Interaction, observation and denotation

A study of dialgebras for program semantics

MSc Thesis *(Afstudeerscriptie)*

written by

Alwin Blok
(born January 19th, 1984 in Kornhorn, The Netherlands)

under the supervision of **Dr. Vincenzo Ciancia**, and submitted to the Board of Examiners in partial fulfillment of the requirements for the degree of

MSc in Logic

at the *Universiteit van Amsterdam.*

Date of the public defense:	**Members of the Thesis Committee:**
September, 2012	Dr. Alexandru Baltag
	Dr. Marcello Bonsangue
	Dr. Vincenzo Ciancia
	Prof. Dr. Benedikt Löwe
	Dr. Alessandra Palmigiano

Institute for Logic, Language and Computation

Abstract

We investigate the use of dialgebras for program semantics. Dialgebras generalise both algebras and coalgebras. As a result dialgebras can model compositional and interactive features of programs, in addition to program state and behaviour over time. We investigate the theory of universal dialgebra and use it to derive canonical denotational semantics from dialgebraic operational specifications. The greatest quotient of a dialgebraic operational semantics provides us with a denotational semantics that characterises behavioural equivalence. Subdialgebras on the other hand can provide small stand–alone denotations for programs. We extend the theory of universal dialgebra with a new result. We define minimisation and simplification sequences for dialgebras. When used with a dialgebraic operational semantics these sequences enable us to compute suitable denotations for programs. The technique can be used to decide equivalence of programs, if equivalence is decidable at all. The thesis is concluded with two examples. We provide a dialgebraic semantics for regular expressions that aligns with the classic semantics in which regular expressions are interpreted as deterministic automata. Finally we give a dialgebraic semantics for the synchronous CCS. In the coalgebraic semantics of the CCS the behaviour of individual processes is modelled. The dialgebraic semantics illustrates that dialgebras can model the behaviour of multiple processes at once, along with their interactions and resulting behaviour.

Contents

Chapter 1

Introduction

1.1 Context and motivation

In programming language theory at least two different attitudes towards semantics are discerned. In the denotational approach each program is interpreted as a particular element of a semantic domain. The elements of this domain are called denotations and a denotational semantics assigns equivalent programs equal denotations. In contrast, an operational semantics considers the computation steps that are to be performed if the program is executed. It interprets a program as the first computation step of its execution, coupled with a residual program that describes the continued process.

Although qualifications of semantics as 'denotational' and 'operational' are common, we prefer to be a bit cautious. There are many approaches to programming languages semantics and one could write a book about their taxonomy. In addition, an accurate classification would involve a philosophical discussion, not a mathematical investigation. That being said, for a proper introduction to this thesis we have to make the notion of a denotational and an operational semantics precise. We do so in this introductory chapter, with a special attention to algebraic and coalgebraic semantics.

Algebras are relevant for several reasons. Foremost, they have been established as a mathematical theory of syntax [6, 7]. Algebraic semantics is commonly associated with denotational semantics and the main topic of the thesis is a generalisation of algebras. In contrast, coalgebras are commonly associated with operational semantics. They have in fact been proposed as a mathematically rigorous theory of operational semantics, c.f. [22]. In addition, the main topic of this thesis is a generalisation of algebras *and* coalgebras.

In this thesis we investigate the use of dialgebras for program semantics. Dialgebras generalise algebras and coalgebras. The increased generality offers a more expressive framework for operational semantics. In particular interaction and input to programs can be intuitively expressed. The theory of universal dialgebra provides the necessary tools for deriving a canonical denotational semantics. In addition it enables the construction of small denotations that do not require a reference to the encompassing semantic domain.

1.1.1 Syntax

What exactly does and what does not constitute syntax is a philosophical issue. It is not our aim to give a full account of the philosophical debate. Instead we focus on the use of algebra as a mathematical theory of syntax.

Computer science tends to have a much more fine–grained view on syntax than mathematics. For a computer program to correctly handle input a much greater attention to detail is required than in mathematical discourse. For example, a computer program that expects textual input has to deal with superfluous parenthesis, indentation, white–space and line–breaks. These issues may not be interesting but in applications they have to be dealt with. In mathematics the issues are not relevant and it tends to start off with a more abstract view of syntax. This distinction motivates the introduction of concrete and abstract syntax.

The concrete syntax of a language is the collection of arrangements of symbols that constitute well–formed terms. The concrete syntax of a (textual) programming language is the set of character sequences that are accepted by the compiler or interpreter of the language. In [15] McCarthy proposed abstract syntax. He observes that the actual representation of program terms is not relevant to the definition and study of their semantics. Rather, what is relevant are the intentional properties of terms; how they may be composed and if possible, how they may be decomposed. Now, it would be impossible to define a nontrivial semantics without an ability to inspect terms so for any practical application it reasonable to assume that terms can be decomposed. The intuition that terms can be composed motivates the formalisation of a concrete syntax as an algebra. Abstracting away from a concrete representation is achieved by considering the concrete syntax up to algebra isomorphism.

A concrete syntax is an algebra. Throughout the thesis we work with a generalised notion of algebra, specifically algebras over an endofunctor. An algebra over an endofunctor $F\colon \mathrm{Set} \to \mathrm{Set}$ is a pair (A, α) of a set A of elements and a composition function $\alpha\colon FA \to A$. The set FA can be thought of as a set of unassembled elements and the composition function α takes unassembled elements to their assembled counterparts. A morphism of algebras $f\colon (X, \chi) \to (Y, \upsilon)$ is a function $f\colon X \to Y$ that is compatible with composition; specifically $\upsilon \circ Ff = f \circ \chi$. Finally, an abstract syntax is an isomorphism class of algebras. For any concrete syntax (A, α) the isomorphism class $\{(B, \beta) \mid (B, \beta) \cong (A, \alpha)\}$ is an abstract syntax.

In practice '*the* concrete syntax' of a (textual) programming language refers to the sequences of characters that are accepted by the compiler or interpreter whereas 'abstract syntax' refers to an intensional description of the syntactic structure, often called an abstract syntax tree (AST). Implementations of programming languages employ a parser function that converts a string of characters to a convenient concrete representation of ASTs. From the theory one would expect that the concrete syntax and the algebra of concretised ASTs are isomorphic. We should note though that in applications the parser typically is not an isomorphism, often it maps programs with minor differences in formatting the same concretised AST.

Abstract syntax provides an intensional description of the syntactic structure of terms. The mathematical theory that is especially well suited to intensional descriptions is category theory. Indeed, an object defined categorically by means of limits, colimits or any typical categorical construction is defined up to isomorphism; up to intensional equivalence if you will. Category theory is also a powerful tool for defining canonical solutions given particular constraints. Now, a syntax is typically defined by a grammar and the set of terms generated by a grammar is

a good example of a canonical construction. For example, consider the grammar for natural numbers.

$$n \in \mathbb{N} \quad ::= \quad \mathsf{Z} \;\big|\; \mathsf{S}(n)$$

The grammar can be seen as specifying a class $\mathcal{C}$ of sets that adhere to a particular constraint: every $N \in \mathcal{C}$ has an element $\mathsf{Z}_N \in N$ and if $n \in N$ then there is an element $\mathsf{S}_N(n) \in N$. This specifies exactly the class of algebras with one constant and one unary operation. The set of terms $\mathbb{N}$ generated by the grammar is the 'smallest yet least identified' set that adheres to this constraint. This can be expressed succinctly as $\mathbb{N}$ being the *initial* algebra of this signature, where an algebra (X, χ) is initial if for every algebra (Y, υ) of the same signature there is a unique algebra morphism $\mathsf{i}_{(Y,\upsilon)} : (X, \chi) \to (Y, \upsilon)$.
For a full account of initial algebras and their motivation we recommend [6]. We introduce the concept here because initial algebras are an excellent formalisation of syntax and abstract syntax. The language generated by a grammar is succinctly described as an initial algebra. Since initial algebras are unique up to isomorphism they abstract away from the concrete representation of terms. Finally, composition functions of initial algebras are isomorphisms, providing an intensional description of both composition and decomposition of terms.

1.1.2 Denotational semantics

In a denotational semantics a program is interpreted as an element of a semantic domain. Assuming that both the collection of program terms and the collection of denotations are sets, a denotational semantics is formalised as a function.

A denotational semantics for a set of terms X is an interpretation function $f : X \to D$ that maps each term x to its denotation $f(x)$ in a semantic domain D.

A denotational semantics $f : X \to D$ establishes a notion of semantic equivalence. Two terms x and y are semantically equivalent if they have the same denotation $f(x) = f(y)$. The semantic equivalence relation induced by f is its *kernel* K_f.

$$\mathrm{K}_f \quad := \quad \{(x, y) \in X \times X \mid f(x) = f(y)\}$$

Conversely, any equivalence relation defines a denotational semantics. For let $R \subseteq X \times X$ be an equivalence relation. One may consider, for example, a relation defined by a proof system. Then $[\_]_R : X \to X/_R$ is a denotational semantics that interprets a term as an equivalence class of terms.

$$[x]_R \quad := \quad \{y \in X \mid (x, y) \in R\}$$

Note that $[\_]_R$ is a surjective function. Now, any surjective function essentially characterises an equivalence relation. Consider a surjection $f : X \to D$. It is isomorphic to the function $[\_]_{\mathrm{K}_f} : X \to X/_{\mathrm{K}_f}$ via the isomorphism $h : X/_{\mathrm{K}_f} \to D$ where $h(S) := f(x)$ for x any element of S. Moreover, any denotational semantics $f : X \to D$ gives rise to a unique surjective denotational semantics that induces the same equivalence relation, simply by restricting its codomain to its image.

To illustrate the concept of a denotational semantics we consider a simple language. We let $n, m \in \mathbb{N}$ range over the natural numbers. The set X of arithmetical expressions is generated by the following grammar.

$$t_1, t_2 \in X \quad ::= \quad n \in \mathbb{N} \;\big|\; t_1 + t_2 \;\big|\; t_1 \times t_2$$

The semantics that we learn in elementary school is a denotational semantics $[\![\_]\!]\colon X \to \mathbb{N}$ that interprets arithmetical expressions as their 'result'. To make this precise, we assume the standard definition of addition and multiplication of natural numbers and denote them by $(+_{\mathbb{N}})$ and $(\times_{\mathbb{N}})$. The denotational semantics $[\![\_]\!]\colon X \to \mathbb{N}$ is defined as follows.

$$\begin{aligned} [\![n]\!] &:= n \\ [\![t_1 + t_2]\!] &:= [\![t_1]\!] +_{\mathbb{N}} [\![t_2]\!] \\ [\![t_1 \times t_2]\!] &:= [\![t_1]\!] \times_{\mathbb{N}} [\![t_2]\!] \end{aligned}$$

To wrap up the example, $[\![2+2]\!] = [\![1+3]\!] = 4$, thus the terms $2+2$ and $1+3$ are semantically equivalent. The example illustrates the concept of a denotational semantics, but it has another noteworthy property. Recall that a syntax is an algebra (X, χ) where X is a set of terms and $\chi\colon FX \to X$ is a composition function. If the domain of a denotational semantics for X can be equipped with a compatible algebra structure then we speak of an algebraic denotational semantics.

Let $f\colon X \to D$ be a denotational semantics for a syntax (X, χ). If there is an algebra structure $\delta\colon FD \to D$ such that $f\colon (X, \chi) \to (D, \delta)$ is an algebra morphism then $f\colon (X, \chi) \to (D, \delta)$ is an algebraic denotational semantics.

Algebraic semantics formalise the concept of compositional semantics. In a compositional semantics the meaning of a term is obtained by composing the meaning of its subterms. The compositional aspect of an algebraic semantics is reflected by its accompanying semantic equivalence, a congruence relation. Moreover any congruence relation induces a surjective algebra morphism.

However, not every denotational semantics is compositional and it can be illuminating to see an example of this. We extend the language of arithmetical expressions with variables and assignment. We let $n, m \in \mathbb{N}$ range over the natural numbers and let $v \in V$ range over a set of variables. The set of terms X' is generated by the following grammar:

$$t_1, t_2 \in X' \quad ::= \quad n \in \mathbb{N} \;\mid\; t_1 + t_2 \;\mid\; t_1 \times t_2 \;\mid\; v \in V \;\mid\; v \leftarrow t_1 \;\mid\; t_1; t_2$$

Intuitively, a term $v \leftarrow t_1$ will be interpreted as t_1 but it will also assign the result of t_1 to the variable v. A term $t_1; t_2$ is interpreted as t_2 however, it does take into account the values assigned to variables as per t_1. We let variables that have not been assigned a value default to 0 thus v_1 and $v_1 \leftarrow 2; v_2$ will both be assigned the denotation 0. We define the denotational semantics formally as a function $[\![\_]\!]\colon X' \to \mathbb{N}$ that takes $t \mapsto \pi_1([\![t, v \mapsto 0]\!])$ where $[\![\_,\_]\!]\colon X' \times \mathbb{N}^V \to \mathbb{N} \times \mathbb{N}^V$ interprets a term and a variable assignment as the 'result' of the term and an updated assignment. The initial assignment $v \mapsto 0$ is the constant function that assigns 0 to every variable. We define $[\![\_,\_]\!]$ as follows.

$$\begin{aligned} [\![v, f]\!] &:= (f(v), f) && \\ [\![n, f]\!] &:= (n, f) && \\ [\![t_1 + t_2, f]\!] &:= (n +_{\mathbb{N}} m,\, h) && \text{where } (n, g) := [\![t_1, f]\!] \text{ and } (m, h) := [\![t_2, g]\!] \\ [\![t_1 \times t_2, f]\!] &:= (n \times_{\mathbb{N}} m,\, h) && \text{where } (n, g) := [\![t_1, f]\!] \text{ and } (m, h) := [\![t_2, g]\!] \\ [\![v_1 \leftarrow t_1, f]\!] &:= (n, h) && \text{where } (n, g) := [\![t_1, f]\!] \text{ and } h\colon v_1 \mapsto n \text{ and } v \mapsto g(v) \text{ if } v \neq v_1 \\ [\![t_1; t_2, f]\!] &:= (m, h) && \text{where } (n, g) := [\![t_1, f]\!] \text{ and } (m, h) := [\![t_2, g]\!] \end{aligned}$$

The semantics $[\![\_]\!] : X' \to \mathbb{N}$ is not algebraic. Consider the terms v_1 and v_2 and the terms $v_1 \leftarrow 1; v_1$ and $v_1 \leftarrow 1; v_2$. Then $[\![v_1]\!] = [\![v_2]\!] = 0$ and obviously $[\![v_1 \leftarrow 1]\!] = [\![v_1 \leftarrow 1]\!] = 1$ but $[\![v_1 \leftarrow 1; v_1]\!] = 1$ and $[\![v_1 \leftarrow 1; v_2]\!] = 0$. Indeed this is not a compositional semantics.

1.1.3 Operational semantics

An operational semantics is intended to model computation steps. For example, in the denotational semantics for arithmetical expressions the 'result' of a term is a suitable denotation. In a programming language the situation is more complicated, we have to deal a.o. with programs that do not terminate. One solution is to interpret every non–terminating program as a particular element of the semantic domain, representing divergence. However this may be too coarse; programs that do not terminate may still generate output, change internal memory configurations, or interact with their environment in general.

Since an operational semantics models computation steps it may be more discerning than a typical denotational semantics. Rather than considering only the final result, the kind– and number of computation steps can be taken into account. For example, two programs that have the same final outcome could be discerned by an operational semantics if they require a distinct amount of computation steps. Note through that there is not one unified notion of a computation step. An often made distinction is between 'small–step' and 'big–step' operational semantics, though what is 'small' and what is 'big' is not rigorously specified. In fact, a denotational semantics such as the one we defined for arithmetical expressions can be seen as a (very) big–step operational semantics that interprets terms in exactly one step. An other extreme is an 'executable operational semantics' in which the size– and order of steps is defined in such detail that the definition constitutes an interpreter or compiler for the language that can be executed on an actual physical machine.

To illustrate the concept we define an operational semantics for the language of arithmetical expressions. We define an interpretation function $[\![\_]\!] : X \to X + \mathbb{N}$ that takes a term t to a reduced expression t', signified by $\iota_1(t')$ or to a final result $n \in \mathbb{N}$ signified by $\iota_2(n)$. Intuitively this semantics reduces the leftmost subterm of the form $n + m$ to $n +_{\mathbb{N}} m$ and can be applied stepwise until there is no more such subterm. Now, it is common for an operational semantics to be defined by a set of transition rules instead of an inductive function definition. We feel that this should be part of the example. Thus, we define two relations $(:) \subseteq X \times \mathbb{N}$ and $(\to) \subseteq X \times X$ by the set of rules below. Incidentally, the way in which transition rules define a relation is similar to how a context free grammar defines a set of terms. The rules specify constraints and induce a class of relations that adhere to them. The intended relation is a canonical element of this class.

$$\frac{}{n : n}$$

$$\frac{t_1 \to t_1'}{t_1 + t_2 \to t_1' + t_2} \qquad \frac{t_1 : n \quad t_2 \to t_2'}{t_1 + t_2 \to t_1 + t_2'} \qquad \frac{t_1 : n \quad t_2 : m}{t_1 + t_2 \to n +_{\mathbb{N}} m}$$

$$\frac{t_1 \to t_1'}{t_1 \times t_2 \to t_1' \times t_2} \qquad \frac{t_1 : n \quad t_2 \to t_2'}{t_1 \times t_2 \to t_1 \times t_2'} \qquad \frac{t_1 : n \quad t_2 : m}{t_1 \times t_2 \to n \times_{\mathbb{N}} m}$$

The relations $(:)$ and $(\to)$ are functional relations with $\mathrm{dom}\,(:) = \mathbb{N}$ and $\mathrm{dom}\,(\to) = X \setminus \mathbb{N}$. Together they form a function $[\![\_]\!] : X \to X + \mathbb{N}$ where $t \mapsto \iota_1(t')$ if $t \to t'$ and $t \mapsto \iota_2(n)$ if

$t : n$. The same function can be defined as follows.

$$\begin{array}{rcl}
[\![n]\!] & := & \iota_2(n) \\
[\![n + m]\!] & := & \iota_1(n +_{\mathbb{N}} m) \\
[\![n + t_2]\!] & := & \iota_1(n + [\![t_2]\!]) \\
[\![t_1 + t_2]\!] & := & \iota_1([\![t_1]\!] + t_2) \text{ if } t_1 \notin \mathbb{N} \\
[\![n \times m]\!] & := & \iota_1(n \times_{\mathbb{N}} m) \\
[\![n \times t_2]\!] & := & \iota_1(n \times [\![t_2]\!]) \\
[\![t_1 \times t_2]\!] & := & \iota_1([\![t_1]\!] \times t_2) \text{ if } t_1 \notin \mathbb{N}
\end{array}$$

The stepwise aspect of the operational semantics is revealed by a simple example. If a term t is such that $[\![t]\!] = \iota_2(n)$ then we are done in a single step. If however $[\![t]\!] = \iota_1(t')$ for some t' then we can continue by interpreting t'. For example, $[\![(1+(2+3))+4]\!] = \iota_1((1+5)+4)$, then $[\![(1+5)+4]\!] = \iota_1(6+4)$, then $[\![6+4]\!] = \iota_1(10)$ and finally $[\![10]\!] = \iota_2(10)$.

The example illustrates the concept of an operational semantics but it has an additional property. It is a coalgebraic semantics. Dual to algebras, coalgebras can be seen as modelling a notion of decomposition, though often a more appropriate intuition is that coalgebras model processes and their observable behaviour as they evolve over time.

A coalgebra is a pair (X, χ) of a set X and an observation function $\chi \colon X \to GX$ where $G \colon \mathrm{Set} \to \mathrm{Set}$ is a functor. The observation function takes elements of X to their behaviour in GX, which may again involve elements of X. A morphism of coalgebras $f : (X, \chi) \to (Y, \upsilon)$ is a function $f : X \to Y$ that is compatible with observations; specifically $Ff \circ \chi = \upsilon \circ f$.

The definition of an algebraic denotational semantics suggests the following analogous definition of a coalgebraic denotational semantics.

Let $f : X \to D$ be a denotational semantics for a set X of terms. If there are coalgebra structures $\chi : X \to FX$ and $\delta : D \to FD$ such that $f : (X, \chi) \to (D, \delta)$ is a coalgebra morphism then $f : (X, \chi) \to (D, \delta)$ is a coalgebraic denotational semantics.

However, this presumes a semantic domain and it adds the requirement that compatible coalgebra structures exist. But as we have mentioned previously, it is not always clear at once how to define a suitable semantic domain for program terms. Now, in the example of an operational semantics above we have defined a coalgebra structure on terms and we have referred to it as an interpretation function. We may think of this interpretation function as a recursively defined denotational semantics. It maps a term x to a 'denotation' that consists of observations but possibly also of other terms, which have to be interpreted in turn to establish the meaning of x. This motivates the following definition of a coalgebraic operational semantics.

A coalgebraic operational semantics for a set X of terms is a function $\chi : X \to FX$.

Recall that a denotational semantics $f : X \to Y$ defines a semantic equivalence relation. This equivalence relation is essentially the surjective denotational semantics obtained by restricting the codomain of f to its image. The question that arises is if and how an *operational* semantics defines an equivalence relation. For coalgebras this is achieved via a notion of behavioural equivalence, well known as bisimulation equivalence. For us, a bisimulation equivalence is the

kernel of a coalgebra morphism thus essentially, it is a surjective coalgebra morphism. Now, typically one considers the greatest such equivalence. Interestingly, the denotational semantics characterised by this equivalence is also obtained via an other conceptual approach. This time, rather than considering equivalences we start with the coalgebraic operational semantics. If we regard the operational interpretation function as a recursively defined denotational semantics as sketched previously, then all we have to do to retrieve a proper denotational semantics is 'unfold' the recursive definition. As it turns out, a repeated unfolding converges to a denotational semantics and the equivalence relation characterises it is the greatest bisimulation equivalence on terms.

The method for deriving a unique denotational semantics just described is very similar to how a grammar defines a language or to how a set of transition rules define a relation. Indeed, one can consider an operational semantics to be the *definition* of a denotational semantics thus obtained.

As we have laid out, given an operational semantics the equivalence on terms is determined by the derived denotational semantics, where two program terms are semantically equivalent if they are mapped to the same element. Thus if we want to *decide* equivalence, if at all possible, we have to compare elements of the semantic domain. To make this possible they have to be assigned a finite, decidable label. This can be achieved by labelling every element with a finite part of the semantic domain that uniquely identifies it. Thus, instead of interpreting terms as elements we can interpret them as subobjects.

In summary, universal coalgebra allows a canonical denotational semantics to be derived from a coalgebraic operational semantics, in addition it provides tools for constructing small, even finite denotations for terms that can be used to determine or even decide equivalence.

1.1.4 Dialgebraic semantics

In the previous section we have introduced operational semantics, though the focus has been on coalgebraic semantics. Coalgebras are suitable for modelling observations, however, interaction and input are not as intuitively modelled. In fact, the only way to model input in a coalgebra is as, well, output. Arguably a less intuitive representation is a small price to pay for gaining access to the theory of universal coalgebra. However, especially if an operational semantics is regarded as a *definition* of a denotational semantics we would prefer a more expressive framework. In particular, we would prefer a framework that allows a clean separation of input and output. To this end we propose to generalise coalgebras to dialgebras and investigate the theory of universal dialgebra.

A dialgebra is a pair (X, χ) of a set X and an interaction–observation function $\chi : FX \to GX$ where $F : \mathrm{Set} \to \mathrm{Set}$ and $G : \mathrm{Set} \to \mathrm{Set}$ are functors. The set FX can be thought of as a set of experiments or interactions, each of which may involve elements from X. The observation function takes experiments to observations in GX, which may again involve elements from X. A morphism $f : (X, \chi) \to (Y, \upsilon)$ of dialgebras is a function $f : X \to Y$ that is compatible with interactions and observations; specifically $\upsilon \circ Ff = Gf \circ \chi$.

Having defined dialgebras we can trace our steps in the previous section and define dialgebraic denotational semantics and dialgebraic operational semantics.

Let $f : X \to D$ be a denotational semantics for a set of terms X. If there are structures $\chi : FX \to GX$ and $\delta : FD \to GD$ such that $f : (X, \chi) \to (D, \delta)$ is a dialgebra morphism then $f : (X, \chi) \to (D, \delta)$ is a dialgebraic denotational semantics.

The remarks that were made about coalgebraic denotational semantics apply here as well. It is often not clear at once how to define a denotational semantics for program terms. Instead, what can be defined with relative ease is how users may interact with a program or how programs may interact with each other and what behaviour results from these interactions. Thus, what is defined instead is a dialgebraic operational semantics.

A dialgebraic operational semantics for a set X of terms is a function $\chi : FX \to GX$.

Note that dialgebras generalise coalgebras, thus any coalgebraic operational semantics is also a dialgebraic operational semantics. Dialgebras are more expressive, they offer more freedom in how a semantics is defined. The additional functor on the domain side of the interpretation function allows input and even composition of programs terms to be expressed, whereas a in a coalgebraic setting one would be forced to model these features as part of the output.

Having discussed coalgebraic semantics it should be clear what we aim for in this thesis. We hope to discuss the techniques in use with coalgebraic operational semantics and generalise them to dialgebras. In particular, we investigate how a canonical denotational semantics is obtained from a dialgebraic operational semantics and how to construct subdialgebras as small denotations for program terms.

1.2 Overview of the thesis

Chapter 1

Given the size of the introduction and the number of concepts introduced we have decided to make the introduction into a proper chapter. It sketches the big picture and the motivation for this work and it includes this particular section entitled "Overview of the thesis".

Chapter 2

In the second chapter we introduce some basic category theory. We have included a discussion of diagrams and category–shaped diagrams. Even though diagrams are pervasive in category theory, they tend to be introduced with varying levels of detail. Beyond diagrams we cover limits, colimits and slice and coslice categories. The reader familiar with category theory may choose to glance through this chapter just shortly.

Chapter 3

In the third chapter we study universal dialgebra. We include some results on limits and factorisations and we introduce subdialgebras and quotient dialgebras. Quotient dialgebras are especially important; the category of quotients of a dialgebraic operational semantics is the category of appropriate denotational semantics. Categories of subdialgebras on the other hand are categories of small denotations for individual terms. We conclude the chapter with a section on dialgebra bisimulations. There are many slightly different notions of bisimulation for coalgebras. We observe that they are most often used either to specify subcoalgebras or to specify a quotient coalgebras. We attempt to clean this up a bit by defining dialgebra bisimulations as joint subdialgebras, which have a the dual notion of joint quotient dialgebras.

Chapter 4

The fourth chapter presents the main development of this thesis. We develop iterative procedures for generating minimal and simple dialgebras. The procedures rely on two new concepts that generalise images and coimages of morphisms, to which we we refer as their base and cobase. The G–base of a morphism is used to form subobjects of objects X that are in some sense contained in GX whereas the F–cobase of a morphism defines quotients of X that are in some sense compatible with quotients of FX. If equivalence is decidable at all, then the minimisation and simplification procedures may be used to compute finite, unique denotations for terms and decide equivalence. We conclude the chapter with a section about the interplay between minimisation and simplification, comparing minimisations of simplifications and simplifications of minimisations.

Chapter 5

In this chapter we present dialgebraic semantics for two systems. The first subject is a classic in computer science. We define a dialgebraic operational semantics for regular expressions and show how the theory may be used to derive small denotations. This exhibits how neatly the dialgebraic semantics resembles the classic semantics that interprets regular expressions as deterministic state machines. The second subject is the calculus of communicating systems. We include the coalgebraic semantics for the synchronous CCS and present a dialgebraic semantics. The coalgebraic semantics models the behaviour of *individual* processes. The dialgebraic

semantics illustrates the use of dialgebras for modelling the behaviour of multiple processes *at once*, including their interactions and the resulting behaviour.

