Categorical Term Rewriting: Monads and Modularity

Christoph Lüth

Doctor of Philosophy
University of Edinburgh
1997

Abstract

Term rewriting systems are widely used throughout computer science as they provide an abstract model of computation while retaining a comparatively simple syntax and semantics. In order to reason within large term rewriting systems, structuring operations are used to build large term rewriting systems from smaller ones. Of particular interest is whether key properties are *modular*, that is, if the components of a structured term rewriting system satisfy a property, then does the term rewriting system as a whole? A body of literature addresses this problem, but most of the results and proofs depend on strong syntactic conditions and do not easily generalize. Although many specific modularity results are known, a coherent framework which explains the underlying principles behind these results is lacking.

This thesis posits that part of the problem is the usual, concrete and syntax-oriented semantics of term rewriting systems, and that a semantics is needed which on the one hand elides unnecessary syntactic details but on the other hand still possesses enough expressive power to model the key concepts arising from the term structure, such as substitutions, layers, redexes etc. Drawing on the concepts of category theory, such a semantics is proposed, based on the concept of a *monad*, generalising the very elegant treatment of equational presentations in category theory. The theoretical basis of this work is the theory of enriched monads.

It is shown how structuring operations are modelled on the level of monads, and that the semantics is compositional (it preserves the structuring operations). Modularity results can now be obtained directly at the level of combining monads without recourse to the syntax at all. As an application and demonstration of the usefulness of this approach, two modularity results for the disjoint union of two term rewriting systems are proven, the modularity of confluence (Toyama's theorem) and the modularity of strong normalization for a particular class of term rewriting systems (non-collapsing term rewriting systems). The proofs in the categorical setting provide a mild generalisation of these results.

Acknowledgements

First of all, I would like to thank my supervisors, Don Sannella and Stefan Kahrs. Together and between themselves, their support, patience and countless suggestions, corrections and ideas kept my research and this thesis on an even keel. For a while, they were ably supported by Andrzej Tarlecki during his stay at Edinburgh.

A very special thanks is due to Neil Ghani, not only for keeping me company, but also for his enthusiasm in this work, which sometimes surpassed my own, which certainly was a great motivation.

Postgraduate study at the LFCS in Edinburgh has been very stimulating, due to the Theory of Computation course, the various "clubs" and talks, and the people present, and I thank everyone involved; I would like to single out Barry Jay for introducing me to category theory in general, and to categorical term rewriting in particular and John Power for his talks about monads and enriched category theory.

Thank you to everybody who read this thesis and suggested improvements, first of all Stefan and Don, but further Neil, Paul-André Melliès, and Burkhart Wolff.

I benefited from a DAAD grant, and Paul Taylor's macro package.

On a more personal note, I would like to thank everybody who made my stay in Edinburgh more pleasant: Simon McPartlin (also for introducing me to the delights of Easter Road), Roope Kaivola, Ally Hume, David Aspinall, all those who played football with me (real or otherwise), my landlords Helmut and Catherine Petzsch, and Martin Köhler (the London correspondent); and also everybody at Bremen for their company, in particular Burkhart, Sabine, and Stefan. My parents have always supported me, for which I am particularly grateful.

Declaration

I declare that this thesis was composed by myself and that the work contained therein is my own, except where explicitly stated otherwise in the text. The main results of chapter 4 have appeared in [50], and the main results of chapter 5 have appeared in [51].

(*Christoph Lüth*)

Table of Contents

Chapter 1

Introduction and Preliminaries

A term rewriting system is a set of directed equations like the following:

$$\begin{aligned} K\cdot x\cdot y &\to x \\ S\cdot x\cdot y\cdot z &\to x\cdot z\cdot(y\cdot z) \end{aligned}$$

This system is in fact one of the first, and still one of the most prominent term rewriting systems, called Combinatory Logic. Ever since its invention by Schönfinkel and Curry in the 1920's, followed by Church's λ-calculus shortly after, term rewriting has been a useful tool in mathematics and (later) computer science. Its use has spread to such diverse areas as

- foundational and meta-mathematical issues (which it was first invented for);
- automated theorem proving;
- implementation of programming languages.

Term rewriting is attractive because it provides an abstract model of computation while retaining a relatively simple syntax and semantics. In fact, this very simplicity also limits the usefulness of term rewriting. Realistic programs or specifications tend to be very large. In order to keep them manageable, they are built from smaller ones using structuring operations; the simplest example of a structuring operation being the disjoint union. The interaction between these structuring operations and term rewriting is not at all clear, precisely because the semantics is too simple to express how rewrites in the component systems can be put together to rewrites in the combined system. In particular, we are interested in whether properties of term rewriting systems such as confluence or termination are preserved by these operations — whether they are *modular*.

The aim of this thesis is to investigate modularity problems in a coherent framework. The means to this end will be a *compositional semantics* for term rewriting systems. By this, we mean the translation of a term rewriting system into a mathematical structure such that we can combine these mathematical structures and obtain the same semantics as if we had combined the corresponding term rewriting systems.

We will demonstrate the viability of our approach by proving some rather strong modularity results known from the literature within our framework.

1.1 Scope, Approach and Structure

Why Semantics?

By giving a semantics to a term rewriting system, we want to reason about its "meaning" independent of its particular syntactic presentation. For example, consider the way we give a semantics to equational presentations in universal algebra. A set of equations like

$$\begin{aligned} x\cdot(y\cdot z) &= (x\cdot y)\cdot z \\ I(x)\cdot x &= 1 \\ 1\cdot x &= x \end{aligned}$$

is on first sight just a collection of syntax, but we can interpret it in the language of sets and functions (by picking a particular set X to interpret all terms, an element $a \in X$ to interpret the constant 1, a unary function $b : X \to X$ to interpret I and a binary function $c : X \times X \to X$ to interpret $\cdot$). When doing so, the above equations entail the equation $x\cdot 1 = x$ (i.e. whatever set and functions we chose, this equation will hold); we say that the equation $x\cdot 1 = x$ is in the *equational theory* of the three equations above. When reasoning about structures described by the three equations (in other words, groups) we want to refer to whole class of equations derivable from those given. This is meant by independence from syntactic presentations. In the same vein, when reasoning about a term rewriting system, we want to refer to all rewrites derivable in it — its so-called *theory*.

Semantics for Term Rewriting Systems

Traditionally, the theory of a term rewriting system is defined rather syntactically, by using concepts such as substitution and contexts (the latter can be thought of as terms with a "hole" given by a special unary symbol $\Box$): given a rule $l \to r$, a substitution σ of the variables in l and r, and a context $C[]$, the left hand side of the rule with the variables substituted by σ and put into context rewrites to the right hand side of the rule with the variables substituted by σ and put into context: $C[\sigma(l)] \to C[\sigma(r)]$. We say $C[\sigma(l)]$ *reduces* to $C[\sigma(r)]$ in one step.

Hence the domain of discourse of the semantics are relations– we talk about sets of objects, here terms, with a binary relation, the reduction relation generated by the rewrite rules, on them. These binary relations are called *abstract reduction systems* (ARS's). While some results can be proven directly at the level of abstract reduction systems, more sophisticated ones and especially those concerning modularity, cannot, since abstract reduction systems lack the expressivity to deal with key concepts such as substitutions, layers, redexes, orthogonality etc. so results relying on these concepts have to be proven directly at the level of the syntax. This approach does not offer any compositionality: we cannot combine rewrites from different systems, just the syntax. This way of defining the semantics of combined systems is referred to as flattening: for a structured term rewriting system, we take the component systems and calculate one large term rewriting system equivalent to the structured system (the flattened system). Hence, when considering whether a certain property is preserved under the disjoint union of two term rewriting systems, we consider rewrites built over the disjoint union of the two signatures and rewrite rules, rather than the combination of rewrites from both systems, and a rather complicated syntactic machinery is set up to extract the rewrites in the component systems (one needs to introduce the notions of layers, special subterms, different colours etc.), for which the assumptions can be applied.

The example of universal algebra above has actually not been chosen accidentally. As universal algebra deals with terms and equations on them, a careful analysis and generalization of the methods of universal algebra will allow us to develop a similar treatment for term rewriting systems; after all, rewrite rules are just directed equations.

Universal Algebra

In universal algebra, there are two ways of giving a semantics to a set of equations (an *equational presentation*):

- One can define an equational calculus on the terms. Essentially, close the equations under substitution, application of operations, transitivity, reflexivity and symmetry; so e.g. if $s = t$ has been derived as valid, $t = s$ is valid as well. Then the semantics of the equational presentation (called the equational theory) is the set of all derivable equations.

- One can interpret the operations in the language of sets and functions as above. In particular, one defines an *algebra* as a set together with a function for each operation symbol. This way we can evaluate a term in the algebra once we fix the values of the variables in the term; and we say an algebra admits an equation $s = t$ if the evaluation of both s and t is the same for all possible values of the variables in s and t. Then the semantics of the equational presentation is the set of those equations admitted by all algebras that admit the equations in the presentation.

In universal algebra, both these semantics coincide. For term rewriting, only the first kind of semantics has been developed at all.

Generalizing Universal Algebra

Where above equations are interpreted in the structure of sets and operations on them, we will interpret term rewriting systems in the language of "sets with structure". Our first task will be to define what exactly we mean by "sets with structure", and it turns out that there is more than one possibility, all of which make perfect sense.

The semantic framework we choose is category theory, because as opposed to set theory (or formal logic, type theory etc.), category theory concentrates on the abstract properties of structures as exhibited by the maps between them rather than their concrete representation. In particular, there is a construction modelling the "closure", for example of a set of operations under composition — and this is precisely the term algebra, the process of building terms from a given set of operations and deriving a theory from a given set of rewrite rules. Moreover, for this construct there is an abstract notion of an "algebra", and hence this construct subsumes both approaches mentioned above.

This construct is called a *monad*, and a particular class of instances of it (monads on the base category of all sets) is well-known to model universal algebra. Our semantics will generalize this treatment of universal algebra along the lines laid down by a particular branch of category theory called *enriched category theory*. The key for the development of an appropriate semantics will be the proper generalization of the underlying concepts from sets to "sets with structure". This will form the core of chapter 2.

Structuring Operations

The above is not *eo ipso* an example of a *compositional* semantics, because it is not clear how to semantically represent the structuring operations. In fact, structuring operations have mostly been considered in the context of algebraic specifications, where the techniques of universal algebra are used to specify programs. Whereas in term rewriting and universal algebra, attention was mostly restricted to the (disjoint and non-disjoint) union, the theory of algebraic specifications has arrived at the characterisation of structuring operations by another concept from category theory, called a *colimit* (which contains unions as a special case), and this allows a very precise characterisation of compositionality: a mapping from syntactic presentations to semantic presentations preserving these colimits. Structuring operations, and how they operate on the semantic presentations (i.e. the monads) will be the scope of chapter 3.

Properties of Term Rewriting Systems

Not every term rewriting system is useful. A system like

$$F(x) \to F(x)$$

that reduces a term just to itself is clearly useless and uninteresting. The system of Combinatory Logic given above is interesting, because it can represent every computable function.

A taxonomy of term rewriting systems is based on the properties they enjoy. Two of the most important properties are

- *confluence*, and
- *strong normalization*.

Confluence means that whenever there are two reductions from the same term to two different terms, we can find another two reductions from these terms both yielding the same term. This means that whichever of the two branching reductions we choose, we arrive at the same result in the end as long as we keep going. Strong normalization means that there is no infinite sequence of one-step reductions— every reduction eventually terminates.

A system which is confluent as well as terminating is *complete*. A complete term rewriting system gives a decision procedure for the equational theory the term rewriting system describes: given two terms s, t, we reduce them until we cannot reduce any further; the resulting terms s', t' are called the *normal form* of s and t. By confluence, a term only has one normal form, and two terms are equivalent in the equational theory iff their normal forms are equal.

In the context of structured systems, the question which structuring operation preserves which property, summed up under the headline of *modularity*, has been attracting a lot of attention — not at least from this thesis, where we will discuss the modularity of confluence in chapter 4 and the modularity of strong normalization in chapter 5. In these two chapters, we will prove known results from term rewriting within our new semantic framework, demonstrating the viability of our approach by simplifying the existing proofs and making them amenable to generalization, as indicated by the slight generalizations of the original results we obtain by re-proving them in our framework.

Category Theory in This Thesis

One of the aims of this thesis is to demonstrate that enriched category theory is a useful basis for the theory of computation, and that the results from enriched category theory can be put to good use in computer science.[1] Hence, this thesis has been written such that the results (if not every detail of the proofs) should be understandable to a reader with a passing acquaintance with category theory (like the first six chapters of [52]), and no knowledge of enriched category theory at all, if the reader is willing to suspend belief at some points and accept some of the categorical results as they are presented, or on the basis of some informal reasoning. I would be delighted if this thesis would entice readers to learn more about category theory, or indeed enriched category theory.

The reader with a good categorical background, however, may find that at some points the simplification of categorical matters borders on the imprecise and erroneous; so where I felt an explanation of the finer points of the category theory was needed, this has been put into separate, dedicated sections. These few sections are marked with a symbol $\underset{\text{Categories}}{\boxed{!}}$ in order to warn the unaware; their content is not essential to an understanding of what follows (under the caveats mentioned above).

1.2 How to read this Thesis

Thesis Outline

In summary, this thesis is structured as follows:

- Chapter 1: The rest of this chapter introduces the notions and concepts of category theory as used in this thesis. Besides the standard concepts

[1] There is a deep reason for this: "ordinary" category theory deals essentially with the equational theory of structures (and maps between them); but computer science is more about computation than equality, and the additional structure offered by the enrichment exactly fits this need.

found in any textbook on the matter, we will draw upon a less commonly known area called enriched category theory. While only briefly recalling the basic notions of category theory, more in order to define the notation than as a self-contained introduction, the subsequent introduction to enriched category theory and 2-categories is slightly more detailed, although the reader is unlikely to obtain an in-depth understanding without consulting the literature. We will further present an extensive review of the categories we will use to model reductions — namely, graphs, relations, preorders and categories. The chapter closes with a few lemmas about the algebra of words over an alphabet $\mathcal{L}$, which will be needed in chapter 3.

- Chapter 2: The theory construction.

  We review how monads on the category of sets model signatures and equational presentations, followed by a review of the theory of enriched monads. We then show how a specialization of the general theory of enriched monads to a particular category of sets-with-structure yields a semantics for term rewriting systems. We close the chapter by showing compositionality of the semantics.

- Chapter 3: Structuring Operations.

  Structuring operations are colimits, constructed from coproducts and coequalizers. We consider the coproduct of two monads, and its construction, and briefly touch on the coequalizer of two monads.

- Chapter 4: Modularity of Confluence.

  We show the modularity of confluence for the coproduct of two monads (Toyama's theorem).

- Chapter 5: Modularity of Strong Normalization.

  We define a subclass of strongly normalizing monads called strongly normalizing under non-deterministic collapses, and show the modularity of strong normalization for this subclass.

- Chapter 6: Conclusion and Further Work.

  We review the literature on categorical term rewriting, relate our work to the literature, and indicate possible directions of further research.

Chapter Dependencies

$$\begin{tikzcd}
 & & & 4 \\
1.3 \text{ to } 1.7 \arrow[r, dotted] & 2 \arrow[r] & 3 \arrow[ur] \arrow[dr] & \\
 & & & 5
\end{tikzcd}$$

where sections 1.3 to 1.7 are intended as a stock of references to be consulted when the results are used, rather than as a straight read.

1.3 Basic Notions of Category Theory

This section recalls some of the basic notions of category theory. It is neither self-contained nor intended to be, and the reader is advised that a knowledge of the basic concepts of category theory (as mentioned in this section) will be necessary to understand what follows (and is unlikely to be gained from this section alone). The standard introductory text is of course [52], but there are a plethora of introductory texts of varying quality.

Readers with a working knowledge of category theory, and a passing knowledge of enriched category theory will be able to skip this section completely, and refer back to it later on whenever they encounter references to those results in this section which are unknown to them. Readers with a working knowledge of category theory, but no knowledge of enriched category theory are invited to consult §1.4; but as mentioned above, this thesis should be readable even with no knowledge of enriched category theory.

We will use the notation of [52] (so readers familiar with the material of the first six chapters of this book will be able to skip this section), except for the following:

- the hom-sets of a category $\mathcal{C}$ are written $\mathcal{C}(x, y)$ rather than $hom(x, y)$;
- the category of functors from $\mathcal{C}$ to $\mathcal{D}$ is written $[\mathcal{C}, \mathcal{D}]$ rather than $\mathcal{D}^{\mathcal{C}}$.

1.3.1 Foundations

Following the approach of [52, Chapter I.6, pg. 21ff.], we shall in the following assume the existence of a *universe* U within which all sets defined in the following live. The universe is a set with certain properties (e.g. it contains the set ω of all finite ordinals and is closed under powersets), the elements of which are called *small sets*. We are not at all concerned with foundational issues here, and shall just note that if a set or category gets "too large" for a particular universe, there is always a larger one which it lives in, and we shall mention these "size conditions" in passing whenever we encounter them.

1.3.2 Categories, Functors and Natural Transformations

A *category* $\mathcal{C}$ is given by

- a set of *objects* $|\mathcal{C}|$,
- and a set of *morphisms* $\mathbf{Mor}_\mathcal{C}$, with two functions $\delta_s, \delta_t : \mathbf{Mor}_\mathcal{C} \to |\mathcal{C}|$ assigning a source and target to every morphism; or alternatively, a family $\{\mathcal{C}(x, y)\}_{x,y \in |\mathcal{C}|}$ of *hom-sets* (we will use both notions interchangeably);
- for every object $x \in |\mathcal{C}|$, an *identity* $\mathbf{1}_x : x \to x$ on x;
- and a composition function, mapping two morphisms $f : x \to y$, $g : y \to z$, to their *composition* $g \cdot f : x \to z$.[2]

such that the composition is associative and the identity its left and right unit. A category $\mathcal{C}$ is called *discrete* if all its morphisms are identities. The *dual* $\mathcal{C}^{\mathrm{op}}$ of the category $\mathcal{C}$ has the same objects, but reversed arrows: $\mathcal{C}^{\mathrm{op}}(y, x) \stackrel{def}{=} \mathcal{C}(x, y)$.

We will use calligraphic letters such as $\mathcal{C}, \mathcal{X}$ as variables for categories and bold letters (as in **Set**) for specific categories. We will further write $x \in \mathcal{C}$ as an abbreviation for $x \in |\mathcal{C}|$, and $f : x \to y$ in $\mathcal{C}$ as an abbreviation for $x, y \in |C|, f \in \mathcal{C}(x, y)$.

A morphism $m : X \to Y$ in $\mathcal{C}$ is *monic* (or a *monomorphism*) iff for all $Z \in \mathcal{C}$ and $f_1, f_2 : Y \to Z$ $m \cdot f_1 = m \cdot f_2$ implies $f_1 = f_2$. A morphism $e : X \to Y$ in $\mathcal{C}$ is *epi* (or an *epimorphism*) iff for all $U \in \mathcal{C}$ and $g_1, g_2 : U \to X$, $g_1 \cdot e = g_2 \cdot e$ implies $g_1 = g_2$. In **Set**, the category of small sets and functions between them (see below), a function is epi iff it is surjective, and monic iff it is injective. Finally, a morphism $f : X \to Y$ is an *isomorphism* iff it is invertible, i.e. if there is a morphism $f^{-1} : Y \to X$ such that $f^{-1} \cdot f = \mathbf{1}_X$ and $f \cdot f^{-1} = \mathbf{1}_Y$. In this case, we also say that the objects X and Y are isomorphic.

A *functor* $F : \mathcal{X} \to \mathcal{Y}$ is given by an object function $F_{Obj} : |\mathcal{X}| \to |\mathcal{Y}|$ and a morphism function $F : \mathbf{Mor}_\mathcal{X} \to \mathbf{Mor}_\mathcal{Y}$, which preserves the source and target of the morphisms, and identities and composition (i.e. $\mathbf{1}_{F_{Obj}(X)} = F\mathbf{1}_X$ and $Fg \cdot Ff = F(g \cdot f)$). By slight abuse of notation, we usually write F for the object function as well. A functor $F : \mathcal{X} \to \mathcal{Y}$ can also be given as a family of

[2] Note that although the author is a computer scientist, composition is written in applicative (not diagrammatic) order.

functions $F_{x,y} : \mathcal{X}(x, y) \to \mathcal{Y}(Fx, Fy)$ for all $x, y \in \mathcal{X}$. Then F is *full* if all $F_{x,y}$ are surjective, and *faithful* if all $F_{x,y}$ are injective.

