X}$ be a category with coproducts, $\Sigma : \mathcal{X} \to \mathcal{X}$ a functor, and I a set.

1. Define $\coprod_I$ on arrows and show that it becomes a functor.

2. Use the universal property of the coproduct to show that there is a natural transformation $\tau : \coprod_I \Sigma \to \Sigma \coprod_I$.

3. Explain τ in case of $\Sigma = \mathcal{P}$.

4. Given a family of coalgebras (X_i, ξ_i), let (X, ξ) be their coproduct as defined in Chapter 2.2.1. Show that $\xi = \tau_X \circ \coprod_I \xi_i$.

That is, the coproduct of systems is given by the coproduct of the transition structures followed by the canonical natural transformation $\coprod_I \Sigma \to \Sigma \coprod_I$.

An **adjunction** between two functors $U : \mathcal{A} \to \mathcal{X}$ and $F : \mathcal{X} \to \mathcal{A}$ is given by a bijection

$$(-)^*_{A,C} : \mathcal{X}(UA, C) \xrightarrow{\sim} \mathcal{A}(A, FC)$$

which is natural in A and C. We write $U \dashv F$ and call U the left adjoint and F the right adjoint. Two different characterisations of adjunctions are the following:

Proposition A.0.10. *The functor $U : \mathcal{A} \to \mathcal{X}$ has a right adjoint iff for each $C \in \mathcal{X}$ there is $FC \in \mathcal{A}$ and $\varepsilon_C : UFC \to C$ in $\mathcal{X}$ such that for any $A \in \mathcal{A}$ and any $c : UA \to C$*

[2] $\coprod_I A_i$ is defined as the colimit of the diagram given by the discrete category $\mathcal{I}$ with I as the set of objects and $D : \mathcal{I} \to \mathcal{A}$ mapping $i \mapsto A_i$.

in $\mathcal{X}$ there is a unique morphism $c^\sharp : A \to FC$ such that the triangle

$$\begin{tikzcd} & UFC \arrow[dl, "\varepsilon C"'] & & FC \\ C & UA \arrow[l, "c"] \arrow[u, "Uc^\sharp"'] & & A \arrow[u, "c^\sharp"'] \end{tikzcd}$$

commutes. Then ε is a natural transformation and F can be extended in a unique way to a functor $\mathcal{X} \to \mathcal{A}$.

Remark. $c^\sharp$ in the proposition was denoted by $c^*_{A,C}$ in the definition of adjunction.

Proposition A.0.11. *The functor $U : \mathcal{A} \to \mathcal{X}$ has a left adjoint iff for each $X \in \mathcal{X}$ there is $FX \in \mathcal{A}$ and $\eta_X : X \to UFX$ in $\mathcal{X}$ such that for any $A \in \mathcal{A}$ and any $v : X \to UA$ there is a unique morphism $v^\sharp : FX \to A$ such that the triangle*

$$\begin{tikzcd} & UFX \arrow[d, "Uv^\sharp"] & & FX \arrow[d, "v^\sharp"] \\ X \arrow[ur, "\eta_X"] \arrow[r, "v"'] & UA & & A \end{tikzcd}$$

commutes. Then η is a natural transformation and F can be extended in a unique way to a functor $\mathcal{X} \to \mathcal{A}$.

Exercise A.0.12. Show that $(- \times I)$ is left adjoint to $(-)^I$ for any set I. (Then $(-)^*_{X,X}$ maps algebras $X \times I \to X$ to the corresponding coalgebras $X \to X^I$, see Chapter 1.3.1.)

Bibliography

[1] P. Aczel and N. Mendler. A final coalgebra theorem. In D. H. Pitt et al, editor, *Category Theory and Computer Science*, volume 389 of *LNCS*, pages 357–365. Springer, 1989.

[2] Peter Aczel. *Non-Well-Founded Sets*. Center for the Study of Language and Information, Stanford University, 1988.