Chapter 6

In the final chapter we shortly conclude our work and we include a number of ideas that require additional research.

Chapter 2

Categorical preliminaries

In this chapter we introduce some of the basic concepts of category theory and the notation that we use throughout the thesis. The chapter is not intended to be a complete reference, instead we refer to the standard references [1, 11, 12].

2.1 Categories and diagrams

Definition 2.1.1 (Quiver)**.** A *quiver* is a directed graph that may have multiple arrows with the same source and target vertices. A quiver $\mathcal{J}$ consists of:

1. A collection of vertices $|\mathcal{J}|$.
   Individual vertices are denoted by A, B, X, Y, Z, where $A : \mathcal{J}$ stresses that A is in $|\mathcal{J}|$.
2. For every pair of vertices X, Y a family of arrows $\mathcal{J}[X, Y]$.
   Individual arrows f, g, h in $\mathcal{J}[X, Y]$ are denoted by $f : X \to Y$. ◊

Quivers are depicted graphically as dots connected by arrows where a dot represents exactly one vertex and a depicted arrow represents exactly one arrow of the quiver.

Definition 2.1.2 (Category)**.** A *category* is a quiver with additional structure. In a category, vertices X, Y are called objects and arrows $f : X \to Y$ are called morphisms. A category $\mathcal{C}$ is a quiver along with:

3. For every object $A : \mathcal{C}$ a designated identity morphism $\mathrm{id}_A : A \to A$.
4. A composition operator $\_ \circ \_$ that maps a pair of morphisms $f : X \to Y$ and $g : Y \to Z$ to a morphism $(g \circ f) : X \to Z$ such that the following conditions are satisfied.

$$\mathrm{id}_Z \circ g \;=\; g \qquad\qquad (f \circ g) \circ h \;=\; f \circ (g \circ h) \qquad\qquad f \circ \mathrm{id}_X \;=\; f \qquad \text{◊}$$

For a morphism $f : A \to B$ we call A the domain of f and B the codomain of f. The collection of morphisms of $\mathcal{C}$ with domain A is denoted by $\mathcal{C}[A, \_]$ and the collection of morphisms with codomain B is denoted by $\mathcal{C}[\_, B]$. The collection of all morphisms of $\mathcal{C}$ is denoted by $\mathcal{C}[\_, \_]$.

Quivers that do not contain any arrows will be called *discrete*. Likewise, a discrete category is a category that has no morphisms other than the identity morphisms.

Definition 2.1.3 (Diagram)**.** A *diagram* is a labelled quiver. Specifically, if $\mathcal{J}$ is a quiver then a diagram *of shape* $\mathcal{J}$ *in a category* $\mathcal{C}$ is a labelling $L\colon \mathcal{J} \to \mathcal{C}$ which assigns to every vertex $A\colon \mathcal{J}$ an object $(LA)\colon \mathcal{C}$ and to every arrow $f\colon A \to B$ of $\mathcal{J}$ a morphism $Lf\colon LA \to LB$ of $\mathcal{C}$. $\Diamond$

Intuitively it is clear that L is a labelled quiver if one thinks of L as a set of ordered pairs of type (A, LA) and (f, Lf). As one would expect, diagrams in a category $\mathcal{C}$ are depicted as quivers with labels added to the dots and arrows. As an example let X, Y, Z be objects of $\mathcal{C}$ and let $f\colon X \to Y$, $h\colon Z \to Y$ and $g\colon X \to Y$ be morphisms of $\mathcal{C}$. Below are examples of diagrams in $\mathcal{C}$ based on the quivers depicted previously.

$$X \qquad Y \qquad\qquad \begin{array}{ccc} & & Z \\ & & \downarrow{\scriptstyle h} \\ X & \xrightarrow[f]{} & Y \end{array} \qquad\qquad \begin{array}{ccc} X & \xrightarrow{g} & Y \\ {\scriptstyle f}\downarrow & & \\ Y & & \end{array} \qquad\qquad X \underset{g}{\overset{f}{\rightrightarrows}} Y$$

Definition 2.1.4 (Commutative diagram)**.** A diagram $L\colon \mathcal{J} \to \mathcal{C}$ is *commutative* if for any two paths with the same start and end object of $\mathcal{J}$, the labels on arrows compose to the same morphism of $\mathcal{C}$.

$$\begin{aligned} \text{For all} \quad LA \xrightarrow{Lf_1} LX \xrightarrow{Lf_2} \cdots \xrightarrow{Lf_n} LB \quad &\text{and} \quad LA \xrightarrow{Lg_1} LY \xrightarrow{Lg_2} \cdots \xrightarrow{Lg_m} LB, \\ LA \xrightarrow{Lf_n \circ \ldots \circ Lf_2 \circ Lf_1} LB \quad &= \quad LA \xrightarrow{Lg_m \circ \ldots \circ Lg_2 \circ Lg_1} LB. \end{aligned}$$ $\Diamond$

A few words on the n and m–ary compositions above are required. For any composition $f_1 \circ \ldots \circ f_n$ we have that $f_1 \circ \ldots \circ f_n = \mathrm{id}_X \circ f_1 \circ \ldots \circ f_n \circ \mathrm{id}_Y$ for suitable X and Y. Whenever we specify a morphism as a composition $(f_1 \circ \ldots \circ f_n)\colon X \to Y$ we define the case $n = 0$ to be id_X in which case, of course, $X = Y$.

As an example of commutative diagrams, note that in the rightmost diagram depicted previously, to state that it commutes implies that $f = g$ whereas this is not the case in the second rightmost diagram.

Given a quiver $\mathcal{J}$ and a category $\mathcal{C}$, one particular family of commutative diagrams is the family of diagonal diagrams.

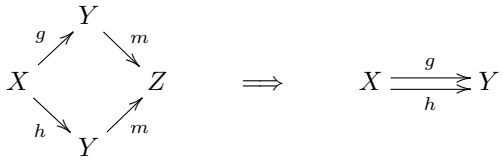
Definition 2.1.5 (Diagonal diagram)**.** For an object $A\colon \mathcal{C}$ the *diagonal diagram* $\Delta_A\colon \mathcal{J} \to \mathcal{C}$ of shape $\mathcal{J}$ assigns to any vertex of $\mathcal{J}$ the object $A\colon \mathcal{C}$ and to any arrow of $\mathcal{J}$ the morphism id_A of $\mathcal{C}$. $\Diamond$

In some sense, a diagonal diagram represents just an object. The fact that it can be of any shape however, makes them a useful tool and we will soon see an example of this.

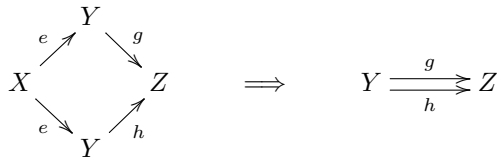
2.2 Morphisms

Definition 2.2.1 (Monomorphism and epimorphism)**.**

1. A *monomorphism* $m : Y \rightarrowtail Z$ is a morphism $m : Y \to Z$ such that for any pair of morphisms $g : X \to Y$ and $h : X \to Y$, if $m \circ g = m \circ h$ then $g = h$. In diagrams, if the diagram below on the left commutes, then so does the diagram on the right.



2. An *epimorphism* $e : X \twoheadrightarrow Y$ is a morphism $e : X \to Y$ such that for any pair $g : Y \to Z$ and $h : Y \to Z$, if $g \circ e = h \circ e$ then $g = h$.



◊

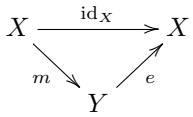
Identity morphisms are monomorphisms. If $\mathrm{id} \circ g = \mathrm{id} \circ h$ then obviously $g = h$. Likewise, if $g \circ \mathrm{id} = h \circ \mathrm{id}$ then $g = h$ thus identity morphisms are epimorphisms, too Moreover, monomorphisms of any category $\mathcal{C}$ are closed under composition, as are epimorphisms. For let $e_1 : X \twoheadrightarrow Y$ and $e_2 : Y \twoheadrightarrow Z$ be epimorphisms. Consider two arbitrary morphisms $f : Z \to A$ and $g : Z \to A$ and assume that $f \circ e_2 \circ e_1 = g \circ e_2 \circ e_1$. Then $f \circ e_2 = g \circ e_2$ because e_1 is an epimorphism and subsequently $f = g$ because e_2 is an epimorphism. Therefore $e_2 \circ e_1$ is an epimorphism. With this in mind it makes sense to define the categories Mono$\mathcal{C}$ and Epi$\mathcal{C}$ as follows.

Definition 2.2.2 (Category of monos, epis)**.** Let $\mathcal{C}$ be a category.

1. The category Mono$\mathcal{C}$ consists of all objects of $\mathcal{C}$ and all monomorphisms between them. Identities and compositions are defined as in $\mathcal{C}$.
2. The category Epi$\mathcal{C}$ consists of all objects of $\mathcal{C}$ and all epimorphisms between them. Identities and compositions are defined as in $\mathcal{C}$. ◊

Recall that identity morphisms are monomorphisms. Another instance of a monomorphism is a morphism $m : X \to Y$ that can be 'undone' by a morphism $e : Y \to X$. If this is the case then we call e a retraction of m and we call m a section of e.

Definition 2.2.3 (Section and retraction)**.** Let $m : X \to Y$ and $e : Y \to X$. If $e \circ m = \mathrm{id}_X$ then e is a *retraction* of m and m is a *section* of e.



◊

A novice may be inclined to think that one can flip the direction of the identity arrow in the diagram above. This is certainly not the case. Asserting that the diagram commutes with the

id_X arrow in the converse direction does indeed assert that $e \circ m = \mathrm{id}_X$ but it also asserts that $m \circ e = \mathrm{id}_Y$ which is *not* implied by commutativity of the original diagram.

A morphism $m : X \to Y$ that has a retraction $e : Y \to X$ is certainly a monomorphism. Consider two morphisms $g : A \to X$ and $h : A \to X$ and assume that $m \circ g = m \circ h$. Then $e \circ m \circ g = e \circ m \circ h$ and because e is a retraction of m we can conclude that $g = h$ which confirms that m is a monomorphism. The dual of this argument shows that any morphism that has a section is an epimorphism.

A more specific case occurs if a morphism $f : X \to Y$ has a retraction $g : Y \to X$ but in addition $f : X \to Y$ is a retraction of $g : Y \to X$. If this is the case then f and g are isomorphisms.

Definition 2.2.4 (Isomorphism)**.** Let $f : X \to Y$ and $g : Y \to X$. Then g is the *inverse* of f and f is the inverse of g if $g \circ f = \mathrm{id}_X$ *and* $f \circ g = \mathrm{id}_Y$. An *isomorphism* $f : X \xrightarrow{\sim} Y$ is a morphism that has an inverse.

$$X \underset{g}{\overset{f}{\rightleftarrows}} Y \qquad \Diamond$$

The inverse of an isomorphism $f : X \xrightarrow{\sim} Y$ is easily seen to be unique and is denoted by $f^{-1} : Y \xrightarrow{\sim} X$. An isomorphism is both a section and a retraction of its inverse therefore it is both a monomorphism and an epimorphism.

So far all properties of morphisms that we have discussed have been defined in terms of commutativity. There has been no mention of an internal structure of objects or morphisms. In category theory properties are commonly defined in this way. The most common category in which objects and morphisms do have an 'inner structure' is the category Set of sets and total functions. Properties of sets and functions often are expressed by referring to their internal structure. Typical examples are injectivity and surjectivity, which are defined by referring to the elements of the domain and codomain of a function. In Set the definitions of monomorphism and epimorphism provide an intentional alternative. A morphism $f : X \to Y$ of Set is injective only if it is a monomorphism, and it is surjective only if it is an epimorphism. The inner structure on sets suggests a particular kind of monomorphism, one that cannot be defined intensionally.

Definition 2.2.5 (Inclusion)**.** In the category Set an *inclusion* $f : X \hookrightarrow Y$ is a morphism $f : X \rightarrowtail Y$ such that $f(x) = x$ for all $x \in X$. $\Diamond$

2.3 Functors

Definition 2.3.1 (Functor)**.** A *functor* $F : \mathcal{C} \to \mathcal{D}$ is a map that takes objects $A : \mathcal{C}$ to objects $(FA) : \mathcal{D}$ and morphisms $f : A \to B$ to morphisms $Ff : FA \to FB$ whilst preserving identities and compositions.

$$F\mathrm{id}_X = \mathrm{id}_{FX} \qquad\qquad F(f \circ g) = Ff \circ Fg \qquad \Diamond$$

A functor $F : \mathcal{C} \to \mathcal{D}$ is *faithful* if for all X, Y of $\mathcal{C}$ the map $F : \mathcal{C}[X, Y] \to \mathcal{D}[FX, FY]$ is injective and F is *full* if for all X, Y of $\mathcal{C}$ the map $F : \mathcal{C}[X, Y] \to \mathcal{D}[FX, FY]$ is surjective. A category $\mathcal{D}$ is a *subcategory* of $\mathcal{C}$ if there is a faithful inclusion functor $I : \mathcal{D} \hookrightarrow \mathcal{C}$. If I is both full and faithful then $\mathcal{D}$ is a *full subcategory* of $\mathcal{C}$.

As the reader will have noticed, for an object A we write functor application as FA rather than the classical $F(A)$. Given a suitable functor G, functor composition will be written as GF. This does not cause any ambiguity: the notation GFA might be read as $G(FA)$ and $(GF)A$ which is $G(F(A))$ and $(G \circ F)(A)$ in classical notation. Finally, we shall use the notation $F^n A$ for iterated applications of F, thus $F^0 A \coloneqq A$ and $F^{m+1} A \coloneqq F F^m A$.

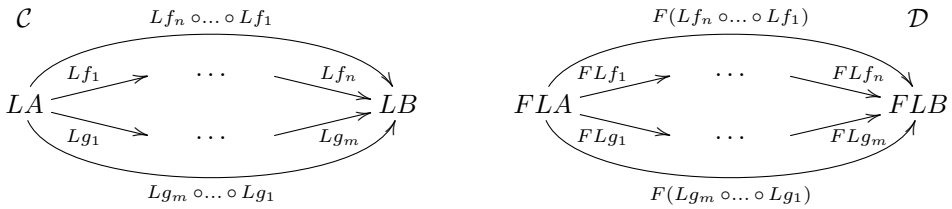
Containers

Intuitively one may think of a functor as a container type. If X and Y are collections of elements then FX and FY are collections of containers, where a container in FX contains elements of X and a container in FY contains elements of Y. Given a morphism $f : X \to Y$ the morphism $Ff : FX \to FY$ takes a container in FX to a container in FY by applying f to every contained element.

Preservation of commutative diagrams

A functor is a map between categories that preserves commutative diagrams. Let $F : \mathcal{C} \to \mathcal{D}$ be a functor and let $L : \mathcal{J} \to \mathcal{C}$ be a diagram of $\mathcal{C}$. Since F takes objects to objects and morphisms to morphisms we can consider what one might call the image of L under F which contains a labelled vertex (A, FLA) for ever labelled vertex (A, LA) of L and a labelled arrow (f, FLf) for every (f, Lf) of L. Hence, we define the image of a diagram $L : \mathcal{J} \to \mathcal{C}$ under a functor $F : \mathcal{C} \to \mathcal{D}$ to be the diagram $FL : \mathcal{J} \to \mathcal{D}$.

Now suppose that the diagram $L : \mathcal{J} \to \mathcal{C}$ commutes. Then its image $FL : \mathcal{J} \to \mathcal{D}$ commutes as well. For consider any two paths between LA and LB of L consisting of labelled arrows $Lf_1, \ldots, Lf_n$ and $Lg_1, \ldots, Lg_m$. Their labels compose to the same morphism $Lf_n \circ \ldots \circ Lf_1 = Lg_m \circ \ldots \circ Lg_1$ of $\mathcal{C}$ by definition of commutativity.



Consider the image of these morphisms under F. Since F is a functor it preserves composition.

$$FLf_n \circ \ldots \circ FLf_1 \quad = \quad F(Lf_n \circ \ldots \circ Lf_1) \quad = \quad F(Lg_m \circ \ldots \circ Lg_1) \quad = \quad FLg_m \circ \ldots \circ FLg_1$$

Thus, any two sequences of labelled arrows of FL compose to the same morphism of $\mathcal{D}$ which shows that the diagram $FL : \mathcal{J} \to \mathcal{D}$ commutes.

2.4 Category shaped diagrams

Consider a quiver $\mathcal{J}$ and a diagram $L : \mathcal{J} \to \mathcal{C}$. Since categories are quivers with additional structure, $\mathcal{J}$ might be a category. If in addition, L preserves identities and compositions then $L : \mathcal{J} \to \mathcal{C}$ is a functor and L is called a category–shaped diagram. Quiver–shaped diagrams are more intuitive than category–shaped diagrams and the graphical diagrams that are common in category theory are depictions of quiver–shaped diagrams. However, category–shaped diagrams

are often easier to work with because they get rid of the distinction between diagrams and functors. Fortunately, any quiver–shaped diagram can be presented as a category–shaped diagram in a way that preserves their commutativity, a construction that will be shown in a minute.

Definition 2.4.1 (Free category)**.** Given a quiver $\mathcal{J}$, the free category $\mathcal{K}$ based on $\mathcal{J}$ is constructed as follows.

- Any vertex $A : \mathcal{J}$ is an object $A : \mathcal{K}$.
- Any path $A \xrightarrow{f_1} B \xrightarrow{f_2} \ldots \xrightarrow{f_n} X$ in $\mathcal{J}$ of length $n \geq 0$ is a morphism of $\mathcal{K}[A, X]$.
- For any object $A : \mathcal{K}$ the morphism id_A is the zero–length path from $A : \mathcal{J}$ to $A : \mathcal{J}$.
- Composition of morphisms is defined as path concatenation:

$$C \xrightarrow{f_1} \cdots \xrightarrow{f_n} X \quad \circ \quad A \xrightarrow{g_1} \cdots \xrightarrow{g_m} C \quad := \quad A \xrightarrow{g_1} \cdots \xrightarrow{g_m} C \xrightarrow{f_1} \cdots \xrightarrow{f_n} X.$$

$\Diamond$

The free category construction allows us to present any quiver–shaped diagram as a category–shaped diagram. Let $L : \mathcal{J} \to \mathcal{C}$ be a quiver–shaped diagram and let $\mathcal{K}$ be the free category on $\mathcal{J}$. Construct a category–shaped diagram $D : \mathcal{K} \to \mathcal{C}$ by putting:

$$\begin{aligned} DX &:= LX \\ D(X \xrightarrow{f_1} \cdots \xrightarrow{f_n} Y) &:= Lf_n \circ \ldots \circ Lf_1 \end{aligned}$$

We would like to use D as a 'drop in' replacement for L. In particular we would like D to commute only if L commutes.

So assume that D does indeed commute. Consider any two paths between some pair of vertices A and B of $\mathcal{J}$. They are morphisms $A \xrightarrow{f_1} \cdots \xrightarrow{f_n} B$ and $A \xrightarrow{g_1} \cdots \xrightarrow{g_m} B$ of $\mathcal{K}$. By commutativity, D maps these morphisms of $\mathcal{K}$ to the same morphism of $\mathcal{C}$ and by definition of D they get mapped to $Lf_n \circ \ldots \circ Lf_1 = Lg_m \circ \ldots \circ Lg_1$ which confirms that L commutes.

Conversely, assume that D does not commute, as witnessed by two paths in $\mathcal{K}$. These paths compose to two parallel morphisms of $\mathcal{K}$. The paths in $\mathcal{K}$ labelled by D compose to distinct morphisms of $\mathcal{C}$. Since D preserves composition it maps the parallel morphisms of $\mathcal{K}$ to the same two distinct morphisms of $\mathcal{C}$. Let the parallel morphisms of $\mathcal{K}$ be $A \xrightarrow{f_1} \cdots \xrightarrow{f_n} B$ and $A \xrightarrow{g_1} \cdots \xrightarrow{g_m} B$. Note that they are paths in $\mathcal{J}$. By definition of D they get mapped to $Lf_n \circ \ldots \circ Lf_1 \neq Lg_m \circ \ldots \circ Lg_1$ which confirms that L does not commute.

2.5 Functor categories

Definition 2.5.1 (Natural transformation)**.** Let $F : \mathcal{C} \to \mathcal{D}$ and $G : \mathcal{C} \to \mathcal{D}$ be functors. A *natural transformation* $\nu : F \dot{\to} G$ is a family of morphisms $(\nu_X : FX \to GX)_{X : \mathcal{C}}$ such that for all morphisms $f : X \to Y$ of $\mathcal{C}$ the following diagram in $\mathcal{D}$ commutes.

$$\begin{array}{ccc} FX & \xrightarrow{\nu_X} & GX \\ {\scriptstyle Ff}\downarrow & & \downarrow{\scriptstyle Gf} \\ FY & \xrightarrow[\nu_Y]{} & GY \end{array}$$

$\Diamond$

The diagram above is called a *naturality square* for ν and a morphism $\nu_X : FX \to GX$ is called the *component* of ν at $X : \mathcal{C}$.

Definition 2.5.2 (Functor category)**.** Let $\mathcal{C}$ and $\mathcal{D}$ be categories. The *functor category* $\mathcal{D}^{\mathcal{C}}$ is defined as follows.

- Every functor $F : \mathcal{C} \to \mathcal{D}$ is an object of $\mathcal{D}^{\mathcal{C}}$.
- Every natural transformation $\nu : F \dot{\to} G$ is a morphism from F to G.
- For every object F the identity morphism $\mathrm{id}_F : F \dot{\to} F$ is the natural transformation $(\mathrm{id}_{FX} : FX \to FX)_{X : \mathcal{C}}$, consisting of identity morphisms of $\mathcal{D}$.
- For any two morphisms $\nu : F \dot{\to} H$ and $\mu : H \dot{\to} G$ the composition $(\mu \circ \nu) : F \dot{\to} G$ is the natural transformation $(\mu_X \circ \nu_X : FX \to GX)_{X : \mathcal{C}}$. ◊

It is not hard to check that compositions and identities are well defined. In particular, for all morphisms $f : X \to Y$ of $\mathcal{C}$ the diagram on the left commutes because $(\mathrm{id}_F)_X := \mathrm{id}_{FX}$. The diagram on the right commutes because it consists of two naturality squares.

$$\begin{array}{ccc} FX & \xrightarrow{(\mathrm{id}_F)_X} & FX \\ {\scriptstyle Ff}\downarrow & & \downarrow{\scriptstyle Ff} \\ FY & \xrightarrow[(\mathrm{id}_F)_Y]{} & FY \end{array} \qquad \begin{array}{ccccc} FX & \xrightarrow{\nu_X} & HX & \xrightarrow{\mu_X} & GX \\ {\scriptstyle Ff}\downarrow & & \downarrow{\scriptstyle Hf} & & \downarrow{\scriptstyle Gf} \\ FY & \xrightarrow[\nu_Y]{} & HY & \xrightarrow[\mu_Y]{} & GY \end{array}$$

Since category–shaped diagrams are just functors, we already have an example of a functor category. Let $\mathcal{J}$ and $\mathcal{C}$ be categories. Objects of the functor category $\mathcal{C}^{\mathcal{J}}$ are category–shaped diagrams $L : \mathcal{J} \to \mathcal{C}$. In the following section we shall see an example of natural transformations between diagrams in a functor category.

2.6 Cones and limits

An appealing feature of category theory is that it offers a way to formalise an intuitive notion of canonicity. Sometimes mathematical structures are said to arise naturally, in that there appears to be only one natural choice among a class of structures that have the required properties. A way to study this intuitive notion of a canonical structure is to study the mathematical structure of the solution space itself. This is a powerful concept. Design in the general sense, might appear to involve lots of arbitrary design decisions (at least, to the unpracticed eye). In liberal arts, designers are typically guided by intuition. A designer might use some formal guidelines but many decisions are made based on intuition and experience of how to achieve the desired effect. If we are able to uncover the mathematical structure of the solution space then design decisions can be formally described and studied. Often, depending on the structure of the solution space may be able to uncover a best, or canonical solution. The key concept in category theory that formalises this intuitive notion of canonicity are initial and final objects.

Definition 2.6.1 (Initial object and final object)**.**

- An *initial object* of a category $\mathcal{C}$ is an object A such that for every object $X : \mathcal{C}$ there is a *unique* morphism $¡_X : A \to X$. If k satisfies just the existence property then A *weakly initial*.

- Dually, a *final object* of $\mathcal{C}$ is an object Z such that for every object $X : \mathcal{C}$ there is a *unique* morphism $!_X : X \to Z$. If k satisfies just the existence property then Z is *weakly final*. $\Diamond$

Often we will speak of *the* initial– and *the* final object of a category. This is justified by the fact that initial– and final objects are unique up to unique isomorphism. Consider a category $\mathcal{C}$ and two initial objects A and B. By definition of initiality of A and B respectively there are morphisms $\mathsf{i}_B : A \to B$ and $\mathsf{i}_A : B \to A$ and thus a morphism $(\mathsf{i}_A \circ \mathsf{i}_B) : A \to A$. However by initiality the identity morphism $\mathrm{id}_A : A \to A$ is the *unique* morphism from A to A thus $\mathsf{i}_A \circ \mathsf{i}_B = \mathrm{id}_A$. The converse holds for B and as such $\mathsf{i}_B : A \to B$ and $\mathsf{i}_A : B \to A$ are isomorphisms as claimed. For clarity, if we speak of *the* initial object of a category $\mathcal{C}$, we refer to a single representative object of the class of initial objects of $\mathcal{C}$.

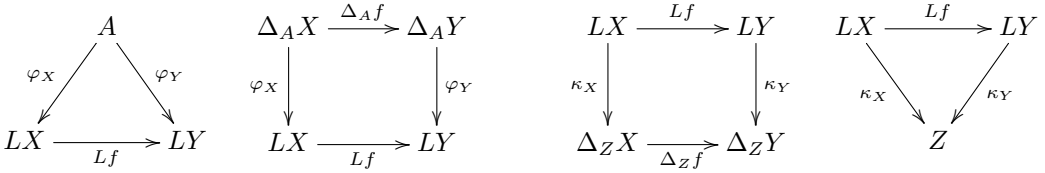
A category can often be constructed such that the 'canonical solution' that we wish to describe is exactly the initial– or final object of this category. One particular example is the category of cones over a diagram.

Definition 2.6.2 (Cones and cocones)**.** Let $L : \mathcal{C}^{\mathcal{J}}$ be a (category–shaped) diagram.

- A *cone* from $A : \mathcal{C}$ to L is a pair (A, φ) where φ is a natural transformation $\varphi : \Delta_A \dot{\to} L$.
- A *cocone* from L to $Z : \mathcal{C}$ is a pair (κ, Z) where κ is a natural transformation $\kappa : L \dot{\to} \Delta_Z$. $\Diamond$

It is easily checked that the definition above amounts to the following:

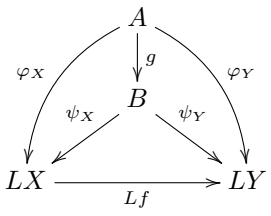
- A cone from $A : \mathcal{C}$ to L is a pair (A, φ) where φ is a family of morphisms $(\varphi_X : A \to LX)_{X : \mathcal{J}}$ such that for every $f : X \to Y$ of $\mathcal{J}$ the two leftmost diagrams below, commute.
- A cocone from L to an object $Z : \mathcal{C}$ is a pair (κ, Z) where κ is a family of morphisms $(\kappa_X : LX \to Z)_{X : \mathcal{J}}$ such that for every $f : X \to Y$ of $\mathcal{J}$ the two rightmost diagrams commute.



Cones and cocones form a category. We define the category of cones, the category of cocones is defined likewise.

Definition 2.6.3 (Category of cones)**.** Let $L : \mathcal{J} \to \mathcal{C}$ be a category–shaped diagram. The *category of cones over* L is defined as follows.