A *natural transformation* $\alpha : F \Rightarrow G : \mathcal{X} \to \mathcal{Y}$ between two functors F and G is given by a family $\{\alpha_x : FX \to GX\}_{x\in\mathcal{X}}$ of arrows in $\mathcal{Y}$, indexed by the objects in $\mathcal{X}$, such that for all $f : X \to Y$ (this is called the *naturality* of α in X):

$$\alpha_Y \cdot Ff = Gf \cdot \alpha_X \tag{1.1}$$

Given two categories $\mathcal{X}, \mathcal{Y}$ the *functor category* $[\mathcal{X}, \mathcal{Y}]$ between them has functors as objects and natural transformations as morphisms, with the obvious composition and the *identity transformation* id_F, given by the identity $\mathbf{1}_{FX}$ for every object $X \in \mathcal{X}$.[3] The *constant functor* $\Delta Y : \mathcal{X} \to \mathcal{Y}$ for $Y \in \mathcal{Y}$ maps every object to Y, and every morphism to the identity on Y. For a morphism $f : Y \to Z$, the natural transformation $\Delta f : \Delta Y \Rightarrow \Delta Z$ is f for every object $X \in \mathcal{X}$. This gives the *diagonal functor* $\Delta(-) : \mathcal{Y} \to [\mathcal{X}, \mathcal{Y}]$.

1.3.3 Representability, Limits and Colimits

A particularly important category is the category **Set** which has all small sets as objects, and the functions between them as morphisms. Then for a small category $\mathcal{C}$ there are the two *hom-functors* for every object $X \in \mathcal{C}$: $\mathcal{C}(X, -) : \mathcal{C} \to \mathbf{Set}$, which on the objects maps Y to $\mathcal{C}(X, Y)$, and maps a morphism $f : Y \to Z$ to the *postcomposition* $f\cdot(-) : \mathcal{C}(X, Y) \to \mathcal{C}(X, Z)$, mapping $g : X \to Y$ to $g\cdot f : X \to Z$; and $\mathcal{C}(-, X) : \mathcal{C}^{\mathrm{op}} \to \mathbf{Set}$, which maps Y to $\mathcal{C}(Y, X)$ and f to $(-)\cdot f$, the *precomposition* with f.

A functor $F : \mathcal{C} \to \mathbf{Set}$ is *representable* if there is an object $X \in \mathcal{C}$ such that for all $Y \in \mathcal{C}$, there is an isomorphism $\mathcal{C}(X, Y) \cong FY$ which is natural in Y, also written as $\mathcal{C}(X, -) \cong F$.

Given a functor $F : \mathcal{J} \to \mathcal{C}$, a *cone to the base F from vertex X* is given by an object $X \in \mathcal{C}$, and a natural transformation $\nu : \Delta X \Rightarrow F$; and a *cone over F to the vertex X* is given by an object $X \in \mathcal{C}$, and a natural transformation $\nu : F \Rightarrow \Delta X$. F has a *limit* if the functor $[\mathcal{J}, \mathcal{C}](\Delta -, F) : \mathcal{C}^{\mathrm{op}} \to \mathbf{Set}$ is representable; the representing object X is called the limit of F, and $c : \Delta X \Rightarrow F$ is the *limiting cone*. Dually, if the functor $[\mathcal{J}, \mathcal{C}](F, \Delta -) : \mathcal{C} \to \mathbf{Set}$ is representable, F has a *colimit* given by a colimiting object X and a colimiting cone $c : F \Rightarrow \Delta X$

[3] The functor category is the first instance of the "size conditions" mentioned above: the functor category $[\mathcal{X}, \mathcal{Y}]$ may be larger than $\mathcal{X}$ and $\mathcal{Y}$; see [52, pg. 41]. To be more precise: a category $\mathcal{C}$ is *small* if both the set of its objects and of its morphisms are small sets, then $[\mathcal{X}, \mathcal{Y}]$ is only small if $\mathcal{X}$ is small.

(sometimes written as a tuple (c, X)). The limiting object for a functor F is sometimes denoted as $\mathit{lim}\, F$, and the colimiting object as $\mathit{colim}\, F$.

We say a functor $H : \mathcal{C} \to \mathcal{D}$ *preserves* limits if the image $Hc : H\mathit{lim} F \Rightarrow HF$ for the limit $(c, \mathit{lim}\, F)$ of a functor $F : \mathcal{J} \to \mathcal{C}$ is a limiting cone for HF.

Given a functor $F : \mathcal{X} \to \mathcal{Y}$, if the functor $\mathcal{Y}(F(-), Y) : \mathcal{X}^{\mathrm{op}} \to \mathbf{Set}$ is representable for every object $Y \in \mathcal{Y}$, F has a *right adjoint*, which is a functor $G : \mathcal{Y} \to \mathcal{X}$, mapping Y to the representing object GY; and we have an isomorphism $\phi : \mathcal{Y}(F-, Y) \cong \mathcal{X}^{\mathrm{op}}(GY, -) \cong \mathcal{X}(-, GY)$ which maps a function $f : Y \to Z$ to the function $\phi(f) : \mathcal{X}(GZ, GY)$. We say F and G form an *adjunction* $F \dashv G : \mathcal{X} \to \mathcal{Y}$ (or *are adjoint*). The existence of an adjunction can be characterised in at least three other ways: given two functors $F : \mathcal{X} \to \mathcal{Y}$, $G : \mathcal{Y} \to \mathcal{X}$, they are adjoint if

- there is a bijection $\mathcal{Y}(FX, Y) \cong \mathcal{X}(X, GY)$ which is natural in $X \in \mathcal{X}$ and $Y \in \mathcal{Y}$;
- there are two natural transformations $\eta : \mathbf{1}_{\mathcal{X}} \to GF$ (the *unit* of the adjunction) and $\varepsilon : FG \Rightarrow \mathbf{1}_{\mathcal{Y}}$ (the *counit* of the adjunction), satisfying the *triangle laws*: $G\varepsilon_Y \cdot \eta_{GY} = \mathbf{1}_{GY}$ and $\varepsilon_{FX} \cdot F\eta_X = \mathbf{1}_{FX}$ (for $X \in \mathcal{X}, Y \in \mathcal{Y}$).
- there is, for every object $X \in \mathcal{X}$, a morphism $\eta_X : X \to GFX$ which is *universal* from X to G: for every other object $Y \in \mathcal{Y}$, and every other morphism $f : X \to GY$, there is a unique morphism $!_f : FX \to Y$ in $\mathcal{X}$ such that $f = G(!_f) \cdot \eta_X$.

All of these formulations are equivalent [52, pg. 81]. An important fact about adjoints is that *left adjoints preserve colimits* and *right adjoint preserve limits* [52, pg. 114].

Limits can also be considered as adjoints: a category $\mathcal{C}$ has a limit for all functors $F : \mathcal{J} \to \mathcal{C}$ (it is *complete*) iff the diagonal functor $\Delta : \mathcal{C} \to [\mathcal{J}, \mathcal{C}]$ has a right adjoint (and dually, it has colimits (is *cocomplete*) iff the diagonal functor has a left adjoint).

1.3.4 Particular Limits and Categories

The empty category $\emptyset$ has no objects and no morphisms. For any category $\mathcal{C}$, there is exactly one functor $! : \emptyset \to \mathcal{C}$. The limit (colimit) of this functor (if it exists) is called the terminal (initial) object in $\mathcal{C}$.

The one-object category $\mathbf{1}$ is has one object $|\mathbf{1}| = \{*\}$ and a single morphism, the identity on it. A functor $F : \mathbf{1} \to \mathcal{C}$ specifies exactly one object of $\mathcal{C}$.

More generally, for any natural number $n \in \mathbb{N}$ there is the discrete category $\mathbf{n}$ which has the ordinal n as objects; then a functor $F : \mathbf{n} \to \mathcal{C}$ specifies n objects in $\mathcal{C}$, and its limit (colimit) is the n-ary *(cartesian) product* (*coproduct*). If these exist for any n, we say $\mathcal{C}$ has all finite products (coproducts). This can be extended to any set v, and if v is infinite, we say $\mathcal{C}$ has infinite products (coproducts). Coproducts are often written in the form $X_1 + X_2 + \cdots + X_n$ or even $\coprod_{i=1}^{n} X_i$ (with the similar notation $X_1 \times X_2 \times \ldots X_n = \prod_{i=1}^{n} X_i$) for products).

The one-arrow category $\longrightarrow$ has two objects with two identity morphisms and one non-identity morphism between them; the two-arrow category $\rightrightarrows$ has two objects with two identities and two non-identity morphisms between them. Then a functor $F :\rightrightarrows \to \mathcal{C}$ specifies two morphisms $f, g : X \to Y$ in $\mathcal{C}$ with the same source and target (called a *fork*) and the limit (colimit) of F is called the *equalizer* (*coequalizer*) of f and g.

(Co-)Products and (co-)equalizers are important because if they exist all other (co-) limits exists, as will be shown in the following section.

1.3.5 Colimits from Coproducts and Coequalizers

One of the most useful theorems about limits says that any limit can actually be constructed from products and equalizers. Since we will need the dual of this construction later on, it is worth spelling out in some detail in the following proposition, which is the dual of Theorem 1 in [52, pg. 109].

Proposition 1.3.1 *Given two categories $\mathcal{J}$ and $\mathcal{C}$, if $\mathcal{C}$ has all coproducts indexed by the set of objects and morphisms of $\mathcal{J}$, and coequalizers for all pairs of arrows, then $\mathcal{C}$ has a colimit for every functor $F : \mathcal{J} \to \mathcal{C}$.*

Proof. Form the coproduct $\coprod_{u:j \to k} Fj$, indexed by all arrows $u : j \to k$ in $\mathcal{J}$.

$$\begin{tikzcd}
& Fj \arrow[dl, "\iota_u"'] \arrow[dr, "\iota_j"] & & \\
\coprod\limits_{u:j\to k} Fj \arrow[rr, shift left, "f"] \arrow[rr, shift right, "g"'] & & \coprod\limits_{i} Fi \arrow[r, "q"] & Q \\
Fj \arrow[u, "\iota_u"] \arrow[rr, "Fu"] & & Fk \arrow[u, "\iota_k"'] &
\end{tikzcd} \tag{1.2}$$

Now for every arrow $u : j \to k$, we define two maps from Fj into the coproduct

$\coprod_{i\in\mathcal{J}} Fi$: f_u is the injection ι_j of Fj into the coproduct, and g_u is Fu followed by the injection ι_k. This gives two unique arrows f, g, such that in diagram 1.2, f makes the upper triangle and g the lower square commute for all arrows $u : j \to k$. Then the coequalizer Q is the colimit of F. The colimiting cone $\mu : F \Rightarrow \Delta Q$ is given by $\mu_j \stackrel{def}{=} \iota_j \cdot q$; which is a cone since q coequalizes f and g, and it is the universal cone, since any other cone $\nu : F \to \Delta P$ necessarily coequalizes f and g, and hence there is a unique morphism $! : Q \to P$. □

1.3.6 Monads

A *monad* $\mathsf{T} = \langle T, \eta, \mu\rangle$ on a category $\mathcal{C}$ is given by a functor $T : \mathcal{C} \to \mathcal{C}$ (the *action* of the monad), and two natural transformations $\eta : \mathbf{1}_\mathcal{C} \Rightarrow T$ and $\mu : TT \Rightarrow T$ (the *unit* and the *multiplication*), such that the *monad laws* hold: $\mu\cdot\eta_T = \mathbf{1}_\mathcal{C}$, $\mu\cdot T\eta = \mathbf{1}_\mathcal{C}$ and $\mu\cdot T\mu = \mu\cdot\mu_T$ (these state that η and μ form a monoid, with η a unit to μ on the left and right, and μ associative).

An *algebra* for a monad T is given by an object $X \in \mathcal{C}$ and a morphism $\alpha : TX \to X$ (called the *structure map* of the algebra), such that $\alpha\cdot\mu_X = \alpha\cdot T\alpha$, and $\alpha\cdot\eta_X = \mathbf{1}_X$. A map between two algebras $\alpha : TX \to X$ and $\beta : TY \to Y$ is a morphism $f : X \to Y$ s.t. $f\cdot\alpha = \beta\cdot Tf$.

Every adjunction $F \dashv G : \mathcal{X} \to \mathcal{Y}$ with unit η and counit υ gives rise to a monad $\mathsf{T} \stackrel{def}{=} \langle GF, \eta, G\upsilon_F\rangle$; on the other hand, every monad T gives rise to a variety of adjunctions, in particular between $\mathcal{C}$ and the Kleisli-category $\mathcal{C}_\mathsf{T}$ (see [52, Section VI.5]), and $\mathcal{C}$ and the Eilenberg-Moore-category $\mathcal{C}^\mathsf{T}$, given by the algebras and morphisms between them. The left adjoint is $F^T : \mathcal{C} \to \mathcal{C}^T$, mapping X to the *free algebra* $\mu_X : TTX \to TX$, whereas the right adjoint $U^T : \mathcal{C}^T \to \mathcal{C}$ maps $\alpha : TX \to X$ to the object X.

If a functor $G : \mathcal{D} \to \mathcal{C}$ has a left adjoint $F : \mathcal{C} \to \mathcal{D}$, and $\mathcal{D}$ is isomorphic to the Eilenberg-Moore category of the resulting monad, we say that G is *monadic*, or even that $\mathcal{D}$ is monadic over $\mathcal{C}$ if G is understood. This means that $\mathcal{D}$ can be generated from $\mathcal{C}$ by a set of operations plus equations on them; examples are the category of groups, **Grp** which is monadic over **Set**, or the category of small categories **Cat** which is monadic over **Grph**. A non-example is **Cat**, which is *not* monadic over **Set**.

The Category of Monads on $\mathcal{C}$

Given two monads $\mathsf{T}_1 = \langle T_1, \eta_1, \mu_1\rangle, \mathsf{T}_2 = \langle T_2, \eta_2, \mu_2\rangle$, a *monad morphism* $\alpha : \mathsf{T}_1 \to \mathsf{T}_2$ is given by a natural transformation $\alpha : T_1 \Rightarrow T_2$ such that α commutes with the unit and multiplication of the two monads as in 1.3 and 1.4, where $(*)$ commutes by naturality of α.

$$
\begin{tikzcd}
 & T_1X \arrow[dd, "\alpha_X"] \\
X \arrow[ur, "\eta_{1,X}"] \arrow[dr, "\eta_{2,X}"'] & \\
 & T_2X
\end{tikzcd}
\qquad
\begin{tikzcd}
 & T_1^2X \arrow[rr, "\mu_{1,X}"] \arrow[dl, "T_1\alpha_X"'] \arrow[dr, "\alpha_{1,T_1X}"] & & T_1X \arrow[dd, "\alpha_X"] \\
T_1T_2X \arrow[dr, "\alpha_{T_2X}"] & (*) & T_2T_1X \arrow[dl, "T_2\alpha_X"] & \\
 & T_2^2X \arrow[rr, "\mu_{2,X}"] & & T_2X
\end{tikzcd}
$$

(1.3) (1.4)

The category $\mathbf{Mon}(\mathcal{C})$ of monads on $\mathcal{C}$ has monads on $\mathcal{C}$ as objects and monad morphisms between them as morphisms.[4]

A particular monad is the identity monad on $\mathcal{C}$, $\mathsf{Id}_{\mathcal{C}} \stackrel{def}{=} \langle \mathbf{1}_{\mathcal{C}}, id_{\mathbf{1}_{\mathcal{C}}}, id_{\mathbf{1}_{\mathcal{C}}} \rangle$ which has the identity on $\mathcal{C}$ as its action, and the identity transformations as both unit and multiplication. By equation 1.3, it is the initial object in $\mathbf{Mon}(\mathcal{C})$.

1.3.7 Kan Extensions

Kan extensions can be thought of as a canonical way of extending the domain of a functor. Given a functor $K : \mathcal{M} \to \mathcal{C}$ and a category $\mathcal{A}$, the precomposition with K defines a functor $(-)\cdot K : [\mathcal{C}, \mathcal{A}] \to [\mathcal{M}, \mathcal{A}]$, which takes every functor $F : \mathcal{C} \to \mathcal{A}$ to $FK : \mathcal{M} \to \mathcal{A}$. The adjoints of $(-)\cdot K$, if they exist, are the Kan extensions. Thus Kan extensions come in two variants, left and right. More explicitly, given functors $K : \mathcal{M} \to \mathcal{C}$ and $T : \mathcal{M} \to \mathcal{A}$, the *left Kan extension* of T along K is given by a functor $\mathrm{Lan}_K T : \mathcal{C} \to \mathcal{A}$ and a natural transformation $\varepsilon : T \Rightarrow \mathrm{Lan}_K T\cdot K$ which is universal, i.e. for any other functor $S : \mathcal{C} \to \mathcal{A}$ and natural transformation $\alpha : T \Rightarrow SK$, there is a unique natural transformation $\sigma : \mathrm{Lan}_K T \Rightarrow S$ such that $\alpha = \sigma K\cdot\varepsilon$. The right Kan extension is given similarly by the right adjoint (see [52, Chapter X] for details).

If $\mathcal{C}$ is cocomplete, the left Kan extension can be constructed as a pointwise colimit [52, Chapter X.5]. For $K : \mathcal{M} \to \mathcal{C}$ and $T : \mathcal{M} \to \mathcal{A}$, the value of the left

[4] Of course, this is not a category unless some size restrictions are placed on $\mathcal{C}$, i.e. $\mathcal{C}$ being small. If $\mathcal{C}$ is not small, $\mathbf{Mon}(\mathcal{C})$ will be a large category. We have to be careful not to treat $\mathbf{Mon}(\mathcal{C})$ in turn as a category of the same size as $\mathcal{C}$ (in particular, $\mathbf{Mon}(\mathbf{Cat})$ is not an object of $\mathbf{Cat}$), but apart from that we can treat $\mathbf{Mon}(\mathcal{C})$ as a normal category.

Kan extension at $c \in \mathcal{C}$ is given by

$$\mathrm{Lan}_K T(c) = colim\,((K \downarrow c) \xrightarrow{P} \mathcal{M} \xrightarrow{T} \mathcal{A}) \tag{1.5}$$

where $(K \downarrow c)$ is the so-called *comma category* [52, Chapter II.6] which has as objects morphisms $f : Kx \to c$ in $\mathcal{C}$, and as morphisms commuting triangles

$$\begin{tikzcd}
Kx \arrow[r, "f"] \arrow[d, "Ku"'] & c \\
Ky \arrow[ur, "g"'] &
\end{tikzcd}$$

and P is the projection functor, taking $f : Kx \to c$ to x and Ku as above to u.

A useful fact about Kan extensions is that extension to the left preserves colimits, and to the right limits respectively. Limits and colimits are given by the left and right adjoint of the diagonal functor Δ (see page 15 above). The composition of the diagonal functor $\Delta : \mathcal{A} \to [\mathcal{C}, \mathcal{A}]$ with the postcomposition $(-)\cdot K : [\mathcal{C}, \mathcal{A}] \to [\mathcal{M}, \mathcal{A}]$ yields a functor which for any $a \in \mathcal{A}$ gives the constant functor $\Delta a : \mathcal{M} \to \mathcal{A}$, hence is equal to $\Delta : \mathcal{A} \to [\mathcal{M}, \mathcal{A}]$.

Then by diagram 1.6, $colim \cdot \mathrm{Lan}_K(-) : [\mathcal{M}, \mathcal{A}] \to \mathcal{A}$ is a left adjoint to

$$\begin{tikzcd}[column sep=huge]
\mathcal{A} \arrow[r, "\Delta"] & {[\mathcal{C}, \mathcal{A}]} \arrow[l, bend right, "colim"'] \arrow[l, bend left, "lim"] \arrow[r, "(-)\cdot K"] & {[\mathcal{M}, \mathcal{A}]} \arrow[l, bend right, "\mathrm{Lan}_K(-)"'] \arrow[l, bend left, "\mathrm{Ran}_K(-)"]
\end{tikzcd} \tag{1.6}$$

$\Delta : \mathcal{A} \to [\mathcal{M}, \mathcal{A}]$ and thus maps any functor to its colimit: for any $F : \mathcal{M} \to \mathcal{A}$,

$$colim\,\mathrm{Lan}_K F \cong colim\,F \tag{1.7}$$

1.3.8 Locally Finitely Presentable Categories

In the category **Set**, there are the finite sets and the finite ordinals; the generalisation of this concept leads to *finitely presentable objects* (see [1, Chapter 1] or [4] for a motivation of the following definitions).