[3] J. Adámek, H. Herrlich, and G. Strecker. *Abstract and Concrete Categories*. John Wiley & Sons, 1990.

[4] M.A. Arbib and E.G. Manes. Adjoint machines, state-behaviour machines, and duality. *Journ. of Pure and Applied Algebra*, 6:313–344, 1975.

[5] A. Asperti and G. Longo. *Categories, Types, and Structures: An Introduction to Category Theory for the Working Computer Scientist*. MIT Press, 1991. Available at `http://www.di.ens.fr/users/longo/download.html`.

[6] S. Awodey and J. Hughes. The coalgebraic dual of Birkhoff's variety theorem. October 2000. Available at `http://www.contrib.andrew.cmu.edu/user/jesse/papers/CoBirkhoff.ps.gz`.

[7] B. Banaschewski and H. Herrlich. Subcategories defined by implications. *Houston Journal of Mathematics*, 2(2):149–171, 1976.

[8] M. Barr and C. Wells. *Category Theory for Computing Science*. Prentice-Hall International, 1990.

[9] Falk Bartels. Generalised coinduction. In A. Corradini, M. Lenisa, and U. Montanari, editors, *Coalgebraic Methods in Computer Science (CMCS'01)*, volume 44.1 of *ENTCS*. Elsevier, 2001.

[10] J. Barwise and L. Moss. *Vicious Circles*. Center for the Study of Language and Information, Stanford University, 1996.

[11] M. Bidoit, R. Hennicker, and A. Kurz. On the duality between observability and reachability. In F. Honsell and M. Miculan, editors, *Foundations of Software Science*

and Computation Structures (FOSSACS'01), volume 2030 of *LNCS*, pages 72–87, 2001.

[12] P. Blackburn, M. de Rijke, and Y. Venema. *Modal Logic*. Cambridge University Press, 2001. See also `http://www.mlbook.org`.

[13] Corina Cîrstea. *Integrating Observations and Computations in the Specification of State-Based, Dynamical Systems*. PhD thesis, University of Oxford, 2000.

[14] E. de Vink and J. Rutten. Bisimulation for probabilistic transition systems: a coalgebraic approach. In *Proceedings of ICALP'97*, volume 1256 of *LNCS*, pages 460–470, 1997.

[15] E.A. Emerson. Temporal and modal logic. In J. van Leeuwen, editor, *Handbook of Theoretical Computer Science*, volume 1. Elsevier, 1990.

[16] Robert 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.

[17] Robert Goldblatt. A calculus of terms for coalgebras of polynomial functors. In A. Corradini, M. Lenisa, and U. Montanari, editors, *Coalgebraic Methods in Computer Science (CMCS'01)*, volume 44.1 of *ENTCS*. Elsevier, 2001.

[18] Robert Goldblatt. What is the coalgebraic analogue of Birkhoff's variety theorem? *Theoretical Computer Science*, To appear.

[19] H. P. Gumm and T. Schröder. Products of coalgebras. *Algebra Universalis*. Accepted for publication.

[20] H. P. Gumm and T. Schröder. Covarieties and complete covarieties. In B. Jacobs, L. Moss, H. Reichel, and J. Rutten, editors, *Coalgebraic Methods in Computer Science (CMCS'98)*, volume 11 of *Electronic Notes in Theoretical Computer Science*, 1998.

[21] H. Peter Gumm. Elements of the general theory of coalgebras. LUATCS'99, 1999.

[22] H. Peter Gumm. Equational and implicational classes of coalgebras. *Theoretical Computer Science*, 260:57–69, 2001.

[23] M. Hennessy and R. Milner. Algebraic laws for nondeterminism. *Journal of the Association for Computing Machinery*, 32:137–161, 1985.

[24] C. Hermida and B. Jacobs. Structural induction and coinduction in a fibrational setting. *Information and Computation*, 145(2):107–152, 1998.

[25] B. Jacobs and J. Rutten. A tutorial on (co)algebras and (co)induction. *EATCS Bulletin*, 62, 1997.