1. Objects are cones $\varphi : \Delta_A \dot{\to} L$.
2. For a pair of cones $\varphi : \Delta_A \dot{\to} L$ and $\psi : \Delta_B \dot{\to} L$ a morphisms from φ to ψ is a morphism $g : A \to B$ of $\mathcal{C}$ such that for all X and Y of $\mathcal{J}$ the following commutes.



3. The identity morphism of a cone $\varphi : \Delta_A \dot{\to} L$ is the morphism $\mathrm{id}_A : A \to A$ of $\mathcal{C}$.
4. Composition of morphisms is defined as composition of the underlying morphisms of $\mathcal{C}$. ◊

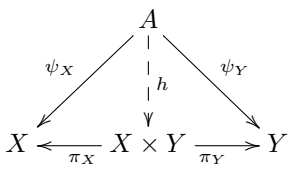
Definition 2.6.4 (Limits and colimits)**.** Let $L : \mathcal{J} \to \mathcal{C}$ be a category–shaped diagram.

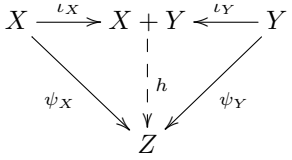
- A *limit* of L is an initial object $\pi : \Delta_A \dot{\to} L$ in the category of cones over L.
  If a cone $\pi : \Delta_A \dot{\to} L$ is weakly initial then π is a *weak limit*.
- A *colimit* of L is a final object $\iota : L \dot{\to} \Delta_Z$ in the category of cocones over L.
  If a cone $\iota : L \dot{\to} \Delta_Z$ is weakly final then ι is a *weak colimit*. ◊

A limit of a diagram $L : \mathcal{J} \to \mathcal{C}$ is a pair (A, π) of an object $A : \mathcal{C}$ and a family of morphisms $(\pi_X : A \to LX)_{X \,:\, \mathcal{J}}$. The morphisms $\pi_X : A \to LX$ are called *projections*, and the notation π_X is reserved for morphisms of initial cones specifically, although we sometimes subscript it with the object LX of $\mathcal{C}$ rather than the vertex X of $\mathcal{J}$. Likewise, a colimit of a a diagram $L : \mathcal{J} \to \mathcal{C}$ is a pair (ι, Z) where ι is a family of morphisms $(\iota_X : LX \to Z)_{X \,:\, \mathcal{J}}$ into an object $Z : \mathcal{C}$, called *injections*. Again, the notation ι_X, or ι_{LX}, is used specifically for morphisms of final cocones. Below, the most common limits and colimits are introduced.

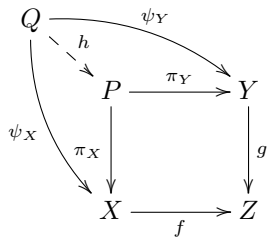
Definition 2.6.5 (Common limits and colimits)**.**

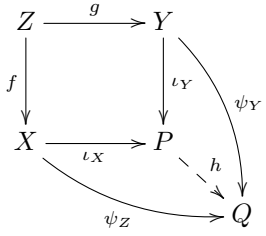
- Let $\mathcal{J}$ be the discrete category with two objects. A diagram $L : \mathcal{J} \to \mathcal{C}$ selects two objects X and Y of $\mathcal{C}$. The binary *product* $X \times Y$ is the initial cone over L. The unique morphism $h : A \to X \times Y$ is denoted by (ψ_X, ψ_Y).





- Again, let $\mathcal{J}$ be the discrete category with two objects so that $L : \mathcal{J} \to \mathcal{C}$ selects two objects X and Y of $\mathcal{C}$. The binary *coproduct* $X + Y$ is the final cocone over L. The unique morphism $h : X + Y \to Z$ is denoted by $[\psi_X, \psi_Y]$.
- Let $\mathcal{J}$ be the free category over the quiver consisting of two arrows with distinct domains and a common codomain. A diagram $L : \mathcal{J} \to \mathcal{C}$ selects three objects and two morphisms $f : X \to Z$ and $g : Y \to Z$ of $\mathcal{C}$. The *pullback* of f and g is the initial cone over L.





- Let $\mathcal{J}$ be the free category over the quiver consisting of two arrows with a common domain but distinct codomains. A diagram $L : \mathcal{J} \to \mathcal{C}$ selects three objects and two morphisms $f : Z \to X$ and $g : Z \to Y$ of $\mathcal{C}$. The *pushout* of f and g is the final cocone over L. ◊

2.7 Comma categories

Definition 2.7.1 (Comma category)**.** Let $\mathcal{D}$, $\mathcal{C}$ and $\mathcal{E}$ be categories and let $F\colon \mathcal{D} \to \mathcal{C}$ and $G\colon \mathcal{E} \to \mathcal{C}$ be functors between them. The comma category $(F{\downarrow}G)$ of F and G is defined as follows.

- Objects of $(F{\downarrow}G)$ are triples (X, f, Y) where X is an object of $\mathcal{D}$, $f\colon FX \to GY$ is a morphism of $\mathcal{C}$ and Y is an object of $\mathcal{E}$.
- A morphism from (X, f, Y) to (X', f', Y') is a pair (g, h) where $g\colon X \to X'$ is a morphism of $\mathcal{D}$ and $h\colon Y \to Y'$ is a morphism of $\mathcal{E}$ such that the following commutes in $\mathcal{C}$.

$$\begin{array}{ccc} FX & \xrightarrow{Fg} & FX' \\ {\scriptstyle f}\downarrow & & \downarrow{\scriptstyle f'} \\ GY & \xrightarrow[Gh]{} & GY' \end{array}$$

- The identity morphism $\mathrm{id}_{(X,f,Y)}$ of an object (X, f, Y) is simply pair $(\mathrm{id}_X, \mathrm{id}_Y)$.
- Composition $(g', h') \circ (g, h)$ is defined by $(g' \circ g, h' \circ h)$. ◊

Definition 2.7.2 (Slice– and coslice categories)**.** Let $\mathbf{1}$ be the category consisting of a single object and its identity morphism. The diagonal functor $\Delta_X : \mathbf{1} \to \mathcal{C}$ for a given category $\mathcal{C}$ picks out a single object X of $\mathcal{C}$.

- The slice category $(\mathcal{C}/X)$ is the comma category $(\mathrm{Id}_{\mathcal{C}}{\downarrow}\Delta_X)$.
- The coslice category $(X/\mathcal{C})$ is the comma category $(\Delta_X{\downarrow}\mathrm{Id}_{\mathcal{C}})$. ◊

Objects of a slice category $(\mathcal{C}/X)$ are triples $(A, f, 1)$ where $f\colon A \to X$ and 1 is the single object of $\mathbf{1}$. Similarly, objects of a coslice category $(X/\mathcal{C})$ are triples $(1, f, A)$ with $f\colon X \to A$. Since 1 is the only object of $\mathbf{1}$ we can simply omit it from the triples. Morphisms from (A, f) to (B, g) in the slice category are tuples (h, id_1) with $h\colon A \to B$ a morphism of $\mathcal{C}$. Morphisms from (f, A) to (g, B) in the coslice category are tuples (id_1, h) again with $h\colon A \to B$ of $\mathcal{C}$. As before we can simply omit id_1, and thus morphisms of the slice and coslice categories $(\mathcal{C}/X)$ and $(X/\mathcal{C})$ are just morphisms of $\mathcal{C}$ such that the two equivalent diagrams on the left and respectively the two diagrams on the right commute.

$$\begin{array}{ccc} A & \xrightarrow{h} & B \\ & {\scriptstyle f}\searrow \quad \swarrow{\scriptstyle g} & \\ & X & \end{array} \qquad \begin{array}{ccc} \mathrm{Id}A & \xrightarrow{\mathrm{Id}h} & \mathrm{Id}B \\ {\scriptstyle f}\downarrow & & \downarrow{\scriptstyle g} \\ \Delta_X 1 & \xrightarrow[\Delta_X \mathrm{id}_1]{} & \Delta_X 1 \end{array} \qquad \begin{array}{ccc} \Delta_X 1 & \xrightarrow{\Delta_X \mathrm{id}_1} & \Delta_X 1 \\ {\scriptstyle f}\downarrow & & \downarrow{\scriptstyle g} \\ \mathrm{Id}A & \xrightarrow[\mathrm{Id}h]{} & \mathrm{Id}B \end{array} \qquad \begin{array}{ccc} & X & \\ & {\scriptstyle f}\swarrow \quad \searrow{\scriptstyle g} & \\ A & \xrightarrow[h]{} & B \end{array}$$

Chapter 3

Universal dialgebra

3.1 Dialgebras

Dialgebras generalise algebras and coalgebras. Algebras model a notion of composition whereas coalgebras model decomposition. A more commonly cited intuition for coalgebras is that they model behaviour and the state of a processes as they evolve over time. Coalgebras can model state, but algebras can be seen as such too. However, the evolution of a 'state' in an algebra occurs by providing input that may contain additional states, something that we otherwise refer to as composition. In coalgebras there is no notion of input, only output. The output may contain residual states, which is reminiscent of decomposition. Since dialgebras generalise algebras and coalgebras both intuitions apply. Formally, dialgebras are defined as follows.

Definition 3.1.1 (Dialgebra)**.** An (F, G)–dialgebra is a pair (X, χ) of an object $X : \mathcal{C}$ and a morphism $\chi : FX \to GX$ where $F : \mathcal{C} \to \mathcal{C}$ and $G : \mathcal{C} \to \mathcal{C}$ are functors. The object $X : \mathcal{C}$ is called the *carrier* and the morphism $\chi : FX \to GX$ is called the *structure* of the dialgebra. The category $\mathcal{C}$ is called the *base category*. ◊

An F-algebra is an (F, Id)–dialgebra and a G-coalgebra is an (Id, G)–dialgebra. For algebras and coalgebras there are appropriate notions of morphisms between them. Dialgebra morphisms generalise these algebra– and coalgebra morphisms. In particular, an F–algebra morphism is a (F, Id)–dialgebra morphism and a G–coalgebra morphism is (Id, G)–dialgebra morphism.

Definition 3.1.2 (Dialgebra morphism)**.** An (F, G)–dialgebra morphism $h : (X, \chi) \to (Y, \upsilon)$ is a morphism $h : X \to Y$ such that $Gh \circ \chi = \upsilon \circ Fh$.

$$\begin{array}{ccc} FX & \xrightarrow{Fh} & FY \\ {\scriptstyle\chi}\downarrow & & \downarrow{\scriptstyle\upsilon} \\ GX & \xrightarrow[Gh]{} & GY \end{array}$$

◊

For a dialgebra (X, χ) the identity morphism $\mathrm{id}_X : X \to X$ is clearly a dialgebra morphism. In addition, dialgebra morphisms may be composed. If $g : (X, \chi) \to (Z, \zeta)$ and $h : (Z, \zeta) \to (Y, \upsilon)$ are dialgebra morphisms then $h \circ g : (X, \chi) \to (Z, \zeta)$ is defined by composing the underlying morphisms $g : X \to Y$ and $h : Y \to Z$ of $\mathcal{C}$. It is easily checked that the composition laws of $\mathcal{C}$ carry over to dialgebra morphisms. Therefore any pair of functors $F : \mathcal{C} \to \mathcal{C}$ and $G : \mathcal{C} \to \mathcal{C}$

defines a category of (F, G)–dialgebras, which we denote by $\mathcal{C}_G^F$.

Given a category $\mathcal{C}_G^F$ we let $\mathrm{U}\colon \mathcal{C}_G^F \to \mathcal{C}$ denote the forgetful functor that takes dialgebras to their carriers $(X, \chi) \mapsto X$ and dialgebra morphisms to the underlying morphisms, that is $(f\colon (X,\chi) \to (Y,\upsilon)) \mapsto (f\colon X \to Y)$. However, we usually denote the underlying morphism of a dialgebra morphism $f\colon (X,\chi) \to (Y,\upsilon)$ by $f\colon X \to Y$ rather than $\mathrm{U}f\colon X \to Y$. To prevent confusion we mention the category explicitly, thus f of $\mathcal{C}_G^F$ versus f of $\mathcal{C}$.

Since dialgebras generalise algebras and coalgebras they model composition and input but also decomposition and behaviour over time. This makes dialgebras well–suited for modelling processes that allow input and output, as proposed in [4]. For example, let X be a collection of elements that are to describe processes, equipped with a dialgebra structure $\delta\colon C \times X \to D \times X$. A tuple $(c, x) \in C \times X$ models a user command c that is to be issued to the process $x \in X$. The structure δ models the computation steps so that $\delta(c, x)$ yields a pair $(d, x') \in D \times X$ that models the output d sent back to the user alongside a residual state $x' \in X$. Note that this is only a simple example. Not only input of constants can be modelled. We may also model passing states as input, for example as a dialgebra structure $\xi\colon X \times C \times X \to D \times X$.

Closure

Intuitions of both algebras and coalgebras may apply to dialgebras, but with the added expressivity some properties of algebras, or coalgebras, are lost. One of those properties is closure. A set $A \subseteq X$ is closed under a functor $F\colon \mathrm{Set} \to \mathrm{Set}$ if $FA \subseteq A$. More generally, it is closed under an algebraic structure $\sigma\colon FX \to X$ if the image $\sigma[FA]$ is a subset of A.

Algebras are necessarily closed under their structure. The structure α of an F–algebra (A, α) is a morphism $\alpha\colon FA \to A$ and thus by virtue of its type $\alpha[FA] \subseteq A$. The carrier of a dialgebra however *need not* be closed under the dialgebra structure. For example, consider a Set_G^F–dialgebra (X, δ) with $FX \coloneqq X^2$ and $GX \coloneqq X + 1$. Its structure may take $(x_1, x_2) \mapsto *$ where $*$ is the element $* \in 1$ not in X. Then obviously $\delta(x_1, x_2) \notin X$ and $\delta[FX] \not\subseteq X$.

A *partial* F–algebra on a set is an algebra (A, α) except that $\alpha\colon FA \to A$ is a *partial* function. Now, a partial algebra structure $\alpha\colon FA \to A$ can be extended to a dialgebra structure $\alpha'\colon FA \to GA$ where $GX \coloneqq X + 1$ by letting α' take $x \mapsto (\iota_1 \circ \alpha)(x)$ if $\alpha(x)$ is defined and $x \mapsto \iota_2(*)$ otherwise.

For example, the set $\mathbb{N}$ of natural numbers is an F–algebra $(\mathbb{N}, \alpha)$ where $FX \coloneqq 1 + X$. An element $a \in F\mathbb{N}$ is either $\iota_1(*)$ for $*$ the only element of 1 or it is $\iota_2(n)$ for $n \in \mathbb{N}$; we typically denote them by Z and $\mathsf{S}n$. The structure $\alpha\colon F\mathbb{N} \to \mathbb{N}$ takes $\mathsf{Z} \mapsto 0$ and $\mathsf{S}n \mapsto n + 1$ where (0) and $(+)$ are to be interpreted in $\mathbb{N}$ as usual. The set $\mathbb{N}$ is closed under these operations: for all $n \in \mathbb{N}$ we have $\alpha(\mathsf{Z}) \in \mathbb{N}$ and $\alpha(\mathsf{S}n) \in \mathbb{N}$. Now consider an additional operator to denote the predecessor of a number. We define $HX \coloneqq 1 + X + X$ and use the notation $H\mathbb{N} = \{\mathsf{Z}, \mathsf{P}n, \mathsf{S}n \mid n \in \mathbb{N}\}$. We would define ξ by extending α with $\mathsf{P}n \mapsto n - 1$. However, $0 - 1 \notin \mathbb{N}$ so we would end up with a partial function. Instead, we define $GX \coloneqq X + 1$ and we define $\xi\colon H\mathbb{N} \to G\mathbb{N}$ by $\mathsf{P}(n+1) \mapsto \iota_1(n)$, $\mathsf{P}0 \mapsto \iota_2(*)$ and $a \mapsto \iota_1(\alpha(a))$ otherwise. We have an example of a dialgebra $(\mathbb{N}, \xi)$ and a method to turn partial algebras into dialgebras.

Density

A concept dual to a closed set is sometimes referred to as a dense set [8]. A set Z is dense under a functor $F\colon \mathrm{Set} \to \mathrm{Set}$ if $Z \subseteq FZ$. More generally, it is dense under a coalgebraic structure $\sigma\colon X \to FX$ if $Z \subseteq \sigma^{-1}[FZ]$. Dual to algebras, coalgebras are necessarily dense under their structure. For, the structure ζ of an F–coalgebra (Z, ζ) is a morphism $\zeta\colon Z \to FZ$ and as such $Z \subseteq \zeta^{-1}[FZ]$.

The carrier of a dialgebra however is generally *not* dense under its structure. We are not aware of a familiar concept dual to partial algebras. However, the technique that we described in the previous subsection is easily dualised. For example, consider a structure (Z, ζ) that is a 'partial' G–coalgebra in the sense that Z is *almost* dense under ζ but still, $Z \not\subseteq \zeta^{-1}[FZ]$. We can complete (Z, ζ) to a Set^F_G dialgebra (Z, ξ) by defining $C \coloneqq \zeta^{-1}[FZ] \setminus Z$ and $FX \coloneqq C + X$. We let ξ take $\iota_1(c) \mapsto c$ and $\iota_1(z) \mapsto \zeta(z)$. Then (Z, ξ) is a dialgebra.

Graphs and hypergraphs

Coalgebras are often used for representing transition graphs. For example, consider a G–coalgebra $(X,)$ where $GX \coloneqq \mathscr{P}(C \times X)$ for some constant set C. This encodes a directed labelled graph (X, E) where X is the set of vertices and $E \subseteq X \times C \times X$ is the set of edges defined by $\{(x, c, y) \mid x \in X \text{ and } (c, y) \in \chi(x)\}$, which is just another way to encode the graph $\mathrm{G}_\chi \coloneqq \{(x, \chi(x)) \mid x \in X\}$ of χ.

This perspective on coalgebras extends to dialgebras. In particular, dialgebras can be seen as transition *hyper*graphs. A hypergraph is a graph in which a single edge can connect an arbitrary number of vertices. Thus a plain (unlabelled, unordered) hypergraph is a pair (X, E) of a set of vertices X and a set of edges $E \subseteq \mathscr{P}X \times \mathscr{P}X$.

A dialgebra (X, ξ) with $\xi\colon X^2 \to \mathscr{P}(C \times X)$ can be seen as a special hypergraph (X, E) where $E \subseteq X^2 \times C \times X$ is defined by $\{((x, y), c, z) \mid (x, y) \in X^2 \text{ and } (c, z) \in \xi(x, y)\}$ in line with the coalgebra example. This encoding of the edges is somewhat arbitrary and one might indeed prefer to define $E \coloneqq \mathrm{G}_\xi$.

Restricting a class of algebras

Consider a carrier X equipped with an algebra structure $\chi\colon FX \to X$. Algebra morphisms provide us with a class of algebras that agree with the structure of (X, χ). We will discuss this in more detail later but for now let us consider algebras (Y, υ) such that there is an algebra morphism $h\colon (X, \chi) \to (Y, \upsilon)$. Even though the existence of an algebra morphism to (Y, υ) ensures that (Y, υ) is compatible with the structure χ on X, we may still want to further restrict this class of algebras.

By extending (X, χ) to a dialgebra it is possible to impose a partitioning on X and consider a class algebras that is compatible with both the partitioning and the structure χ. Assume a partitioning on X has been defined via a morphism $q\colon X \twoheadrightarrow C$, representing each partition by an element $c \in C$. We can extend the algebra structure $\chi\colon FX \to X$ to a dialgebra structure $\xi\colon FX \to X \times C$ via $\xi \coloneqq (\chi, q \circ \chi)$. Then any dialgebra (Y, υ) that is the codomain of a dialgebra morphism $h\colon (X, \xi) \to (Y, \upsilon)$ agrees with the partitioning q on X. In addition, given any such dialgebra morphism we can retrieve an algebra morphism $h\colon (X, \chi) \to (Y, \pi_1 \circ \upsilon)$

so that the algebra structure $\pi_1 \circ \upsilon$ is compatible with the algebra structure χ on X whilst respecting the partitioning.

Restricting a class of coalgebras

Consider a G–coalgebra (X, χ) on a set of states X. Coalgebra morphisms $m : (Y, \upsilon) \to (X, \chi)$ provide us with a class of coalgebras (Y, υ) that are compatible with the structure on X. As with the previous subsection, in some situations we would like to further restrict this class of coalgebras. By extending (X, χ) to a dialgebra, it is possible to enforce that a subobject of the carrier X is in the image of any of the morphisms $h : (Y, \upsilon) \to (X, \chi)$, effectively imposing a lower bound on the size Y. Incidentally, generated subcoalgebras 'rooted' at a particular state can be obtained using this very technique.

Consider a Set_H coalgebra (X, χ). We can extend its structure χ to a dialgebra structure $\delta : FX \to GX$ where $FX := 1 + X$ and $GX := \mathscr{P}X + HX$. Let δ take $\iota_1(*) \mapsto X'$ and $\iota_2(x) \mapsto \iota_2(\chi(x))$ where X' is a subset of X that we wish to enforce. We can now consider the class of dialgebras (Y, ξ) such that there is a morphism $h : (Y, \xi) \to (X, \delta)$. This ensures that for all such dialgebras (Y, ξ) we have $h^{-1}[X'] \subseteq Y$. And finally, from any such dialgebra (Y, ξ) we can retrieve a coalgebra (Y, υ) by letting υ take $y \mapsto \xi(\iota_2(y))$.

Multi–dialgebras

The previous subsections on restricting classes of algebras and coalgebras use a technique that is an instance of a more general scheme. Since dialgebras generalise algebras and coalgebras they allow both algebra structures and coalgebra structures to be expressed at once. More generally, dialgebra structures themselves may be composed of algebra–, coalgebra and dialgebra structures. Extending a structure with an additional algebra–, coalgebra or dialgebra structure is a powerful technique for restricting a class of dialgebras as sketched in the previous examples. The technique is best explained through a variant of dialgebras that we call multi–dialgebras.

Definition 3.1.3 (Multi–dialgebra)**.** Let $(F_i, G_i)_i$ be a family of pairs of functors $F_i : \mathcal{C} \to \mathcal{C}$ and $G_i : \mathcal{C} \to \mathcal{C}$. An $(F_i, G_i)_i$ dialgebra is a pair $(X, (\chi_i)_i)$ of a carrier $X : \mathcal{C}$ and a family of dialgebra structures $(\chi_i : F_i \to G_i)_i$. A morphism $h : (X, (\chi_i)_i) \to (Y, (\upsilon_i)_i)$ of multi–dialgebras is a morphism $h : X \to Y$ where $h : (X, \chi_i) \to (Y, \upsilon_i)$ is a dialgebra morphism for all indices i.

$$\begin{array}{ccc} F_iX & \xrightarrow{F_ih} & F_iY \\ {\scriptstyle \chi_i}\downarrow & & \downarrow{\scriptstyle \upsilon_i} \\ G_iX & \xrightarrow[G_ih]{} & G_iY \end{array}$$

◊

Intuitively, multi–dialgebra morphisms are more restricted than dialgebra morphisms. Indeed, multi–dialgebra morphisms $h : (X, (\chi_1, \chi_2)) \to (Y, (\upsilon_1, \upsilon_2))$ are those morphisms $h : X \to Y$ such that both $h : (X, \chi_1) \to (Y, \upsilon_1)$ and $h : (X, \chi_2) \to (Y, \upsilon_2)$ are dialgebra morphisms. However, any category of multi–dialgebras embeds into a category of (F, G)–dialgebras for a suitable pair of functors F and G.

Proposition 3.1.4. Consider a family of pairs of functors $(F_i, G_i)_i$ on a base category $\mathcal{C}$ and let $\mathcal{D}$ denote the category of $(F_i, G_i)_i$ multi–dialgebras.

1. Then $\mathcal{D}$ is a full subcategory of $\mathcal{C}_G^F$ where $FX \coloneqq \coprod_i F_i X$ and $GX \coloneqq \coprod_i G_i X$.
2. In addition $\mathcal{D}$ is a full subcategory of $\mathcal{C}_G^F$ where $FX \coloneqq \prod_i F_i X$ and $GX \coloneqq \prod_i G_i X$.

Proof. We consider the first item shortly. We have to show that there is a faithful inclusion functor $I : \mathcal{D} \hookrightarrow \mathcal{C}_G^F$. Thus let I take objects $(X, (\chi)_i) \mapsto (X, \coprod_i \chi_i)$ and morphisms $h \mapsto h$. Then $(Ih) : (X, \coprod_i \chi_i) \to (Y, \coprod_i \upsilon_i)$ is a morphism of $\mathcal{C}_G^F$ because products and likewise coproducts are computed pointwise. In particular, for every a family of morphisms $((\iota_i \circ \upsilon_i) : FY \to GY)_i$ there is a unique morphism from FY to GY by the definition of coproduct. This unique morphism is the the copairing $[\iota_i \circ \upsilon_i]_i$, which is $\coprod_i \upsilon_i$. Likewise $[\iota_i \circ F_i h]_i = Fh$, $[\iota_i \circ G_i h]_i = Gh$ and $[\iota_i \circ \chi_i]_i = \coprod_i \chi_i$. □

Since categories of multi–dialgebras embed into categories of dialgebras we will only work with the latter. We can at any point use the proposition above to combine multiple dialgebra structures on the same carrier into one.

3.2 Limits and colimits

Algebras have products and coalgebras are have coproducts. In particular, limits in a category $\mathcal{C}^F$ of algebras are obtained by equipping limits of $\mathcal{C}$ with appropriate algebra structures. For example, consider two algebras (A, α) and (B, β) of $\mathcal{C}^F$ and assume that $\mathcal{C}$ has products. By definition, the product $A \times B$ is the carrier of the unique algebra structure $k = (\alpha \circ F\pi_A,\ \beta \circ F\pi_B)$

$$\begin{array}{ccccc} FA & \xleftarrow{F\pi_A} & F(A \times B) & \xrightarrow{F\pi_B} & FB \\ \downarrow{\scriptstyle \alpha} & & \downarrow{\scriptstyle k} & & \downarrow{\scriptstyle \beta} \\ A & \xleftarrow[\pi_A]{} & A \times B & \xrightarrow[\pi_B]{} & B \end{array}$$

In addition $(A \times B, k)$ is the product of (A, α) and (B, β) in $\mathcal{C}^F$. For consider an algebra (X, χ) and two algebra morphisms $f : (X, \chi) \to (A, \alpha)$ and $g : (X, \chi) \to (B, \beta)$. By the definition of products there is a unique morphism $u : X \to A \times B$ such that $\pi_A \circ u = f$ and $\pi_B \circ u = g$ in $\mathcal{C}$, which extends to a unique algebra morphism $u : (X, \chi) \to (A \times B, k)$.

Under the assumption that F preserves colimits there is a similar result for coproducts of algebras. If (A, α) and (B, β) are F–algebras and F preserves coproducts then $F(A + B)$ is the coproduct of FA and FB and $(A + B, k)$ is the coproduct of (A, α) and (B, β) where $k : F(A + B) \to A + B$ is the unique morphism $[\iota_X \circ \alpha,\ \iota_Y \circ \beta]$.

The arguments can be generalised to other limits and colimits without much effort and are easily dualised for coalgebras to obtain the results of [20] section 4. By combining the dual results we obtain the following theorem for limits and colimits of dialgebras, restating theorem 13 of [23].

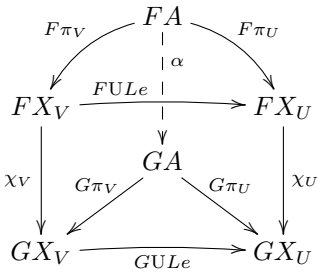
Theorem 3.2.1.