A partially ordered set $(I, \leq)$ is called *directed* if for any two objects $j, k \in I$ there is an object $l \in I$ such that $j \leq l$, $k \leq l$. A *directed colimit* is the colimit of a functor $D : (I, \leq) \to \mathcal{C}$ where $(I, \leq)$ is a directed poset considered as a category. An object $k \in \mathcal{C}$ is *finitely presentable* (or fp) if its hom-functor $\mathcal{C}(k, -) : \mathcal{C} \to$ **Set** preserves directed colimits. A category $\mathcal{C}$ is *locally finitely* *presentable* (also written as lfp) if its cocomplete, and there is a set A of finitely presentable objects such that every object is a directed colimit of objects from A. The subcategory of $\mathcal{C}$ given by the fp objects is written as $\mathcal{C}_{\mathrm{fp}}$. Intuitively, fp objects are "finite objects", and a category is lfp if it can be generated from its finite objects.

Examples of finitely presentable objects are finite sets in **Set**, finite graphs in **Grph**, finite preorders in **Pre**, and categories with a finite set of objects and morphisms which are generated from a finite set of morphisms by closing under composition and quotienting with a finite number of equations on them in **Cat**. All of these categories (**Set**, **Grph**, **Pre**, **Cat**) are locally finitely presentable. A category which is not lfp is **CPO**, the category of complete partial orders and continuous functions. For more examples see [1, Chapter 1].

An equivalent characterization of lfp is via generators: a *strong generator* $\mathcal{G}$ of a category $\mathcal{C}$ is a small full subcategory $J : \mathcal{G} \hookrightarrow \mathcal{C}$ such that $f : X \to Y$ is an isomorphism iff for all $G \in \mathcal{G}$, $\mathcal{C}(G, f) : \mathcal{C}(G, X) \to \mathcal{C}(G, Y)$ is an isomorphism. Then $\mathcal{C}$ is lfp iff $\mathcal{C}$ is cocomplete and has a strong generator [1, Theorem 1.11]. An example of this is the functor category $[\mathcal{C}, \mathbf{Set}]$ for any small category $\mathcal{C}$: by the Yoneda lemma the representable functors form a small set of generators in $[\mathcal{C}, \mathbf{Set}]$, hence $[\mathcal{C}, \mathbf{Set}]$ is lfp.

An equivalent concept to directed colimits are filtered colimits, which are often more convenient to use. A category $\mathcal{J}$ is *filtered* if

(i) for any two objects two objects $i, j \in \mathcal{J}$, there is an object $k \in \mathcal{J}$ and morphisms $f : i \to k, g : j \to k$, and

(ii) for any two parallel arrows $r, s : i \to j$, there is an arrow $q : j \to k$ such that $q{\cdot}r = q{\cdot}s$.

A filtered colimit is the colimit of a functor $F : \mathcal{J} \to \mathcal{C}$ with $\mathcal{J}$ filtered. A category $\mathcal{C}$ has all filtered colimits iff it it has all directed colimits [1, Theorem 1.5] (and hence a functor preserves filtered colimits iff it preserves directed colimits).

If we liken coequalizers to quotients and coproducts to the disjoint union, filtered or directed colimits can be thought of as a completeness property; this is reflected by the fact that a category has directed colimits iff it has colimits of chains [1, Corollary 1.7].

1.3.9 Weakly Filtered Colimits

Weakly filtered categories are a non-standard concept which is introduced here, because the canonical diagrams we will encounter will not be filtered, merely weakly filtered. A weakly filtered category lacks the second property of a filtered category:

Definition 1.3.2 (Weakly Filtered Categories and Colimits) A category $\mathcal{J}$ is *weakly filtered* if for every two objects $i, j \in \mathcal{J}$, there is an object $k \in \mathcal{J}$ and morphisms $f : i \to k, g : j \to k$.

The colimit of a functor $F : \mathcal{J} \to \mathcal{C}$ is called weakly filtered if the category $\mathcal{J}$ is weakly filtered.

Not every weakly filtered category is filtered, since there can be parallel arrows $r, s : i \to j$ but there is no arrow $q : j \to k$ with $q \cdot r = q \cdot s$. Hence, weakly filtered colimits are a strictly weaker notion than filtered or directed colimits, but if a functor preserves coequalizers of parallel arrows and preserves filtered colimits, it will also preserve all weakly filtered colimits.

To show this lemma, we will first extend the weakly filtered category $\mathcal{J}$ to a filtered category. Then any weakly filtered functor can be made into a filtered one by its left Kan extension along the inclusion; and if we can show that the functor preserves this Kan extension if it preserves coequalizers, then by equation 1.7 it will preserve any weakly filtered diagram.

The filtered completion $\mathcal{J}_0$ of a weakly filtered category will have, for all objects $j \in \mathcal{J}$ an object $\overline{\jmath}$ and a morphism from j to $\overline{\jmath}$ which coequalizes all parallel arrows into j, thus satisfying the second condition for filtered categories. To have the resulting category still satisfy the first condition, we further need to add in exactly one morphism between $\overline{\jmath}$ and $\overline{k}$ iff there is at least one between j and k.

Definition 1.3.3 (The Filtered Completion) Given a weakly filtered category $\mathcal{J}$, its *filtered completion* $\mathcal{J}_0$ is defined as the category with

- objects: $\{j, \overline{\jmath} \mid j \in \mathcal{J}\}$
- morphisms:

$$\begin{array}{rl} f : i \to j & \text{for } f : i \to j \text{ in } \mathcal{J} \\ \iota_{i,j} : i \to \overline{\jmath} & \text{for } i, j \in \mathcal{J} \text{ if there is } f : i \to j \text{ in } \mathcal{J} \\ \kappa_{i,j} : \overline{\imath} \to \overline{\jmath} & \text{if there is } f : i \to j \text{ in } \mathcal{J} \end{array}$$

  where for all $f, g : i \to j$, $\overline{f} = \overline{g}$. Further, $\mathcal{J}_0(\overline{\imath}, j) \stackrel{def}{=} \emptyset$ for all $i, j \in \mathcal{J}$.

Composition in $\mathcal{J}_0$ is defined as follows (for $i, j, k \in \mathcal{J}$):

1. for $f : i \to j, g : j \to k$, $g \cdot f$ is given by composition in $\mathcal{J}$;

2. for $\kappa_{i,j} : \bar{\imath} \to \bar{\jmath}, \kappa_{j,k} : \bar{\jmath} \to \overline{k}$, $\kappa_{j,k} \cdot \kappa_{i,j} \stackrel{def}{=} \kappa_{i,k}$ (where $\kappa_{i,k}$ is given by the composition $g \cdot f$ of the morphisms $f : i \to j$ and $g : j \to k$ giving rise to $\kappa_{i,j}$ and $\kappa_{j,k}$);

3. for $f : i \to j$, $\iota_{j,k} : j \to \overline{k}$, $\iota_{j,k} \cdot f \stackrel{def}{=} \iota_{i,k}$ (where $\iota_{i,k}$ is given by $g \cdot f : i \to k$ in $\mathcal{J}$);

4. for $\iota_{i,j} : i \to \bar{\jmath}$, $\kappa_{j,k} : \bar{\jmath} \to \overline{k}$, $\kappa_{j,k} \cdot \iota_{i,j} \stackrel{def}{=} \iota_{i,k}$ (where $\iota_{i,k}$ is given by the composition $g \cdot f$ of $f : i \to j$, $g : j \to k$ giving rise to $\iota_{i,j}$ and $\kappa_{j,k}$).

The identities are $\mathbf{1}_j$ and $\kappa_{j,j}$ (given by $\mathbf{1}_j$).

Let $\overline{\mathcal{J}}$ be the full subcategory of $\mathcal{J}_0$ given by all objects $\bar{\jmath} \in \mathcal{J}_0$ for $j \in \mathcal{J}$; this subcategory is ismorphic to $J(\mathcal{J})$ (where J is the functor from page 42 below) taking a category to a preorder by identifying all arrows with the same source and target.

The functor $I : \mathcal{J} \to \mathcal{J}_0$ is the inclusion $\mathcal{J} \hookrightarrow \mathcal{J}_0$.

Lemma 1.3.4 If $\mathcal{J}$ is weakly filtered, then $\mathcal{J}_0$ is filtered.

Proof. First, we show that for all $j, k \in \mathcal{J}_0$ there is $i \in \mathcal{J}_0, p : j \to i, q : k \to i$ in $\mathcal{J}_0$. We distinguish four cases: firstly, $j, k \in \mathcal{J}$, then because $\mathcal{J}$ is weakly filtered, there is $i \in \mathcal{J}$, and $p : j \to i, q : k \to i$ in $\mathcal{J}$ and hence in $\mathcal{J}_0$; secondly, if $\bar{\jmath}, \overline{k} \in \overline{\mathcal{J}}$, then since $\mathcal{J}$ is weakly filtered, there is $i \in \mathcal{J}$, and $\kappa_{j,i} : \bar{\jmath} \to \bar{\imath}, \kappa_{k,i} : \overline{k} \to \bar{\imath}$ in $\mathcal{J}_0$; thirdly, if $j \in \mathcal{J}, \overline{k} \in \overline{\mathcal{J}}$, then there is $i \in \mathcal{J}$, and $p : j \to i, q : k \to i$ in $\mathcal{J}$, and hence $\bar{\imath} \in \mathcal{J}_0, \iota_{j,i} : j \to \bar{\imath}, \kappa_{k,i} : \overline{k} \to \bar{\imath}$; and finally $\bar{\jmath} \in \mathcal{J}, k \in \overline{\mathcal{J}}$, which is symmetric to the third case.

It remains to show that for any two arrows $r, s : i \to j$ in $\mathcal{J}_0$, there is an arrow $q : j \to k$ in $\mathcal{J}_0$ such that $q \cdot r = q \cdot s$. We have three cases: firstly, if $i, j \in \mathcal{J}$, then $q \stackrel{def}{=} \iota_{j,j}$ with $\iota_{j,j} \cdot r = \iota_{j,j} \cdot s = \iota_{j,i}$; secondly, if $\bar{\imath}, \bar{\jmath} \in \overline{\mathcal{J}}$, then $r = s = \kappa_{i,j}$, hence $q \stackrel{def}{=} \mathbf{1}_{\bar{\jmath}}$; thirdly, if $i \in \mathcal{J}, \bar{\jmath} \in \overline{\mathcal{J}}$, then $r = s = \iota_{i,j}$, hence $q \stackrel{def}{=} \mathbf{1}_{\bar{\jmath}}$. A fourth case, $\bar{\imath} \in \overline{\mathcal{J}}, j \in \mathcal{J}$, can be excluded by definition of $\mathcal{J}_0$. □

We can now turn any weakly filtered functor $F : \mathcal{J} \to \mathcal{C}$ into a filtered functor by taking its Kan extension along $I : \mathcal{J} \to \mathcal{J}_0$; by isomorphism 1.7, the colimit will remain the same. If a functor $T : \mathcal{C} \to \mathcal{D}$ preserves this Kan extension and filtered colimits, then it will preserve weakly filtered colimits. Under some mild additional assumptions, it turns out that in addition to preservation of filtered colimits, preservation of coequalizers is sufficient.

The following are the additional assumptions we need to make. They roughly say that the diagrams are finitely generated, and are formulated in quite a strong way such that they still hold for the diagrams we consider (in chapter 3), while making rather weak assumptions on the functor F in question for an easy proof. For example, the requirement below that the hom-sets be at most countably infinite could be dropped if we required the functor to preserve coequalizers of arbitrary sets of parallel morphisms, but since the diagrams in question have countably infinite hom-sets we considered the weaker assumption that the functor preserves coequalizers of pairs of parallel arrows.

Definition 1.3.5 ((Weakly) ω-Filtered Categories) A (weakly) filtered category $\mathcal{J}$ is called *(weakly) ω-filtered*, if

(i) for all $j \in \mathcal{J}$, $card(|(\mathcal{J} \downarrow j)|) = card(\{i \mid \exists f : i \to j \text{ in } \mathcal{J}\}) \leq \aleph_0$ i.e. for any object j there are at most countably infinitely many objects from which there is a morphism into j

(ii) for all $i, j \in \mathcal{J}$, $card(\mathcal{J}(i,j)) \leq \aleph_0$, i.e. all hom-sets are at most countably infinite.

Lemma 1.3.6 Given a weakly ω-filtered category $\mathcal{J}$, a functor $F : \mathcal{J} \to \mathcal{C}$ where $\mathcal{C}$ is cocomplete, and a functor $T : \mathcal{C} \to \mathcal{D}$ which preserves coequalizers and filtered colimits, then T preserves the left Kan extension along $I : \mathcal{J} \to \mathcal{J}_0$:

$$T\mathrm{Lan}_I F \cong \mathrm{Lan}_I TF \tag{1.8}$$

Proof. Since $\mathcal{C}$ is cocomplete, the left Kan extension can be constructed pointwise by equation 1.5: hence, for $j \in \mathcal{J}_0$, $\mathrm{Lan}_I F(j)$ is given as

$$\mathrm{Lan}_I F(j) = colim\left((I \downarrow j) \xrightarrow{P} \mathcal{J} \xrightarrow{F} \mathcal{C}\right) \tag{1.9}$$

We will now investigate how cones over FP look, and construct a colimiting one. We will then show the lemma by showing that T preserves this colimiting cone.

We first show that for all $i, j \in \mathcal{J}$, there is an object $S \in \mathcal{C}$ and a morphism $q : Fj \to S$ in $\mathcal{C}$ such that q is the coequalizer of all arrows $Fk : Fi \to Fj$ for $k : i \to j$ in $\mathcal{J}$, written

$$(S, q) = coeq\left(\{Fk : Fi \to Fj \mid k : i \to j \text{ in } \mathcal{J}\}\right)$$

and that T preserves this coequalizer. Since all hom-sets in $\mathcal{J}$ are at most countably infinite, we can enumerate the morphisms in $\mathcal{J}(i,j)$ as $f_1, f_2 \ldots, f_n, \ldots$. If

$J(i,j)$ is empty, then $S \overset{def}{=} Fj$ and $q \overset{def}{=} \mathbf{1}_{Fj}$. Otherwise, S and q are given as the colimit $\underset{n<\omega}{colim}\, S_n$ of the ω-chain $(S_n)_{n<\omega}$ with $\iota_n : S_{n-1} \to S_n$ defined as

$$\begin{array}{rclrcl} S_0 & \overset{def}{=} & Fj & q_0 & \overset{def}{=} & \mathbf{1}_{Fj} \\ (S_{n+1}, \iota_{n+1}) & \overset{def}{=} & coeq\,(q_n \cdot Ff_{n+1}, q_n \cdot Ff_0) & q_{n+1} & \overset{def}{=} & \iota_{n+1} \cdot q_n \end{array}$$

where we write (S,q) for the coequalizing object and morphism of the two arrows $q_n \cdot Ff_{n+1}$ and $q_n \cdot Ff_0$. Then for all $m < \omega$, $i, j < m$, $q_m \cdot Ff_i = q_m \cdot Ff_j$, hence the colimit (S,q) of the chain makes all the arrows equal. It remains to show this is the coequalizer of all morphisms in $\mathcal{J}(i,j)$, which is shown by the universal property: suppose there is an object $X \in \mathcal{C}$ and a morphism $g : Fj \to X$ such that for all $h_1, h_2 : i \to j$ in $\mathcal{J}$, $g \cdot Fh_1 = g \cdot Fh_2$, then we construct a unique morphism $!_g : S \to X$ such that $g = !_g \cdot q$. The morphism $!_g$ is given by a cone $\nu_n : S_n \to X$ over the chain such that for all $n < \omega$, $\nu_n \cdot q_n = g$. This cone is defined inductively as follows: $\nu_0 \overset{def}{=} g$, with $\nu_0 \cdot q_0 = g$; and ν_{n+1} is given by ν_n as follows: with $g \cdot Ff_{n+1} = g \cdot Ff_0$, we have $\nu_n \cdot q_n \cdot Ff_{n+1} = \nu_n \cdot q_n \cdot Ff_0$, and since (S_{n+1}, q_{n+1}) is the coequalizer of $q_n \cdot Ff_{n+1}$ and $q_n \cdot Ff_0$, there is a unique ν_{n+1} such that $\nu_{n+1} \cdot \iota_{n+1} = \nu_n$. Then we have $\nu_{n+1} \cdot q_{n+1} = \nu_{n+1} \cdot \iota_{n+1} \cdot q_n = \nu_n \cdot q_n = g$, as required. Since T preserves filtered colimits, it will preserve the colimit of the chain, and since it further preserves coequalizers, it will preserve the coequalizer of all arrows:

$$\begin{array}{l} T(coeq\,(\{Fk : Fi \to Fj \mid k : i \to j \text{ in } \mathcal{J}\})) \\ \cong coeq\,(\{TFk : TFi \to TFj \mid k : i \to j \text{ in } \mathcal{J}\}) \end{array} \tag{1.10}$$

We now turn to the cone over FP. Formally, a cone $\nu : FP \Rightarrow \Delta x$ is given by morphisms $\nu_f : FP(f) \to x$ in $\mathcal{C}$, indexed by objects f of $(I \downarrow j)$, which are in turn morphisms $f : i \to j$ in $\mathcal{J}_0$ where i is in the image of the inclusion functor I and thus in $\mathcal{J}$, such that for all morphism $u : f \to g$ (where $g : k \to j$) in $(I \downarrow j)$, given by morphisms $u : i \to k$ in $\mathcal{J}$ such that $g \cdot u = f$, $\nu_g \cdot FP(u) = \nu_f$, which amounts to $\nu_g \cdot Fu = \nu_f$.

We first show that any cone $\nu : FP \Rightarrow \Delta x$ over FP is determined by a *single* morphism $\nu : Fj \to x$ in $\mathcal{C}$ which if $j \in \overline{\mathcal{J}}$ has to make all parallel arrows into j equal: i.e. for all $f_1, f_2 : i \to j$ in $\mathcal{J}$, $\nu \cdot Ff_1 = \nu \cdot Ff_2$.

We consider two cases: $j \in \mathcal{J}$, and $\overline{\jmath} \in \overline{\mathcal{J}}$. For the first, the identity $\mathbf{1}_j : Ij \to j$ is an object of $(I \downarrow j)$, giving a morphism $\nu_{\mathbf{1}_j} : j \to x$, and moreover it is terminal— for any other $f : Ii \to j$, there is a unique morphism in $(I \downarrow j)$, given by f, into $\mathbf{1}_j : Ij \to j$. Hence, any $\nu_f : FP(f) \to x$ has to filter through

$\nu_{\mathbf{1}} : j \to j$: $\nu_f = \nu_{\mathbf{1}} \cdot FP(f)$, which decodes as

$$\begin{tikzcd}
Fi \arrow[rr, "\nu_f"] \arrow[dr, "Ff"'] & & x \\
& Fj \arrow[ur, "\nu_{\mathbf{1}}"'] &
\end{tikzcd}$$

hence $\nu_{\mathbf{1}}$ determines the value of ν_f for all other $f : i \to j$.

For the second case, an object $(I \downarrow \overline{\jmath})$ is a morphism $Ii \to \overline{\jmath}$ in $\mathcal{J}_0$, which is $\iota_{i,j}$, given by some $f : i \to j$ in $\mathcal{J}$. The identity $\mathbf{1} : j \to j$ gives a weakly terminal object $\iota_{j,j}$, because for any $f : i \to j$, $\iota_{j,j} \cdot f = \iota_{i,j}$; but this not unique, since there can be another $g : i \to j$, which gives the same object in $(I \downarrow j)$ (namely, $\iota_{i,j}$), but a different morphism in $(I \downarrow j)$ into $\iota_{j,j}$, given by g: $\iota_{j,j} \cdot g = \iota_{i,j}$. Now both f and g give rise to the same object $\iota_{i,j}$ in $(I \downarrow j)$ and hence the same morphism $\nu_{\iota_{i,j}} : Fi \to x$ for the cone ν, which is also given by both f and g filtering through $\nu_{\mathbf{1}}$:

$$\begin{tikzcd}
Fi \arrow[rr, "\nu_{\iota_{i,j}}"] \arrow[dr, shift left, "Fg"] \arrow[dr, shift right, "Ff"'] & & x \\
& Fj \arrow[ur, "\nu_{\mathbf{1}}"'] &
\end{tikzcd}$$

Hence if for all $f_1, f_2 : i \to j$, $Ff_1 \cdot \nu_{\mathbf{1}} = Ff_2 \cdot \nu_{\mathbf{1}}$, then $\nu_{\mathbf{1}} : Fj \to x$ gives rise to a cone over FP.