[26] Bart Jacobs. Hompage at `http://www.cs.kun.nl/~bart/`.

[27] Bart Jacobs. Automata and behaviours in categories of processes. Technical Report CS-R9607, CWI, 1996.

[28] Bart Jacobs. Objects and classes, co-algebraically. In B. Freitag, C. B. Jones, C. Lengauer, and H.-J. Schek, editors, *Object-Orientation with Parallelism and Persistence*, pages 83–103. Kluwer Acad. Publ., 1996.

[29] Bart Jacobs. *Categorical Logic and Type Theory.* Elsevier, 1998.

[30] Bart Jacobs. The temporal logic of coalgebras via galois algebras. Technical Report CSI-R9906, Computing Science Institute Nijmegen, 1999.

[31] Bart Jacobs. Towards a duality result in coalgebraic modal logic. In Horst Reichel, editor, *Coalgebraic Methods in Computer Science (CMCS'00)*, volume 33 of *Electronic Notes in Theoretical Computer Science*, pages 163–198, 2000.

[32] Bart Jacobs. Many-sorted coalgebraic modal logic: a model-theoretic study. *Theoretical Informatics and Applications*, 35(1):31–59, 2001.

[33] Fred Kröger. *Temporal Logic of Programs.* Springer, 1987.

[34] A. Kurz and R. Hennicker. On institutions for modular coalgebraic specifications. *Theoretical Computer Science.* To appear.

[35] A. Kurz and D. Pattinson. Coalgebras and modal logics for parameterised endofunctors. Technical Report SEN-R0040, CWI, 2000. `http:www.cwi.nl/~kurz`.

[36] Alexander Kurz. *Logics for Coalgebras and Applications to Computer Science.* PhD thesis, Ludwig-Maximilians-Universität München, 2000. `http://www.informatik.uni-muenchen.de/~kurz`.

[37] Alexander Kurz. Modal logic is dual to equational logic. January 2000. `http://www.informatik.uni-muenchen.de/~kurz`.

[38] Alexander Kurz. A co-variety-theorem for modal logic. In *Advances in Modal Logic 2.* Center for the Study of Language and Information, Stanford University, 2001. Selected Papers from 'Advances in Modal Logic 2', Uppsala, 1998.

[39] Alexander Kurz. Modal rules are co-implications. In A. Corradini, M. Lenisa, and U. Montanari, editors, *Coalgebraic Methods in Computer Science (CMCS'01)*, volume 44.1 of *ENTCS.* Elsevier, 2001.

[40] Alexander Kurz. Specifying coalgebras with modal logic. *Theoretical Computer Science*, 260:119–138, 2001.

[41] J. Lambek and P. Scott. *Introduction to Higher Order Categorical Logic*, volume 7. Cambridge University Press, Cambridge, England, 1986.

[42] Saunders Mac Lane. *Category Theory for the Working Mathematician*. Springer, 1971.

[43] Z. Manna and A. Pnueli. *The Temporal Logic of Reactive and Concurrent Systems*. Springer, 1992.

[44] Lawrence Moss. Coalgebraic logic. *Annals of Pure and Applied Logic*, 96:277–317, 1999.

[45] Dirk Pattinson. Semantical principles in the modal logic of coalgebras. In *Proceedings 18th International Symposium on Theoretical Aspects of Computer Science (STACS 2001)*, LNCS, Berlin, 2001. Springer. Also available as technical report at `http://www.informatik.uni-muenchen.de/~pattinso/`.

[46] Andrew M. Pitts. Categorical logic. In S. Abramsky, D. M. Gabbay, and T. S. E. Maibaum, editors, *Handbook of Logic in Computer Science*, volume 6. Oxford University Press, 1995. Available at `ftp://ftp.cl.cam.ac.uk/papers/ap/catl.ps.gz`.

[47] 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 *Electronic Notes in Theoretical Computer Science*, 1998.