1. If $\mathcal{C}$ has limits of shape $\mathcal{J}$ and G preserves them then $\mathcal{C}_G^F$ has limits of shape $\mathcal{J}$.
2. If $\mathcal{C}$ has colimits of shape $\mathcal{J}$ and F preserves them then $\mathcal{C}_G^F$ has colimits of shape $\mathcal{J}$.

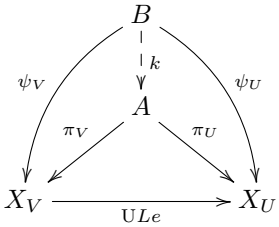
Proof. We prove the first item. Let $L : \mathcal{J} \to \mathcal{C}^F_G$ be a diagram in $\mathcal{C}^F_G$. Then $\mathrm{U}L : \mathcal{J} \to \mathcal{C}$ is a diagram in $\mathcal{C}$ and so are $F\mathrm{U}L$ and $G\mathrm{U}L$. For clarity, if L labels a vertex $V : \mathcal{J}$ with (X, χ) then $\mathrm{U}L$ labels V with X, $F\mathrm{U}L$ labels V with FX and $G\mathrm{U}L$ labels V with GX. For readability we shall write (X_V, χ_V) for the dialgebra LV where $V : \mathcal{J}$.

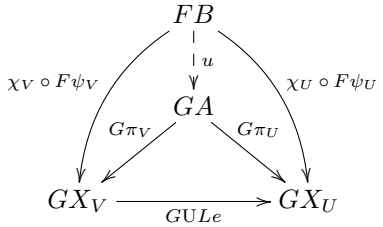
Let (A, π) be the limit of $\mathrm{U}L$ (in $\mathcal{C}$) and recall that π is a family of morphisms $(\pi_V : A \to X_V)_{V:\mathcal{J}}$. We wish to show that there is a unique dialgebra structure $\alpha : FA \to GA$ such that the projections $\pi_V : A \to X_V$ extend to dialgebra morphisms $\pi_V : (A, \alpha) \to (X_V, \chi_V)$ and that $((A, \alpha), \pi)$ is the limit of L in $\mathcal{C}^F_G$.

Let us write $G\pi$ for the family of morphisms $(G\pi_V : GA \to GX_V)_{V:\mathcal{J}}$. Observe that $(GA, G\pi)$ is the initial cone over $G\mathrm{U}L$ (in $\mathcal{C}$) because G preserves limits. Now consider the cone $(FA, F\pi)$ over $F\mathrm{U}L$ (in $\mathcal{C}$). It extends to a cone $(FA, \chi_V \circ F\pi_V : FA \to GX_V)_{V:\mathcal{J}}$ over $G\mathrm{U}L$. Since $(GA, G\pi)$ is the limit of $G\mathrm{U}L$ there is a unique morphism from $(FA, \chi_V \circ F\pi_V)_{V:\mathcal{J}}$ to $(GA, G\pi)$. By definition this is a morphism $\alpha : FA \to GA$ in $\mathcal{C}$ such that the projections $\pi_V : A \to X_V$ extend to dialgebra morphisms $\pi_V : (A, \alpha) \to (X_V, \chi_V)$. For every for every arrow $e : V \to U$ in $\mathcal{J}$ the following diagram in $\mathcal{C}$ commutes.



It remains to show that $((A, \alpha), \pi)$ is the limit of L in $\mathcal{C}^F_G$. So consider any other cone $((B, \beta), \psi)$ with ψ a family of morphisms $(\psi_V : (B, \beta) \to (X_V, \chi_V))_{V:\mathcal{J}}$. Since $(A, \pi_V : A \to X_V)_{V:\mathcal{J}}$ is the limit of $\mathrm{U}L$ (in $\mathcal{C}$) there is a unique morphism of cones from $(B, \psi_V : B \to X_V)_{V:\mathcal{J}}$ to (A, π) thus there is a unique morphism $k : B \to A$ in $\mathcal{C}$. Consider the cone $(FB, \chi_V \circ F\varphi_V)_{V:\mathcal{J}}$ over $G\mathrm{U}L$. Recall that $(GA, G\pi)$ is the initial cone over $G\mathrm{U}L$, so that there is a unique morphism $u : FB \to GA$ in $\mathcal{C}$ and thus $Gk \circ \beta = u = \alpha \circ Fk$ which makes $k : (B, \beta) \to (A, \alpha)$ and $\varphi_V : (B, \beta) \to (X_V, \chi_V)$ for $V : \mathcal{J}$ dialgebra morphisms.





□

In summary, the limits of $\mathcal{C}$ that are preserved by F can be equipped with a unique dialgebra structure to obtain limits in $\mathcal{C}^F_G$, and colimits in $\mathcal{C}$ that are preserved by G extend uniquely to colimits in $\mathcal{C}^F_G$. Now as indicated by [19, 20] in a category $\mathcal{C}_G$ of coalgebras limits of $\mathcal{C}$ that are not preserved by G may exist. However they are not obtained just by equipping the limit in $\mathcal{C}$ with a coalgebra structure. It may very well be possible to to dualise the results in [19] and obtain the more general results about limits and colimits in dialgebras that are not obtained by equipping limits and colimits in the base category with a dialgebra structure.

3.3 Subdialgebras and quotient dialgebras

Intuitively both subobjects of– and quotients of an object are 'smaller versions' of that object. Their smallness however is realised in distinct ways. We motivate this intuition, but we use the notion of elements of objects as we have in Set. Now, the codomain of a surjective function $f : X \to Y$ may be smaller than its domain, for there may be elements $x_1, x_2 \in X$ that are identified via $f(x_1) = f(x_2)$. Even if X and Y have the same cardinality an intuition of smallness applies, in the sense that Y can be obtained from X by merging and renaming elements. On the other hand, the domain of an injective function $g : X \to Y$ may be smaller than its codomain, for there may be elements $y \in Y$ for which there is no $x \in X$ with $g(x) = y$. Again, even if X and Y have the same cardinality an intuition of smallness applies, for X may be obtained by renaming elements of Y while discarding those that are not in the image of f. Thus the distinct notions of smallness are distinguished by considering surjective functions and injective functions separately. The approach is generalised to other categories as follows.

In general, the category $\mathrm{Sub}\mathcal{C}$ of subobjects of a category $\mathcal{C}$ is a suitably chosen subcategory of $\mathrm{Mono}\mathcal{C}$. Dually, the category of quotients $\mathrm{Quot}\mathcal{C}$ is a suitably chosen subcategory of $\mathrm{Epi}\mathcal{C}$. The chosen subcategories depend on $\mathcal{C}$. In some cases $\mathrm{Sub}\mathcal{C}$ is $\mathrm{Mono}\mathcal{C}$ and $\mathrm{Quot}\mathcal{C}$ is $\mathrm{Epi}\mathcal{C}$ but for some categories this does not yield the desired result. As may be expected from the introduction above, for Set it does. The category SubSet is MonoSet in its entirety. Likewise, QuotSet is EpiSet. For a category $\mathcal{C}_G^F$ of dialgebras the categories $\mathrm{Sub}(\mathcal{C}_G^F)$ and $\mathrm{Quot}(\mathcal{C}_G^F)$ are defined as follows.

Definition 3.3.1 (Categories of subdialgebras and quotient dialgebras)**.**

1. An object (X, χ) of $\mathrm{Sub}(\mathcal{C}_G^F)$ is an object of $\mathcal{C}_G^F$.
2. A morphism $m : (Y, \upsilon) \to (X, \chi)$ of $\mathrm{Sub}(\mathcal{C}_G^F)$ is a morphism of $\mathcal{C}_G^F$ where $m : Y \to X$ is a morphism of $\mathrm{Sub}\mathcal{C}$.

1. An object (X, χ) of $\mathrm{Quot}(\mathcal{C}_G^F)$ is an object of $\mathcal{C}_G^F$.
2. A morphism $e : (X, \chi) \to (Y, \upsilon)$ of $\mathrm{Quot}(\mathcal{C}_G^F)$ is a morphism of $\mathcal{C}_G^F$ where $e : X \to Y$ is a morphism of $\mathrm{Quot}\mathcal{C}$. ◊

Note that $\mathrm{Sub}(\mathcal{C}_G^F)$ is indeed a subcategory of $\mathrm{Mono}(\mathcal{C}_G^F)$, dually, $\mathrm{Quot}(\mathcal{C}_G^F)$ is a subcategory of $\mathrm{Epi}(\mathcal{C}_G^F)$. If $e : (X, \chi) \to (Y, \upsilon)$ is a morphism of $\mathrm{Quot}(\mathcal{C}_G^F)$ and g and h of $\mathcal{C}_G^F$ are such that $g \circ e = h \circ e$ then $g = h$ in $\mathcal{C}$ because $e : X \to Y$ is in $\mathrm{Quot}\mathcal{C}$ and thus epic.

This definition may be surprising in that the *objects* of the categories $\mathrm{Sub}\mathcal{C}$ and $\mathrm{Quot}\mathcal{C}$ are just objects of $\mathcal{C}$. Instead, the we refer to the *morphisms* of $\mathrm{Sub}\mathcal{C}$ as subobjects and to the morphisms of $\mathrm{Quot}\mathcal{C}$ as quotients. This is convenient if we consider subobjects and quotients of a particular object $X : \mathcal{C}$. For, a subobject *of* an object $X : \mathcal{C}$ is an *object* $m : A \rightarrowtail X$ of the slice category $(\mathrm{Sub}\mathcal{C}/X)$ and a quotient of $X : \mathcal{C}$ is an object $e : X \twoheadrightarrow Z$ of the coslice category $(X/\mathrm{Quot}\mathcal{C})$. However, if clear from the context we refer to an *object* $A : \mathcal{C}$ as a subobject of $X : \mathcal{C}$ if there is a morphism $m : A \rightarrowtail X$ in $\mathrm{Sub}\mathcal{C}$ and similarly for quotients.

Definition 3.3.2 (Subobjects and quotients)**.**

1. A *subobject* is a morphism of $\mathrm{Sub}\mathcal{C}$.
2. A *subobject of* $X : \mathcal{C}$ is an object $m : A \rightarrowtail X$ of $(\mathrm{Sub}\mathcal{C}/X)$.
1. A *quotient* is a morphism of $\mathrm{Quot}\mathcal{C}$.
2. A *quotient of* $X : \mathcal{C}$ is an object $e : X \twoheadrightarrow Y$ of $(X/\mathrm{Quot}\mathcal{C})$. $\Diamond$

Subdialgebras and quotient dialgebras are defined accordingly. A subdialgebra of a $\mathcal{C}_G^F$ dialgebra (X, χ) is an object $m : (A, \alpha) \rightarrowtail (X, \chi)$ of $(\mathrm{Sub}(\mathcal{C}_G^F)/(X, \chi))$. A quotient dialgebra of (X, χ) is an object $e : (X, \chi) \twoheadrightarrow (Z, \zeta)$ of $((X, \chi)/\mathrm{Quot}(\mathcal{C}_G^F))$.

Semilattices

If a category of subobjects has pullbacks then subobjects form a meet semilattice, likewise quotients form a join semilattice if their category has pushouts. We shortly investigate pullbacks of subdialgebras and pushouts of quotient dialgebras. The following well known lemma allows a more specific analogue of theorem 3.2.1 for pullbacks of subdialgebras and pushouts of quotient dialgebras.

Proposition 3.3.3.

1. Let (P, π_X, π_Z) be the pullback of a pair of morphisms $f : X \to Y$ and $g : Z \to Y$. If g is monic then $\pi_X : P \to X$ monic. If in addition f is monic then $f \circ \pi_X = g \circ \pi_Y$ is monic.
2. Let (ι_Z, ι_Y, P) be the pushout of a pair of morphisms $f : X \to Y$ and $g : X \to Z$. If g is epic then $\iota_Y : Y \to P$ is epic. If in addition f is epic then $f \circ \pi_X = g \circ \pi_Y$ is epic.

Proof. We prove the second item. Consider two arbitrary functions $h : P \to Q$ and $k : P \to Q$ such that $h \circ \iota_Y = k \circ \iota_Y$. Then certainly $h \circ \iota_Y \circ f = k \circ \iota_Y \circ f$. Then $h \circ \iota_Z \circ g = k \circ \iota_Z \circ g$ because $\iota_Y \circ f = \iota_Z \circ g$. But then $h \circ \iota_Z = k \circ \iota_Z$ because g is epic. We already had $h \circ \iota_Y = k \circ \iota_Y$. In proposition 3.6.2 we will show that injections are jointly which allows us to conclude that $h = k$, thus ι_Y is an epimorphism. Finally, if in addition f is epic then $\iota_Y \circ f = \iota_Z \circ g$ is epic because epimorphisms are closed under composition. $\square$

This proposition establishes that if a category $\mathcal{C}$ has pullbacks then $\mathrm{Mono}\mathcal{C}$ has pullbacks. Likewise if $\mathcal{C}$ has pushouts then so does $\mathrm{Epi}\mathcal{C}$. In fact, it is reasonable to require that $\mathrm{Sub}\mathcal{C}$ has pullbacks if $\mathcal{C}$ has them and that $\mathrm{Quot}\mathcal{C}$ has pushouts if $\mathcal{C}$ does. Henceforth, if keep the base category abstract, then we assume that these requirements are met. Using theorem 3.2.1 we now obtain the following.

Proposition 3.3.4. Consider a category $\mathcal{C}_G^F$ of dialgebras.

1. If G preserves pullbacks then subdialgebras form a meet semilattice under pullback.

   If $\mathcal{C}$ is Set and G preserves pullbacks then subdialgebras are closed under intersection.
2. If F preserves pushouts then quotient dialgebras form a join semilattice under pushout.

Note that in Set subobjects are isomorphic to inclusions and the pullback of two inclusions $m : X \hookrightarrow Z$ and $n : Y \hookrightarrow Z$ is the inclusion $d : X \cap Y \hookrightarrow Z$ where $d := m \circ \pi_X = n \circ \pi_Y$. Thus if $m : (X, \chi) \hookrightarrow (Z, \zeta)$ and $n : (Y, \upsilon) \hookrightarrow (Z, \zeta)$ are subdialgebras and G preserves pullbacks then d extends uniquely to a subdialgebra of (Z, ζ).

The above does not in general imply that subdialgebras do not have a meet if G does not preserve pullbacks. The remark that we made near theorem 3.2.1 applies here as well. If G preserves pullbacks then the pullback of subdialgebras is obtained from the pullback in the base category by extending it with a unique dialgebra structure. But if G does not preserve pullbacks then pullback of subdialgebras may still exist. However it is not obtained simply by taking the pullback of the underlying morphisms.

Quotient dialgebras as semantics

Consider a set of terms X equipped with a dialgebraic operational semantics $\delta : FX \to GX$. In Set every equivalence relation is essentially a surjective function. The category of quotients $(X/\mathrm{Quot}\,\mathrm{Set})$ is essentially the category of equivalence relations on the set of terms. By considering $((X,\delta)/\mathrm{Quot}\,(\mathrm{Set}^F_G))$ rather than $(X/\mathrm{Quot}\,\mathrm{Set})$ we restrict attention to equivalences that are compatible with the operational semantics, thus we restrict attention to behavioural equivalences on terms.

We assume that we are interested only in denotational semantics that discern distinct behaviour. Under this assumption the quotients of (X,δ) are its suitable denotational semantics. Perhaps a more striking way to express this is that if we wish to define a denotational semantics for (X,δ) then $((X,\delta)/\mathrm{Quot}\,(\mathrm{Set}^F_G))$ is the solution space. Defining a semantics then involves selecting an object of this category. Since good design is about the absence of arbitrary design decisions we can use the structure of the solution space and select a canonical object. In particular, we are interested in the final object of the category. We discuss such objects in the next section and in chapter 4.

Subdialgebras as denotations

Quotients may be small intuitively, but for nontrivial languages they are infinite. Any interesting language has infinitely many behaviourally distinct terms, therefore all its quotients have infinitely many elements. However, we are often not interested in the entire semantic domain at once. We can restrict attention to its subdialgebras. In particular, subdialgebras of the semantic domain provide small denotations for terms.

Assume that we have settled on a quotient $[\![\_]\!] : (X,\delta) \to (Z,\zeta)$ of an operational semantics. Then two terms $p, q \in X$ are semantically equivalent if $[\![p]\!] = [\![q]\!]$. However, if Z is infinite then we can not naively check whether this is the case. Instead we need finite, decidable representations of $[\![p]\!]$ and $[\![q]\!]$. What we can do is interpret terms not as elements of Z but as subdialgebras. If a semantics is at all decidable then it may be possible to construct for each element of the semantic domain a finite subdialgebra that uniquely represents it. Moreover, we may use the technique outlined in section 3.1 to help us with this task. Consider a term $p \in X$ and its interpretation $[\![p]\!] \in Z$. We can construct for p a class of possible denotations by extending (Z,ζ) to a dialgebra $(Z,\, p_{[\![p]\!]} + \zeta)$ where $p_{[\![p]\!]} : 1 \to Z$ takes $* \mapsto [\![p]\!]$. Now $(\mathrm{Sub}\,(\mathrm{Set}^F_G)/(Z,\, p_{[\![p]\!]} + \zeta))$ is a category of denotations for p. For clarity, objects of this category are subdialgebras of $(Z,\, p_{[\![p]\!]} + \zeta)$ that are enforced to contain at least $[\![p]\!]$. In typical situations the initial object of this category is an appropriate and finite denotation for p that may be compared against similar denotations of other terms. We will pay special attention to such objects in the the next section and we investigate how to construct subdialgebras whilst ensuring that they are suitable denotations in chapter 4.

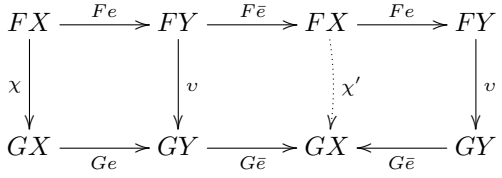
3.4 Factorisations

A factorisation of a morphism $f : X \to Y$ is an object Z and a pair of morphisms $e : X \to Z$ and $h : Z \to Y$ in $\mathcal{C}$ such that $f = h \circ e$. In this section we study whether factorisations in a base category extend to factorisations of dialgebras. We start with sections and retractions. In particular, if a dialgebra morphism has a section in the base category then that section extends to a dialgebra morphism.

Proposition 3.4.1.

1. If $e : (X, \chi) \to (Y, \upsilon)$ is a dialgebra morphism and $e : X \to Y$ has a section $\bar{e} : Y \rightarrowtail X$ then there is a dialgebra structure χ' such that $\bar{e} : (Y, \upsilon) \rightarrowtail (X, \chi')$ is a dialgebra morphism. Consequently $\bar{e} \circ e$ is a dialgebra morphism $(\bar{e} \circ e) : (X, \chi) \to (X, \chi')$.
2. If $m : (X, \chi) \to (Y, \upsilon)$ is a dialgebra morphism and $m : X \to Y$ has a retraction $\bar{m} : Y \twoheadrightarrow X$ then there is a dialgebra structure υ' such that $\bar{m} : (Y, \upsilon') \twoheadrightarrow (X, \chi)$ is a dialgebra morphism. Consequently $\bar{m} \circ m$ is a dialgebra morphism $(m \circ \bar{m}) : (X, \upsilon') \to (Y, \upsilon)$.

Proof. We prove the first item. Define $\chi' := G\bar{e} \circ \upsilon \circ Fe$. We have to show that $\chi' \circ F\bar{e} = G\bar{e} \circ \upsilon$. Now $\chi' \circ F\bar{e}$ is $G\bar{e} \circ \upsilon \circ Fe \circ F\bar{e}$ by definition of χ', which is $G\bar{e} \circ \upsilon$ because $\bar{e}$ is a retraction of e. For the second item define $\upsilon' := Gm \circ \chi \circ F\bar{m}$. Then $\chi \circ F\bar{m} = G\bar{m} \circ Gm \circ \chi \circ F\bar{m} = G\bar{m} \circ \upsilon'$.



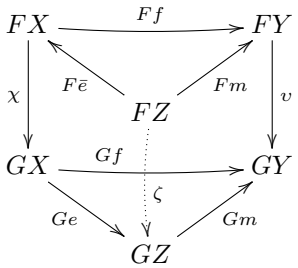
□

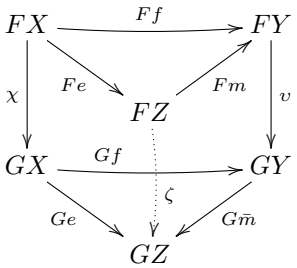
Factorisations $f = m \circ e$ in the underlying category of dialgebra morphisms extend to factorisations of dialgebra morphisms if e has a retraction or if m has a section.

Proposition 3.4.2. Let $f : (X, \chi) \to (Y, \upsilon)$ be a morphism of $\mathcal{C}_G^F$ such that $f = m \circ e$ in $\mathcal{C}$.

1. If e has a section then there are dialgebra structures χ' and ζ such that $e : (X, \chi') \to (Z, \zeta)$ and $m : (Z, \zeta) \to (Y, \upsilon)$ are dialgebra morphisms.
2. If m has a retraction then there are dialgebra structures ζ and υ' such that $e : (X, \chi) \to (Z, \zeta)$ and $m : (Z, \zeta) \to (Y, \upsilon')$ are dialgebra morphisms.
3. If e has a section and m has a retraction then there is a dialgebra structure ζ such that $e : (X, \chi) \to (Z, \zeta)$ and $m : (Z, \zeta) \to (Y, \upsilon)$ are dialgebra morphisms. If in addition F preserves epis and G preserves monos then there is a unique such structure.

Proof. Assume that $e : X \to Z$ has a section $\bar{e} : Z \rightarrowtail X$. Then $f \circ \bar{e} = m \circ e \circ \bar{e}$ and as such $f \circ \bar{e} = m$. Therefore we have $Ff \circ F\bar{e} = Fm$ and $Gf = Gm \circ Ge$. Since $f : (X, \chi) \to (Y, \upsilon)$ is a dialgebra morphism we have $\upsilon \circ Ff = Gf \circ \chi$. Define $\zeta : FZ \to GZ$ by $\zeta := Ge \circ \chi \circ F\bar{e}$. Then the diagram on the left commutes and $m : (Z, \zeta) \to (Y, \upsilon)$ is a dialgebra morphism.





The diagram on the right corresponds to the dual case.

In general it does not hold that $e\colon (X,\chi) \to (Z,\zeta)$ is a dialgebra morphism. However, we can define $\chi' \coloneqq G\bar{e} \circ \zeta \circ Fe$. Then $Ge \circ \chi' = Ge \circ G\bar{e} \circ \zeta \circ Fe = \zeta \circ Fe$ so that $e\colon (X,\chi') \to (Z,\zeta)$ *is* a dialgebra morphism.

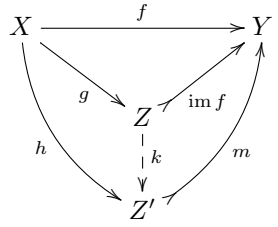
The second item is dual to the first. As for the third item, it follows from the above there are dialgebra structures ζ_1, ζ_2 such that $e\colon (X,\chi) \to (Z,\zeta_2)$ and $m\colon (Z,\zeta_1) \to (Y,\upsilon)$ are dialgebra morphisms. Then $Gm \circ \zeta_2 \circ Fe = Gm \circ Ge \circ \chi = Gf \circ \chi$ and $Gm \circ \zeta_1 \circ Fe = \upsilon \circ Fm \circ Fe = \upsilon \circ Ff$. Thus $Gm \circ \zeta_2 \circ Fe = Gm \circ \zeta_1 \circ Fe$ because f is a dialgebra morphism. But then $\zeta_2 = \zeta_1$ because Ge is epic and Gm is monic. □

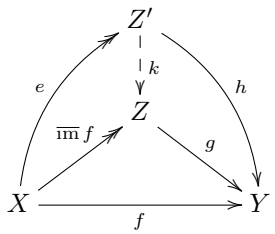
In the category Set every epimorphism has a section and every monomorphism has a retraction. Moreover, all functors $F\colon \mathrm{Set} \to \mathrm{Set}$ preserve epimorphisms and monomorphisms, except possibly for monomorphisms with domain $\emptyset$. Thus a factorisation $f = m \circ e$ in Set of a dialgebra morphism f extends to a unique factorisation in Set^F_G if e is epic and m monic, unless perhaps $X = \emptyset$.

We may use proposition 3.4.2 to define quotient dialgebras and subdialgebras by factorising dialgebra morphisms. Similar to limits and colimits there is a notion of a canonical factorisation. If we pay restrict attention to quotients and subobjects then we arrive at a generalisation of the set–theoretic image.

Definition 3.4.3 (Image and coimage)**.**

1. The *image* $(\mathrm{im}\, f)\colon Z \rightarrowtail Y$ of a morphism $f\colon X \to Y$ is the least subobject such that there is a factorisation $f = (\mathrm{im}\, f) \circ g$. Specifically, for any subobject $m\colon Z' \rightarrowtail Y$ and factorisation $f = m \circ h$ there is a unique morphism $k\colon Z \to Z'$ that makes the diagram on the left commute.





2. The *coimage* $(\overline{\mathrm{im}}\, f)\colon X \twoheadrightarrow Z$ of a morphism $f\colon X \to Y$ is the greatest quotient such that there is a factorisation $g \circ (\overline{\mathrm{im}}\, f) = f$. Specifically, for any quotient $e\colon X \twoheadrightarrow Z'$ and factorisation $h \circ e = f$ there is a unique morphism $k\colon Z' \to Z$ that makes the diagram on the right above commute. ◊

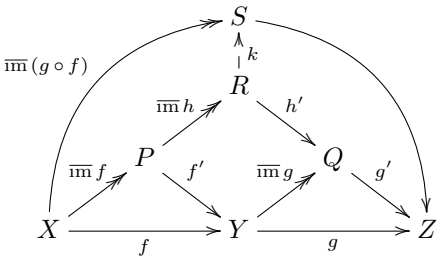
Thus the image of a morphism $f\colon X \to Y$ is the least subobject of Y through which f factorises. The coimage of f is the greatest quotient of X through which f factorises. Images and coimages generalise the set–theoretic image of a function. In particular, the image $\{f(x) \mid x \in X\}$ of a function f is isomorphic to the domain of $\mathrm{im}\, f$ which in Set is isomorphic to the codomain of $\overline{\mathrm{im}}\, f$.

Intuitively, a composition of morphisms $g \circ f$ identifies more than a morphism f alone. So one would expect that the coimage of a morphism $g \circ f$ is a quotient of the coimage of f. The following proposition confirms that this intuition is correct.

Proposition 3.4.4. Let $f : X \to Y$ and $g : Y \to Z$ be morphisms of a category $\mathcal{C}$.

1. If $\mathcal{C}$ has images then $\operatorname{im}(g \circ f)$ is a subobject of $\operatorname{im} g$.
2. If $\mathcal{C}$ has coimages then $\overline{\operatorname{im}}(g \circ f)$ is a quotient of $\overline{\operatorname{im}} f$.

Proof. We prove the second item. Consider $\overline{\operatorname{im}} f : X \twoheadrightarrow P$ and $\overline{\operatorname{im}} g : Y \twoheadrightarrow Q$. Let $f' : P \to Y$ and $g' : Q \to Z$ be such that $f' \circ \overline{\operatorname{im}} f = f$ and $g' \circ \overline{\operatorname{im}} g = g$. Define $h : P \to Q := \overline{\operatorname{im}} g \circ f'$ and consider consider $\overline{\operatorname{im}} h : P \twoheadrightarrow R$ and $h' : R \to Q$ such that $h' \circ \overline{\operatorname{im}} h' = h$.



Then $g' \circ h' \circ \overline{\operatorname{im}} h \circ \overline{\operatorname{im}} f = g \circ f$ and therefore there is a unique morphism $k : R \to S$ where S is the codomain of $\overline{\operatorname{im}}(g \circ f)$. The morphism $k \circ \overline{\operatorname{im}} h$ is a morphism from $\overline{\operatorname{im}} f$ to $\overline{\operatorname{im}}(g \circ f)$, that is, $\overline{\operatorname{im}}(g \circ f)$ is a quotient of $\overline{\operatorname{im}} f$. □

Images and coimages of a base category extend to images and coimages of dialgebra morphisms as indicated in proposition 3.4.2.