It is now clear what a colimiting cone over FP is; namely, for $j \in \mathcal{J}$, it is given by $\nu \overset{def}{=} \mathbf{1}_{Fj}$, and for $j \in \overline{\mathcal{J}}$, ν is the coequalizer in $\mathcal{C}$ of the image $Ff, Fg : Fi \to Fj$ of parallel arrows $f, g : i \to j$ into j under F. This reflects the intuition behind the construction of $\mathcal{J}_0$ and the Kan extension: on the objects of $\mathcal{J}$, nothing changes, but the additional objects of $\mathcal{J}_0$ get mapped to the coequalizing objects, making $\mathcal{J}_0$ filtered.

We are now going to construct a colimiting cone over FP, and show that T preserves this construction. For the first case, $j \in \mathcal{J}$, this is rather trivial: with $\nu \overset{def}{=} \mathbf{1}_{Fj}$, $T(\nu) = \mathbf{1}_{TFj}$ because any functor preserves identities.

For the second case, let $j \in \overline{\mathcal{J}}$. Since $\mathcal{J}$ is weakly ω-filtered, the objects of $(I \downarrow j)$ can be enumerated as a sequence $j_1, j_2, \ldots, j_n, \ldots$. We now define a chain $(S_i)_{i<\omega}$ the colimit of which will be a colimiting cone over FP. As above, we write $(S, q) = \mathit{coeq}\,(M)$ for the coequalizing object and morphism of a set M of

$$
\begin{tikzcd}
 & & Fj = S_0 \arrow[r, "\nu"] \arrow[d, "\sigma_1"] & x \\
Fj_1 \arrow[r, shift left=1.5] \arrow[r] \arrow[r, shift right=1.5] & Fj \arrow[ur, equal, "q_0"] \arrow[r, "q_1"] \arrow[dr, "q_2"] \arrow[ddr, "q_3"'] & S_1 \arrow[ur, "\mu_1"] \arrow[d, "\sigma_2"] & \\
Fj_2 \arrow[u, shift left=1, start anchor=north east, end anchor=south] \arrow[u, shift right=1, start anchor=north east, end anchor=south] & & S_2 \arrow[uur, "\mu_2"] \arrow[d, "\sigma_3"] & \\
Fj_3 \arrow[uur, shift left=1.5] \arrow[uur] \arrow[uur, shift right=1.5] & & S_3 \arrow[uuur, "\mu_3"'] &
\end{tikzcd}
\tag{1.11}
$$

morphisms with the same source and target:

$$
\begin{aligned}
(S_0, \sigma_0) &\stackrel{def}{=} (Fj, \mathbf{1}_{Fj}) \\
q_0 &\stackrel{def}{=} \mathbf{1}_{Fj} \\
(S_{n+1}, \sigma_{n+1}) &\stackrel{def}{=} coeq\,(\{q_n{\cdot}Ff \mid f : j_n \to j \text{ in } \mathcal{J}\}) \\
q_{n+1} &\stackrel{def}{=} \sigma_{n+1}{\cdot}q_n
\end{aligned}
$$

(see also diagram 1.11). The colimit of this chain is also a colimit cone over FP:

$$colim\, FP \cong \underset{n<\omega}{colim}\, S_n$$

This is shown by its universal property: given any other cone over FP, which as shown above is given by a morphism $\nu : Fj \to x$ such that $\nu{\cdot}Ff = \nu{\cdot}Fg$ for all $f, g : i \to j$, there is a unique cone morphism $!_\nu : \underset{n<\omega}{colim}\, S_n \to x$. This morphism is constructed by constructing a cone over the chain $(S_i)_{i<\omega}$, given by morphisms $\mu_i : S_i \to x$ such that $\nu = \mu_n{\cdot}q_n$, as follows:

- $\mu_0 \stackrel{def}{=} \nu$, with $\nu = \mu_0{\cdot}\mathbf{1}_{Fj} = \mu_0{\cdot}q_0$;

- given $\mu_n : S_n \to x$, we have $\nu = \mu_n{\cdot}q_n$ and for any $f, g : j_n \to j$, $\nu{\cdot}Ff = \nu{\cdot}Fg$. Hence $\mu_n{\cdot}q_n{\cdot}Ff = \mu_n{\cdot}q_n{\cdot}Fg$, so by the universal property of S_{n+1} there is $\mu_{n+1} : S_{n+1} \to x$ s.t. $\mu_{n+1}{\cdot}\sigma_n = \mu_n$ (which makes μ_i a cone over the chain), and further

  $$\mu_{n+1}{\cdot}q_{n+1} = \mu_{n+1}{\cdot}\sigma_{n+1}{\cdot}q_n = \mu_n{\cdot}q_n = \nu$$

  as required (see also diagram 1.11).

It remains to show that T preserves the colimiting cone just constructed. Since T preserves filtered colimits, it in particular preserves colimits of ω-chains, hence

$$T \operatorname*{\mathit{colim}}_{n<\omega} S_n \cong \operatorname*{\mathit{colim}}_{n<\omega} TS_n$$

Further, by equation 1.10, T preserves coequalizers of sets of morphisms, hence

$$\begin{aligned} TS_n &= T(\mathit{coeq}\,(\{q_n \cdot Ff \mid f : j_n \to j \text{ in } \mathcal{J}\})) \\ &\cong \mathit{coeq}\,(\{Tq_n \cdot TFf \mid f : j_n \to j \text{ in } \mathcal{J}\}) \end{aligned}$$

This describes the colimiting cone for $(I \downarrow j) \xrightarrow{P} \mathcal{J}_0 \xrightarrow{TF} \mathcal{D}$, which gives $\mathrm{Lan}_I TF$. Hence, for all $j \in \mathcal{J}_0$, $T\mathrm{Lan}_I F(j) \cong \mathrm{Lan}_I TF(j)$. □

Lemma 1.3.7 A functor $T : \mathcal{C} \to \mathcal{D}$, where $\mathcal{C}$ is cocomplete, preserves weakly ω-filtered colimits if it preserves filtered colimits and coequalizers.

Proof. Let $\mathcal{J}$ be a weakly ω-filtered category and $F : \mathcal{J} \to \mathcal{C}$, then we have

$$\begin{array}{rcll} T(\mathit{colim}\, F) & \cong & T\,\mathit{colim}\,\mathrm{Lan}_I F & \text{by isomorphism 1.7} \\ & \cong & \mathit{colim}\, T\mathrm{Lan}_I F & \text{since } T \text{ preserves filtered colimits,} \\ & & & \text{and } \mathrm{Lan}_I F \text{ is filtered by lemma 1.3.4} \\ & \cong & \mathit{colim}\,\mathrm{Lan}_I TF & \text{by isomorphism 1.8} \\ & \cong & \mathit{colim}\, TF & \text{by isomorphism 1.7} \end{array}$$

□

We call a functor F *finitary* if it preserves filtered colimits, and *strongly finitary* if it preserves weakly ω-filtered colimits.[5] A monad is called (strongly) finitary if its action is (strongly) finitary, and the category given by strongly finitary monads and all monad morphisms between them is written as $\mathbf{Mon}_{Fin}(\mathcal{C})$.

1.4 Basic Notions of Enriched Category Theory

The following is an introduction to those concepts and notions of enriched category theory which shall be needed later on. No proofs are given. The reader is referred to the standard text in the field [36] for an in-depth presentation of the theory, although some readers may find the concise precision of the text too demanding, and it covers far more material than is required here. [4, Chapter 6] gives a more detailed introduction to the area, covering all the concepts presented here, and is the recommended companion to this introduction. Finally, [46] contains a good motivation, and a very readable introduction to enriched category theory.

[5] A note on the terminology: since not all weakly filtered colimits are filtered, preservation of weakly filtered colimits is a stronger requirement than preservation of filtered colimits, hence we call those functors strongly finitary.

1.4.1 Symmetric monoidal closed categories

A *monoidal* category $\mathcal{V} = (\mathcal{V}_0, \otimes, I, a, l, r)$ is given by

- a category $\mathcal{V}_0$ called the *underlying category*,
- a functor $\otimes : \mathcal{V}_0 \times \mathcal{V}_0 \to \mathcal{V}_0$ called the *multiplication*,
- an object $I \in \mathcal{V}_0$ called the *unit*,
- for all $x, y, z \in \mathcal{V}_0$, an isomorphism $a_{x,y,z} : (x \otimes y) \otimes z \to x \otimes (y \otimes z)$ called the *associativity*, which is natural in x, y and z,
- and for all $x \in \mathcal{V}_0$, isomorphism $l_x : I \otimes x \to x$ and $r_x : x \otimes I \to x$ called *left* and *right unit* respectively, which are natural in x,

making diagrams 1.12 and 1.13 commute for all objects W, X, Y and Z. Dia-

$$\begin{tikzcd}
((W \otimes X) \otimes Y) \otimes Z \arrow[r, "a"] \arrow[d, "a \otimes \mathbf{1}_Z"'] & (W \otimes X) \otimes (Y \otimes Z) \arrow[r, "a"] & W \otimes (X \otimes (Y \otimes Z)) \\
(W \otimes (X \otimes Y)) \otimes Z \arrow[rr, "a"'] & & W \otimes ((X \otimes Y) \otimes Z) \arrow[u, "\mathbf{1}_W \otimes a"']
\end{tikzcd} \tag{1.12}$$

$$\begin{tikzcd}
(X \otimes I) \otimes Y \arrow[rr, "a"'] \arrow[dr, "r \otimes \mathbf{1}_Y"'] & & X \otimes (I \otimes Y) \arrow[dl, "\mathbf{1}_X \otimes l"] \\
& X \otimes Y &
\end{tikzcd} \tag{1.13}$$

grams of isomorphisms such as these are called *coherence axioms*. They ensure that every diagram built out of the left and right unit, associativity and multiplication commutes (see [52, Chapter VII]). If the isomorphisms are all identities, the monoidal category $\mathcal{V}$ is called *strict*. In that case, we shall allow ourselves to omit the identities when denoting $\mathcal{V}$, as we shall when these are canonical isomorphisms: for example, $(\mathbf{Set}, \times, \{*\})$ is the monoidal category of small sets with the cartesian product and the unit set (and the canonical isomorphisms), and $(\mathbf{Set}, +, \emptyset)$ is the monoidal category of small sets with the coproduct and the empty set. Other examples of monoidal categories include the category **Grph** of graphs (see §1.5.1 below), the category **Cat** of small categories (see §1.5.3 below), the category $\boldsymbol{\rightarrow}$, and more generally, ordered sets with finite intersections.

A monoidal category $\mathcal{V}$ is *symmetric* if there is natural isomorphism $s_{x,y} : x \otimes y \to y \otimes x$, subject to certain other coherence axioms and the symmetry axiom $s_{y,x} \cdot s_{x,y} = \mathbf{1}_{x \otimes y}$.

A monoidal category $\mathcal{V}$ is *closed* if the functor $- \otimes X : \mathcal{V}_0 \to \mathcal{V}_0$ has a right adjoint $[X, -] : \mathcal{V}_0 \to \mathcal{V}_0$ for every object X, with the counit $\varepsilon_X : [X, Y] \otimes X \to Y$ called the *evaluation* at X.

If the multiplication is the cartesian product of the two objects, and the unit the terminal object, with the associativity and unit the canonical isomorphisms, then $\mathcal{V}$ is *cartesian*; and if it is closed, it is a *cartesian closed category.*

In the examples above, **Set** is cartesian closed, **Grph** is closed with respect to the "tensor product" of two graphs as defined below, but not with respect to the cartesian product, and **Cat** is closed with respect to both the cartesian and another monoidal product.

For a monoidal category $\mathcal{V}$, a morphism $f : I \to X$ in $\mathcal{V}_0$ can be regarded as an element of X; under the mild assumption that $\mathcal{V}_0$ is locally small (i.e. all hom-sets are small) there is a functor $V \stackrel{def}{=} \mathcal{V}_0(I, -) : \mathcal{V}_0 \to \mathbf{Set}$ which gives the "underlying set of objects" for the objects of $\mathcal{V}$. For example, for a graph $\mathcal{G}$, $V\mathcal{G}$ is (isomorphic to) its set of vertices, and for a category $\mathcal{C}$, $V\mathcal{C}$ is its set of objects.

1.4.2 $\mathcal{V}$-categories

The monoidal structure is all one needs to do enriched category theory: we can now replace the set of morphisms between any two objects with an object from $\mathcal{V}_0$, and obtain $\mathcal{V}$-categories, $\mathcal{V}$-functors, and $\mathcal{V}$-natural transformations analogously.

If we want to explicitly distinguish categories as defined above from $\mathcal{V}$-categories we are just about to define and 2-categories later on, we will call them *ordinary* categories (and similarly, ordinary functors, natural transformations, adjunctions etc.)

Let $\mathcal{V}$ be a monoidal category. A *$\mathcal{V}$-category* $\mathcal{C}$ is given by

- a set $|\mathcal{C}|$ of *objects*,
- for every two objects $X, Y \in \mathcal{C}$, an object $\mathcal{C}(X, Y) \in \mathcal{V}_0$, called the *hom-object*,
- for all $X, Y, Z \in \mathcal{C}$, a morphism $c_{X,Y,Z} : \mathcal{C}(Y, Z) \otimes \mathcal{C}(X, Y) \to \mathcal{C}(X, Z)$ in $\mathcal{V}_0$ called the *composition*,
- and for all $X \in \mathcal{C}$, a morphism $i_X : I \to \mathcal{C}(X, X)$ in $\mathcal{V}_0$ called the *identity* on X,

such that the composition is associative and the identity a unit on the left and right.

Given two $\mathcal{V}$-categories $\mathcal{C}$ and $\mathcal{D}$, a *$\mathcal{V}$-functor* $F : \mathcal{C} \to \mathcal{D}$ is given by

- an object function $F_{Obj} : |\mathcal{C}| \to |\mathcal{D}|$, and
- for any two objects $X, Y \in \mathcal{C}$, a morphism in $\mathcal{V}_0$

$$F_{X,Y} : \mathcal{C}(X, Y) \to \mathcal{D}(F_{Obj}(X), F_{Obj}(Y))$$

which respect the composition and identity.

As a matter of convenience, we usual do not distinguish between F and F_{Obj}, with the tacit understanding that we mean $F_{Obj}(X)$ in the previous notation when writing $F(X)$ for an object $X \in \mathcal{C}$.

Given two $\mathcal{V}$-functors $F, G : \mathcal{C} \to \mathcal{D}$, a $\mathcal{V}$-natural transformation $\alpha : F \Rightarrow G$ is given by a family of morphisms $\{\alpha_X : I \to \mathcal{D}(FX, GX)\}_{X \in \mathcal{C}}$ indexed by objects in $\mathcal{C}$, which is natural in the sense that diagram 1.14 commutes for all X and Y.

$$\begin{tikzcd}
& \mathcal{C}(X,Y)\otimes I \arrow[r, "G_{X,Y}\otimes\alpha_X"] & \mathcal{D}(GX,GY)\otimes\mathcal{D}(FX,GX) \arrow[dr, "c_{FX,GX,GY}"] & \\
\mathcal{C}(X,Y) \arrow[ur, "r^{-1}_{X,Y}"] \arrow[dr, "l^{-1}_{X,Y}"'] & & & \mathcal{D}(FX,GY) \\
& I\otimes\mathcal{C}(X,Y) \arrow[r, "\alpha_Y\otimes F_{X,Y}"'] & \mathcal{D}(FY,GY)\otimes\mathcal{D}(FX,FY) \arrow[ur, "c_{FX,FY,GY}"'] &
\end{tikzcd} \tag{1.14}$$

We shall in the following assume that $\mathcal{V}$ is an arbitrary but fixed symmetric monoidal closed category. Also, functors between $\mathcal{V}$-categories are always understood to be $\mathcal{V}$-functors, and natural transformations between $\mathcal{V}$-functors are always $\mathcal{V}$-natural transformations.

1.4.3 2-categories

The category of all small $\mathcal{V}$-categories, $\mathcal{V}$-functors, and $\mathcal{V}$-natural transformations between them has as much structure as **Cat**, the category of all small ordinary categories, so all constructions on ordinary categories which can be expressed in terms of functors and natural transformations in **Cat** are also possible on $\mathcal{V}$-categories. To make this precise, the concept of a 2-category will be introduced.

A 2-category is a category enriched over the category of all small categories with the cartesian product, the one-object-category **1** as unit and the canonical isomorphisms; a 2-functor is **Cat**-enriched functor, and so on. In more detail, a 2-category $\mathcal{C}$ has

- a set $|\mathcal{C}|$ of *objects*,
- for every two objects $X, Y \in \mathcal{C}$, a hom-category $\mathcal{C}(X, Y)$;

- for all $X, Y, Z \in \mathcal{C}$, a bifunctor $\circ_{X,Y,Z} : \mathcal{C}(Y,Z) \times \mathcal{C}(X,Y) \to \mathcal{C}(X,Z)$;
- and for all $X \in \mathcal{C}$, an object $\mathbf{1}_X \in \mathcal{C}(X,X)$ in the hom-category;

A 2-category has two kinds of composition: the composition within each hom-category, called *vertical composition*, and the composition given by the bifunctor $\circ_{X,Y,Z}$, called *horizontal composition*. The functoriality of $\circ$ means that it has to preserve the vertical composition; this is also called the *interchange law*. The objects of the hom-categories are also called the 1-cells of $\mathcal{C}$, and the morphisms of the hom-categories the 2-cells .

By taking the objects $|\mathcal{C}|$ of $\mathcal{C}$, and the objects $|\mathcal{C}(X,Y)|$ as hom-sets between X and Y, with the object function of $\circ$ as the composition, we obtain the *underlying ordinary category* $\mathcal{C}_0$ of $\mathcal{C}$; and similarly, we obtain an underlying ordinary functor F_0 for a 2-functor F.

The horizontal and vertical composition allows us to draw "pasting diagrams" [40, 66] to reason about the equality of 2-cells. Also, given a morphism $f : X \to Y$ and a 2-cell $\alpha \in \mathcal{C}(Y,Z)$, we can compose α with f, written as $f \multimap \alpha \stackrel{def}{=} \mathbf{1}_f \circ \alpha$.[6] As an example of this, here is the diagram defining 2-naturality: given two 2-functors $F, G : \mathcal{C} \to \mathcal{D}$, a 2-natural transformation $\nu : F \Rightarrow G$ is, for every object $X \in \mathcal{C}$, a 1-cell $\nu_X \in \mathcal{D}(FX, GX)$; then diagram 1.14 is a diagram of functors, which on the object level imply the usual naturality, and on the morphism level commutativity of diagram 1.15. Since in general $\mathcal{V}$-natural transformations form

$$\begin{tikzcd} FX \arrow[r, shift left=2, "Ff"] \arrow[r, shift right=2, "Fg"'] \arrow[r, phantom, "\Downarrow F\alpha"] \arrow[d, "\nu_X"'] & FY \arrow[d, "\nu_Y"] \\ GX \arrow[r, shift left=2, "Gf"] \arrow[r, shift right=2, "Gg"'] \arrow[r, phantom, "\Downarrow G\alpha"] & GY \end{tikzcd} \tag{1.15}$$

a $\mathcal{V}$-object, there is also a notion of 2-cells between 2-natural transformations: a *modification* $\gamma : \nu \to \mu : F \Rightarrow G : \mathcal{C} \to \mathcal{D}$ is given by a family of 2-cells $\{\gamma_X\}_{X \in \mathcal{X}}$

[6] This is also sometimes referred to as *tadpole composition*, since this is what it is supposed to look like in a diagram. Incidentally, it is possible to define a 2-category in terms of this tadpole compositions: a set of objects, a set of morphisms between objects, and a set of 2-cells between morphisms, with an identity morphism for every object, an identity 2-cell for every morphism, a (horizontal) composition between the morphisms, and a tadpole composition "on the left" between any morphism $f : Y \to X$ and 2-cell $\alpha : g \Rightarrow h : Y \to Z$, and "on the right" between any 2-cell α and morphism $u : Z \to U$, which is subject to ten equations[82, pg.52].

indexed by objects in $\mathcal{C}$ satisfying diagram 1.16.[7]

$$\begin{tikzcd}
FX \arrow[r, "Ff"] \arrow[d, shift right=4, "\nu_X"'] \arrow[d, shift left=4, "\mu_X"] \arrow[d, phantom, "\overset{\gamma_X}{\Rightarrow}"] & FY \arrow[d, shift right=4, "\nu_Y"'] \arrow[d, shift left=4, "\mu_Y"] \arrow[d, phantom, "\overset{\gamma_Y}{\Rightarrow}"] \\
GX \arrow[r, "Gf"] & GY
\end{tikzcd} \tag{1.16}$$

2-categories are more than just another enriched category; they are the "natural" setting for constructions such as adjunctions [40] and monads [85], the whole theory of which can be entirely formulated in terms of 1-cells and 2-cells in a 2-category.[8] Since small $\mathcal{V}$-categories, $\mathcal{V}$-functors and $\mathcal{V}$-natural transformations between form a 2-category $\mathcal{V}$-**Cat**, we can deduce how to define adjunctions and monads for $\mathcal{V}$-categories as specializations of the more general case.