[48] Horst Reichel. *Initial computability, algebraic specifications, and partial algebras*. Oxford, Clarendon Press, 1987.

[49] Horst Reichel. An approach to object semantics based on terminal co-algebras. *Mathematical Structures in Computer Science*, 5(2):129–152, June 1995.

[50] Grigore Roşu. Personal Communication, August 1999.

[51] Grigore Roşu. *Hidden Logic*. PhD thesis, University of California at San Diego, 2000.

[52] Grigore Roşu. Equational axiomatizability for coalgebra. *Theoretical Computer Science*, 260:229–247, 2001.

[53] Martin Rößiger. Languages for coalgebras on datafunctors. In J. Rutten B. Jacobs, editor, *Coalgebraic Methods in Computer Science (CMCS'00)*, volume 19 of *Electronic Notes in Theoretical Computer Science*, pages 55–76, 1999.

[54] Martin Rößiger. Coalgebras and modal logic. In Horst Reichel, editor, *Coalgebraic Methods in Computer Science (CMCS'00)*, volume 33 of *Electronic Notes in Theoretical Computer Science*, pages 299–320, 2000.

[55] Martin Rößiger. *Coalgebras, Clone Theory, and Modal Logic*. PhD thesis, Dresden University of Technology, 2000.

[56] Martin Rößiger. From modal logic to terminal coalgebras. *Theoretical Computer Science*, 260:209–228, 2001.

[57] J. Rutten and D. Turi. On the foundations of final semantics: Non-standard sets, metric spaces, partial orders. Report CS-R9241, CWI, Amsterdam, 1992.

[58] J. Rutten and D. Turi. Initial algebra and final coalgebra semantics for concurrency. Report CS-R9409, CWI, Amsterdam, 1994.

[59] J. J. M. M. Rutten. A structural co-induction theorem. Technical Report CS R 9346, CWI, Amsterdam, 1993.

[60] J. J. M. M. Rutten. A calculus of transition systems (towards universal coalgebra). In A. Ponse, M. de Rijke, and Y. Venema, editors, *Modal Logic and Process Algebra*, volume 53 of *CSLI Lecture Notes*. Center for the Study of Language and Information, Stanford University, 1995.

[61] J. J. M. M. Rutten. Automata and coinduction - an exercise in coalgebra, 1998.

[62] J. J. M. M. Rutten. Coalgebra, concurrency, and control. Report SEN-R9921, CWI, Amsterdam, 1999.

[63] J. J. M. M. Rutten. Behavioural differential equations: A coinductive calculus of streams, automata, and power series. Report SEN-R0023, CWI, Amsterdam, 2000.

[64] J. J. M. M. Rutten. Universal coalgebra: A theory of systems. *Theoretical Computer Science*, 249:3–80, 2000. First appeared as technical report CS R 9652, CWI, Amsterdam, 1996.

[65] Krister Segerberg. An essay in classical modal logic. *Filosofiska Studier 13*, 1971.

[66] Colin Stirling. Modal and temporal logics. In S. Abramsky, D. M. Gabbay, and T. S. E. Maibaum, editors, *Handbook of Logic in Computer Science*, volume 2. Oxford University Press, 1992.

[67] 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, 1998.

[68] Johan van Benthem. *Modal Correspondence Theory*. PhD thesis, University of Amsterdam, 1976.

[69] Johan van Benthem. *Modal Logic and Classical Logic*. Bibliopolis, Naples, 1983.

[70] Steven J. Vickers. *Topology Via Logic*, volume 5. Cambridge University Press, 1988.

[71] Wolfgang Wechler. *Universal Algebra for Computer Scientists*, volume 25 of *EATCS Monographs on Theoretical Computer Science*. Springer, 1992.

[72] James 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 *Electronic Notes in Theoretical Computer Science*, 1998.

[73] James Worrell. *On Coalgebras and Final Semantics.* PhD thesis, Oxford University Computing Laboratory, 2000.