Proposition 3.4.5. Consider a category $\mathcal{C}^F_G$ of dialgebras.

1. If a dialgebra morphism f has an image factorisation $f = (\operatorname{im} f) \circ e$ in $\mathcal{C}$ and e has a section then f has an image factorisation in $\mathcal{C}^F_G$.
2. If a dialgebra morphism f has a coimage factorisation $f = m \circ (\overline{\operatorname{im}} f)$ in $\mathcal{C}$ and m has a retraction f has a coimage factorisation in $\mathcal{C}^F_G$.
3. Any category Set^F_G has image and coimage factorisations.

Semilattices

We may use proposition 3.4.2 to extend image and coimage factorisations in a base category to image and coimage factorisations of dialgebra morphisms. Recall that in section 3.3 we have discussed the meet of subdialgebras and the join of quotient dialgebras. The join of two subobjects and the meet of two quotients are not defined as colimits and limits. Instead they are canonical factorisations as follows.

Definition 3.4.6 (Join and meet)**.**

1. The join of two subobjects $m : A \rightarrowtail X$ and $n : B \rightarrowtail X$ is the least subobject $s : Y \to X$ such that there are morphisms g, h and factorisations $m = s \circ g$ and $n = s \circ h$
2. The meet of two quotients $e : X \twoheadrightarrow A$ and $f : X \twoheadrightarrow B$ is the greatest quotient $q : X \twoheadrightarrow Y$ such that there are morphisms g, h and factorisations $g \circ q = e$ and $h \circ q = f$. ◊

In the presence of coproducts and products the join of two subobjects m and n is $\mathrm{im}[m, n]$ and the meet of two quotients e and f is $\overline{\mathrm{im}}(e, f)$. This suggests the following.

Proposition 3.4.7. Consider a category $\mathcal{C}_G^F$ of dialgebras.

1. If F preserves coproducts and $\mathcal{C}_G^F$ has images then the join of two subdialgebras exists and gives rise to a subdialgebra of (X, χ). Consequently, if $\mathcal{C}$ is Set and F preserves coproducts then subdialgebras are closed under union.
2. If G preserves products and $\mathcal{C}_G^F$ has coimages then the meet of two quotient dialgebras exists and gives rise to a quotient of (X, χ).

Again, this does not suggest that subdialgebras do not have a join if F does not preserve coproducts. In fact, in chapter 4 we develop a technique for generating minimal dialgebras which may also be employed to generate the join of subdialgebras. However, if F does preserve coproducts then the join of two subdialgebras can simply be obtained by equipping the carrier of the join in the base category with a unique dialgebra structure.

3.5 Minimal and simple dialgebras

In universal algebra and coalgebra it is common to study initial and final objects. Indeed, in many categories of algebras and coalgebras initial and final objects exist. In categories of dialgebras the situation is more delicate. In fact, many categories of dialgebras do not have a final object. Intuitively, all dialgebras of such categories can be extended with elements that are behaviourally distinct from all others so that the cardinality the carriers is not bounded.

Proposition 3.5.1. The category Set_G^F with $FX := X^2$ and $GX := 2$ where 2 is the two element set $\{0, 1\}$ does not have a final object.

Proof. Assume for a contradiction that (Z, ζ) is the final object of Set_G^F. Define $Z' := Z \cup \{z'\}$ with $z' \notin Z$ and define $\xi \colon FZ' \to GZ'$ as follows.

$$\begin{array}{lcll} (x, y) & \mapsto & 1 & \text{if } x = z' \text{ and } y = z' \\ (x, y) & \mapsto & 0 & \text{if } x = z' \text{ or } y = z' \\ (x, y) & \mapsto & \zeta(z, y) & \text{otherwise} \end{array}$$

Now (Z', ξ) is a Set_G^F dialgebra and by finality of (Z, ζ) there is a morphism $h \colon (Z', \xi) \to (Z, \zeta)$. Since h is a dialgebra morphism $\zeta \circ Fh = Gh \circ \xi = \mathrm{id}_2 \circ \xi = \xi$. Now let $z \in Z$ such that $h(z') = z$. Note that $z \in Z$ and thus $z \in Z'$. Then $(\zeta \circ Fh)(z', z') = \zeta(z, z) = (\zeta \circ Fh)(z', z)$. However, $\xi(z', z') = 1$ and $\xi(z', z) = 0$ which contradicts that h is a dialgebra morphism. □

At first sight this may seem discouraging. Given a coalgebraic operational semantics, the morphism into the final coalgebra defines a canonical denotational semantics. The lack of a final dialgebra would seem to suggest that a canonical denotational semantics for a dialgebraic operational semantics is problematic. However, final objects offer much more than we require. A final coalgebra provides a semantic domain not just for a single coalgebraic operational semantics, but for *any* operational semantics of the same signature. In addition, morphisms into it are often not surjective, which implies that it contains denotations that are not associated with terms.

Rather than considering finality and likewise initiality of dialgebras over–all, we restrict attention to final quotients and initial subobjects of a given dialgebra. These restrictions result in categories that meet our needs and in which initial and final objects exist.

Definition 3.5.2 (Minimal objects and simple objects)**.**

1. The least subobject of an object $X : \mathcal{C}$ is the initial object of the category $(\mathrm{Sub}\mathcal{C}/X)$.
   A minimal object is an object A that is the domain of a least subobject $m : A \rightarrowtail X$.
2. The greatest quotient of an object $X : \mathcal{C}$ is the final object of the category $(X/\mathrm{Quot}\mathcal{C})$.
   A simple object is an object Z that is de codomain of a greatest quotient $e : X \twoheadrightarrow Z$. $\Diamond$

Minimal objects are truly minimal in that they have no proper subobjects. Indeed, any subobject of a minimal object is an isomorphism. Likewise any quotient of a simple object is an isomorphism.

Proposition 3.5.3. Consider a category $\mathcal{C}$.

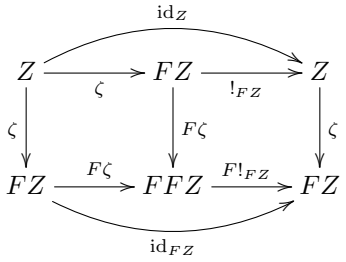
1. If A is minimal and $n : B \rightarrowtail A$ is a subobject then it is an isomorphism.
2. If Z is simple and $e : Z \twoheadrightarrow Y$ is a quotient then it is an isomorphism.

There is a well known result about initial algebras and final coalgebras, known as Lambek's lemma. It states that the structure of an initial algebra is an isomorphism and dually, so is the structure of a final coalgebra.

Proposition 3.5.4 (Lambek's lemma)**.**

1. If (A, α) is an initial algebra then α is an isomorphism.
2. If (Z, ζ) is a final coalgebra then ζ is an isomorphism.

Proof. We prove the second item. Observe that $\zeta : (Z, \zeta) \to (FZ, F\zeta)$ is a coalgebra morphism. By finality of (Z, ζ) there is a unique coalgebra morphism $!_{(FZ,F\zeta)}$. For readability we denote its underlying morphism by $!_{FZ}$. Again by finality there is a unique coalgebra morphism $\mathrm{id}_{(Z,\zeta)} = !_{(Z,\zeta)}$ whose underlying morphism is id_Z. Thus $!_{FZ} \circ \zeta = \mathrm{id}_Z$ and $!_{FZ}$ is a retraction of ζ but also $F!_{FZ} \circ F\zeta = \mathrm{id}_{FZ}$. Since $!_{(FZ,F\zeta)}$ is a coalgebra morphism we have $\zeta \circ !_{FZ} = F!_{FZ} \circ F\zeta$ which is id_{FZ} so $!_{FZ}$ is also a section of ζ and ζ is an isomorphism.



□

Inspired by Lambek's lemma we investigate the structure of minimal algebras and simple coalgebras. Unlike initial algebras and final coalgebras these structures are not in general isomorphisms. However a similar weaker but similar result does hold.

Proposition 3.5.5. Consider a category $\mathcal{C}^F$ of algebras and a category $\mathcal{C}_G$ of coalgebras.

1. If (A, α) is a minimal algebra then $\mathrm{im}\,\alpha$ is an isomorphism.
2. If (Z, ζ) is a simple coalgebra then $\overline{\mathrm{im}}\,\zeta$ is an isomorphism.

Proof. Consider the second item. Let $h : Z \to Y$ be such that $\zeta = h \circ \overline{\mathrm{im}}\,\zeta$ and define $\upsilon : Y \to FY$ by $\upsilon := h \circ \overline{\mathrm{im}}\,\zeta$. Then $\overline{\mathrm{im}}\,\zeta : (Z, \zeta) \to (Y, \upsilon)$ is a quotient dialgebra. If (Z, ζ) is simple then $\overline{\mathrm{im}}\,\zeta$ is an isomorphism by proposition 3.5.3. □

In Set and in other category where the morphism m of a factorisation $\zeta = m \circ \overline{\mathrm{im}}\,\zeta$ is a monomorphism, this implies that the structure of a simple coalgebra is a monomorphism. Likewise, if (A, α) is a minimal algebra and $\alpha = \mathrm{im}\,\alpha \circ e$ with e an epimorphism then α is an epimorphism. Consequently, in Set the structure of a minimal algebra is surjective and the structure of a simple coalgebra is injective.

Neither Lambek's lemma nor the weaker result on minimal algebras and simple coalgebras extend to dialgebra structures. This is illustrated by the following counterexamples.

For a simple dialgebra consider $(\mathbb{N}, \zeta)$ where $FX := X + X$, $GX := X + 1$ and $\zeta : F\mathbb{N} \to G\mathbb{N}$ is defined by $\iota_i(0) \mapsto \iota_2(*)$ and $\iota_i(n+1) \mapsto n$ for both $i = 1$ and $i = 2$. Then $(\mathbb{N}, \zeta)$ is simple but ζ is not injective: $\zeta(\iota_1(n)) = \zeta(\iota_2(n))$ for all $n \in \mathbb{N}$.

For a minimal dialgebra consider $(\mathbb{N}, \xi)$ where $\xi : F\mathbb{N} \to G\mathbb{N}$ with $FX := 1 + X$ and $GX := X + 1$. Define ξ by $\iota_1(*) \mapsto \iota_1(0)$ and $\iota_2(n) \mapsto \iota_1(n+1)$. Then $(\mathbb{N}, \xi)$ is minimal. It is almost an algebra but its structure has an extended codomain. Then $\mathrm{im}\,\xi$ is isomorphic to the injection $\iota_1 : \mathbb{N} \rightarrowtail \mathbb{N} + 1$ which is not an isomorphism.

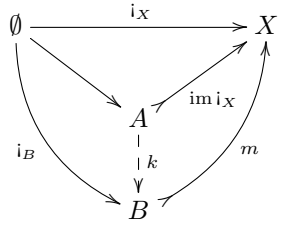
However, the counterexamples do not defeat the purpose of minimal and simple dialgebras. In fact they have many properties that are intuitively similar to their algebraic and coalgebraic counterparts. These properties are just not as easily expressed in terms of their structure. For example, the intuition that in a minimal dialgebra all states can be constructed is correct, even though it is a bit more complicated to express what it means for a state of a dialgebra to be constructed. Shortly, in a Set^F algebra (A, α) an element $a \in A$ is constructed from $c \in FA$ if $\alpha(c) = a$ whereas an element $x \in X$ of a Set^F_G dialgebra (X, χ) is constructed from $c \in FX$ if x is 'occurs' in $\chi(c)$. For instance, if $GX := X \times X$ then x occurs in (y, z) if $y = x$ or $z = x$.

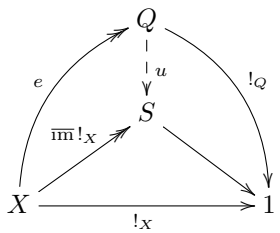
The intuition that elements of X can occur in elements of GX will be made precise in the next chapter where we investigate minimal and simple dialgebras more thoroughly. We conclude this section by stating as additional results about factorisations and minimal and simple objects.

Proposition 3.5.6. Consider a category $\mathcal{C}$ and an object X of $\mathcal{C}$.

1. The image $\mathrm{im}\,\mathsf{i}_X$ of the initial morphism $\mathsf{i}_X : \emptyset \to X$ is the least subobject of X.
2. The coimage $\overline{\mathrm{im}}\,!_X$ of the final morphism $!_X : X \to 1$ is the greatest quotient of X.

Proof. We prove the first item. Consider any subobject $m : B \rightarrowtail X$ of $X : \mathcal{C}$. We have to show that there is a unique morphism of subobjects $k : \mathrm{im}\,\mathsf{i}_X \to m$. By initiality of $\emptyset$ there is a unique morphism $\mathsf{i}_B : \emptyset \to B$. Initial morphisms are unique by definition therefore $m \circ \mathsf{i}_B = \mathsf{i}_X$. By the definition of image there is a unique morphism $k : A \to B$ which is the unique morphism $k : \mathrm{im}\,\mathsf{i}_X \to m$. The second item is dual. We obtain for every quotient $e : X \twoheadrightarrow Q$ a unique morphism of quotients $u : e \to \overline{\mathrm{im}}\,!_X$.





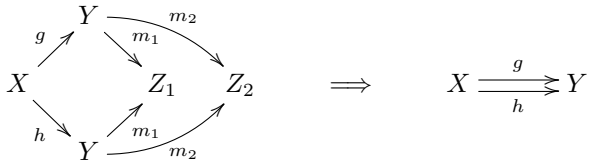
□

3.6 Bisimulations

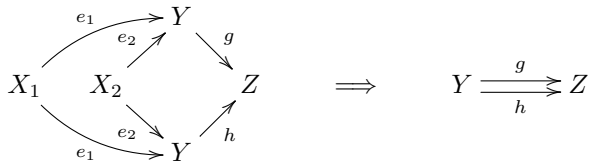
In the literature on universal algebra and coalgebra, congruence relations and bisimulation relations play a central role. However confusingly, there are quite a few related, yet different definitions to be found [21].
Most commonly a *bisimulation* between two coalgebras (X, χ) and (Y, υ) is defined to be a subobject $m : R \rightarrowtail X \times Y$ such that R has a coalgebra structure making $(\pi_1 \circ m) : R \to X$ and $(\pi_2 \circ m) : R \to Y$ coalgebra morphisms. On the other hand, *bisimilarity* between (X, χ) and (Y, υ) is more commonly defined as the existence of a pair of coalgebra morphisms $g : (X, \chi) \to (Z, \zeta)$ and $h : (Y, \upsilon) \to (Z, \zeta)$ to a common coalgebra (Z, ζ), in which case a bisimulation relation may be defined as the pullback of the two morphisms $g : X \to Z$ and $h : Y \to Z$ in the base category.
In extension, we notice that bisimulations are often used with one out of two particular goals. A bisimulation relation $R \rightarrowtail X \times X$ might be used to define a subcoalgebra of a particular coalgebra (X, χ). More generally, a bisimulation relation $R \rightarrowtail X \times Y$ may be used to define a subcoalgebra of (X, χ) and a subcoalgebra of (Y, υ) whose structures are in agreement. On the other hand, a bisimulation equivalence may be used to specify a quotient coalgebra rather than a subcoalgebra. We attempt to distinguish those uses and specify bisimulations as a generalisation of subobjects so that they have a dual notion which generalises quotients. In doing so we depend on a generalisation of monomorphisms and epimorphisms.

Definition 3.6.1 (Jointly monic and jointly epic morphisms)**.**

1. A pair of morphisms $m_1 : Y \to Z_1$ and $m_2 : Y \to Z_2$ is *jointly monic* if for any pair of morphisms $g : X \to Y$ and $h : X \to Y$, if $m_1 \circ g = m_1 \circ h$ and $m_2 \circ g = m_2 \circ h$ then $g = h$.



2. A pair of morphisms $e_1 : X_1 \to Y$ and $e_2 : X_2 \to Y$, is *jointly epic* if for any pair $g : Y \to Z$ and $h : Y \to Z$, if $g \circ e_1 = h \circ e_1$ and $g \circ e_2 = h \circ e_2$ then $g = h$.



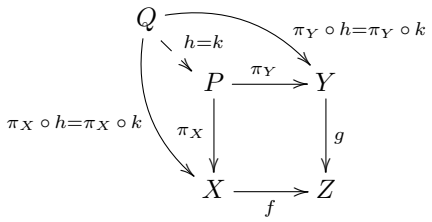
◊

In a category that has products, a jointly monic pair $m : Z \to X$ and $n : Z \to Y$ gives rise to the monomorphism $(m, n) : Z \rightarrowtail X \times Y$ and dually, a jointly epic pair $e : X \to Z$ and $f : Y \to Z$ gives rise to the epimorphism $[e, f] : X + Y \twoheadrightarrow Z$ if $\mathcal{C}$ has coproducts. More generally, projections are jointly monic and injections are jointly epic, which we state for pullbacks and pushouts.

Proposition 3.6.2.

1. Let P be the pullback of $f : X \to Z$ and $g : Y \to Z$. The projections $\pi_X : P \to X$ and $\pi_Y : P \to Y$ such that $f \circ \pi_X = g \circ \pi_Y$ are jointly monic.
2. Let P be the pushout of $f : Z \to X$ and $g : Z \to Y$. The injections $\iota_X : X \to P$ and $\iota_Y : Y \to P$ such that $\iota_X \circ f = \iota_Y \circ g$ are jointly epic.

Proof. We prove the first item. Consider an arbitrary pair of morphisms $h : Q \to P$ and $k : Q \to P$. We have to show that if $\pi_X \circ h = \pi_X \circ k$ and $\pi_Y \circ h = \pi_Y \circ k$ then $h = k$. Since P is a pullback of f and g we have $f \circ \pi_X = g \circ \pi_Y$ and by the definition of pullback $h = k$ is the unique morphism such that $f \circ \pi_X \circ h = g \circ \pi_Y \circ h$ and $f \circ \pi_X \circ k = g \circ \pi_Y \circ k$.

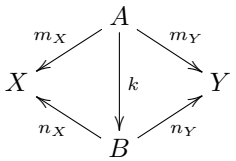


□

If $\mathcal{C}$ is a category such that Sub$\mathcal{C}$ and Quot$\mathcal{C}$ are defined to be Mono$\mathcal{C}$ and Epi$\mathcal{C}$ we may as well call a jointly monic pair of morphisms a *joint subobject* and call a jointly epic pair of morphisms a *joint quotient*. If Sub$\mathcal{C}$ and Quot$\mathcal{C}$ do not coincide with Mono$\mathcal{C}$ and Epi$\mathcal{C}$ then joint subobjects and joint quotients have to be defined more carefully.

Definition 3.6.3 (Category of joint subobjects)**.**

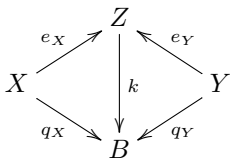
1. A *joint subobject* (A, m_X, m_Y) of $X : \mathcal{C}$ and $Y : \mathcal{C}$ is a jointly monic pair of morphisms $m_X : A \to X$ and $m_Y : A \to Y$ of a suitable subclass of jointly monic morphisms of $\mathcal{C}$.
2. A joint subobject morphism $k : (A, m_X, m_Y) \to (B, n_X, n_Y)$ is a morphism $k : A \to B$ of Sub$\mathcal{C}$ such that $n_X \circ k = m_X$ and $n_Y \circ k = m_Y$.



◊

Definition 3.6.4 (Category of joint quotients)**.**

1. A *joint quotient* (e_X, e_Y, Z) of objects $X : \mathcal{C}$ and $Y : \mathcal{C}$ is a jointly epic pair of morphisms $e_X : X \to Z$ and $e_Y : Y \to Z$ of a suitable subclass of jointly epic morphisms of $\mathcal{C}$.
2. A joint quotient morphism $k : (e_X, e_Y, Z) \to (q_X, q_Y, B)$ is a morphism $k : Z \to B$ of Quot$\mathcal{C}$ such that $k \circ e_X = q_X$ and $k \circ e_Y = q_Y$.



◊

Recall that a subobject $m : A \rightarrowtail X$ of an object $X : \mathcal{C}$ is a morphism in a suitably chosen subcategory of Mono$\mathcal{C}$. This subcategory is typically defined by specifying a property that the monomorphisms must satisfy. Defining joint subobjects involves carefully generalising this property. However, doing so requires getting into the details of subobjects and quotients of particular categories. We do not let ourselves be distracted by these issues and will assume that for base categories of dialgebras joint subobjects and joint quotients coincide with joint monomorphisms and joint epimorphisms.

Of special interest to us are joint subdialgebras and joint quotient dialgebras. Tracing our steps in the previous section we generalise the categories of subdialgebras and quotient dialgebras as follows.

Definition 3.6.5 (Category of joint subdialgebras)**.**

1. A *joint subdialgebra* of two dialgebras (X, χ) and (Y, υ) is a joint subobject (A, π_X, π_Y) of X and Y in $\mathcal{C}$ such that A has a dialgebra structure α that makes $\pi_X : (A, \alpha) \to (X, \chi)$ and $\pi_Y : (A, \alpha) \to (Y, \upsilon)$ dialgebra morphisms.

$$\begin{array}{ccccc} FX & \xleftarrow{F\pi_X} & FA & \xrightarrow{F\pi_Y} & FY \\ {\scriptstyle\chi}\downarrow & & \downarrow{\scriptstyle\alpha} & & \downarrow{\scriptstyle\upsilon} \\ GX & \xleftarrow[G\pi_X]{} & GA & \xrightarrow[G\pi_Y]{} & GY \end{array}$$

2. A joint subdialgebra morphism $k : ((A, \alpha), m_X, m_Y) \to ((B, \beta), n_X, n_Y)$ is a dialgebra morphism $k : (A, \alpha) \to (B, \beta)$ that makes the following diagram in $\mathcal{C}_G^F$ commute.

$$\begin{array}{ccccc} & & (A,\alpha) & & \\ & {\scriptstyle m_X}\swarrow & \downarrow{\scriptstyle k} & \searrow{\scriptstyle m_Y} & \\ (X,\chi) & & & & (Y,\upsilon) \\ & {\scriptstyle n_X}\nwarrow & & \nearrow{\scriptstyle n_Y} & \\ & & (B,\beta) & & \end{array}$$

◊

With this we have redefined bisimulations: A joint subdialgebra is exactly a bisimulation as defined in [9, 20, 23]. In addition, the existence of a joint quotient dialgebra defined dually, is remarkably close to the definition of bisimilarity as the existence of two morphisms to a common codomain.

Definition 3.6.6 (Category of joint quotient dialgebras)**.**

1. A *joint quotient dialgebra* of (X, χ) and (Y, υ) is a joint quotient (ι_X, ι_Y, Z) of X and Y in $\mathcal{C}$ such that Z has a dialgebra structure ζ that makes $\iota_X : (X, \chi) \to (Z, \zeta)$ and $\iota_Y : (Y, \upsilon) \to (X, \chi)$ dialgebra morphisms.

$$\begin{array}{ccccc} FX & \xrightarrow{F\iota_X} & FZ & \xleftarrow{F\iota_Y} & FY \\ {\scriptstyle\chi}\downarrow & & \downarrow{\scriptstyle\zeta} & & \downarrow{\scriptstyle\upsilon} \\ GX & \xrightarrow[G\iota_X]{} & GZ & \xleftarrow[G\iota_Y]{} & GY \end{array}$$

2. A joint quotient dialgebra morphism $k : (e_X, e_Y, (Z, \zeta)) \to (q_X, q_Y, (B, \beta))$ is a dialgebra morphism $k : (A, \alpha) \to (B, \beta)$ that makes the following diagram in $\mathcal{C}_G^F$ commute.

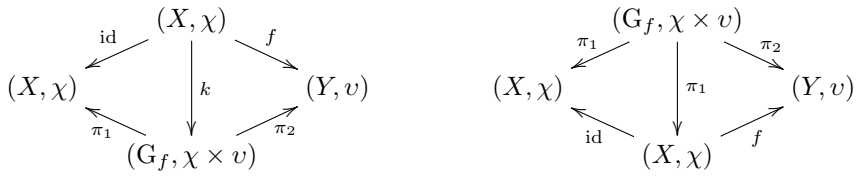
$$\begin{array}{ccccc} & & (Z,\zeta) & & \\ & {\scriptstyle e_X}\nearrow & \downarrow{\scriptstyle k} & \nwarrow{\scriptstyle e_Y} & \\ (X,\chi) & & & & (Y,\upsilon) \\ & {\scriptstyle q_X}\searrow & & \swarrow{\scriptstyle q_Y} & \\ & & (B,\beta) & & \end{array}$$

◊

Note that joint subdialgebras are indeed jointly monic pairs of dialgebra morphisms and joint quotient dialgebras are jointly epic pairs of dialgebra morphisms. This follows from the underlying pairs of morphisms being jointly monic and jointly epic, respectively.

In universal coalgebra there are quite a few theorems that relate subcoalgebras, quotient coalgebras and bisimulations. We shortly discuss some of them for dialgebras.

- Let (A, α) be a dialgebra and let $\mathrm{id} : (A, \alpha) \to (A, \alpha)$ denote the identity morphism on (A, α). Then $((A, \alpha), \mathrm{id}, \mathrm{id})$ is a joint subdialgebra, i.e. a bisimulation. If the base category is Set then A is isomorphic to the diagonal relation $\Delta_A := \{(x, x) \mid x \in A\}$ via $k : x \mapsto (x, x)$ which has an inverse $\pi_1 = \pi_2$ given by the projections out of Δ_A. The isomorphism extends to a dialgebra isomorphism that makes $((\Delta_A, \alpha^2), \pi_1, \pi_2)$ a bisimulation and $k : ((A, \alpha), \mathrm{id}, \mathrm{id}) \xrightarrow{\sim} ((\Delta_A, \alpha^2), \pi_1, \pi_2)$ a bisimulation isomorphism.
- Dually, if (Z, ζ) is a dialgebra then $(\mathrm{id}, \mathrm{id}, (Z, \zeta))$ is a joint quotient dialgebra.
- If $m : (A, \alpha) \to (X, \chi)$ is a subdialgebra then $((A, \alpha), m, m)$ is a bisimulation. If the base category is Set then this bisimulation is isomorphic to $((\Delta_A, \alpha^2), m \circ \pi_1, m \circ \pi_2)$.
- If $e : (X, \chi) \to (Z, \zeta)$ is a quotient dialgebra then $(e, e, (Z, \zeta))$ is a joint quotient dialgebra.
- Let $f : (X, \chi) \to (Y, \upsilon)$ be a dialgebra morphism in $\mathcal{C}_G^F$. Because $\mathrm{id} : (X, \chi) \to (X, \chi)$ is both monic and epic $((X, \chi), \mathrm{id}, f)$ is a bisimulation and $(f, \mathrm{id}, (Y, \upsilon))$ is a joint quotient dialgebra. If $\mathcal{C}$ is Set then there is an injection $k : X \rightarrowtail \mathrm{G}_f$ into the graph $\mathrm{G}_f := \{(x, f(x)) \mid x \in X\}$ of f via $x \mapsto (x, f(x))$. The morphism k extends to a dialgebra morphism $k : (X, \chi) \to (\mathrm{G}_f, \chi \times \upsilon)$ and to a morphism of bisimulations $k : ((X, \chi), \mathrm{id}, f) \to ((\mathrm{G}_f, \chi \times \upsilon), \pi_1, \pi_2)$. In addition, it has a retract $\pi_1 : \mathrm{G}_f \to X$ which extends to a dialgebra morphism and to a morphism of bisimulations $\pi_1 : ((\mathrm{G}_f, \chi \times \upsilon), \pi_1, \pi_2) \to ((X, \chi), \mathrm{id}, f)$.