1.4.4 $\mathcal{V}$ is a $\mathcal{V}$-category

The closed structure of $\mathcal{V}$ makes $\mathcal{V}$ itself a $\mathcal{V}$-category: more precisely, if $\mathcal{V}_0$ is closed, we have a bijection

$$\mathcal{V}_0(X \otimes Y, Z) \cong \mathcal{V}_0(X, [Y, Z])$$

Let $X \stackrel{def}{=} I$, and recall the definition of V as $V \stackrel{def}{=} \mathcal{V}_0(I, -)$, then

$$\mathcal{V}_0(Y, Z) \cong \mathcal{V}_0(I \otimes Y, Z) \cong \mathcal{V}_0(I, [Y, Z]) = V[Y, Z] \tag{1.17}$$

so the $\mathcal{V}$-object $[Y, Z]$ is the lifting of the hom-set between Y and Z through V; and we can define the $\mathcal{V}$-category $\mathcal{V}$:

- the objects are the same as $\mathcal{V}_0$: $|\mathcal{V}| \stackrel{def}{=} |\mathcal{V}_0|$;
- for any two objects X, Y, $\mathcal{V}(X, Y) \stackrel{def}{=} [X, Y]$;
- the composition is given by under the bijection

$$\mathcal{V}_0([Y, Z] \otimes [X, Y], [X, Z]) \cong \mathcal{V}_0(([Y, Z] \otimes [X, Y]) \otimes X, Z)$$

[7] This means that the 2-category **Cat**-**Cat** of small 2-categories, 2-functors and 2-natural transformations is enriched over **Cat**-**Cat**, and thus a "3-category", and this process can be carried on to n-categories, but we shall refrain from doing so here, since little is known about the medical implications.

[8] For example, the construction of an adjunction from a monad by the Eilenberg-Moore category $\mathcal{C}^T$ for a monad T on a $\mathcal{V}$-category $\mathcal{C}$ can be described as the left adjoint for the 2-functor mapping every category to the identity monad on it; and the Kleisli construction is just the dual of this case.

as the composition of

$$([Y,Z]\otimes[X,Y])\otimes X \xrightarrow{a} ([Y,Z]\otimes([X,Y]\otimes X)) \xrightarrow{\mathbf{1}\otimes\varepsilon_X} ([Y,Z]\otimes Y) \xrightarrow{\varepsilon_Y} Z$$

- and the identities are given by the image of $l : I\otimes X \to X$ under the bijection $\mathcal{V}_0(I,[X,X]) \cong \mathcal{V}_0(I\otimes X, X)$

1.4.5 Enriched Functor Categories

Under the assumption that $\mathcal{V}$ is complete, and again certain size conditions, given two $\mathcal{V}$-categories $\mathcal{C}$,$\mathcal{D}$, there is $\mathcal{V}$-category $\mathcal{V}[\mathcal{C},\mathcal{D}]$, the underlying ordinary category of which is $\mathcal{V}$-$\mathbf{Cat}(\mathcal{C},\mathcal{D})$, the category of $\mathcal{V}$-functors and natural transformations between them; for the details of this construction, we refer to [36, Chapter 2] or [4, Proposition 6.3.1].

1.4.6 Limits and Colimits for $\mathcal{V}$-categories

The generalization of limits from ordinary category theory to the enriched case is more subtle. Recall that a limit for an ordinary functor $F:\mathcal{J}\to\mathcal{C}$ is defined in terms of the representability of the functor $[\mathcal{J},\mathcal{C}](\Delta-,F):\mathcal{C}^{\mathrm{op}}\to\mathbf{Set}$. The problem is that it is not possible to define the constant functor $\Delta Y:\mathcal{J}\to\mathcal{C}$ in the enriched context, because we would have to give, for all $J,K\in\mathcal{J}$, a morphism $\Delta X_{J,K}$ in $\mathcal{V}_0$ filtering through the identity on X (diagram 1.18); for the ordinary

$$\begin{tikzcd} \mathcal{J}(J,K) \arrow[rr, "\Delta X_{J,K}"] \arrow[dr, "m"'] & & \mathcal{C}(X,X) \\ & I \arrow[ur, "i_X"'] & \end{tikzcd} \tag{1.18}$$

case, $I=\{*\}$ is terminal in $\mathbf{Set}$, so m is the unique morphism $!_{\mathcal{J}(J,K)}$ into $\{*\}$, but in general this map m need not exist.

It turns out that the appropriate generalization of this concept is to allow *weighted* or *indexed* limits. A weight (or index) for a diagram given by a functor $F:\mathcal{J}\to\mathcal{C}$ is a functor $G:\mathcal{J}\to\mathcal{V}$. Then the limit of F weighted or indexed by G, exists if the functor $[\mathcal{J},\mathcal{V}](G,\mathcal{C}(X,F-))$ exists for all $X\in\mathcal{C}$ and admits a representation

$$\mathcal{C}(X, lim_G F)\cong[\mathcal{J},\mathcal{V}](G,\mathcal{C}(X,F-))$$

We shall not pursue this theory here, since we do not need it below. The interested reader is referred to [4, Section 6.4] for a more lucid motivation of the construction, and to [36] for the finer points of the theory. For the reader interested in examples, [38] presents constructions like comma-categories and lax limits as (simple) weighted limits in **Cat**.

Like ordinary limits and colimits, weighted limits (colimits) are preserved by right (left) adjoints (which are enriched adjunctions, of course). For ordinary categories, we can weigh every diagram with $\Delta(\{*\}) : \mathcal{J} \to \mathbf{Set}$ (for any $\mathcal{J}$), recovering the conical limits from §1.3.3.

1.4.7 Tensors and Cotensors

In the category **Set**, sets of cardinality n represent n-tuples of elements under the following bijection:

$$\mathbf{Set}(\coprod_n \{*\}, Z) \cong \prod_n \mathbf{Set}(\{*\}, Z) \tag{1.19}$$

There is no reason that a similar bijection should hold in $\mathcal{V}_0$; hence, to represent "tuples" of elements the notions of *tensor* and *cotensor* are introduced. In a $\mathcal{V}$-category $\mathcal{A}$, a the *tensor* of an object $A \in \mathcal{A}$ and $X \in \mathcal{V}$ is an object $V \otimes A$ s.t. for all $B \in \mathcal{A}$,

$$\mathcal{A}(X \otimes A, B) \cong \mathcal{V}(X, \mathcal{A}(A, B))$$

(Note the overloading of the symbol $\otimes$.) Dually the *cotensor* of A and X is an object $X \pitchfork A$ s.t.

$$\mathcal{A}(A, X \pitchfork B) \cong \mathcal{V}(X, \mathcal{A}(A, B))$$

If these objects exist for all A and X, we say $\mathcal{A}$ is tensored (and cotensored, respectively). Tensors and cotensors are the simplest examples of weighted limits: the cotensor $X \pitchfork A$ is the limit of the functor $F : \mathbf{1} \to \mathcal{A}$ (where $\mathbf{1}$ is the one-object $\mathcal{V}$-category), weighted by $G : \mathbf{1} \to \mathcal{V}$ with $G(\{*\}) = X$. Proposition 1.3.1 now generalizes, a $\mathcal{V}$-category $\mathcal{A}$ has all weighted colimits if it has all coproducts, coequalizers and is cotensored. Further, if $U : \mathcal{A} \to \mathcal{B}$ has a left adjoint, then U preserves cotensors as well as weighted colimits; and on the other hand, if the underlying ordinary functor $U_0 : \mathcal{A}_0 \to \mathcal{B}_0$ has an ordinary left adjoint, and $U : \mathcal{A} \to \mathcal{B}$ preserves cotensors, then U has a left adjoint.

1.4.8 Change of base

Let $\mathcal{V} = (\mathcal{V}_0, \otimes, I)$ and $\mathcal{W} = (\mathcal{W}_0, \oplus, J)$ be two monoidal closed categories, than a morphism between them is given by

- a functor $F : \mathcal{V}_0 \to \mathcal{W}_0$,

- for all objects $X, Y \in \mathcal{V}$, a morphism $\tau_{X,Y} : FX \oplus FY \to F(X \otimes Y)$ in $\mathcal{W}_0$, which is natural in X and Y, and
- a morphism $\varepsilon : J \to F(I)$ in $\mathcal{W}_0$.

respecting the associativity and the left and right unit (and the symmetry, if it is a morphism between symmetric monoidal closed categories).

A morphism like the above induces a 2-functor $F^* : \mathcal{V}\text{-}\mathbf{Cat} \to \mathcal{W}\text{-}\mathbf{Cat}$, which is defined as follows: let $\mathcal{C}$ be a $\mathcal{V}$-category, then the $\mathcal{W}$-category $\mathcal{D} \stackrel{def}{=} F^*(\mathcal{C})$ has the same objects as $\mathcal{C}$; for any two objects, $\mathcal{D}(X, Y) \stackrel{def}{=} F(\mathcal{C}(X, Y))$, the composition is given by

$$F(\mathcal{C}(Y, Z)) \oplus F(\mathcal{C}(X, Y)) \xrightarrow{\tau} F(\mathcal{C}(Y, Z) \otimes \mathcal{C}(X, Y) \xrightarrow{Fc_{X,Y,Z}} F(\mathcal{C}(X, Z))$$

and the identities for $X \in \mathcal{D}$ are given by

$$J \xrightarrow{\quad\varepsilon\quad} F(I) \xrightarrow{Fi_X} F(\mathcal{C}(X, X))$$

An application of this is that for any symmetric monoidal closed category $\mathcal{V}$, there is a 2-functor $U : \mathcal{V}\text{-}\mathbf{Cat} \to \mathbf{Cat}$, defined as the lifting of the functor $V \stackrel{def}{=} \mathcal{V}_0(I, -) \to \mathbf{Set}$ taking a $\mathcal{V}$-category $\mathcal{C}$ to its *underlying ordinary category* $U\mathcal{C}$.

If $\mathcal{V}$ is cocomplete, this functor has a left adjoint[9] $\mathbf{F}_{\mathcal{V}}(-) : \mathbf{Cat} \to \mathcal{V}\text{-}\mathbf{Cat}$, mapping an ordinary category $\mathcal{C}$ to its *free $\mathcal{V}$-category* $\mathbf{F}_{\mathcal{V}}(\mathcal{C})$; briefly, the construction is as follows: the functor $V : \mathcal{V}_0 \to \mathbf{Set}$ has an ordinary left adjoint $F : \mathbf{Set} \to \mathcal{V}_0$, which for a set X is $\coprod_X I$, the X-th copower of the unit in $\mathcal{V}_0$, and we have a bijection

$$\mathcal{V}_0(\coprod_X I, Y) \cong \prod_X \mathcal{V}_0(I, Y) \cong \mathbf{Set}(X, \mathcal{V}_0(I, Y)) \tag{1.20}$$

For an ordinary category $\mathcal{C}$, the free $\mathcal{V}$-category has the same objects as $\mathcal{C}$, and as hom-objects the image of the hom-sets under F. Again, we refer to the literature for the details of the construction ([4, Section 6.4] or [36, Section 2.5]).

1.5 Particular Categories

We shall in the following investigate the category of graphs, the category of binary relations, and the category of small categories, since they will play an important rôle in modelling reductions later on.

[9]To be precise, a left adjoint 2-functor.

1.5.1 Graphs and the Category Grph

This section reviews the definitions of a graph, the category of all graphs, paths in a graph and the relation between graphs and categories. A closer examination of the category of all graphs reveals that it has a closed structure but it is not cartesian closed. We will further investigate the relation between graphs and relations, and categories and pre-orders.

A Warning

In traditional graph theory, there are various definitions of graphs. Here, we prefer (of course) the definition from [52]. The reader with a background in graph theory rather than category theory should be aware of the following discrepancies between our definition and those found in graph theory: firstly, our graphs are *directed* (but we still say they have edges, not arcs, as is sometimes done); and secondly, our graphs can have more than one edge between two given vertices (graphs like these are sometimes referred to as *multigraphs* [92], although this term does not seem to be standard).

Graphs and the Category Grph

A *graph* $\mathcal{G} = (G_0, G_1, \delta_s, \delta_t)$ is given by a set of *vertices* G_0, a set of *edges* G_1, and two functions $\delta_s, \delta_t : G_0 \to G_1$ assigning a source and a target vertex to every edge.

Given a graph $\mathcal{G}$ as above, we write $V(\mathcal{G})$ for its set of vertices G_0 and $E(\mathcal{G})$ for its set of edges G_1. Further, we write $e : x \to y \in E(\mathcal{G})$ as an abbreviation of the statement that e is an edge in $\mathcal{G}$ with source x and target y.

A *graph morphism* $f : \mathcal{G} \to \mathcal{H}$ from a graph $\mathcal{G}$ to a graph $\mathcal{H}$ is given by two functions $f_V : V(\mathcal{G}) \to V(\mathcal{H})$ and $f_E : E(\mathcal{G}) \to E(\mathcal{H})$ respecting the source and target mapping, i.e.

$$\begin{array}{rcl} \delta_s' \cdot f_E & = & f_V \cdot \delta_s \\ \delta_t' \cdot f_E & = & f_V \cdot \delta_t \end{array} \tag{1.21}$$

The category **Grph** consists of graphs as objects and graph morphisms as morphisms, with the evident identities and compositions.

More abstractly, a graph is a functor from the category $\rightrightarrows$ to the category **Set** of sets. The category **Grph** is the functor category $[\rightrightarrows, \mathbf{Set}]$, and equation 1.21 just says that graph morphisms are natural transformations. This abstract view tells us that **Grph** is lfp, and it also tells us the generators in **Grph** (see §1.3.8 on page 19): the representable functors. Since the category $\rightrightarrows$ has two objects, there are two[10] representable functors, giving two fp objects, namely the graph $\mathcal{I}$, and the graph $\longrightarrow$ which has two vertices and a single edge between them.

Paths in a Graph

A *path* in a graph $\mathcal{G}$ is given by a sequence $<X_1, e_1, X_2, \ldots, X_{n+1}>$ where $n \geq 0$, $X_j \in V(\mathcal{G}), e_i \in E(\mathcal{G})$ and $\delta_s(e_i) = X_i, \delta_t(e_i) = X_{i+1}$ for $j = 1, \ldots, n+1$ and $i = 1, \ldots, n$. The set of all paths in $\mathcal{G}$ is denoted by $\mathcal{G}^*$.

For a path $p = <X_1, e_1, X_2, \ldots, X_{n+1}>$, its source and target are defined as $\mathtt{s}(p) \stackrel{def}{=} X_1$, $\mathtt{t}(p) \stackrel{def}{=} X_{n+1}$. For two paths $p = <X_1, e_1, X_2, \ldots, X_{n+1}>$, $q = <Y_1, f_1, Y_2, \ldots, Y_{m+1}>$ such that $X_{n+1} = Y_1$, their *concatenation* (composition) is defined as

$$p :: q \stackrel{def}{=} <X_1, e_1, X_2, \ldots, X_{n+1}, f_1, Y_2, \ldots, Y_{m+1}>$$

The *length* of a path is the number of edges it contains:

$$|<X_1, e_1, X_2, \ldots, X_{n+1}>| \stackrel{def}{=} n$$

In a non-empty path $<X_1, e_1, X_2, \ldots, X_{n+1}>$ (i.e. $n > 0$), we can omit the vertices X_i, since they are implicitly given by the edges. Thus, in the following, we write $<e_1, e_2, \ldots, e_n>$ for a non-empty path $<\delta_s(e_1), e_1, \delta_t(e_1), e_2, \delta_t(e_2), \ldots, \delta_t(e_n)>$; and for the empty path $<X>$ with $X \in V(\mathcal{G})$, we write $\mathtt{id}_X$.

A *path congruence* on the paths $\mathcal{G}^*$ is an equivalence relation $\equiv \subseteq \mathcal{G}^* \times \mathcal{G}^*$ which is compatible with path composition, i.e. the following two implications hold:

$$p \equiv q \Rightarrow \mathtt{s}(p) = \mathtt{s}(q), \mathtt{t}(p) = \mathtt{t}(q) \tag{1.22}$$

and for all paths $p, q, r, s \in \mathcal{G}^*$ such that $\mathtt{t}(p) = \mathtt{s}(q)$ and $\mathtt{t}(r) = \mathtt{s}(s)$,

$$p \equiv r, q \equiv s \Rightarrow p :: q \equiv r :: s \tag{1.23}$$

Free Categories and Underlying Graphs

For a graph $\mathcal{G}$, its *free category* $\mathcal{F}(\mathcal{G})$ has the vertices $V(\mathcal{G})$ of $\mathcal{G}$ as objects, and paths in $\mathcal{G}$ as morphisms, with $\mathtt{id}_X$ as identities and concatenation as composition. This assignment extends to a functor $\mathcal{F} : \mathbf{Grph} \to \mathbf{Cat}$.

Every small category $\mathcal{C}$ has an *underlying graph*, the vertices of which are the objects of $\mathcal{C}$, and the edges are the morphisms of $\mathcal{C}$ (with source and target as in $\mathcal{C}$). This extends to a functor $U : \mathbf{Cat} \to \mathbf{Grph}$, which is the right adjoint of the functor $\mathcal{F}$.

[10] Up to isomorphism.

By this adjunction, to define a functor $M : \mathcal{F}(\mathcal{G}) \to \mathcal{X}$, it is sufficient to define a graph morphism $d : \mathcal{G} \to U\mathcal{X}$. Then, d is called a *diagram* in $\mathcal{X}$.

A Monoidal Closed Structure on Grph

Grph has a cartesian product, but is not cartesian closed. **Grph** also has a tensor product, which leads to a closed structure in which the internal homs are "transformations" – just like natural transformations on **Cat**, but without the naturality condition.

Given two graphs $\mathcal{G}, \mathcal{H}$, their *cartesian product* $\mathcal{G} \times \mathcal{H}$ is defined as the graph with

$$\begin{aligned} V(\mathcal{G} \times \mathcal{H}) &\stackrel{def}{=} V(\mathcal{G}) \times V(\mathcal{H}) \\ E(\mathcal{G} \times \mathcal{H}) &\stackrel{def}{=} E(\mathcal{G}) \times E(\mathcal{H}) \end{aligned}$$

with the obvious source and target mappings. But this definition does not lead to a closed structure; there is no right adjoint for the functor $- \times \mathcal{G}$. There is one for a different monoidal structure, though:

Definition 1.5.1 (Tensor Product of Graphs) Given two graphs $\mathcal{G}, \mathcal{H}$, their *tensor product* $\mathcal{G} \otimes \mathcal{H}$ is defined as the graph with

$$\begin{aligned} V(\mathcal{G} \otimes \mathcal{H}) &\stackrel{def}{=} V(\mathcal{G}) \times V(\mathcal{H}) \\ E(\mathcal{G} \otimes \mathcal{H}) &\stackrel{def}{=} V(\mathcal{G}) \times E(\mathcal{H}) + E(\mathcal{G}) \times V(\mathcal{H}) \end{aligned}$$

with the source mapping defined as follows:

$$\delta_s((x, e)) \stackrel{def}{=} \begin{cases} (x, \delta_s(e)) & \text{if } x \in V(\mathcal{G}), e \in E(\mathcal{H}) \\ (\delta_s(e), x) & \text{if } e \in E(\mathcal{G}), x \in V(\mathcal{H}) \end{cases}$$

and the target mapping defined analogously.

The unit of the monoidal structure is the one-element graph $\mathcal{I}$ with one vertex, no edges and empty source and target mappings: $\mathcal{I} \stackrel{def}{=} (\{*\}, \emptyset, !, !)$. Note that this is not the unit of the cartesian structure, since for any graph $\mathcal{G}$, $\mathcal{I} \times \mathcal{G}$ has an empty set of edges; rather, the unit for the cartesian structure is given by $\mathbf{1} \stackrel{def}{=} (\{*\}, \{*\}, \mathbf{1}_{\{*\}}, \mathbf{1}_{\{*\}})$.

When showing that the structure given by the tensor product and the unit as defined above is indeed monoidal structure (and even closed), the main emphasis will be on the edges of the graphs involved, since on the vertices, this is the cartesian product on **Set** (which is well-known to be monoidal and closed).

Lemma 1.5.2 **Grph** $\stackrel{def}{=}$ $(\mathbf{Grph}, \otimes, \mathcal{I})$ is a symmetric monoidal category.

Proof. The proof of associativity of the tensor requires distributivity of the coproduct (disjoint union) over the cartesian product in **Set**, plus associativity and commutativity of the two. Hence, for three graphs $\mathcal{G}, \mathcal{H}, \mathcal{K}$, the set of edges (for the vertices, associativity is just associativity of the product in **Set**), is given by

$$\begin{array}{ll} & E((\mathcal{G} \otimes \mathcal{H}) \otimes \mathcal{K}) \\ = & V(\mathcal{G}) \times V(\mathcal{H}) \times E(\mathcal{K}) + (V(\mathcal{G}) \times E(\mathcal{H}) + E(\mathcal{G}) \times V(\mathcal{H})) \times V(\mathcal{K}) \\ \cong & E(\mathcal{G}) \times V(\mathcal{H}) \times V(\mathcal{K}) + V(\mathcal{G}) \times E(\mathcal{H}) \times V(\mathcal{K}) + V(\mathcal{G}) \times V(\mathcal{H}) \times E(\mathcal{K}) \\ \cong & V(\mathcal{G}) \times (V(\mathcal{H}) \times E(\mathcal{K}) + E(\mathcal{H}) \times E(\mathcal{H}))E(\mathcal{G}) \times V(\mathcal{H}) \times V(\mathcal{K}) \\ = & E(\mathcal{G} \otimes (\mathcal{H} \otimes \mathcal{K})) \end{array}$$

The third line gives the set of edges of three edges as a polynomial in the product and sum on **Set**. This polynomial allows the "pointwise" verification of the coherence diagrams 1.12 and 1.13; e.g. for four graphs $\mathcal{G}_1, \mathcal{G}_2, \mathcal{G}_3, \mathcal{G}_4$, the edges of the tensor product can be given as a similar polynomial, which is the result of either way of chasing round diagram 1.12 (see [52, pg. 159]). Symmetry again follows from symmetry of the product and coproduct on **Set**. □

This leads to the closed structure on **Grph**: a transformation between graph morphisms is essentially like a natural transformation but without the naturality condition (for which to be stated the composition of edges would be needed).

Definition 1.5.3 (Transformation) Given two graph morphisms $f, g : \mathcal{G} \to \mathcal{H}$ from a graph $\mathcal{G}$ to a graph $\mathcal{H}$, a *transformation* $\alpha : f \Rightarrow g$ is a set of edges in $\mathcal{H}$ indexed by vertices in $\mathcal{G}$, such that for all $x \in V(\mathcal{G})$ there is a $\alpha_x : f_V(x) \to g_V(x) \in E(\mathcal{H})$.

We obtain a graph representing the function space between two graphs, the so-called internal hom in **Grph**:

Definition 1.5.4 The *internal hom* (or function space graph) of two graphs $\mathcal{G}, \mathcal{H}$ is the graph $[\mathcal{G}, \mathcal{H}]$ which has as its vertices all graph morphisms from $\mathcal{G}$ to $\mathcal{H}$, and as edges all transformations between these graph morphisms, with the obvious source and target mappings.

We now want to show that the internal hom indeed represents the function space, i.e. for any given graph $\mathcal{G}$, the functor $[\mathcal{G}, -] : \mathbf{Grph} \to \mathbf{Grph}$ is right adjoint to the tensor product with $\mathcal{G}$.

Lemma 1.5.5 For any graph $\mathcal{G}$, the functor $[\mathcal{G}, -] : \mathbf{Grph} \to \mathbf{Grph}$ is right adjoint to the functor $- \otimes \mathcal{G} : \mathbf{Grph} \to \mathbf{Grph}$. Hence, the monoidal category $\mathbf{Grph} \stackrel{def}{=} (\mathbf{Grph}, \otimes, \mathcal{I})$ is closed.

Proof. For all graphs $\mathcal{H}$, there is a graph morphism $e_{\mathcal{H}} : [\mathcal{G}, \mathcal{H}] \otimes \mathcal{G} \to \mathcal{H}$ which is natural in $\mathcal{H}$ and universal from $- \otimes \mathcal{G}$ to $\mathcal{H}$, defined as follows:

$$
\begin{aligned}
e_{\mathcal{H}V}(m, x) &\stackrel{def}{=} m_E(x) \\
e_{\mathcal{H}E}(\alpha, y) &\stackrel{def}{=} \alpha_x : m_E(y) \to n_E(y) \quad \text{for } \alpha : m \Rightarrow n : \mathcal{G} \to \mathcal{H},\ y \in V(G) \\
e_{\mathcal{H}E}(m, a) &\stackrel{def}{=} m_E(a) \quad \text{for } m : \mathcal{G} \to \mathcal{H},\ a \in E(\mathcal{H})
\end{aligned}
$$

Showing that this is a graph morphism, and natural in $\mathcal{H}$ is fairly routine. To show that it is universal, we have to show that for all graphs $\mathcal{K}$ and graph morphisms $f : \mathcal{K} \otimes \mathcal{G} \to \mathcal{H}$, there is a unique $\hat{f} : \mathcal{K} \to [\mathcal{G}, \mathcal{H}]$, called the *currying* of f, such that

$$
\begin{tikzcd}
{[\mathcal{G}, \mathcal{H}] \otimes \mathcal{G}} \arrow[r, "e_{\mathcal{H}}"] & \mathcal{H} \\
\mathcal{K} \otimes \mathcal{G} \arrow[u, "\hat{f} \otimes \mathbf{1}_{\mathcal{G}}"] \arrow[ur, "f"'] &
\end{tikzcd}
$$

$\hat{f}$ is defined as follows: on vertices, it maps a morphism $m : \mathcal{K} \to \mathcal{H}$ to a a graph morphism $\hat{f}(m) : \mathcal{G} \to \mathcal{H}$ which itself is given by on the vertices by $\hat{f}(m)_V(x) \stackrel{def}{=} f_V(m, x)$ and on the edges by $\hat{f}(m)_E(e) \stackrel{def}{=} f_E(m, e)$; and on edges, $\hat{f}$ maps a transformation $\alpha : m \Rightarrow n : \mathcal{K} \to \mathcal{H}$ to a transformation $\beta : \hat{f}(m) \Rightarrow \hat{f}(n) : \mathcal{G} \to \mathcal{H}$, defined as $\beta_x \stackrel{def}{=} f_E(\alpha, x)$. The requirement that this is a graph morphism amounts to the universality (any other such graph morphism is equal to $\hat{f}$). $\square$

In the preceding, we have concentrated on the monoidal structure on the objects; of course, one has to show that $\otimes$ is in fact a functor. For two graph morphisms $f : \mathcal{G} \to \mathcal{G}', g : \mathcal{H} \to \mathcal{H}'$, $f \otimes g$ is a graph morphism $f \otimes g : \mathcal{G} \otimes \mathcal{H} \to \mathcal{G}' \otimes \mathcal{H}'$ which on the edges is defined is defined as follows (on the vertices, it is the obvious definition):

$$
\begin{aligned}
(f \otimes g)_E \langle x, e : y \to y' \rangle &\stackrel{def}{=} \langle f_V(x), e_E \rangle \\
(f \otimes g)_E \langle e : x \to x', y \rangle &\stackrel{def}{=} \langle f_E(e), y_V \rangle
\end{aligned}
$$

The internal hom $[G, -]$ on morphisms maps $f : \mathcal{H} \to \mathcal{K}$ to the postcomposition $f \cdot (-)$ with f, much like its analogue $\mathbf{Set}(X, -) : \mathbf{Set} \to \mathbf{Set}$ in the category of small sets.

1.5.2 Binary Relations and the category Rel

A *binary relation* R on a set X is a subset of the product $X \times X$, written (X, R). A relation morphism $f : (X, R) \to (Y, S)$ is a function $f : X \to Y$ which is monotone, i.e. for all $x, y \in X$, $xRy \Rightarrow (fx)S(fy)$, where xRy is the infix notation for $(x, y) \in R$ which will be used in the following. This gives the category **Rel** of relations and morphisms between them; it has, just as graphs, a cartesian product $R \times S$ and a tensor product $R \otimes S$ defined as follows:

$$\begin{aligned}
(X, R) \times (Y, S) &\stackrel{def}{=} (X \times Y, \{((x_1, y_1), (x_2, y_2)) \mid x_1 R x_2 \ \wedge \ y_1 S y_2\}) \\
(X, R) \otimes (Y, S) &\stackrel{def}{=} (X \times Y, \{((x_1, y_1), (x_2, y_2)) \mid \ (x_1 = x_2 \ \wedge \ y_1 S y_2) \vee \\
&\qquad\qquad\qquad\qquad\qquad\qquad\quad (x_1 R x_2 \ \wedge \ y_1 = y_2)\})
\end{aligned}$$

The unit for the cartesian product is given by the one-element all-relation $\mathbf{1} \stackrel{def}{=} (\{*\}, \{(*, *)\})$, and the unit for the tensor product, the one-element empty relation $\mathcal{I} \stackrel{def}{=} (\{*\}, \emptyset)$. We obtain two monoidal structures on **Rel**, and **Rel** is closed with respect to both of these, but with two different relations on the functions: given (X, R) and (Y, S), then the function space between them will be the set $[X, Y]$ of all functions from X to Y which are monotone w.r.t.. R and S. If we order the functions pointwise

$$f < g \Leftrightarrow \forall x \in X.\, fx\ S\ gx$$

then the relation $([X, Y], <)$ is the internal hom with respect to the tensor product. If on the other hand we order the functions implicational

$$f \succ g \Leftrightarrow \forall x, y \in X.\, xRy \Rightarrow fx\ S\ gy$$

then the relation $([X, Y], \succ)$ is the internal hom with respect to the cartesian product.

Closures

Given two relations (X, R) and (X, S), their *relational product* is defined as

$$(X, R; S) \stackrel{def}{=} (X, \{(x, z) \mid \exists y. xRy \wedge ySz\})$$

The *reflexive closure* of a relation (X, R) is defined as

$$(X, R)^{=} \stackrel{def}{=} (X, R \cup \{(x, x) \mid x \in X\})$$

and the *transitive closure* $(X, R)^{+}$ as the smallest relation closed under the relational product $(X, R) \subseteq (X, R; R)$. Both the reflexive and the transitive closure of a relation can be given by monads on the category **Rel**, the algebras being reflexive and transitive relations, respectively. They can be combined into a single monad, giving for a relation (X, R) its transitive-reflexive closure $(X, R)^{*}$.

Hence, there is the following system of adjunctions between the categories of reflexive, transitive and reflexive-transitive relations. A relation which is reflexive and transitive is also called a *preorder*, with **Pre** the category of preorders:

$$\begin{tikzcd}
\mathbf{Rel} \arrow[r, shift left=2, "(-)^="] \arrow[r, phantom, "\perp"] \arrow[d, shift left=2, "(-)^+"] \arrow[d, phantom, "\dashv"] & \mathbf{RRel} \arrow[l, hook, shift left=2] \arrow[d, shift left=2, "(-)^+"] \arrow[d, phantom, "\dashv"] \\
\mathbf{TRel} \arrow[u, hook, shift left=2] \arrow[r, shift left=2, "(-)^R"] \arrow[r, phantom, "\perp"] & \mathbf{Pre} \arrow[u, hook, shift left=2] \arrow[l, hook, shift left=2]
\end{tikzcd}$$

In this case, the two monads can be combined in both possible ways, making the diagram commute; in general, this is not the case (e.g. if we replace the reflexive by the symmetric closure above). The reflexive-transitive closure of a relation (X, R) is written as $(X, R)^*$, and satisfies the following property with respect to the relational product

$$(X, R \cup S)^* = (X, R^*; S^*)^* \tag{1.24}$$

Relations and Graphs

A relation (X, R) can be considered as a graph $G(R) \stackrel{def}{=} (X, R, \pi_1, \pi_2)$; hence a relation is equivalent to a graph with at most one edge between any two given vertices, and we can even say that **Rel** is a subcategory of **Grph**. On the other hand, given a graph $\mathcal{G}$, we obtain a binary relation $|\mathcal{G}| \subseteq V(\mathcal{G}) \times V(\mathcal{G})$ on the vertices of $\mathcal{G}$, defined by $(x, y) \in |\mathcal{G}| \Leftrightarrow \exists p : x \to y \in E(\mathcal{G})$. This extends to a functor $G : \mathbf{Rel} \to \mathbf{Grph}$, which is left adjoint to the inclusion $\mathbf{Rel} \hookrightarrow \mathbf{Grph}$, so **Rel** is a *reflective subcategory* of **Grph**.

Similarly, a preorder can be considered a category with at most one morphism between any two morphisms, with the identities and composition given by the reflexivity and the transitivity. The adjunction between **Rel** and **Grph** can be lifted to categories and preorders: for a category $\mathcal{C}$, the relation given by the underlying graph of $\mathcal{C}$ is a preorder, giving a functor $J : \mathbf{Cat} \to \mathbf{Pre}$ by $J(\mathcal{C}) \stackrel{def}{=} |U\mathcal{C}|$ which identifies all morphisms between the same source and target, which is left adjoint to the inclusion $I : \mathbf{Pre} \hookrightarrow \mathbf{Cat}$. Moreover, the counit of this adjunction is the identity

$$J(I(X)) = X \tag{1.25}$$

since any preorder X considered as a category will have at most one morphism between any two objects:

In summary we obtain the following system of commuting adjunctions:

$$
\begin{tikzcd}
\mathbf{Rel} \arrow[r, shift left=1.5, "(-)^*"] \arrow[r, phantom, "\perp"] \arrow[d, hook, shift right=1.5] \arrow[d, phantom, "\vdash"] & \mathbf{Pre} \arrow[l, hook', shift left=1.5] \arrow[d, hook, shift right=1.5] \arrow[d, phantom, "\vdash"] \\
\mathbf{Grph} \arrow[u, shift right=1.5, "|-|"'] \arrow[r, shift left=1.5, "\mathcal{F}"] \arrow[r, phantom, "\perp"] & \mathbf{Cat} \arrow[u, shift right=1.5, "J"'] \arrow[l, shift left=1.5, "U"]
\end{tikzcd}
$$

1.5.3 Small Categories and the Category Cat

In this section, we will investigate the internal structure of the category **Cat** of all small categories. The results in this section are more or less standard.

Epimorphisms and Monomorphisms in Cat

An *epimorphism* in **Cat** is a functor which is surjective on objects, and surjective under closure on morphisms[11], i.e. given $F : \mathcal{X} \to \mathcal{Y}$, for all $\beta \in \mathbf{Mor}_{\mathcal{Y}}$ there are $\alpha_1, \dots, \alpha_n \in \mathbf{Mor}_{\mathcal{X}}$ such that $\beta = F(\alpha_n) \cdot \ldots \cdot F(\alpha_1)$. A *monomorphism* in **Cat** is a functor which is injective on objects, and injective on morphisms (i.e. it is faithful); we call such functors injective in the following.

Closed Monoidal Structures on Cat

The cartesian product of two categories $\mathcal{X}, \mathcal{Y}$ is called a *product category*, and has as objects and morphisms pairs of objects and morphisms from $\mathcal{X}$ and $\mathcal{Y}$ (with the obvious pointwise definitions for identity and composition):

$$
\begin{aligned}
|\mathcal{X} \times \mathcal{Y}| &\overset{def}{=} |\mathcal{X}| \times |\mathcal{Y}| \\
\mathcal{X} \times \mathcal{Y}((x_1, y_1), (x_2, y_2)) &\overset{def}{=} \mathcal{X}(x_1, x_2) \times \mathcal{Y}(y_1, y_2)
\end{aligned}
$$

Further, **Cat** is cartesian closed: for any category $\mathcal{X}$, the functor $- \times \mathcal{X} : \mathbf{Cat} \to \mathbf{Cat}$ mapping $\mathcal{Y}$ to the product category $\mathcal{Y} \times \mathcal{X}$ has a right adjoint $[\mathcal{X}, -] : \mathbf{Cat} \to \mathbf{Cat}$ mapping $\mathcal{Y}$ to the functor category $[\mathcal{X}, \mathcal{Y}]$.

Cat also inherits the tensor product from **Grph**. For two categories, $\mathcal{X}$, $\mathcal{Y}$, their *tensor product* $\mathcal{X} \otimes \mathcal{Y}$ has the same objects as the product category, but as morphisms sequences of pairs of objects and morphisms $<(\alpha_1, \beta_1), \dots, (\alpha_n, \beta_n)>$ such that for $i = 1, \dots, n$, either $\alpha_i \in \mathcal{X}, \beta_i \in \mathcal{Y}(x_i, y_i)$ with $\beta_i \neq \mathbf{1}_{x_i}$ and if $i < n$ then $\alpha_{i+1} \in \mathcal{X}(x_{i+1}, y_{i+1}), \beta_{i+1} \in \mathcal{Y}$ with $\alpha_i = x_{i+1}$ and $y_i = \beta_{i+1}$, or the

[11] This does not mean it is full, as one might perhaps expect.

other way around, i.e. $\alpha_i \in \mathcal{X}(x_i, y_i), \beta_i \in \mathcal{Y}$ with $\alpha_i \neq \mathbf{1}_{x_i}$ and if $i < n$ then $\alpha_{i+1} \in \mathcal{X}, \beta_{i+1} \in \mathcal{Y}(x_{i+1}, y_{i+1})$ with $y_i = \alpha_{i+1}$ and $\beta_i = x_{i+1}$. The source of the path above is the source of (α_1, β_1), which is (α_1, x_1) if $\alpha_1 \in \mathcal{X}, \beta_1 \in \mathcal{X}(x_1, y_1)$, and (x_1, β_1) otherwise; the target is defined analogously. The identity is given by the empty sequence, and composition by concatenation and composition in $\mathcal{X}$ and $\mathcal{Y}$. This monoidal structure is closed: the functor $- \otimes \mathcal{X} : \mathbf{Cat} \to \mathbf{Cat}$ has a right adjoint. This is best explained in terms of the internal hom resulting from it: given two functors $F, G : \mathcal{X} \to \mathcal{Y}$, a *transformation* $\alpha : F \Rightarrow G$ between them is a family $\{\alpha_X : FX \to GX\}_{X \in \mathcal{X}}$ of arrows in $\mathcal{Y}$, indexed by the objects in $\mathcal{X}$. (In other words, a natural transformation without the naturality requirement 1.1). Taking the functors between $\mathcal{X}$ and $\mathcal{Y}$ as objects, we obtain the category of functors and transformations between them, denoted as $[\![\mathcal{X}, \mathcal{Y}]\!]$, and we have the adjunction $- \otimes \mathcal{X} \dashv [\![\mathcal{X}, -]\!]$.

In summary, we obtain the two closed monoidal categories $\mathbf{Cat}_C \stackrel{def}{=} (\mathbf{Cat}, \times, \mathbf{1})$ and $\mathbf{Cat}_S \stackrel{def}{=} (\mathbf{Cat}, \otimes, \mathbf{1})$.

Coproducts and Coequalizers in Cat

In order to construct colimits in **Cat**, using proposition 1.3.1 we need coproducts and coequalizers. The first is easy: the coproduct of two categories $\mathcal{C}$ and $\mathcal{D}$ is given by their disjoint union, the objects and morphisms of which are the disjoint union of the objects and morphisms of $\mathcal{C}$ and $\mathcal{D}$, respectively.