- Let $f : (X, \chi) \to (Z, \zeta)$ and $g : (Y, \upsilon) \to (Z, \zeta)$ be dialgebra morphisms and let R be the pullback of $f : X \to Z$ and $g : Y \to Z$. By proposition 3.6.2 $\pi_X : R \to X$ and $\pi_Y : R \to Y$ are jointly monic. If in addition G preserves pullbacks then by theorem 3.2.1 R has a unique dialgebra structure ρ such that π_X and π_Y dialgebra morphisms and $((R, \rho), \pi_X, \pi_Y)$ a bisimulation.
  Note that in Set the kernel $\mathrm{K}_f = \{(x, y) \mid f(x) = f(y)\}$ of a function $f : X \to Z$ is the pullback of f along itself, thus if G preserves pullbacks then K_f is (the carrier of) a bisimulation.

Much of the theory of bisimulations and joint quotient dialgebras can be stated by generalising the theory of subdialgebras and quotient dialgebras. Indeed without too much effort the propositions of the previous sections on subdialgebras and quotient dialgebras can be adapted to obtain similar results about bisimulations and joint quotients. We refer to [20] for results about coalgebra bisimulations and to [23] in which they are stated for dialgebras.

Chapter 4

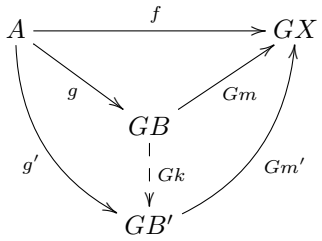
Minimisation and Simplification

In this chapter we introduce two new concepts that generalise images and coimages of morphisms. We employ these concepts in a minimisation procedure and a simplification procedure for dialgebras. The chapter is concluded with a short investigation of the interplay between minimisation and simplification.

4.1 Bases and cobases

The *base* of a morphism is a generalisation of the image of a morphism, where the image of a morphism its Id–base. Intuitively, the G–base of a morphism $f : A \to GX$ is a subobject of X that is in some sense compatible with the subobject $\operatorname{im} f$ of GX.

Definition 4.1.1 (Base)**.** Let $G : \mathcal{C} \to \mathcal{C}$ be a functor and let X be an object of $\mathcal{C}$. The *G–base* of a morphism $f : A \to GX$ is the least subobject $m : B \rightarrowtail X$ such that f factorises through Gm. Specifically, if $f = Gm \circ g$ then for any subobject $m' : B' \rightarrowtail X$ with a factorisation $f = Gm' \circ g'$ there is a unique morphism $k : B \to B'$ such that $m = m' \circ k$ and $Gk \circ g' = g$. We use the notation $\operatorname{im}_G f$ to denote the G–base $m : B \rightarrowtail X$ of a morphism $f : A \to GX$.



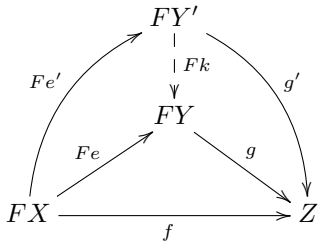
◊

In Set the G–base of a function $f : A \to GX$ is the set of elements of X that are 'contained' in elements of $\operatorname{dom} \operatorname{im} f$. This is best understood by considering the base of inclusions. For example, if $GX := X \times X$ and $m : 1 \hookrightarrow GX$ is the inclusion of the singleton set $\{(x, y)\}$ into GX then the G–base of m is the set $B = \{x, y\}$. This generalises to subsets of GX, for example if $A := \{(x, y), (x, z), (z, x)\}$ then the base of the inclusion $m : A \hookrightarrow GX$ is the set $\{x, y, z\}$. More generally, for morphisms $f : A \to GX$ of Set the G–base is obtained exactly as one would expect, by extracting and collecting all elements of X from elements of $f[A]$. For common instantiations of G the base $(\operatorname{im}_G f) : B \hookrightarrow X$ of f is as follows.

$$
\begin{array}{lcll}
GX & \coloneqq & C & B = \emptyset \\
GX & \coloneqq & X & B = f[A] \\
GX & \coloneqq & X + C & B = \{x \mid \iota_1(x) \in f[A]\,\} \\
GX & \coloneqq & X + X & B = \{x \mid \iota_1(x) \in f[A] \text{ or } \iota_2(x) \in f[A]\,\} \\
GX & \coloneqq & X \times C & B = (\pi_1 \circ f)[A] \\
GX & \coloneqq & X \times X & B = (\pi_1 \circ f)[A] \cup (\pi_2 \circ f)[A] \\
GX & \coloneqq & \mathscr{P}X & B = \bigcup f[A]
\end{array}
$$

Dual to the base of a morphism, the F–cobase generalises the concept of a coimage. Specifically, the coimage of a morphism is the Id–cobase of that morphism. Intuitively, the F–cobase of a morphism $f\colon FX \to X$ is a partitioning of X that is in some sense compatible with the partitioning $\overline{\mathrm{im}}\, f$ of FX.

Definition 4.1.2 (Cobase)**.** Let $F\colon \mathcal{C} \to \mathcal{C}$ be a functor and let X be an object of $\mathcal{C}$. The *F–cobase* of a morphism $f\colon FX \to Z$ is the greatest quotient $e\colon X \twoheadrightarrow Y$ such that f factorises through Fe. Specifically, if $f = g \circ Fe$ then for any quotient $e'\colon X \twoheadrightarrow Y'$ with a factorisation $f = g' \circ Fe'$ there is a unique morphism $k\colon Y' \to Y$ such that $g' \circ Fk = g$ and $e = k \circ e'$. We use the notation $\overline{\mathrm{im}}_F\, f$ to denote the F–cobase $e\colon X \twoheadrightarrow Y$ of a morphism $f\colon FX \to Z$.



◊

The G–base of a morphism can be used to extract contents of containers, a concept that is easy to grasp. The dual concept of a cobase may require a bit of additional attention. To get an intuition it is easiest to consider Set where an epimorphism $f\colon FX \twoheadrightarrow Z$ is essentially a partitioning of FX. Indeed, Z is isomorphic to the set of equivalence classes $(FX)/_f \coloneqq \{[a]_f \mid a \in FX\}$ where $[a]_f \coloneqq \{b \in FX \mid f(a) = f(b)\}$. Now let $e\colon X \twoheadrightarrow Y$ denote the cobase $\overline{\mathrm{im}}_F\, f$. Then e is essentially a partitioning of X and Fe a partitioning of FX. The requirement that f factorises through Fe enforces that $(FX)/_{Fe}$ is compatible with $(FX)/_f$. The existence of a unique morphism enforces that $X/_e$ is the coarsest partitioning on X such that this is the case.

Let us go through an example. Consider a morphism $f\colon FX \twoheadrightarrow Z$ of Set where $X \coloneqq \{s, r, t\}$ and $FX \coloneqq X^2$. We let $Z = 2$ be a two element set and we let f take $(t, \_) \mapsto 0$, $(\_, t) \mapsto 0$ and $(\_, \_) \mapsto 1$ otherwise. Then Z is isomorphic to $(FX)/_f$.

$$
(FX)/_f \quad = \quad \{ \quad \{(s,s),(r,r),(s,r),(r,s)\}, \quad \{(t,s),(t,r),(s,t),(r,t),(t,t)\} \quad \}
$$

The F–cobase of f is a quotient $e\colon X \twoheadrightarrow Y$ for some Y which is essentially a partitioning of X. Since X is a three element set there are essentially three candidates for Y, that is, $Y \cong 1$, $Y \cong 2$ or $Y \cong X$. By definition e is the greatest quotient of X such that f factorises through $Fe\colon FX \to FY$, This excludes the possibility that $Y \cong 1$, for $F1 \cong 1$ and $f\colon FX \twoheadrightarrow Z$ certainly

does not factorise through 1. So let us try $Y \cong 2$. There are three ways to partition X into two parts, corresponding to three essential candidates for $e : X \twoheadrightarrow 2$. Defined in terms of equivalence classes we have the following candidates.

$$X/_{e_1} = \{\{s,r\},\{t\}\} \qquad X/_{e_2} = \{\{s,t\},\{r\}\} \qquad X/_{e_3} = \{\{t,r\},\{s\}\}$$

For ease of writing let us assume that the codomain of each e_i is in fact $X/_{e_i}$. Now, if some e_i is the F–cobase of f then there is a morphism g such that $Fe_i \circ g = f$. A quick examination reveals that e_2 and e_3 are out, for $Fe_2(s,s) = Fe_2(t,t)$ and $Fe_3(r,r) = Fe_3(t,t)$ whereas $f(s,s) \neq f(t,t)$ and $f(r,r) \neq f(t,t)$. However Fe_1 *is* compatible with f via a morphism $g : F(X/_e) \to Z$ as follows.

$$(\{t\},\{t\}) \mapsto 0 \qquad (\{t\},\{s,r\}) \mapsto 1 \qquad (\{s,r\},\{t\}) \mapsto 1 \qquad (\{s,r\},\{s,r\}) \mapsto 1$$

Let us investigate this a bit further. Intuitively the morphism e_1 defines a partitioning of X such that Fe_1 is compatible with f whereas e_2 and e_3 do not. Specifically, each of the candidates e_i defines a partitioning of FX via Fe_i. We can find the appropriate candidate by comparing $(FX)/_{Fe_i}$ to $(FX)/_f$ for each e_i. Working out the details gives us the following.

$$\begin{array}{lcl}
(FX)/_{Fe_1} & = & \{\ \{(s,s),(r,r),(s,r),(r,s)\},\quad \{(t,s),(t,r)\},\quad \{(s,t),(r,t)\},\quad \{(t,t)\}\ \} \\
(FX)/_{Fe_2} & = & \{\ \{(s,s),(t,t),(s,t),(t,s)\},\quad \{(r,s),(r,t)\},\quad \{(s,r),(t,r)\},\quad \{(r,r)\}\ \} \\
(FX)/_{Fe_3} & = & \{\ \{(t,t),(r,r),(t,r),(r,t)\},\quad \{(s,r),(s,t)\},\quad \{(r,s),(t,s)\},\quad \{(s,s)\}\ \}
\end{array}$$

As one can check, $(FX)/_f$ can be partitioned further into $(FX)/_{Fe_1}$ but it cannot be partitioned into either of $(FX)/_{Fe_2}$ or $(FX)/_{Fe_2}$. Thus, F_{e_1} is compatible with f, whereas F_{e_2} and F_{e_3} are not.

4.2 Closure and density

In chapter 3 we have made a remark that dialgebras are not closed under their structure. Shortly, if (X,δ) is a Set^F_G dialgebra with $FX \coloneqq 1 + X$ and $GX \coloneqq X + 1$ then $\delta(\iota_2(x))$ is not an element of X. Indeed, it is an element of GX. However, we have not been entirely accurate. In some sense, dialgebras *are* closed under their structure. Shortly, if δ takes $\iota_2(x) \mapsto \iota_1(y)$ then δ *is* defined on $\iota_2(y)$ of FX.

Recall that a subset $A \subseteq X$ is closed under a function $\sigma : FX \to X$ if $\sigma[FA] \subseteq A$. Let us put this in proper category theoretic terms. A subobject $m : A \rightarrowtail X$ is closed under $\sigma : FX \to X$ if $\mathrm{im}\,(\sigma \circ Fm)$ factorises through $m : A \rightarrowtail X$, i.e. there is a morphism u such that $\mathrm{im}\,(\sigma \circ Fm) = m \circ u$. Note that any such u is necessarily unique because m and $\mathrm{im}\,(\sigma \circ Fm)$ are monomorphisms. Moreover, it implies that $m : A \rightarrowtail X$ extends to a subdialgebra $m : (A,\alpha) \rightarrowtail (X,\sigma)$ where $\alpha \coloneqq u \circ e$ with e such that $\mathrm{im}\,(\sigma \circ Fm) \circ e = \sigma \circ Fm$ as per the definition of the image. With the generalisation of the image of a morphism to the base of a morphism we can define closure under a dialgebraic structure. In addition, density under a dialgebra structure turns arises as the dual of dialgebraic closure.

Definition 4.2.1 (Closure and density)**.**

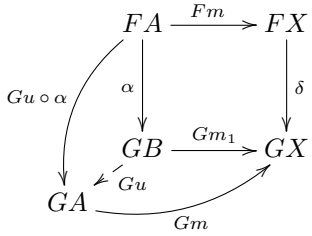
1. A subobject $m : A \rightarrowtail X$ is closed under $\delta : FX \to GX$ if $\mathrm{im}_G\,(\delta \circ Fm)$ factors through m.
2. A quotient $w : X \twoheadrightarrow Z$ is dense under $\delta : FX \to GX$ if $\overline{\mathrm{im}}_F\,(Gw \circ \delta)$ factors through w. $\Diamond$

Now that we have a generalised concept of closure we can generalise the story above to dialgebras. As one would hope, we find that if a subobject is closed under a dialgebraic structure then it is a subdialgebra and again, if a quotient is dense under a dialgebra structure then it is a quotient dialgebra.

Proposition 4.2.2.

1. If a subobject $m\colon A \rightarrowtail X$ is closed under $\sigma\colon FX \to GX$ then m is a subdialgebra.
2. If a quotient $w\colon X \twoheadrightarrow Z$ is dense under $\sigma\colon FX \to GX$ then w is a quotient dialgebra.

Proof. We prove the first item. Define $m_1\colon B \to X$ by $m_1 \coloneqq \mathrm{im}_G\,(\delta \circ Fm)$. By definition of base there is a morphism $\alpha\colon FA \to FB$ such that $Gm_1 \circ \alpha = \delta \circ Fm$. By definition of closure there is a morphism $u\colon B \to A$ such that $m \circ u = m_1$. Note that u is unique because m and m_1 are monic. Now $m\colon (A,\, Gu \circ \alpha) \rightarrowtail (F, \delta)$ is a subdialgebra.



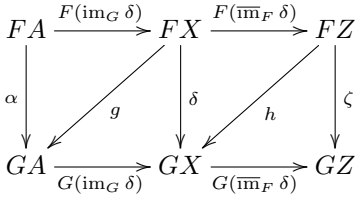
□

In section 3.5 we have included proposition 3.5.5, which in a way is a restricted version of Lambek's lemma for minimal algebras and simple coalgebras. Moreover we have implied that this proposition is not applicable to dialgebras. But with the generalised notion of images and coimages at hand we can conclude this section with a generalised version of this very proposition.

Proposition 4.2.3.

1. If (X, δ) is a dialgebra then $\mathrm{im}_G\,\delta$ is a subdialgebra of (X, δ).
   If (X, δ) is minimal then $\mathrm{im}_G\,\delta$ is an isomorphism.
2. If (X, δ) is a dialgebra then $\overline{\mathrm{im}}_F\,\delta$ is a quotient dialgebra of (X, δ).
   If (X, δ) is simple then $\overline{\mathrm{im}}_F\,\delta$ is an isomorphism.

Proof. We prove all at once. Let g be such that $G(\mathrm{im}_G\,\delta) \circ g = \delta$ and let h be such that $\delta = h \circ F(\overline{\mathrm{im}}_F\,\delta)$. Thus $\mathrm{im}_G\,\delta$ is a subobject of X and $\overline{\mathrm{im}}_F\,\delta$ is a quotient of X.



Define $\alpha \coloneqq g \circ F(\mathrm{im}_G\,\delta)$ and $\zeta \coloneqq G(\overline{\mathrm{im}}_F\,\delta) \circ h$ Then $(\mathrm{im}_G\,\delta)\colon (A, \alpha) \rightarrowtail (X, \delta)$ is a subdialgebra and $(\overline{\mathrm{im}}_F\,\delta)\colon (X, \delta) \twoheadrightarrow (Z, \zeta)$ is a quotient dialgebra. If $\mathrm{im}_G\,\delta$ is not an isomorphism then it is a proper subdialgebra so that (X, δ) is not minimal. Likewise, if $\overline{\mathrm{im}}_F\,\delta$ is not an isomorphism then it is a proper quotient dialgebra and (X, δ) is not simple. □

4.3 Minimisation and simplification

In this section we develop a minimisation and simplification procedures dialgebras. The procedures generalise existing techniques for minimising algebras and simplifying coalgebras. With a small adjustment the minimisation procedure can be used not just to minimise dialgebras but also to generate least subdialgebras encompassing a specified subobject of the carrier. This technique applies to coalgebras as well, where it can be used for generating subcoalgebras that are 'rooted' at a particular state or set of states. Dually, the simplification procedure may be used to generate the greatest quotient of a dialgebra that is compatible with a predefined partitioning on the carrier.

We focus on minimisation first. The simplification procedure that we develop is perfectly dual, however minimisation may be much easier intuitively. Let us consider a $\mathcal{C}_G^F$ dialgebra. Our aim is to construct the least subdialgebra of (X, δ); recall that this is the initial object $m : (A, \alpha) \rightarrowtail (X, \delta)$ of $(\mathrm{Sub}(\mathcal{C}_G^F)/(X, \delta))$. Now before we proceed let us think about the solution space for a moment. If (A, α) is to be a subdialgebra of (X, δ) then $m : A \rightarrowtail X$ is a subobject of X thus the category $(\mathrm{Sub}\mathcal{C}/X)$ encompasses the solution space.

Our procedure iteratively improves an approximation to the carrier A of the minimal dialgebra, starting from $m_0 = \mathfrak{i}_X : \emptyset \rightarrowtail X$ until we reach a fixed point. If we restrict attention to inclusions in Set then this corresponds to a chain of approximants $\emptyset = A_0 \subseteq A_1 \subseteq A_2 \ldots$, for which X is an upper bound. For a $\mathcal{C}_G^F$ dialgebra in general, we generate a sequence of subobjects $\mathfrak{i}_X = m_0 \to m_1 \to m_2 \to \ldots$ that is connected by a sequence of morphisms $(k_i : m_i \to m_{i+1})_i$ of $(\mathrm{Sub}\mathcal{C}/X)$. The upper bound for this sequence is the object $\mathrm{id}_{\mathrm{id}_X}$ of $(\mathrm{Sub}\mathcal{C}/X)$.

Definition 4.3.1 (Minimisation sequence)**.** The minimisation sequence of a $\mathcal{C}_G^F$ dialgebra (X, δ) is the sequence $(m_i : A_i \rightarrowtail X)_i$ of *objects* of $(\mathrm{Sub}\mathcal{C}/X)$ defined as follows.

$$\begin{aligned} m_0 &:= \mathfrak{i}_X : \emptyset \rightarrowtail X \\ m_{i+1} &:= \mathrm{im}_G(\delta \circ Fm_i) \end{aligned} \qquad \Diamond$$

The simplification sequence is perfectly dual to the minimisation sequence. Thus the simplification procedure improves an approximation to the carrier Z of the simple dialgebra, starting from $w_0 = !_X : X \twoheadrightarrow 1$ until a fixed point is reached. If we restrict attention to Set then the morphism w_0 creates the coarsest possible partitioning of X. At each step this partitioning is refined according to the dialgebra structure δ. We start with $Gw_0 \circ \delta$, which induces a partitioning on FX. The cobase of $Gw_0 \circ \delta$ corresponds to the coarsest possible partitioning on X that is compatible with this partitioning of FX. Thus, for a $\mathcal{C}_G^F$ dialgebra in general, we generate a weakly decreasing sequence of quotients $!_X = w_0 \to w_1 \to w_2 \to \ldots$ of X with as lower bound the object $\mathrm{id}_{\mathrm{id}_X}$ of $(X/\mathrm{Quot}\mathcal{C})$.

Definition 4.3.2 (Simplification sequence)**.** The simplification sequence of a $\mathcal{C}_G^F$ dialgebra (X, δ) is the sequence $(w_i : X \twoheadrightarrow Z_i)_i$ of *objects* of $(X/\mathrm{Quot}\mathcal{C})$ defined as follows.

$$\begin{aligned} w_0 &:= !_X : X \twoheadrightarrow 1 \\ w_{i+1} &:= \overline{\mathrm{im}}_F(Gw_i \circ \delta) \end{aligned} \qquad \Diamond$$

The following lemma establishes that the sequences just defined are connected by sequences of morphisms of $(\mathrm{Sub}\mathcal{C}/X)$ and of $(X/\mathrm{Quot}\mathcal{C})$ respectively. Thus, the minimisation sequence of a dialgebra (X, δ) is a weakly increasing sequence of subobjects with upper bound X and its simplification sequence is a weakly decreasing sequence of quotients with lower bound X.

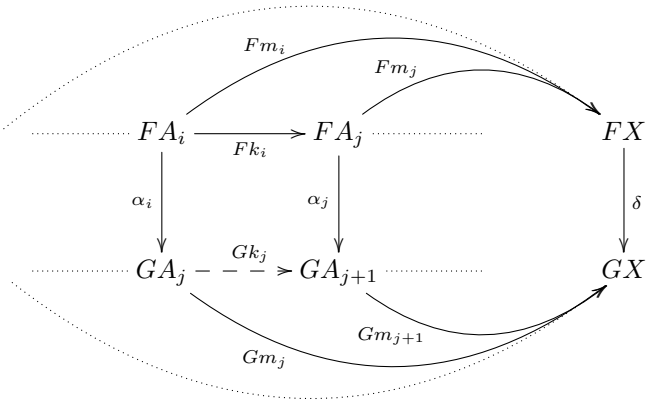
Consequently, fixed points exist and we will soon show that the fixed points are minimal and simple dialgebras respectively.

Lemma 4.3.3. Let (X, δ) be a $\mathcal{C}_G^F$ dialgebra and let $(m_i : A_i \rightarrowtail X)_i$ and $(w_i : X \twoheadrightarrow Z_i)_i$ as in definition 4.3.1.

1. Then $(m_i : A_i \rightarrowtail X)_i$ is connected by a sequence of morphisms $(k_i : m_i \to m_{i+1})_i$ of $(\mathrm{Sub}\mathcal{C}/X)$. In addition there is a sequence of morphisms $(\alpha_i : FA_i \to GA_{i+1})_i$ such that $Gk_{i+1} \circ \alpha_i = \alpha_{i+1} \circ Fk_i$ in $\mathcal{C}$.
2. Likewise $(w_i : X \twoheadrightarrow Z_i)_i$ is connected by a sequence $(k_i : w_{i+1} \to w_i)_i$ of morphisms of $(X/\mathrm{Quot}\mathcal{C})$. In addition there is a sequence of morphisms $(\zeta_i : FZ_{i+1} \to GZ_i)_i$ such that $\zeta_i \circ Fk_{i+1} = Gk_i \circ \zeta_{i+1}$ in $\mathcal{C}$.

Proof. We prove the first item. Define $(\alpha_i : FA_i \to GA_{i+1})_i$ by letting $\alpha_i : FA_i \to GA_{i+1}$ be a morphism such that $Gm_{i+1} \circ \alpha_i = \delta \circ Fm_i$. Its existence is implied by the definition of the base. We proceed by induction, simultaneously showing that there is a unique connecting morphism $k_j : A_j \to A_{j+1}$ such that $m_j = m_{j+1} \circ k_j$ and $Gk_{j+1} \circ \alpha_j = \alpha_{j+1} \circ Fk_j$ in $\mathcal{C}$ for all ordinals j.

- For $j = 0$ recall that $A_0 = \emptyset$. We have $m_0 = \mathfrak{i}_X$ by definition and $k_0 = \mathfrak{i}_{A_1}$ and $m_1 \circ k_0 = m_0 = \mathfrak{i}_X$ by uniqueness initial morphisms.
- For $j = i + 1$ the induction hypothesis is $m_j \circ k_i = m_i$. By definition of α_i and α_j we have $Gm_j \circ \alpha_i = \delta \circ Fm_i$ and $Gm_{j+1} \circ \alpha_j = \delta \circ Fm_j$. We have to show that there is a unique morphism $k_j : A_j \to A_{j+1}$ such that the following diagram commutes.



Now $m_j = \mathrm{im}_G(\delta \circ Fm_i)$ which is $\mathrm{im}_G(\delta \circ Fm_j \circ Fk_i)$ by the induction hypothesis. By proposition 3.4.4 the image of a morphism $g \circ f$ is a subobject of a morphism g. This generalises to the base of morphisms, thus $\mathrm{im}_G(\delta \circ Fm_j \circ Fk_i)$ is a subobject of $\mathrm{im}_G(\delta \circ Fm_j)$ via a unique morphism $k_i : A_i \to A_j$ such that $m_j \circ k_i = m_i$. □

Theorem 4.3.4 (Minimisation and simplification)**.** Consider a $\mathcal{C}_G^F$ dialgebra (X, δ).

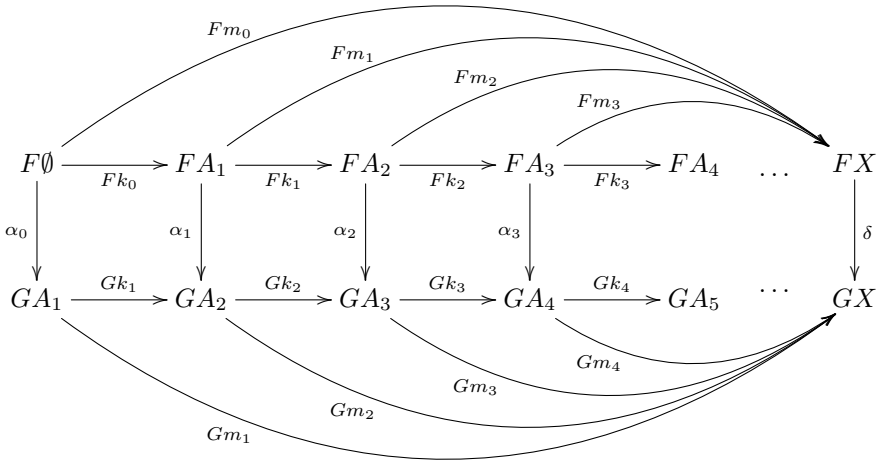
1. Let $(m_i : A_i \rightarrowtail X)_i$ be as in definition 4.3.1 and let $(k_i : m_i \to m_{i+1})_i$ be as in lemma 4.3.3. The least subdialgebra of (X, δ) is the morphism m_{j+1} where j is such that $k_j : m_j \to m_{j+1}$ is an isomorphism.
2. Let $(w_i : X \twoheadrightarrow Z_i)_i$ be as in definition 4.3.2 and let $(k_i : w_{i+1} \to w_i)_i$ be as in lemma 4.3.3. The greatest quotient of (X, δ) is the morphism w_j where j is such that $k_j : w_{j+1} \to w_j$ is an isomorphism.

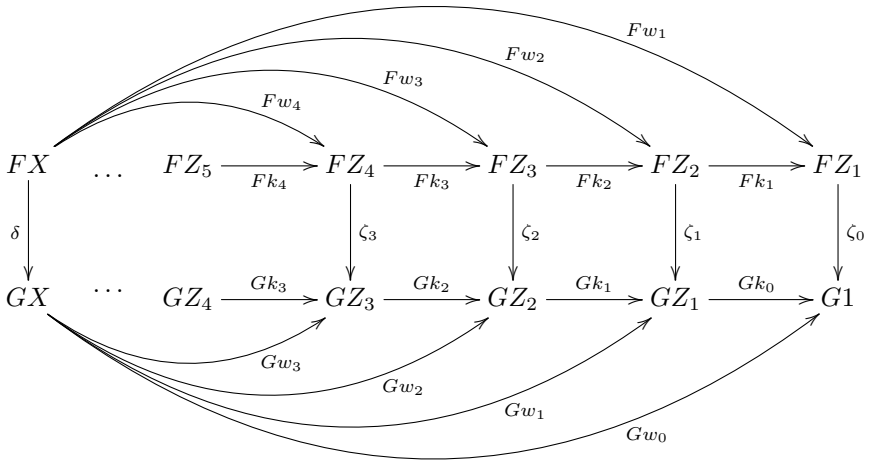
Proof. We prove the first item. Let us write m for m_{j+1} and A for A_{j+1}. We have to show that there is a dialgebra structure $\alpha : FA \to GA$ such that $m : (A, \alpha) \rightarrowtail (X, \delta)$ is a subdialgebra. Since $k_j : m_j \to m$ is an isomorphism it has has an inverse k_j^{-1} with an underlying morphism $k_j^{-1} : A \to A_j$. Thus let $(\alpha_i : FA_i \to GA_{i+1})_i$ be as in lemma 4.3.3 and define $\alpha : FA \to GA$ by $\alpha \coloneqq \alpha_j \circ k_j^{-1}$. Then $m : (A, \alpha) \rightarrowtail (X, \delta)$ is a subdialgebra.

It remains to show that $m : (A, \alpha) \rightarrowtail (X, \delta)$ is the *least* subdialgebra. So consider any other subdialgebra $s : (Y, \upsilon) \rightarrowtail (X, \delta)$. We have to show that there is a unique morphism $h : (Y, \upsilon) \to (A, \alpha)$. To this end we consider the minimisation sequence $(n_i : B_i \rightarrowtail Y)_i$ of (Y, υ) and we show by induction that there exists a sequence of morphisms $(h_i : A_i \rightarrowtail B_i)_i$, simultaneously showing that $s \circ n_j \circ h_j = m_j$ for all ordinals j to load the induction.

- For $j = 0$ we have $A_0 = \emptyset = B_0$ thus $h_0 : A_0 \to B_0$ is the initial identity morphism $\mathsf{i}_\emptyset$. Moreover, $m_0 = \mathsf{i}_X$ thus $s \circ n_0 \circ h_0 = m_0$ by uniqueness of initial morphisms.
- For $j = i + 1$ consider $m_j \coloneqq \mathrm{im}_G(\delta \circ Fm_i)$ and $n_j \coloneqq \mathrm{im}_G(\upsilon \circ Fn_i)$. Let β_i be such that $Gn_j \circ \beta_i = \upsilon \circ Fn_i$ as by the definition of the base. By induction hypothesis there is a morphism $h_i : A_i \rightarrowtail B_i$ such that $s \circ n_i \circ h_i = m_i$ thus we have $Fs \circ Fn_i \circ Fh_i = Fm_i$ and then because s is a subdialgebra we have $\delta \circ Fs \circ Fn_i \circ Fh_i = Gs \circ \upsilon \circ Fn_i \circ Fh_i$ so that we have $Gs \circ Gn_j \circ \beta_i \circ Fh_i = \delta \circ Fm_i$. Now, $m_j = \mathrm{im}_G(\delta \circ Fm_i)$ and therefore there is a unique morphism $h_j : Aj \to Bj$ such that $s \circ n_j \circ h_j = m_j$.

□




Fw_1
Fw_2
Fw_3
Fw_4
FX
$\dots$
FZ_5
Fk_4
FZ_4
Fk_3
FZ_3
Fk_2
FZ_2
Fk_1
FZ_1
δ
ζ_3
ζ_2
ζ_1
ζ_0
GX
$\dots$
GZ_4
Gk_3
GZ_3
Gk_2
GZ_2
Gk_1
GZ_1
Gk_0
$G1$
Gw_3
Gw_2
Gw_1
Gw_0

4.4 Quotients of subdialgebras

In the previous section we have introduced procedures for generating least subdialgebras and greatest quotients. We have stressed the importance of greatest quotients of operational semantics several times. They are the intended denotational semantics. The main use of subdialgebras is to limit the size of the operational semantics or of quotients thereof. Often it is possible to generate for a term an appropriate finite subdialgebra of the operational semantics. If this is the case then then their greatest quotients can be computed by using the simplification sequence. If the subdialgebras were chosen appropriately then equality of the quotients coincides with behavioural equivalence of terms. In many cases, especially in automata theory, generating the least subdialgebras that contains the respective terms suffices. However, for dialgebras in general this may not yield correct results, which is what we investigate in this section.

Proposition 4.4.1. Consider a Set_G^F dialgebra (X, χ).

1. If F preserves pushouts and (A, α) is a subdialgebra of (X, χ) then any quotient of (A, α) is a subdialgebra of a quotient of (X, χ).
2. If G preserves pullbacks and (Z, ζ) is a quotient of (X, χ) then any subdialgebra of (Z, ζ) is a quotient of a subdialgebra of (X, χ).

Proof. We prove the first item. Let $m : (A, \alpha) \rightarrowtail (X, \chi)$ and let $q : (A, \alpha) \twoheadrightarrow (Y, \upsilon)$. If F preserves pushouts then by theorem 3.2.1 Set_G^F has them. Thus let $(\iota_X, \iota_Y, (Z, \zeta))$ be the pushout of m and q. Then $\iota_X : (X, \chi) \to (Z, \zeta)$ is epic because q is epic. In Set and more generally in any topos the pushout of a mono along a morphism is again a mono, c.f. lemma 18 of [10]. Thus $\iota_Y : (Y, \upsilon) \to (Z, \zeta)$ is monic because m is monic. □

Intuitively, if the interaction functor of a dialgebra preserves pushouts then there cannot be interdependence between states. The pushout of two initial morphisms $\mathsf{i}_A : \emptyset \to A$ and $\mathsf{i}_B : \emptyset \to B$ is the coproduct $A + B$ and for coproduct preserving functors it is especially clear that they prevent interdependence. For, if F is such that $F(A + B) \cong FA + FB$ then experiments on A are necessarily disjoint from experiments on B. More generally, the use of a pushout preserving functor effectively enforces that the behaviour of a state is determined only by the state itself and the states that are reachable from it. Indeed, it enforces that any subdialgebra that contains a particular state is sufficiently large to determine the behaviour of that state state in the encompassing dialgebra.

Proposition 4.4.2. Consider a $\mathcal{C}_G^F$ dialgebra (X, χ) and let (A, α) be a subdialgebra of (X, χ). In general a quotient of (A, α) *need not* be a subobject of a quotient of (X, χ).

Proof. As a counterexample we consider a Set_G^F dialgebra (X, ξ) where $FX := 1 + 1 + X \times X$ and $GX := X + X + C \times X + 1$ with $C := \{1, 2\}$. We let $X := \{r, s, t\}$ and define ξ as follows.

$$\begin{array}{rcl} \iota_1(*) & \mapsto & \iota_1(r) \\ \iota_2(*) & \mapsto & \iota_2(t) \\ \iota_3(r, s) & \mapsto & \iota_3(1, r) \\ \iota_3(s, r) & \mapsto & \iota_3(2, s) \\ \iota_3(\_, \_) & \mapsto & \iota_4(*) \end{array}$$

Now (X, χ) is a simple dialgebra. The domain of its least subdialgebra is the dialgebra (A, α) with $A = \{r, t\}$ where α takes $\iota_1(*) \mapsto \iota_1(r)$, $\iota_2(*) \mapsto \iota_2(t)$ and $\iota_3(\_, \_) \mapsto \iota_4(*)$. Thus it is easy to see that r and t are behaviourally equivalent in (A, α). The only proper quotient of (A, α)

is its greatest quotient $q\colon (A,\alpha) \twoheadrightarrow (Z,\zeta)$ that takes $r \mapsto z$ and $t \mapsto z$ where $z \in Z$ is the only element of Z. Now, recall that (X,ξ) is also a simple dialgebra. Clearly (Z,ζ) is not isomorphic to (X,ξ). □

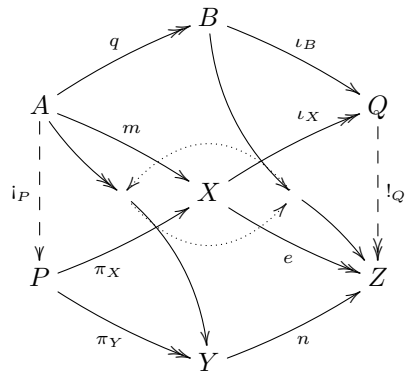
There are cases where the signature functors do not meet the requirements of proposition 4.4.1, yet where the consequences do hold. For example, consider a Set_G^F dialgebra (X,χ) with $FX \coloneqq 1 + X \times X$. We can define a Set_H^F dialgebra (X,ξ) with $HX \coloneqq X + GX$ where ξ is defined by $\iota_1(*) \mapsto (\chi \circ \iota_1)(*)$ and $\iota_2(x,y) \mapsto (\iota_2 \circ \chi)(x)$. Now F does not preserve pushouts and G and H may not preserve pullbacks. However, there is no interdependence between sates; ξ just ignores the second component of the tuple in an experiment $\iota_2(x,y)$. The example illustrates that the signature of a dialgebra can prevent interaction, but it cannot enforce it.

This result implies that we cannot in general correctly decide equivalence of terms by minimising and then simplifying. To be precise, the technique may yield false positives. Interestingly, in automata theory this is *the* technique to decide equivalence of automata. A closer inspection reveals that if such automata are modelled as Set_G^F dialgebras then F does indeed preserve pushouts.

The following theorem establishes that if F preserves pushouts or if G preserves pullbacks then the order of simplification and minimisation is irrelevant. This is especially useful if minimal subdialgebras are finite, in which case their greatest quotient can be computed and equivalence can be decided.

Theorem 4.4.3. Consider a Set_G^F dialgebra (X,δ). Let (A,α) be its least subdialgebra and let (Z,ζ) be its greatest quotient. If F preserves pushouts or G preserves pullbacks then the greatest quotient of (A,α) is the least subdialgebra of (Z,ζ).

Proof. Let $m\colon (A,\alpha) \rightarrowtail (Z,\zeta)$ and $e\colon (X,\chi) \twoheadrightarrow (Z,\zeta)$, let $q\colon (A,\alpha) \twoheadrightarrow (B,\beta)$ be the greatest quotient of (A,α) and let $n\colon (Y,\upsilon) \rightarrowtail (Z,\zeta)$ be the least subdialgebra of (Z,ζ). Then, let (P,π_X,π_Y) be the pullback of n and e in Set and let (ι_B,ι_X,Q) be the pushout of m and q in Set. Note that π_Y and ι_X are epic whereas π_X and ι_B are monic.



If F preserves pullbacks then π_X is a subdialgebra. Since (A,α) is the least subdialgebra of (X,χ) there is a morphism $\mathsf{i}_P\colon A \rightarrowtail P$ underlying a unique dialgebra morphism so that $(\pi_Y \circ \mathsf{i}_P)$ is a dialgebra morphism. By proposition 3.4.5 $\mathrm{im}(\pi_Y \circ \mathsf{i}_P)$ and because (Y,υ) is minimal it is an isomorphism. We have $\mathrm{im}(!_Q \circ \iota_B) \cong \mathrm{im}(e \circ m) \cong n \circ \mathrm{im}(\pi_Y \circ \mathsf{i}_P)$ and therefore $(B,\beta) \cong (Y,\upsilon)$. Otherwise G preserves pushouts and $(!_Q \circ \iota_B)$ is a dialgebra morphism. Then $\overline{\mathrm{im}}(!_Q \circ \iota_B)$ is a dialgebra morphism and it is an isomorphism because (B,β) is simple. Then $\overline{\mathrm{im}}(\pi_Y \circ \mathsf{i}_P) \cong \overline{\mathrm{im}}(e \circ m) \cong \overline{\mathrm{im}}(!_Q \circ \iota_B) \circ q$ hence $(B,\beta) \cong (Y,\upsilon)$. □

Chapter 5

Two Examples

In this chapter we present two small examples to exhibit the use of dialgebras for operational semantics. The first example is a classic. We present a dialgebraic operational semantics for regular expressions that reveals how closely theory of dialgebras relates to the classic theory where regular expressions are interpreted as deterministic automata. As a second example we discuss the calculus of communicating systems (CCS). We include the coalgebraic semantics for the synchronous CCS and a new dialgebraic semantics. The coalgebraic semantics models the behaviour of *individual* processes whereas the dialgebraic semantics models the behaviour of multiple processes at once, including their interactions and the behaviour that results.

5.1 Regular expressions

Regular expressions are used to describe patterns in sequences of characters. Equivalently, they define a language; a set of finite strings that match a particular pattern. We let $a, b \in A$ range over a finite set of symbols. The signature of the syntax of regular expressions extends the signature of a boolean algebra with concatenation $r \cdot s$ and repetition r^*.

Definition 5.1.1 (Syntax)**.**

$$r, s \in X \quad ::= \quad a \in A \;\mid\; \epsilon \;\mid\; \top \;\mid\; \bot \;\mid\; \neg r \;\mid\; r \vee s \;\mid\; r \wedge s \;\mid\; r \cdot s \;\mid\; r^* \qquad \Diamond$$

There is a denotational semantics that interprets a regular expression as a language. Here, a language is a set of stings, each string consisting of characters $a \in A$. We will denote such strings by $\vec{a}, \vec{b} \in A^*$ where A^* denotes the set of all strings over A. The empty string is denoted by ϵ, strings of length one are denoted by a, b and string concatenation is denoted by $\vec{a} \cdot \vec{b}$, overloading the notation used for regular expressions. Finally, repeated concatenation of strings $(\vec{a})^n$ for $n \in \mathbb{N}$ is defined by $(\vec{a})^0 \coloneqq \epsilon$ and $(\vec{a})^{m+1} \coloneqq \vec{a} \cdot (\vec{a})^m$.

The denotational semantics $\mathcal{L} : X \to \mathscr{P}A^*$ interprets regular expressions r as their language.

Definition 5.1.2 (Language)**.**

$$\begin{aligned}
\mathcal{L}(a) &:= \{a\} \\
\mathcal{L}(\epsilon) &:= \{\epsilon\} \\
\mathcal{L}(\bot) &:= \emptyset \\
\mathcal{L}(\top) &:= A^* \\
\mathcal{L}(\neg r) &:= A^* \setminus \mathcal{L}(r) \\
\mathcal{L}(r \vee s) &:= \mathcal{L}(r) \cup \mathcal{L}(s) \\
\mathcal{L}(r \wedge s) &:= \mathcal{L}(r) \cap \mathcal{L}(s) \\
\mathcal{L}(r \cdot s) &:= \{\vec{a} \cdot \vec{b} \mid \vec{a} \in \mathcal{L}(r) \text{ and } \vec{b} \in \mathcal{L}(s)\} \\
\mathcal{L}(r^*) &:= \{(\vec{a})^n \mid \vec{a} \in \mathcal{L}(r) \text{ for } n \in \mathbb{N}\}
\end{aligned}$$ ◊

The main drawback of this semantics is that it does not provide a method to compute equivalence. Two regular expressions r and s are equivalent if $\mathcal{L}(r) = \mathcal{L}(s)$, but many regular expressions have infinite languages which makes a naive comparison problematic. There is an alternative denotational semantics that works around this problem by interpreting expressions as finite recognisers for the language.

Definition 5.1.3 (DFA)**.** A deterministic finite automaton (DFA) is a tuple $(Q, A, \delta, q_0, \chi_F)$ where Q is a finite set of states, A is a finite set of symbols, $\delta : Q \times A \to Q$ is a transition function, $q_0 \in Q$ is the start state of the DFA and $\chi_F : Q \to 2$ is the characteristic function of a set $F \subseteq Q$ of accepting states. ◊

DFAs can be modelled as dialgebras. A carrier $X : \mathcal{C}$ and a tuple of functions $(\chi_i : F_i X \to G_i X)_i$ can be turned into a $\mathcal{C}^{F'}_{G'}$ dialgebra $(X, \coprod_i \chi_i)$ with $F'X := \coprod_n F_n$ and $G'X := \coprod_n F_n$ which allows for the following definition.

Definition 5.1.4 (DFA, dialgebraically)**.** A DFA is a Set^F_G dialgebra $(Q,\ p_{q_0} + \delta + \chi_F)$ where $FX := 1 + (A \times X) + X$ and $GX := X + X + 2$. The morphism $p_{q_0} : 1 \to Q$ points out the start state via $* \mapsto q_0$. The morphism δ is the transition function as in definition 5.1.3 and $\chi_F : Q \to 2$ for $2 := \{0, 1\}$ is the characteristic function of a set $F \subseteq Q$ of accept states. For the DFA to be finite Q must be a finite set of states. ◊

Recently there has been an interest in using 'derivatives' of grammars for parsing algorithms [16]. For regular expressions derivatives have been defined in [3]. The more recent [18] shows their relevance in applications. Now the derivative of a regular expression with respect to a character is again a regular expression and the derivative function is defined inductively on terms. In addition we can inductively define a function $\nu : X \to 2$ that signifies whether the language of a regular expression contains the empty string. We can group these functions together and obtain a dialgebraic operational semantics for regular expressions.

Definition 5.1.5 (Operational semantics)**.** Let $FX \coloneqq (A \times X) + X$ and $GX \coloneqq X + 2$. The dialgebraic operational semantics for regular expressions is the Set_G^F dialgebra $(X,\ \delta + \nu)$ where $\delta \colon A \times X \to X$ and $\nu \colon X \to 2$ are as follows.

$$
\begin{array}{rcl@{\qquad}rcl}
\delta(a,a) & \coloneqq & \epsilon & & & \\
\delta(a,b) & \coloneqq & \bot \text{ if } a \neq b & \nu(a) & \coloneqq & 0 \\
\delta(a,\epsilon) & \coloneqq & \bot & \nu(\epsilon) & \coloneqq & 1 \\
\delta(a,\bot) & \coloneqq & \bot & \nu(\bot) & \coloneqq & 0 \\
\delta(a,\top) & \coloneqq & \top & \nu(\top) & \coloneqq & 1 \\
\delta(a,\neg r) & \coloneqq & \neg\delta(a,r) & \nu(\neg r) & \coloneqq & \text{not } \nu(r) \\
\delta(a, r \vee s) & \coloneqq & \delta(a,r) \vee \delta(a,s) & \nu(r \vee s) & \coloneqq & \nu(r) \text{ or } \nu(s) \\
\delta(a, r \wedge s) & \coloneqq & \delta(a,r) \wedge \delta(a,s) & \nu(r \wedge s) & \coloneqq & \nu(r) \text{ and } \nu(s) \\
\delta(a, r \cdot s) & \coloneqq & (\delta(a,r) \cdot s) \text{ if not } \nu(r) & \nu(r \cdot s) & \coloneqq & \nu(r) \text{ and } \nu(s) \\
 & & (\delta(a,r) \cdot s) \vee \delta(a,s) \text{ otherwise} & \nu(r^*) & \coloneqq & 1 \\
\delta(a, r^*) & \coloneqq & \delta(a,r) \cdot r^* & & &
\end{array}
$$

◊

The theory of dialgebras can be used to derive from the operational semantics a canonical denotational semantics for regular expressions. In particular, the underlying morphism of the greatest quotient of the operational semantics is the canonical denotational semantics.

Definition 5.1.6 (Canonical denotational semantics)**.** Let $(X,\ \delta + \nu)$ be the operational semantics for regular expressions and let $[\![\_]\!] \colon (X,\ \delta + \nu) \twoheadrightarrow (Z, \zeta)$ be its greatest quotient. The canonical denotational semantics for regular expressions is the underlying function $[\![\_]\!] \colon X \to Z$. Two regular expressions r and s are semantically equivalent if $[\![r]\!] = [\![s]\!]$. ◊

This does not give us a method to decide equivalence. For, in order to *compute* whether r and s are equivalent we have to compute whether $[\![r]\!] = [\![s]\!]$. However, both $(X,\ \delta + \nu)$ and (Z, ζ) are infinite, as there are infinitely many behaviourally distinct regular expressions. Thus, to compute whether $[\![r]\!] = [\![s]\!]$ we still need a way to represent them as finite structures. We achieve this by interpreting terms r and s not as elements $[\![r]\!]$ and $[\![s]\!]$ of Z but as deterministic automata instead.

Observe that the signatures of $(X,\ \delta + \nu)$ and (Z, ζ) are *almost* the same as the signature of our dialgebraic DFAs. In fact, for every $r \in X$ and for every $z \in Z$ we can define a deterministic automaton by adding a constant function $p_r \colon 1 \to X$ and $p_z \colon 1 \to Z$ to $\delta + \nu$ and ζ respectively.

Definition 5.1.7. Let r be a regular expression and let $(X,\ \delta + \nu)$ be the dialgebraic operational semantics. Define the constant function $p_r \colon 1 \to X$ by $* \mapsto r$. The term automaton $\mathcal{T}(r)$ of a regular expression is defined as follows.

$$\mathcal{T}(r) \quad \coloneqq \quad (X,\ p_r + \delta + \nu)$$

◊

It is easy to check that adding a constant function $p_r \colon 1 \to X$ to a dialgebra does not affect behavioural equivalence. Therefore for any dialgebra $\mathcal{T}(r) = (X,\ p_r + \delta + \nu)$ the carrier of its greatest quotient is Z. Moreover, the structure is obtained by adding a constant function to (Z, ζ) as follows.

Proposition 5.1.8. Let r be a regular expression and let (Z, ζ) be the greatest quotient of the operational semantics. The greatest quotient of $\mathcal{T}(r)$ is the dialgebra $(Z,\ p_{[\![r]\!]} + \zeta)$ where $p_{[\![r]\!]}$ takes $* \mapsto [\![r]\!]$.

In addition, for any simplified dialgebra $(Z,\, p_{[\![r]\!]} + \zeta)$ the element $[\![r]\!] \in Z$ is present in all its subdialgebras, where for ease of writing we assume the underlying morphisms to be inclusions. Again, this is easy to verify. Indeed, as per the minimisation procedure $[\![r]\!]$ will be part of the second approximation.

Proposition 5.1.9. Let r and s be regular expressions and assume that $[\![r]\!] = [\![s]\!]$. If (B, β) is a subdialgebra of $(Z,\, p_{[\![r]\!]} + \zeta)$ then $[\![p]\!] = [\![s]\!]$ is an element of B.

We can now decide equivalence. In particular, we *can* compute the greatest quotient of a minimised term automaton. The minimisation technique of section 4.3 and the inductive definition of the operational semantics are sufficient to construct minimised term automata. For regular expressions this is always finite so that we can use the simplification procedure to compute its greatest quotient. This motivates the following denotational semantics for regular expressions.

Definition 5.1.10. Let $\mathrm{M}(Y, \upsilon)$ denote the least subdialgebra of a dialgebra (Y, υ) and let $\mathrm{S}(Y, \upsilon)$ denote its greatest quotient. Define $\mathcal{M}(r)$ to take a regular expression r to its simplified, minimised term automaton.

$$\mathcal{M}(r) \quad := \quad \mathrm{SM}(\mathcal{T}(r)) \qquad \Diamond$$

Take note of the fact that for any regular expression r the automaton $\mathcal{M}(r)$ is finite and computable. Now, since the composition functor of a dialgebraic DFA preserves pushouts, theorem 4.4.3 applies. Thus the computable denotational semantics $\mathcal{M}(\_)$ coincides with the canonical denotational semantics of definition 5.1.6.

Theorem 5.1.11. Let r and s be regular expressions. They are semantically equivalent if and only if their simplified, minimised term automata coincide.

$$[\![r]\!] = [\![s]\!] \quad \Longleftrightarrow \quad \mathcal{M}(r) = \mathcal{M}(s)$$

Proof. First, if $[\![r]\!] = [\![s]\!]$ then $\mathrm{S}(\mathcal{T}(r)) = (Z,\, p_{[\![r]\!]} + \zeta) = (Z,\, p_{[\![s]\!]} + \zeta) = \mathrm{S}(\mathcal{T}(s))$ Hence, $\mathrm{MS}(\mathcal{T}(r)) = \mathrm{MS}(\mathcal{T}(s))$ which is $\mathcal{M}(s)$ by theorem 4.4.3. Conversely, assume that $\mathcal{M}(r) = \mathcal{M}(s)$. Again, this implies $\mathrm{MS}(\mathcal{T}(r)) = \mathrm{MS}(\mathcal{T}(s))$, that is, $\mathrm{M}(Z,\, p_{[\![r]\!]} + \zeta) = \mathrm{M}(Z,\, p_{[\![s]\!]} + \zeta)$. By proposition 5.1.9 $[\![r]\!]$ and $[\![s]\!]$ are elements of this automaton. Moreover, its structure takes $\iota_1(*) \mapsto [\![r]\!] = [\![s]\!]$ because it is a subdialgebra of both $(Z,\, p_{[\![r]\!]} + \zeta)$ and $(Z,\, p_{[\![s]\!]} + \zeta)$. Thus we conclude $[\![r]\!] = [\![s]\!]$. □

5.2 Calculus of communicating systems

In this section we develop a dialgebraic semantics for the calculus of communicating systems (CCS) [17]. The CCS is a suitable subject for investigating dialgebraic semantics for interactive systems, where in particular we wish to model interaction between processes and the resulting behaviour by a dialgebraic operational semantics. Terms of the CCS represent processes that may or may not interact with each other. Any interaction that does occur, takes place by sending and receiving messages. We consider the synchronous CCS, where any such send and receive interaction must take place in a synchronised fashion.

Definition 5.2.1 (Syntax)**.** Let $a, b \in A$ range over a countable set of names. Processes $p, q, r, s \in X$ and actions $\alpha, \beta \in A$ are generated by the following grammars.

$$\begin{array}{lcl} \alpha \in C & ::= & \tau \;\mid\; \overline{a} \;\mid\; a \\ p \in X & ::= & \epsilon \;\mid\; \alpha.p \;\mid\; p_1 + p_2 \;\mid\; p_1 \parallel p_2 \end{array} \qquad \Diamond$$

Intuitively, a process $a.p$ (*recieve* a then p) waits until it receives an a from a parallel process, after which it continues as p. Similarly, a process $\overline{a}.p$ (*send* a then p) waits until it has delivered an a to a parallel process and then continues as p. A process $\tau.p$ does not wait. It does an 'internal step' and continues as p.

Coalgebraic semantics

We first consider the coalgebraic operational semantics for the CCS. However, given the intention to model communicating processes, 'operational' may be a bit of a misnomer. At least, it does not accurately model computation steps that are performed during execution. For example, a process $a.p$ executed in isolation, will be waiting indefinitely for a message a. However, it will not recieve this message and it will never continue as p as suggested by this operational semantics.

Definition 5.2.2 (Coalgebraic operational semantics)**.** Let $HX := \mathscr{P}(C \times X)$. The coalgebraic operational semantics for the CCS is the coalgebra (X, δ) where $\delta \colon X \to HX$ is defined as follows.

$$\begin{array}{rcl}
\epsilon & \mapsto & \emptyset \\
\alpha.p & \mapsto & \{(\alpha,\, p)\} \\
p + q & \mapsto & \delta(p) \cup \delta(q) \\
p \parallel q & \mapsto & \{(\alpha,\, p' \parallel q\,) \mid (\alpha, p') \in \delta(p)\} \\
& \cup & \{(\alpha,\, p \parallel q') \mid (\alpha, q') \in \delta(q)\} \\
& \cup & \{(\tau,\, p' \parallel q') \mid (c, p') \in \delta(p) \text{ and } (\overline{c}, q') \in \delta(q)\} \\
& \cup & \{(\tau,\, p' \parallel q') \mid (\overline{c}, p') \in \delta(p) \text{ and } (c, q') \in \delta(q)\}
\end{array}$$

◊

The operational semantics for the CCS is commonly presented by a set of transition rules rather than an inductive function definition. For completeness we include the corresponding transition rules below. The rules define a relation $(\to) \subseteq X \times (C \times X)$ where the notation $p \xrightarrow{\alpha} q$ is used in place of $p \to (\alpha, q)$. Then δ takes $p \mapsto (\alpha, q)$ if and only if $p \xrightarrow{\alpha} q$.

$$\frac{}{\alpha.p \xrightarrow{\alpha} p} \qquad \frac{p \xrightarrow{\alpha} p'}{p + q \xrightarrow{\alpha} p'} \qquad \frac{q \xrightarrow{\alpha} q'}{p + q \xrightarrow{\alpha} q'}$$

$$\frac{p \xrightarrow{\alpha} p'}{p \parallel q \xrightarrow{\alpha} p' \parallel q} \qquad \frac{q \xrightarrow{\alpha} q'}{p \parallel q \xrightarrow{\alpha} p \parallel q'} \qquad \frac{p \xrightarrow{c} p' \qquad q \xrightarrow{\overline{c}} q'}{p \parallel q \xrightarrow{\tau} p' \parallel q'} \qquad \frac{p \xrightarrow{\overline{c}} p' \qquad q \xrightarrow{c} q'}{p \parallel q \xrightarrow{\tau} p' \parallel q'}$$

There are many different notions of behavioural equivalence for CCS terms and for process calculi in general. We consider what is referred to as strong bisimilarity, which is the standard notion of coalgebraic behavioural equivalence as applied to the operational semantics above. Thus, terms are strongly bisimilar if and only if there is a coalgebra morphism that identifies them. In contrast, in a so called weak bisimulation a single internal τ–step is considered to be equivalent with multiple consecutive such steps. Usually weak bisimulations are defined based on the operational semantics defined previously as a non–standard bisimulation. Incidentally, it is certainly possible to define an alternative operational semantics for which weak bisimulation equivalence does coincide with the standard notion of coalgebraic behavioural equivalence.