The construction of the coequalizer is a bit more involved. Given two functors $F, G : \mathcal{X} \to \mathcal{Y}$, their coequalizer in **Cat** is a category $\mathcal{Z}$ and a functor $Q : \mathcal{Y} \to \mathcal{Z}$,

$$\begin{tikzcd} \mathcal{X} \arrow[r, shift left, "F"] \arrow[r, shift right, "G"'] & \mathcal{Y} \arrow[r, "Q"] & \mathcal{Z} \end{tikzcd}$$

with $\mathcal{Z}$ defined as follows (see also [24, Chapter I.1]):

- Its *objects* are given by the coequalizer in **Set** of the two object functions $|F|, |G|$ of F and G respectively; in other words, the objects of $\mathcal{Y}$ quotiented by the equivalence closure $\equiv_O$ of the relation $\sim$ defined as

$$x \sim y \Leftrightarrow \exists z \in Y. Fz = x, Gz = y$$

- Its *morphisms* are given as follows: we define the graph $\mathcal{Z}_0$ which has

$$\begin{array}{lrcl} \text{Vertices:} & V(\mathcal{Z}_0) & \stackrel{def}{=} & |\mathcal{Y}|/{\equiv_O} \\ \text{Edges:} & E(\mathcal{Z}_0) & \stackrel{def}{=} & \{\alpha : [x] \to [y] \mid \alpha \in \mathcal{Y}(x, y)\} \end{array}$$

  The equivalence relation $\equiv_M$ on ${\mathcal{Z}_0}^*$ is defined as the smallest path congruence on ${\mathcal{Z}_0}^*$ such that

(i) for all $\alpha \in \mathcal{Y}(x,y), \beta \in \mathcal{Y}(y,z)$, $\texttt{<}\alpha, \beta\texttt{>} \equiv_M \texttt{<}\beta{\cdot}\alpha\texttt{>}$

(ii) for all $\alpha \in \mathbf{Mor}_{\mathcal{X}}$, $\texttt{<}F\alpha\texttt{>} \equiv_M \texttt{<}G\alpha\texttt{>}$

Then for $X, Y \in \mathcal{Z}$, the set of morphisms is

$$\mathcal{Z}(X,Y) \stackrel{def}{=} \{\alpha \in \mathcal{Z}_0{}^* \mid \mathtt{s}(\alpha) = X, \mathtt{t}(\alpha) = Y\}/{\equiv_M}$$

For an object $X \in \mathcal{Y}$, the identity on X is the equivalence class of the empty path $\mathbf{1}_X \stackrel{def}{=} [\mathtt{id}_X]$. The composition of two morphisms $[A], [B]$ is given by $[B]{\cdot}[A] \stackrel{def}{=} [A :: B]$. This is well-defined because $\equiv_M$ is a path congruence.

The functor $Q : \mathcal{Y} \to \mathcal{Z}$ maps objects to their equivalence classes, and morphisms to the equivalence class of the singleton sequence: for $x \in \mathcal{Y}$ and $\alpha \in \mathcal{Y}(x,y)$,

$$\begin{aligned} Q(x) &\stackrel{def}{=} [x] \\ Q(\alpha) &\stackrel{def}{=} [\texttt{<}\alpha\texttt{>}] \end{aligned}$$

1.6 Basic Principles of Term Rewriting

In this section, we will briefly recall those basic concepts of term rewriting which we will need in this thesis. For a more detailed introduction, we refer the reader to e.g. [41, 10].

1.6.1 Abstract Reduction Systems and Term Rewriting Systems

The following definition is slightly paraphrased from [41]:

Definition 1.6.1 An *abstract reduction system* is a structure $\mathcal{A} = \langle A, \{\to_\alpha\}_{\alpha\in I}\rangle$ consisting of a carrier set A and a family of binary relations $\to_\alpha \subseteq A \times A$, called reduction or rewrite relations.

Fix a countably infinite set V of variables.[12] A *substitution* is a map $\sigma : T_\Omega(V) \to T_\Omega(V)$ such that $\sigma(f(t_1, \ldots, t_n)) = f(\sigma t_1, \ldots, \sigma t_n)$ for all operations $f \in \Omega$. A *context* is a term $C \in T_{\Omega\uplus\{\Box\}}(V)$ with exactly one occurrence of a special constant $\Box$. A context is generally denoted by $C[\,]$. If $t \in T_\Omega(V)$, the result of substituting t for $\Box$ in a context $C[\,]$ is written $C[t]$.

[12] Which in [41] is actually taken to be part of the signature.

A *rewrite rule* $r : t \to s$ is given by two terms $t, s \in T_\Omega(V)$. A rewrite rule gives rise to *reduction steps* $C[\sigma(t)] \to_r C[\sigma(s)]$, for all contexts $C[\,]$, and substitutions σ.

Then for every term rewriting system $\Theta = (\Omega, R)$, given by a signature Ω and a set of rewrite rules R, there is an abstract reduction system (the *one-step reduction relation*) $(T_\Omega(V), \{\to_r\}_{r\in R})$.

The *many-step reduction relation* $\{\twoheadrightarrow_r\}$ is defined as the transitive-reflexive closure of $\{\to_r\}$, describing the reduction of term s to another term t in 0 or more steps. The one-step reduction relation $\to_R$ is the union of $\to_r$ for all $r \in R$ (and similarly $\twoheadrightarrow_R$). Further, the equivalence closure of $\twoheadrightarrow_R$, denoted $=_R$, is called the convertibility relation generated by R, and describes the equational theory generated by the term rewriting system, or equivalently, the theory of the rewrite rules considered as equations.

1.6.2 Commutativity, Confluence and Strong Normalization

Properties of term rewriting systems such as confluence and strong normalisation can be defined on the level of binary relations (see §1.5.2 above). This has the advantage that one can prove lemmas (such as the Hindley-Rosen lemma below) directly at this more abstract level.

Definition 1.6.2 (Commuting Relations) A binary relation $\to_R \subseteq X \times X$ *commutes* with a binary relation $\to_S \subseteq X \times X$ if for all $x, y, z \in X$ such that $x \to_R y$ and $x \to_S z$, there is a $u \in X$ such that

$$\begin{tikzcd}
 & x \arrow[dl, "R"'] \arrow[dr, "S"] & \\
y \arrow[dr, two heads, "S"'] & & z \arrow[dl, two heads, "R"] \\
 & u &
\end{tikzcd}$$

(where $\twoheadrightarrow_R$ is the reflexive-transitive closure of $\to_R$).

A binary relation is *confluent* if it commutes with itself. If above $y \to_{S^=} u$, $y \to_{R^=} u$ we say that $\to_R$ *subcommutes* with $\to_S$. If both $\to_R$ and $\to_S$ above are confluent, then commutativity means that their union is confluent as well:

Lemma 1.6.3 (Hindley-Rosen [71]) Given two confluent binary relations $\to_R \subseteq X \times X$, $\to_S \subseteq X \times X$ where $\to_R$ commutes with $\to_S$, then $\to_R \cup \to_S$ is confluent.

Definition 1.6.4 (Strong Normalization) A binary relation $\to_R \subseteq X \times X$ is called *strongly normalizing* (*terminating*) if there is no infinite sequence

$$x_1 \to_R x_2 \to_R x_3 \to_R \dots$$

Finally, the notions of confluence and strong normalization carry over to term rewriting systems: a term rewriting system $\Theta = (\Omega, R)$ is confluent if the one-step reduction relation $\to_R$ is confluent, and a term rewriting system $\Theta = (\Omega, R)$ is strongly normalizing (or terminating) if $\to_R$ is strongly normalizing.

A binary relation (or term rewriting system) which is both confluent and strongly normalizing is called *complete*.

1.7 Words and Languages

This section contains a few definitions and lemmas dealing with the algebra of words over a language. These are technical preliminaries for §3.2, so the reader may want to defer studying these definitions until that point.

Let $\mathcal{L}^*$ be the language of words over an alphabet $\mathcal{L}$. We write ε for the empty word in $\mathcal{L}^*$, and juxtaposition for the concatenation of words. Further, $\mathcal{L}^+$ is the sublanguage of non-empty words: $\mathcal{L}^+ \stackrel{def}{=} \mathcal{L}^* \setminus \{\varepsilon\}$. $\leq \subseteq \mathcal{L}^* \times \mathcal{L}^*$ is the *prefix ordering* on words: $w_1 \leq w_2$ iff there is $u \in \mathcal{L}^*$ such that $w_1 u = w_2$. $<$ is the *strict prefix ordering*: $w_1 < w_2$ iff there is $u \in \mathcal{L}^+$ such that $w_1 u = w_2$. The *length* of a word $w \in \mathcal{L}^*$ is defined as follows: $|\varepsilon| \stackrel{def}{=} 0$, $|jw'| = 1 + |w'|$ for $j \in \mathcal{L}, w' \in \mathcal{L}^*$

Lemma 1.7.1 Let $v, w, x, y \in \mathcal{L}^*$ such that $vw = xy$, then one of the following three is true: $v < x$, $x = v$, or $x < v$.

Proof. By induction on the length of w and a careful case distinction [47]. □

The notion of a decomposition as defined in the following, and the lemmas to go with it, seem not to be standard, hence we go into perhaps more detail than the substance of the material warrants.

A *decomposition* of a word $w \in \mathcal{L}^*$ is given by words $x, y, z \in \mathcal{L}^+$ such that $xyz = w$. We will need the following facts about decompositions:

Lemma 1.7.2 Given two decompositions $w = rst$ and $w = xyz$, at least one of the following six cases is true:

(i) there is $u \in \mathcal{L}^*$ s.t. $w = rsuyz$;

(ii) there is $u \in \mathcal{L}^*$ s.t. $w = xyust$;

(iii) there are $u, v \in \mathcal{L}^*$ s.t. $v \leq y$, $xv = r$, $vs = yu$ and $ut = z$;

(iv) there are $u, v \in \mathcal{L}^*$ s.t. $u \leq s$, $ru = x$, $uy = sv$ and $vz = t$;

(v) there are $u, v \in \mathcal{L}^*$ s.t. $ruyvt = w$;

(vi) there are $u, v \in \mathcal{L}^*$ s.t. $xusvz = w$.

In case (i) and (ii), we say s and y are *independent*; in case (iii) and (iv), we say they are *overlapping*; in case (v) we say s *contains* y, and in case (vi) that y *contains* s.

Proof. Using lemma 1.7.1, we do a case distinction on the strict prefix ordering:

1. $x < r$ then $\exists v' \in \mathcal{L}^*.xv' = r$, with the following sub-cases:
   (a) $xy \leq r$ then $\exists u' \in \mathcal{L}^*.xyu' = r$: case (ii) with $u \stackrel{def}{=} u'$;
   (b) $r < xy$ then $\exists u' \in \mathcal{L}^*.ru' = xy$, with the following two sub-cases:
      i. $xy \leq rs$ then $\exists w' \in \mathcal{L}^*.xyw' = rs$: case (iii) with $u \stackrel{def}{=} w', v \stackrel{def}{=} v'$;
      ii. $rs < xy$ then $\exists w' \in \mathcal{L}^*.rsw' = xy$: case (vi) with $u \stackrel{def}{=} v'$, $v \stackrel{def}{=} u'$ as given.
2. $x = r$ then we have the following sub-cases:
   (a) $y = s$: cases (iii) and (iv) with $u, v \stackrel{def}{=} \varepsilon$;
   (b) $y < s$ then $\exists u' \in \mathcal{L}^*.su' = y$: cases (iii) and (v) with $u \stackrel{def}{=} \varepsilon$, $v \stackrel{def}{=} u'$.
   (c) $s < y$ then $\exists u' \in \mathcal{L}^*.yu' = s$: cases (iv) and (vi) with $u \stackrel{def}{=} \varepsilon$, $v \stackrel{def}{=} u'$.
3. $r < x$: symmetric to the first case, covering cases (i), (iv) and (v).

□

We will below be interested in cases in which we can make the following useful simplification:

Lemma 1.7.3 Given two decompositions $w = rst$ and $w = xyz$ of w such that $||s| - |y|| \leq 1$, then cases (v) and (vi) are subsumed by cases (iii) and (iv). In other words, s and y are either independent or overlapping.

Proof. Assume that s contains y, then there are $u, v \in \mathcal{L}^*$ such that $ruyvt = w$, then $uyv = s$, and $|u| + |y| + |v| = |s|$, $|u| + |v| = |s| - |y| \leq 1$, hence $|u| = 0$ or $|v| = 0$. $|u| = 0$ implies $u = \varepsilon$, and this is covered by case (iii) with $v' \stackrel{def}{=} \varepsilon$, $u' \stackrel{def}{=} s$; similarly, $|v| = 0$ implies $v = \varepsilon$, which is covered by case (iv) with $v' \stackrel{def}{=} u$, $u' \stackrel{def}{=} \varepsilon$. □

Chapter 2

A Compositional Semantics for Term Rewriting Systems

Signatures and equational presentations (i.e. signatures together with sets of equations) can be modelled by monads on the category **Set** of all small sets. Here is how a monad $\mathsf{T} = \langle T, \eta, \mu \rangle$ on **Set** captures the way terms are built:

- For a set X, we can consider TX to be the term algebra (the set of terms built over the variables X);
- then the unit $\eta_X : X \to TX$ describes how to make elements of X into variables (i.e. terms);
- and the multiplication $\mu : TTX \to TX$ describes how to substitute terms for variables.

The monad laws mean that this substitution is associative, and that substituting a term into a variable, and a variable for itself, yields the identity (in an informal notation, $x[t/x] = t$ and $t[x/x] = t$). The naturality of η and μ mean that the process of taking an element of X to a variable has be uniform over all sets X (i.e. does not depend on the set X). It turns out this is all one needs to do universal algebra (see [53]).

The key issue of this chapter will be the generalization of this construction to "sets with structure", by endowing the set TX of terms with a reduction structure between its elements. We will now try to sketch what this structure could be before describing the difficulties we are likely to encounter during the generalization. This will be the main motivation for the rest of this chapter, which will deal with resolving the issues and questions raised in the following two paragraphs.

Sets With Structure

What do we mean by "set with structure"? The intuitive picture is of a set as collection of elements (denoting terms), then the structure consists of some sort of "arrows" in between them (denoting reductions). The exact definition depends on the kind of reduction structure we are interested in. In particular, we can distinguish between *one-step* and *many-step* reductions, and between *named* and *unnamed* reductions (with the latter, any two reductions between the same two terms are equal). Each of these structures corresponds to one concrete instantiation of the abstract theory.

The Monad Construction

The generalization of the monad construction is more subtle than a cursory first glance might suggest[1], because we have to make explicit some of the implicit properties of **Set** used in the modelling of the equational presentations.

First, in the category **Set** we can completely forget about the internal structure of the objects, since every set X is determined by the morphisms into it, which in turn form a set. Abstracting from the properties of objects is a key concept in category theory, so if we generalize our objects to "sets with structure", we have to generalize the set of morphisms (*hom-sets* in category theory parlance) to sets-with-structure of morphisms as well; this means our construction has to be an *enriched* monad in the sense of §1.4.

The second realization is that terms are just composed operations. Consequently, contexts (sets of variables to build terms from) and arities (the number of arguments of an operation) are essentially the same, and have to modelled by the same objects of the base category.[2] In **Set**, these are finite sets (as contexts) or natural numbers or finite ordinals (as arities). In categories of sets-with-structure, this rôle will be taken, roughly speaking, by finite sets-with-structure. This means the arities of operations can be more than just natural numbers, and will have interesting ramifications on the rewrite rules, since it will allow rewrites between variables.

Compositionality

Giving a "semantics" to a formal system such as a term rewriting systems or equational specification amounts to constructing a mapping from the syntactic

[1] In particular, it is not sufficient to merely change the base category of the monad.

[2] One particular point here is that the arities are modelled by objects of the base category in the first place, something which may not be clear *a priori*.

presentation of these systems into a mathematical structure such as relations (for term rewriting systems) or sets and operations on them (for equational presentations). One then aims to understand the formal system by reflecting the behaviour of the predefined structure back to the formal system; e.g. an equation which holds in all algebras satisfying an equational presentation follows from the equations of the presentation.

A semantics is called *compositional* if the mapping preserves structuring operations, such as the disjoint union. The structuring operations, and their definitions on both the syntactic and the semantic representations will be the focus of the next chapter, but in this chapter we will show that our semantics satisfies a property which makes it preserve most structuring operations; namely, it is a left adjoint.

Structure of this Chapter

The rest of this chapter is structured as follows:

- In §2.1, we will recall the categorical treatment of universal algebra by monads on the category **Set** of all small sets. Besides motivation, this will provide us with the objects of sets with structure.

- We will then in §2.2 present the basic principles of the theoretical foundation of this work as developed by Kelly and Power in [39]. This section is optional, and can be skipped by readers feeling uneasy about enriched category theory.

- In §2.3, we will instantiate the general theory of §2.2 to our particular needs, explaining the general notions (as far as needed) in the process. We will then develop a term construction, ultimately yielding a monad on the category **Pre** of preorders modelling a term rewriting system.

- In §2.4, we will develop some properties of the monad as needed in the later chapters.

- Finally, in §2.5, we will show the semantics to be compositional in the sense that it is left adjoint; the precise meaning of this, and the structuring operations themselves will be discussed in the next chapter.

The main section of this chapter is §2.3; the two sections before are motivation and definitions leading up to that section. The hurried reader may choose to restrict his attention in these sections to definition 2.1.2 and §2.1.2, which introduce notations used extensively in the following.

2.1 Signatures and Equational Presentations

In this section, we recall the well known notions of universal algebra and their categorical treatment, such as the definitions of signatures, signature morphisms, algebras and algebra homomorphisms (the "semantic" treatment of signatures) in §2.1.1, and the construction of the term algebra (the "syntactic" treatment) in §2.1.2. The fact that the term algebra is free over other algebras of the same signature corresponds to an adjunction between the category of algebras for the signature, and the category of all small sets. This naturally leads to the modelling a signature by a monad on **Set**. In §2.1.3 we will develop the notion of the internal signature of a monad; the mapping of a signature to the modelling monad, and of a monad to its internal signature will be adjoint. Finally, as an aside in §2.1.4 we will sketch the modelling of equational presentations.

The reader familiar with e.g. the material in the first three chapters of [12] will be able to skip §2.1.1 and §2.1.2, but should have a brief glance at definitions 2.1.1 and 2.1.2 to pick up our notation.

2.1.1 Signatures and Algebras

Signatures and Signature Morphisms

Signatures or operator domains are sets of operators equipped with an arity. Morphisms are maps between them respecting these arities; and signatures and the morphisms between them give rise to the category **Sig**.

Definition 2.1.1 (Signature) A *finitary signature* or *operator domain* Ω is given by a map $\Omega : \mathbb{N} \to \mathbf{Set}$, giving for a natural number $n \in \mathbb{N}$ the set $\Omega_n \stackrel{def}{=} \Omega(n)$ of operations of *arity* n.

In the literature, the name operator domain is mainly used for the traditional single-sorted case [8, 53], and signature for the many-sorted case used in the context of algebraic specifications [12]. We will stick to the second term. Also, normally signatures are defined "the other way around", as a set of operations Ω together with a function $ar : \Omega \to \mathbb{N}$ assigning an arity to each operation. This is equivalent to definition 2.1.1 in the sense that given a signature as $\Omega' : \mathbb{N} \to \mathbf{Set}$, one can always define a set of operators Ω together with an arity function $ar : \Omega \to \mathbb{N}$, and vice versa, but definition 2.1.1 allows an easier development of the theory.

If the arities of the operations are not restricted to finite ordinals, one obtains *infinitary signatures*. Using monads, these can also be treated in much the same way as the finitary ones[3], but the resulting monads do not enjoy some of the properties needed later on (see section 2.4.1). Moreover, they do not seem of great practical relevance; hence, we restrict ourselves to finitary signatures, and in the following, mean finitary signatures whenever we merely say signature.

Given two signatures Ω and Σ, a *signature morphism* $\sigma : \Omega \to \Sigma$ is a family of maps $\{\sigma_n : \Omega(n) \to \Sigma(n)\}_{n \in \mathbb{N}}$. This gives the category **Sig** with signatures as objects, and signature morphisms as morphisms. More abstractly, if we consider $\mathbb{N}$ as a discrete category, **Sig** is the functor category $[\mathbb{N}, \mathbf{Set}]$; a signature morphism is a natural transformation, but since $\mathbb{N}$ is discrete, the naturality condition is vacuous.

When writing signatures, we will use subscripts to denote the arities; e.g. writing $\Omega = \{\mathtt{F}_1, \mathtt{K}_2\}$ means there is a signature $\Omega = \{1 \mapsto \{\mathtt{F}\}, 2 \mapsto \{\mathtt{K}\}\}$.

Algebras for a Signature

Given a signature Ω, a Ω-algebra $A = (A_S, \{\omega_A\}_{\omega \in \Omega})$, is given by a set A_S, called the *carrier set* and, for all operations $\omega \in \Omega_n$, a function $\omega_A : A^n \to A$.

Given two Ω-algebras A, B, a *homomorphism* between them is given by a map $f : A_S \to B_S$ respecting the operations in Ω: for all $\omega \in \Omega_n$, and for all $a_1, \ldots, a_n \in A_S$,

$$\omega_B(f(a_1), \ldots, f(a_n)) = f(\omega_A(a_1, \ldots, a_n)) \tag{2.1}$$

Alternatively, this requirement can be written (without using elements of A_S) as $\omega_B \cdot f^n = f \cdot \omega_A$ (where f^n is the n-fold product of f with itself, rather than the n-fold composition of f with itself). We obtain, for a signature Ω, the category Ω-**Alg** of Ω-algebras.

2.1.2 The Term Algebra

Very crudely spoken, a term is a syntactic entity generated by the signature. Terms have the structure of an algebra, and moreover a very important one (it is "freely generated", meaning there is exactly one mapping to any other algebra). Another way to regard terms is as composed operations; then the set of all terms is the closure of the operations under composition.