Definition 5.2.3. Let (X, δ) be the coalgebraic operational semantics of definition 5.2.2 and let $[\![\_]\!]\colon (X,\delta) \twoheadrightarrow (Z,\zeta)$ be its greatest quotient. Two processes p and q are semantically equivalent if $[\![p]\!] = [\![q]\!]$. $\Diamond$

Any coalgebraic semantics for program terms imposes a notion of process identity. Let us assume a G–coalgebra (X, δ). The transition function δ models the state of processes over time. Any state $x \in X$ evolves in one time–step to a collection of states $\delta(x) \in GX$. Now if $GX := \mathscr{P}X$ then x evolves in one step to a set of states. The usual intuitive interpretation of this is that x is a state of a non–deterministic process that may evolve to any one of the states $x' \in \delta(x)$. Thus the process is in one single state at any given time, but it is is not determined in which particular state. This intuition is strongly present in the above semantics for the CCS. However, there is nothing that precludes us from another intuitive interpretation of a structure $\delta\colon X \to \mathscr{P}X$. We could in fact let it model concurrency instead of non–determinism. In that case, a transition $x \mapsto X'$ is to be understood as a process that 'splits' at state x into a series of new processes, one for each $x' \in X'$. Thus now the process can be in multiple states concurrently at any given point in time. It is an interesting exercise to define a coalgebraic semantics for the CCS where the powerset functor models concurrency rather than non–determinism. This is however not our motivation for the above discussion. Instead, we intend to advice the reader that any such intuitive understanding of state and process identity may be problematic in a dialgebraic semantics.

Dialgebraic semantics

Since dialgebras allow input and output to be separated the first thing that comes to mind is to define a dialgebraic operational semantics for the CCS in which received messages are modelled as input and sent messages are modelled as output. We would consider a Set^F_G dialgebra (X, γ) with $FX := X + X \times A$ and $GX := X + A \times X$. Note that in such a semantics process identity and non–determinism is modelled similar as in the coalgebraic semantics. Indeed, such a semantics models the subjective world–view of each process as it evolves over time. Interestingly, a dialgebraic operational semantics can define the interactions of processes and the behaviour that results from a third person point of view instead.

Definition 5.2.4 (Dialgebraic operational semantics)**.** Let $FX := X + X^2$ and $GX := \mathscr{P}X + \mathscr{P}(X^2)$. The dialgebraic operational semantics for the synchronous CCS is the dialgebra $(X,\, \delta_1 + \delta_2)$ where $\delta_1\colon X \to \mathscr{P}X$ and $\delta_1\colon X^2 \to \mathscr{P}(X^2)$ are defined as follows.

$$
\begin{aligned}
\delta_1(\epsilon) &:= \emptyset \\
\delta_1(\alpha.p) &:= \{p\} \text{ if } \alpha = \tau \\
&\quad\ \emptyset \text{ otherwise} \\
\delta_1(p+q) &:= \delta_1(p) \cup \delta_1(q) \\
\delta_1(p \,\|\, q) &:= \{p' \,\|\, q \mid p' \in \delta_1(p)\} \cup \\
&\quad\ \{p \,\|\, q' \mid q' \in \delta_1(q)\} \cup \\
&\quad\ \{p' \,\|\, q' \mid (p', q') \in \delta_2(p, q)\}
\end{aligned}
\qquad
\begin{aligned}
\delta_2(\epsilon, q) &:= \emptyset \\
\delta_2(\alpha.p,\, \beta.q) &:= \{(p,q)\} \text{ if } \{\alpha,\beta\} = \{\overline{a}, a\} \text{ for some } a \\
&\quad\ \emptyset \text{ otherwise} \\
\delta_2(p+q,\, r) &:= \delta_2(p,r) \cup \delta_2(q,r) \\
\delta_2(p \,\|\, q,\, r) &:= \{(p' \,\|\, q,\, r') \mid (p', r') \in \delta_2(p,r)\} \cup \\
&\quad\ \{(p \,\|\, q', r') \mid (q', r') \in \delta_2(q,r)\} \\
\delta_2(p,q) &:= \{(p', q') \mid (q', p') \in \delta_2(q,p)\}
\end{aligned}
\qquad \Diamond
$$

The transition functions of the semantics above can be defined by a set of transition rules. For completeness we present the set of rules below. The reader can check that $p \to p'$ only if there is a $p' \in \delta_1(p)$ and $(p, q) \to (p', q')$ only if there is a $(p', q') \in \delta_2(p, q)$.

$$\frac{}{\tau.p \to p} \qquad \frac{p \to p'}{p + q \to p'} \qquad \frac{q \to q'}{p + q \to q'}$$

$$\frac{p \to p'}{p \parallel q \to p' \parallel q} \qquad \frac{q \to q'}{p \parallel q \to p \parallel q'} \qquad \frac{(p, q) \to (p', q')}{p \parallel q \to p' \parallel q'}$$

$$\frac{}{(a.p,\, \overline{a}.q) \to (p, q)} \qquad \frac{(p, r) \to (p', r')}{(p + q, r) \to (p', r')} \qquad \frac{(q, r) \to (q', r')}{(p + q, r) \to (q', r')}$$

$$\frac{(p, r) \to (p', r')}{(p \parallel q,\, r) \to (p' \parallel q,\, r')} \qquad \frac{(q, r) \to (q', r')}{(p \parallel q,\, r) \to (p \parallel q',\, r')} \qquad \frac{(q, p) \to (q', p')}{(p, q) \to (p', q')}$$

Intuitively, a δ_2 transition $(p, q) \mapsto R$ indicates that two processes p and q are able to communicate with each other. They may do so in multiple ways, each of which leads to a pair $(p', q') \in R$ where p' is the continuation of p and q' is the continuation of q. Be aware though that this intuition of p evolving to p' and q evolving to q' is just that. The dialgebraic semantics does not enforce process identity and one may just as well decide to regard q' a residual of p and p' a residual of q.

Given the dialgebraic operational semantics the denotational semantics is defined in the usual way, as the greatest quotient of the operational semantics. The denotational semantics thus obtained coincides with the coalgebraic semantics.

Definition 5.2.5. Let $(X, \delta_1 + \delta_2)$ be the dialgebraic operational semantics for the CCS as per definition 5.2.4 and let $(\!|\_|\!) : (X, \delta) \twoheadrightarrow (Z, \zeta)$ be its greatest quotient. Two processes p and q are semantically equivalent if $(\!|p|\!) = (\!|q|\!)$. $\Diamond$

To prove that the dialgebraic semantics and the coalgebraic semantics coincide we prove something more general. Any coalgebra morphism out of the coalgebraic operational semantics is a dialgebra morphism out of the dialgebraic operational semantics. Conversely, any *quotient* of the dialgebraic operational semantics is a quotient of the coalgebraic operational semantics.

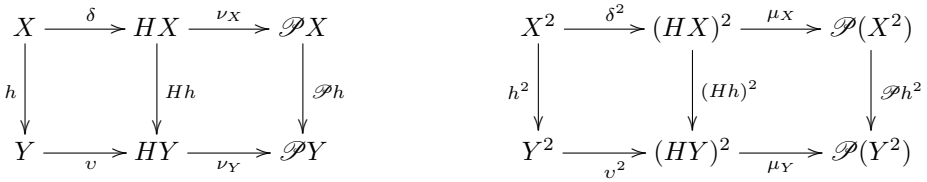
Lemma 5.2.6. Let (X, δ) be the coalgebraic operational semantics and let $(X, \delta_1 + \delta_2)$ be the dialgebraic operational semantics for the CCS. If $h : X \to Y$ extends to a coalgebra morphism $h : (X, \delta) \twoheadrightarrow (Y, \upsilon)$ then it extends to a dialgebra morphism $h : (X, \delta_1 + \delta_2) \to (Y, \xi)$.

Proof. In what follows, let $HX := \mathscr{P}(C \times X)$, $FX := X + X^2$ and $GX := \mathscr{P}X + \mathscr{P}(X^2)$. Assume that $h : (X, \delta) \to (Y, \upsilon)$ is a coalgebra morphism and thus $\upsilon \circ h = Hh \circ \delta$. We have to show that there is a dialgebra structure $\xi : FY \to GY$ such that $h : (X, \delta_1 + \delta_2) \to (Y, \xi)$ is a dialgebra morphism. To this end we define two natural transformations $\nu_X : HX \dot{\to} \mathscr{P}X$ and $\mu_X : (HX)^2 \dot{\to} \mathscr{P}(X^2)$ as follows.

$$\begin{aligned} \nu_X(S) &:= \{p \mid (\tau, p) \in S\} \\ \mu_X(R, S) &:= \{(p, q) \mid (\alpha,\, p) \in R \text{ and } (\beta,\, q) \in S \text{ where } \{\alpha, \beta\} = \{a, \overline{a}\} \text{ for some } a \in A\} \end{aligned}$$

Now define $\xi := \xi_1 + \xi_2$ where $\xi_1 := (\nu_Y \circ \upsilon)$ and $\xi_2 := (\mu_Y \circ \upsilon^2)$. We have to show that $(\xi_1 + \xi_2) \circ (h + h^2) = (\mathscr{P}h + \mathscr{P}h^2) \circ (\delta_1 + \delta_2)$. We start with $\xi_1 \circ h = \mathscr{P}h \circ \delta_1$. By definition

$\xi_1 \circ h$ is $\nu_Y \circ \upsilon \circ h$ which is $\mathscr{P}h \circ \nu_X \circ \delta$ by naturality of ν. Thus it is sufficient to show that $\nu_X \circ \delta = \delta_1$ which is follows by induction on the structure of terms.



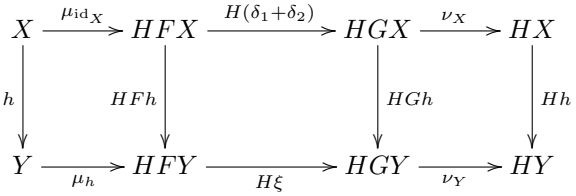
On to $\xi_2 \circ h^2 = \mathscr{P}h^2 \circ \delta_2$. By definition $\xi_2 \circ h^2$ is $\mu_Y \circ \upsilon^2 \circ h^2$ which is $\mathscr{P}h^2 \circ \mu_X \circ \delta^2$. Thus it is sufficient to show that $\mu_X \circ \delta^2 = \delta_2$ which can be shown by induction. □

Lemma 5.2.7. Let (X, δ) be the coalgebraic operational semantics and let $(X, \delta_1 + \delta_2)$ be the dialgebraic operational semantics. If the morphism $h : X \twoheadrightarrow Y$ extends to a dialgebra morphism $h : (X, \delta_1 + \delta_2) \twoheadrightarrow (Y, \xi)$ then it extends to a coalgebra morphism $h : (X, \delta) \twoheadrightarrow (Y, \upsilon)$.

Proof. Again, let $HX := \mathscr{P}(C \times X)$, $FX := X + X^2$ and $GX := \mathscr{P}X + \mathscr{P}(X^2)$. We have to show that there is a coalgebra structure $\upsilon : Y \to HY$ such that $h : (X, \delta) \to (Y, \upsilon)$ is a coalgebra morphism. We define a natural transformations $\mu : \mathrm{Id} \dot{\to} HF$ and a natural transformation $\nu : HG \dot{\to} H$. However we index μ by objects of $(X/\mathrm{Quot}\,\mathrm{Set})$ rather than by objects of Set. Thus, μ is not a natural transformation from $\mathrm{Id}_{\mathrm{Set}}$ to $HF : \mathrm{Set} \to \mathrm{Set}$ but rather from $\mathrm{Id}_{\mathcal{C}}$ to $HF : \mathcal{C} \to \mathcal{C}$ where $\mathcal{C}$ abbreviates $(X/\mathrm{Quot}\,\mathrm{Set})$. This is sufficient, for we only require naturality 'with respect to' quotients $h : X \twoheadrightarrow Y$ of $(X/\mathrm{Quot}\,\mathrm{Set})$.

$$\begin{aligned} \mu_h(y) &:= \{(\tau, \iota_1(y)),\ (a,\ \iota_2(y, h(\overline{a}.\epsilon)),\ (\overline{a},\ \iota_2(y, h(a.\epsilon)) \mid a \in A\} \\ \nu_Y(S) &:= \{(\alpha, x) \mid (\alpha, \iota_1(U)) \in S \text{ and } x \in U\} \cup \{(\alpha, x) \mid (\alpha, \iota_2(R)) \in S \text{ and } (x, \_) \in R\} \end{aligned}$$

Now define υ by $\upsilon := \nu_Y \circ H(\xi) \circ \mu_h$. We have to show that $\upsilon \circ h = Hh \circ \delta$. Since h is a dialgebra morphism $\xi \circ Fh = Gh \circ (\delta_1 + \delta_2)$ and by naturality of μ and ν the following diagram commutes. Thus, it is sufficient to show that $\delta = \nu_X \circ H(\delta_1 + \delta_2) \circ \mu_{\mathrm{id}_X}$ which is a tedious but straightforward exercise.



□

With lemma 5.2.6 and 5.2.7 in place we can conclude that the dialgebraic semantics does indeed coincide with the coalgebraic semantics.

Theorem 5.2.8. Coalgebraic behavioural equivalence coincides with dialgebraic behavioural equivalence.

$$[\![s]\!] = [\![p]\!] \quad \Longleftrightarrow \quad (\!|s|\!) = (\!|p|\!)$$

Chapter 6

Further research

In this thesis we have investigated the use of dialgebras for program semantics. We have discussed the theory of universal dialgebra and its applications to operational semantics. In particular, the greatest quotient of a dialgebraic operational semantics provides us with a canonical denotational semantics that characterises behavioural equivalence. Subdialgebras on the other hand may be used as suitable denotations for individual terms.

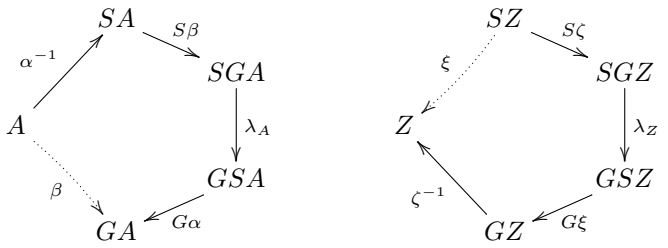
The main result is a generalisation of the minimisation and simplification procedures for algebras and coalgebras. We have generalised images and coimages of morphisms to what we have called the base and the cobase of a morphism. Intuitively, the G–base of a morphism can be used to extract elements of X from elements of GX whereas the F–cobase of a morphism may be used to retrieve a partitioning on X that is in some sense compatible with a partitioning on FX. The notion of a base and a cobase have allowed us to define the minimisation and simplification sequences for dialgebras, generalising both the minimisation procedure for algebras and the simplification procedure for coalgebras. Dialgebra minimisation and simplification can be used with a dialgebraic operational semantics to derive small, suitable denotations for terms. If equivalence is decidable at all then the technique allows program equivalence to be decided.

Finally, we have included two examples. A dialgebraic semantics for regular expressions illustrates how neatly the dialgebraic approach matches with the classic semantics where regular expressions are interpreted as deterministic automata. The dialgebraic semantics for the synchronous CCS that we have included illustrates that dialgebras are able to model multiple processes and their interaction and behaviour at once, in contrast with the coalgebraic semantics that models the behaviour of processes individually.

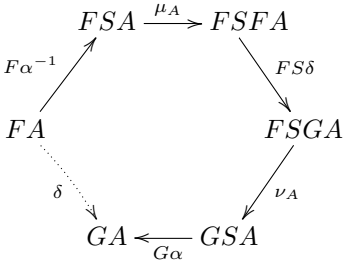
Induction and compositionality

A topic that we have barely touched upon is *how* an operational semantics on a set of terms can be defined. A *coalgebraic* operational semantics (A, β) on a set of terms A given as an initial S–algebra (A, α) can be defined by means of a natural transformation $\lambda_A : SGA \dot{\to} GSA$ c.f. [2]. The type of this natural transformation enforces an induction scheme for defining a coalgebra structure β on terms, defined *inductively* by $\beta := G\alpha \circ \lambda_A \circ S\beta \circ \alpha^{-1}$. In addition, the final coalgebra (Z, ζ) for the semantics (A, β) thus obtained is an algebraic semantics (Z, ξ),

where the algebra structure $\xi : SZ \to Z$ is defined *coinductively* by $\xi := \zeta^{-1} \circ G\xi \circ \lambda_Z \circ S\zeta$.



It is clear to us that a dialgebraic operational semantics (A, δ) on a set of terms (A, α) given as an initial S–algebra can be defined by *two* natural transformations $\mu_A : FSA \dot{\to} FSFA$ and $\nu_A : GSGA \dot{\to} GSA$. The dialgebra structure $\delta : FA \to GA$ is then defined *inductively* by $\delta := G\alpha \circ \nu_A \circ FS\delta \circ \mu_A \circ F\alpha^{-1}$. Intuitively, μ_A specifies how to construct experiments from decomposed terms whereas ν_A specifies how the outcome of those experiments is to be recombined.



However, not all dialgebraic semantics can be inductively defined by this schema. Be aware though that the same remark applies to the single natural transformation $\lambda_A : SGA \dot{\to} GSA$ in the coalgebraic case, which intuitively, can only depend on the topmost constituent of a term and the topmost constituents of the behaviour observed from its immediate subterms.

For the dialgebraic case it is not yet clear if, and how a pair of natural transformations defines an algebra structure on the carrier of a simple dialgebra. Whereas in a final coalgebra the inverse of the coalgebra structure can be used, this is not a possibility in a simple (or final) dialgebra. The problem does not appear to be too difficult though. A hint as to how this may be solved lies in the proof of theorem 5.2.8 where we have used natural transformations to compare a Set_G^F dialgebra to a Set_H coalgebra and vice versa. We expect that the same technique can be used to retrieve an algebra structure on the carrier of a simple dialgebra, thus retrieving a compositional semantics. On the other hand, the inductive and coinductive definitions for the coalgebraic case employ Lambek's lemma. We may attempt to use the restricted version of proposition 3.5.5 and especially proposition 4.2.3 to generalise the technique to simple and minimal dialgebras, presumably by switching to natural transformations indexed by subobjects and quotients.

Base and cobase construction

Our main result, the minimisation and simplification sequences for dialgebras, makes heavy use of the base and the cobase of morphisms. Constructing the base of a morphism of Set appears an easy task that agrees with the intuition of containers and their contents. Still, we would like to discuss these constructions more thoroughly and investigate how the base of a morphism is

constructed in other categories, especially in toposes.

In contrast with the base, cobase constructions are surprisingly involved, even in Set. A concrete algorithm that constructs the cobase of a morphism by some other method than brute force should be provided. Before we found the simplification sequence, we had been working on a technique to compare dialgebras to coalgebras and use coalgebra simplification instead. One of the main components of this technique involved generating all experiments with a particular element in their *base*, which can be achieved by constructing multi–hole contexts for elements of a set FX and then 'plugging the hole' with a particular element $x \in X$. Inspired by those experiments we think that a more thorough investigation of the cobase can reveal connections between congruences, contexts and also with the base of morphisms. Note, a.o. that the F–cobase of an F–algebra structure is an algebra morphism and thus its kernel is a congruence relation. Finally, there appears to be a connection between the *cobase* of a morphism and contexts and with this a connection with 'derivatives of functors' [13, 14] which is a fascinating story all by itself.

Suitable denotations

In chapter 4 we have posed sufficient conditions for the minimisation of a simplification to coincide with the simplification of a minimisation. This provides a method for computing denotations of terms, if possible at all, by simplifying the least subdialgebra of the operational semantics that contains this term. However, if the conditions are not met then the method may identify behaviourally distinct terms. It should be investigated what is the smallest subdialgebra large enough to characterise the behaviour of a term in general, so that a general method for deciding equivalence can be developed.

Sorted dialgebras

Most of the propositions stated are general enough to apply to dialgebras on multi–sorted sets. However, it should be interesting to investigate instances of multi–sorted dialgebras. We have the feeling that this will result in insights into the relation between algebras, dialgebras and coalgebras. This may not work out, but let us provide a rough sketch.

Sorted algebras can be used to retrieve a compositional semantics from a semantics that is not compositional, where intuitively the sort of a term is used to restrict the contexts in which it may be placed (see e.g. [5] for our source of inspiration). An other way of seeing this is that the sorts are essentially equivalence classes of contexts with respect to which the term has the same meaning. With this in mind, we think that a Set_G^F dialgebra may be modelled as a multi–sorted algebra or dually, as a multi–sorted coalgebra. *Very* roughly, given a dialgebra structure $\delta : FX \to GX$ an idea would be to equip elements x with sorts, where sorts are appropriate quotients of elements of FX that contain x.

Simplification of input

The separation of input and output in dialgebras has an additional benefit that requires further investigation. Consider a dialgebra structure $\delta : C \times X \to D \times X$. The structure imposes a behavioural equivalence on states $x \in X$. However, every *state* also imposes an equivalence on the experiments $\{(c, x) \mid c \in C\}$ that involve it. Equivalently in this case, every state x defines a structure $\delta_x : C \to D \times X$ that takes $c \mapsto \delta(c, x)$ thus imposing an equivalence on $C \cong \{(c, x) \mid c \in C\}$. In actual implementations this is very useful. For example, we have been

able to use it in an implementation of the semantics of definition 5.1.5 to allow for an infinite alphabet while keeping denotations of terms finite and computable. An attempt to generalise this technique to other signatures should prove useful to applications and should result in a better understanding of dialgebras.

Dualise and generalise

While working with dialgebras we have found a recurring theme. Many results had been known in universal algebra before they were dualised to universal coalgebra. When a pair of dual results is known an attempt can be made to find a restricted class of algebras to which results from universal coalgebra apply and a dual restricted class of coalgebras to which results from universal algebra apply. The results may then be merged into a general result for dialgebras. This approach is illustrated by theorem 3.2.1. Finding the simplification and minimisation sequences was not as straightforward however, where rather than restricting the class of algebras and coalgebras, the concept of image and a coimage had to be generalised. Since the theory of universal algebra and universal coalgebra is well developed there are *many* opportunities for generalisations to dialgebras. Foremost this includes varieties and covarieties, equations and coequations, free and cofee dialgebras, and monads and comonads. We would very much like to see some of those topics investigated.

Bibliography

[1] S. Awodey. *Category Theory*. Oxford Logic Guides. Clarendon Press, 2006.

[2] F. Bartels. *On Generalised Coinduction and Probabilistic Specication Formats (Distributive laws in coalgebraic modelling)*. PhD thesis, Vrije Universiteit Amsterdam, 2004.

[3] Janusz A. Brzozowski. Derivatives of regular expressions. *J. ACM*, 11(4):481–494, October 1964.

[4] Vincenzo Ciancia. Interaction and observation, categorically. In Alexandra Silva, Simon Bliudze, Roberto Bruni, and Marco Carbone, editors, *Proceedings Fourth Interaction and Concurrency Experience*, volume 59 of *EPTCS*, pages 25–36, 2011.

[5] Marcelo Fiore, Gordon Plotkin, and Daniele Turi. Abstract syntax and variable binding (extended abstract). In *Proceedings 14th LICS*, pages 193–202. IEEE Computer Science Press, 1999.

[6] J. A. Goguen and J. W. Thatcher. Initial algebra semantics. In *Proceedings of the 15th Annual Symposium on Switching and Automata Theory*, SWAT '74, pages 63–77, Washington, DC, USA, 1974. IEEE Computer Society.

[7] J. A. Goguen, J. W. Thatcher, E. G. Wagner, and J. B. Wright. Initial algebra semantics and continuous algebras. *J. ACM*, 24(1):68–95, January 1977.

[8] Andrew D. Gordon. A tutorial on co-induction and functional programming. In *Glasgow Functional Programming Workshop*, pages 78–95. Springer, 1994.

[9] Bart Jacobs and Jan Rutten. A tutorial on (co)algebras and (co)induction. *EATCS Bulletin*, 62:62–222, 1997.

[10] Stephen Lack and Pawel Sobocinski. Toposes are adhesive. In Andrea Corradini, Hartmut Ehrig, Ugo Montanari, Leila Ribeiro, and Grzegorz Rozenberg, editors, *Graph Transformations*, volume 4178 of *Lecture Notes in Computer Science*, pages 184–198. Springer Berlin/Heidelberg, 2006.

[11] S.M. Lane. *Categories for the Working Mathematician*. Graduate Texts in Mathematics. Springer, 1998.

[12] F.W. Lawvere and S.H. Schanuel. *Conceptual Mathematics: A First Introduction to Categories*. Cambridge University Press, 1997.

[13] Conor Mcbride. The derivative of a regular type is its type of one-hole contexts (extended abstract), 2001.

[14] Conor McBride. Clowns to the left of me, jokers to the right (pearl): dissecting data structures. In *Proceedings of the 35th annual ACM SIGPLAN-SIGACT symposium on Principles of programming languages*, POPL '08, pages 287–295, New York, NY, USA, 2008. ACM.

[15] John McCarthy. Towards a mathematical science of computation. In Cicely M. Popplewell, editor, *Information Processing 62: Proceedings of IFIP Congress 1962*, pages 21–28, Amsterdam, 1963. North-Holland.

[16] Matthew Might, David Darais, and Daniel Spiewak. Parsing with derivatives: a functional pearl. In *Proceedings of the 16th ACM SIGPLAN international conference on Functional programming*, ICFP '11, pages 189–195, New York, NY, USA, 2011. ACM.

[17] Robin Milner. *A Calculus of Communicating Systems*, volume 92 of *Lecture Notes in Computer Science*. Springer, 1980.

[18] Scott Owens, John Reppy, and Aaron Turon. Regular-expression derivatives re-examined. *Journal of Functional Programming*, 19(2):173–190, March 2009.

[19] John Power and Hiroshi Watanabe. An axiomatics for categories of coalgebras. *Electronic Notes in Theoretical Computer Science*, 11(0):158 – 175, 1998. CMCS '98, First Workshop on Coalgebraic Methods in Computer Science.

[20] J.J.M.M. Rutten. Universal coalgebra: a theory of systems. *Theoretical Computer Science*, 249(1):3 – 80, 2000. Modern Algebra.

[21] Sam Staton. Relating coalgebraic notions of bisimulation. *Logical Methods in Computer Science*, 7(1), 2011.

[22] Daniele Turi and Jan Rutten. On the foundations of final coalgebra semantics: non-well-founded sets, partial orders, metric spaces. *Mathematical Structures in Computer Science*, 8(05):481–540, 1998.

[23] George Voutsadakis. Universal dialgebra: Unifying universal algebra and coalgebra. *Far East Journal of Mathematical Sciences*, 44(1):1 – 53, 2009.