The Term Algebra Construction

Given a set X of variables, the term algebra $T_\Omega(X)$ is the smallest set which contains all variables $x \in X$ and is closed under application of the operations in

[3]This is one of the advantages of the monad treatment.

Ω. Put formally (this also fixes the notation we use for terms in the following)[4]

Definition 2.1.2 (Term Algebra) Given a signature Ω and a set X, the *term algebra* $T_\Omega(X)$ is the smallest set satisfying the following implications:

$$\frac{}{\text{'}x \in T_\Omega(X)}\; x \in X \qquad\qquad \frac{t_1, \ldots, t_n \in T_\Omega(X)}{\omega(t_1, \ldots, t_n) \in T_\Omega(X)}\; \omega \in \Omega_n$$

The *set of variables* of a term $t \in T_\Omega(X)$ is defined inductively as follows:

$$\begin{aligned} var(\text{'}x) &\stackrel{def}{=} \{x\} \\ var(\omega(t_1, \ldots, t_n)) &\stackrel{def}{=} \bigcup_{i=1,\ldots,n} var(t_i) \end{aligned}$$

We are now going to extend construction 2.3.4 to a monad. In order to prove the monad properties, we need a structural induction scheme.

Structural induction

To prove that a predicate holds for all terms, a method called *structural induction* is used:

Proposition 2.1.3 *For a predicate R on $T_\Omega(X)$, we write $t \models R$ (R holds for t) if $t \in R$ for $t \in T_\Omega(X)$. Then $t \models R$ for all $t \in T_\Omega(X)$ (R holds for all terms) if the following two implications hold:*

$$\begin{aligned} x \in X &\Rightarrow \text{'}x \models R \\ \omega \in \Omega_n, t_1 \models R, \ldots, t_n \models R &\Rightarrow \omega(t_1, \ldots, t_n) \models R \end{aligned}$$

The first of these is called the induction base*, the second the* induction step.

Proof. This is proven by reducing it to natural induction. For the details, see e.g. [12, pg. 19]. We define the size of a term as the number of operation symbols in it; then for a predicate R define a predicate q on the natural numbers such that $q(n)$ holds iff. R holds for all terms of size n or less; then show that if the two implications above hold, $q(n)$ holds for all $n \in \mathbb{N}$, and hence for all terms, by natural induction on n. □

[4] The reader may wonder about the reason for notation $\text{'}x$: later on we will consider terms built over terms, and we will have to distinguish terms like $\text{'}x$ in $T_\Omega(X)$ and $\text{''}x$ in $T_\Omega(T_\Omega(X))$, or even $\text{'}\mathtt{K}(\text{'}x)$ and $\mathtt{K}(\text{''}x)$ in $T_\Omega(T_\Omega(X))$.

Freeness of the Term Algebra

The term algebra is free in the class of all algebras over a set X, meaning there is exactly one homomorphism from it to any other algebra A once the assignment of the variables in X has been fixed. This homomorphism lets us evaluate terms from $T_\Omega(X)$ in A.

Lemma 2.1.4 Given a set X, an Ω-algebra A, and a map $\sigma : X \to A_S$ (called an assignment of the variables in X), there is a unique map $\overline{\sigma} : T_\Omega(X) \to A_S$ which is an algebra homomorphism, called the *homomorphic extension* of σ, satisfying

$$\begin{tikzcd} X \arrow[r, "\eta_X"] \arrow[dr, "\sigma"'] & T_\Omega(X) \arrow[d, "\overline{\sigma}"] \\ & A_S \end{tikzcd} \tag{2.2}$$

diagram 2.2, where η_X is the inclusion of the variables into the term algebra, defined as

$$\eta_X(x) \stackrel{def}{=} \text{'}x$$

Proof. $\overline{\sigma}$ is defined as follows:

$$\begin{aligned} \overline{\sigma}(\omega(t_1, \ldots, t_n)) &\stackrel{def}{=} \omega_A(\overline{\sigma}(t_1), \ldots, \overline{\sigma}(t_n)) \\ \overline{\sigma}(\text{'}x) &\stackrel{def}{=} \sigma(x) \end{aligned}$$

The second of these equations makes diagram 2.2 commute. Any homomorphism has to satisfy equation 2.1 which is the first equation. Hence $\overline{\sigma}$ is uniquely defined.

□

In fact, the inclusion η_X is the unit of an adjunction between the category of all sets, and the category of all Ω-algebras, given by

$$F_\Omega \dashv U_\Omega : \mathbf{Set} \to \Omega\text{-}\mathbf{Alg} \tag{2.3}$$

where the left adjoint F_Ω maps any set X to the term algebra $T_\Omega(X)$, and the right adjoint U_Ω maps any Ω-algebra A to its carrier set A_S. The uniqueness of the homomorphic extension is exactly the universal property of the unit of the adjunction.

A Signature is Modelled by a Monad

Like every adjunction, this gives rise to a monad which as argued in the introduction exactly captures the way terms are built. We will first describe the left adjoint F_Ω in more detail:

Definition 2.1.5 The functor $F_\Omega : \mathbf{Set} \to \Omega\text{-}\mathbf{Alg}$ maps a set X to $T_\Omega(X)$, and a map $f : X \to Y$ in $\mathbf{Set}$ to its *lifting* $f^* : T_\Omega(X) \to T_\Omega(Y)$, defined as follows:

$$\begin{aligned} f^*(\omega(t_1, \dots, t_n)) &\stackrel{def}{=} \omega(f^*(t_1), \dots, f^*(t_n)) \quad \text{where } \omega \in \Omega_n \\ f^*('x) &\stackrel{def}{=} 'fx \end{aligned}$$

That f^* is an Ω-homomorphism (i.e. respects the operations) is obvious. That the assignment $f : X \to Y$ to $f^* : T_\Omega(X) \to T_\Omega(Y)$ is functorial (preserves composition and identities) is shown by routine structural induction.

The counit of the adjunction is a natural transformation $\varepsilon_A : F_\Omega(U_\Omega(A)) \to A$ for every Ω-algebra A, which can be thought of as the evaluation of terms t in the algebra A. It is defined as follows:

$$\begin{aligned} \varepsilon_A('x) &\stackrel{def}{=} x \\ \varepsilon_A(\omega(x_1, \dots, x_n)) &\stackrel{def}{=} \omega_A(\varepsilon_A(x_1), \dots, \varepsilon_A(x_n)) \end{aligned} \tag{2.4}$$

Proposition 2.1.6 *Every signature Ω gives rise to a monad $\mathsf{T}_\Omega \stackrel{def}{=} \langle T_\Omega, \eta, \mu\rangle$ on* $\mathbf{Set}$*, where T_Ω maps X to the term algebra $T_\Omega(X)$, and $f : X \to Y$ to its lifting $f^* : T_\Omega(X) \to T_\Omega(Y)$. The multiplication is given by $\mu \stackrel{def}{=} U_\Omega \varepsilon_{F_\Omega}$, or more explicitly*

$$\begin{aligned} \mu_X(\omega(t_1, \dots, t_n)) &= \omega(\mu_X(t_1), \dots, \mu_X(t_n)) \\ \mu_X('x) &= x \end{aligned}$$

This monad actually subsumes the adjunction 2.3, because one can show that the Eilenberg-Moore-category of algebras of this monad and the category Ω-**Alg** of Ω-algebras are isomorphic; hence the categorical notion of an algebra subsumes the notion of an Ω-algebra. This even works for equational presentations, so "finitary universal algebra is the study of finitary monads on **Set**" [53, pg. 42].

Finitariness

The monad T_Ω satisfies an important continuity condition: it is *finitary*. To understand this condition, consider the term algebra $T_\Omega(X)$ over an infinite set

X. Since every operation $\omega \in \Omega_n$ can only take finitely many arguments, every term $t \in T_\Omega(X)$ can only contain finitely many variables from X; and hence, instead of building the term algebra over the infinite set X, we can also build the term algebras over all finite subsets X_0 of X and take the union of these:

$$T_\Omega(X) = \bigcup_{X_0 \subseteq X} T_\Omega(X_0)$$

Categorically, we say that the set X is the colimit of the directed diagram of its finite subsets; and above equation means that the functor T_Ω has to preserve directed colimits or in other words is finitary.

The monad T_Ω arising from a signature even is strongly finitary, since the absence of equations means precisely that the action T_Ω preserves coequalizers and hence weakly filtered colimits (see lemma 1.3.7), whereas the monad $\mathsf{T}_{(\Omega,E)}$ arising from an equational presentation will not preserve coequalizers and hence be merely finitary.

That T_Ω is finitary will be proven in greater detail in §2.4.1 where the present case is treated as a corollary.

2.1.3 The Internal Language of a Monad

Not only does every signature give rise to a monad, a monad gives rise to a signature as well. In analogy to the situation with the simply typed λ-calculus and cartesian closed categories [45], this signature is called the *internal language* of the monad.

Internal Signatures

Definition 2.1.7 (Internal Signature) The *internal signature* of a finitary monad $\mathsf{S} = \langle S, \eta, \mu \rangle$ on **Set** is given by

$$\Sigma(\mathsf{S})(n) \stackrel{def}{=} \bigcup_{card(X)=n} S(X)$$

The unit and multiplication of the monad give the infrastructure to talk about terms. Specifically, an operation f of arity n is given by an element $f \in SX$ with $card(X) = n$. $card(X) = n$ means that X is isomorphic to n copies of the one-element set $\mathbf{1}$. A map $\mathbf{1} \to Z$ represents exactly one element of Z, hence a map $X \to Z$ corresponds to an n-tuple of elements of Z under the following bijection:

$$\mathbf{Set}(X, Z) \cong \mathbf{Set}(\coprod_n \mathbf{1}, Z) \cong \prod_n \mathbf{Set}(\mathbf{1}, Z) \cong \prod_n Z \tag{2.5}$$

Given $t_1, \ldots, t_n$ in Z, we use the notation $[t_1, \ldots, t_n]$ to denote the morphism from X to Z given by the previous bijection. If these n elements are operations themselves (i.e. $Z = SY$ for some Y), the *composition of the operation* f with these operations is given by the image of f under the following two morphisms:

$$SX \xrightarrow{S[t_1, \ldots, t_n]} SSY \xrightarrow{\quad \mu_Y \quad} SY$$

Alternatively, if we think of $f, t_1, \ldots, t_n$ as terms, this defines the *substitution of the variables* from X in f with the terms $t_1, \ldots, t_n$. Hence, both the substitution of variables and the composition of operations are modelled by the same construction. By slight abuse of notation, we write $f[f_1, \ldots, f_n]$ for this substitution.

The Adjunction $F \dashv U$

The mappings of a signature Ω to the monad T_Ω, and of a monad S to its internal signature $\Sigma(\mathsf{S})$ form an adjunction between the category of signatures and the category of finitary monads on **Set**. This adjunction between the category of syntactic and semantic presentations justifies our calling the semantics compositional: structuring operations like the coproduct are colimits, and a left adjoint functor preserves these. We will further elaborate on this point below.

We are now going to construct this adjunction. The category **Sig** has been defined above and the category $\mathbf{Mon}_{Fin}(\mathbf{Set})$ has been given in §1.3.6 on page 17, so we first need to extend the two maps to proper functors:

Definition 2.1.8 The functor $F : \mathbf{Sig} \to \mathbf{Mon}_{Fin}(\mathbf{Set})$ maps a signature Ω to the monad T_Ω, and a signature morphism $\sigma : \Omega \to \Sigma$ to its *lifting*, the monad morphism $\widehat{\sigma} : T_\Omega \Rightarrow T_\Sigma$, which is defined pointwise for $X \in \mathbf{Set}$ as follows:

$$\begin{aligned} \widehat{\sigma}_X(e(t_1, \ldots, t_n)) &\stackrel{def}{=} (\sigma e)(\widehat{\sigma}_X(t_1), \ldots, \widehat{\sigma}_X(t_n)) \\ \widehat{\sigma}_X(\text{'}x) &\stackrel{def}{=} \text{'}x \end{aligned}$$

The functor $U : \mathbf{Mon}_{Fin}(\mathbf{Set}) \to \mathbf{Sig}$ maps a finitary monad $\mathsf{T} = \langle T, \eta, \mu \rangle$ to its internal signature $\Sigma(\mathsf{T})$, and a monad morphism $\sigma : \mathsf{T} \Rightarrow \mathsf{S}$ to a signature morphism $U\sigma : \Sigma(\mathsf{T}) \to \Sigma(\mathsf{S})$ defined by

$$(U\sigma)_n(f) \stackrel{def}{=} \sigma_X(f) \text{ for } f \in TX \text{ with } card(X) = n \tag{2.6}$$

It is easy to see that $\widehat{\sigma}$ is natural in X and a monad morphism. $U\sigma$ is a signature morphism by construction. We further have to show the functoriality of F and U, i.e. preservation of identity and composition. For F, this means that given a signature Ω, $\widehat{\mathbf{1}_\Omega} = id_{\mathsf{T}_\Omega}$, which is proven by showing $\widehat{\mathbf{1}_\sigma}(t) = t$ for all

$X \in \mathbf{Set}$ and $t \in T_\Omega(X)$ by (an easy) structural induction on t, and that given two signature morphisms $\sigma : \Omega \to \Sigma, \tau : \Sigma \to \Sigma'$, $\widehat{\tau}\cdot\widehat{\sigma} = \widehat{\tau\cdot\sigma}$, which is proven by showing that $\widehat{\tau}(\widehat{\sigma}(t)) = \widehat{\tau\cdot\sigma}(t)$ for all $X \in \mathbf{Set}$ and $t \in T_\Omega(X)$. For U, we have to show that $U(id_{\mathsf{T}}) = \mathbf{1}_{\mathcal{L}(\mathsf{T})}$, which follows just by definition ($id_{\mathsf{T},X}(f) = f$ for all $X \in \mathbf{Set}, t \in TX$ in equation 2.6), and that $U(\tau\cdot\sigma) = U\tau\cdot U\sigma$, which again follows from equation 2.6.

Lemma 2.1.9 The two functors F and U form an adjunction $F \dashv U : \mathbf{Sig} \to \mathbf{Mon}_{Fin}(\mathbf{Set})$.

Proof. We show adjointness by constructing a unit $\upsilon_\Omega : \Omega \to \Sigma(\mathsf{T}_\Omega)$ which is universal from Ω to U.

The unit is a signature morphism $\upsilon_\Omega : \Omega \to UT_\Omega$ which is defined as $\upsilon_{\Omega,n}(\omega) \stackrel{def}{=} \omega(\text{'}0, \ldots, \text{'}n-1)$ (with the natural number n considered as the ordinal $n = \{0, \ldots, n-1\}$). Clearly υ is natural in Ω. To show that it is universal from Ω to U, we have to show that given any monad $\mathsf{S} = \langle S, \zeta, \xi\rangle$, and a signature morphism $\nu : \Omega \to \Sigma(\mathsf{S})$, there is a unique monad morphism $!_\nu : \mathsf{T}_\Omega \Rightarrow \mathsf{S}$ such that

$$(U!_\nu)\cdot\upsilon_\Omega = \nu \tag{2.7}$$

which we define as follows (where $\omega \in \Omega_n$, hence $\nu(\omega) \in SY$ with $card(Y) = n$):

$$\begin{aligned} !_{\nu,X}(\omega(t_1, \ldots, t_n)) &\stackrel{def}{=} \xi_X\cdot S[!_{\nu,X}(t_1), \ldots, !_{\nu,X}(t_n)](\nu(\omega)) \\ !_{\nu,X}(\text{'}x) &\stackrel{def}{=} \zeta_X(x) \end{aligned}$$

This definition is unique, since the second line makes $!_\nu$ satisfy equation 2.7, and the first line makes it a monad morphism. □

The counit of this adjunction consists of monad morphisms $\varepsilon_T : T_{\Sigma(\mathsf{T})} \Rightarrow \mathsf{T}$. We can think of ε_T as evaluating terms built over the internal signature of T in T, and we say that the monad T admits an equation $t_1 = t_2$ if $\varepsilon_T(t_1) = \varepsilon_T(t_2)$, leading to the concept of an internal language consisting of the internal signature and the set of equations admitted by the monad (see below). The counit is given by $\varepsilon_T = !_{\mathbf{1}_T}$, or more explicitly:

$$\begin{aligned} \varepsilon_{T,X}(f(t_1, \ldots, t_n)) &= \mu_X\cdot T[\varepsilon_{T,X}(t_1), \ldots, \varepsilon_{T,X}(t_n)](f) \\ \varepsilon_{T,X}(\text{'}x) &= \eta_X(x) \end{aligned}$$

We will below need to refer to this being natural in T, so let us conclude by spelling out what this means: if we have a monad morphism $\sigma : \mathsf{T} \Rightarrow \mathsf{S}$, we have for all sets X:

$$\begin{tikzcd} T_{\Sigma(\mathsf{T})}X \arrow[r, "\varepsilon_{T,X}"] \arrow[d, "\widehat{U\sigma}_X"'] & TX \arrow[d, "\sigma_X"] \\ T_{\Sigma(\mathsf{S})}X \arrow[r, "\varepsilon_{S,X}"'] & SX \end{tikzcd} \tag{2.8}$$

The Internal Language at Work

We will below need the property that lifted morphisms commute with the substitution, i.e. given $g : Y \to Z$, $t_1, \ldots, t_n \in TY$, and $t \in TX$ with $card(X) = n$, then

$$g^*(t[t_1, \ldots, t_n]) = t[g^*(t_1), \ldots, g^*(t_n)] \tag{2.9}$$

This can be proven by an easy structural induction on the term t, but we can prove it directly and more elegantly in the internal signature. Equation 2.9 translates as

$$g^*(\mu_Y(T[f_1, \ldots, f_n](t))) = \mu_Z(T[g^*t_1, \ldots, g^*t_n](t))$$

By applying g^* to both sides of bijection 2.5 we get $[g^*t_1, \ldots, g^*t_n] = g^*[t_1, \ldots, t_n]$. The proof then is given by diagram 2.10, where triangle (1) commutes because T is a functor preserving composition (recall that $g^* = Tg$), and square (2) commutes because μ is a natural transformation.

$$\begin{tikzcd} TX \arrow[r, "{T[t_1, \ldots, t_n]}"] \arrow[dr, "{T[g^*t_1, \ldots, g^*t_n]}"'] \arrow[dr, phantom, "(1)"] & TTY \arrow[r, "\mu_Y"] \arrow[d, "g^{**}"] \arrow[dr, phantom, "(2)"] & TY \arrow[d, "g^*"] \\ & TZ \arrow[r, "\mu_Z"'] & TZ \end{tikzcd} \tag{2.10}$$

2.1.4 Equational Presentations

We have shown that every signature can be modelled by a finitary monad on the category **Set**. In fact, monads are more general and can be used to model *equational presentations*, which consist of a signature and a set of equations on the derived terms. We shall briefly review this material, but leave the reader to consult the standard references [69, 53, 52] for more details.

Definition 2.1.10 (Equations and Equational Presentations) Let Ω be a signature. A (finitary) Ω-*equation* is of the form $X \vdash t = s$ where X is a (finite) set and $t, s \in T_\Omega(X)$. An *equational presentation* $\mathcal{A} = (\Omega, E)$ consists of a signature Ω and a set E of Ω-equations.

Given an equational presentation $\mathcal{A} = (\Omega, E)$ one defines the relations $\sim_X$ on the set of terms $T_\Omega(X)$ as the least equivalence relation generated by

$$\frac{(X \vdash t = s) \in E \quad \theta : X \to T_\Omega(Y)}{\overline{\theta}(t) \sim_Y \overline{\theta}(s)}$$

$$\frac{f \in \Omega_n \quad t_i \sim_X u_i \text{ for } i = 1, \ldots, n}{f(t_1, \ldots, t_n) \sim_X f(u_1, \ldots, u_n)}$$

where $\overline{\theta}$ is the homomorphic extension (see lemma 2.1.4) of θ, i.e. the function substituting $\theta(x)$ for each variable x in a term. The term algebra construction generalizes from signatures to equational presentations by mapping a set X to the term algebra $T_\Omega(X)$ quotiented by the equivalence relation $\sim_X$

$$T_\mathcal{A}(X) = T_\Omega(X)/\sim_X$$

Lemma 2.1.11 Given an equational presentation $\mathcal{A}$, the map $X \mapsto T_\mathcal{A}(X)$ extends to a finitary monad $\mathsf{T}_\mathcal{A}$ on the category **Set**.

Proof. Essentially the same as T_Ω proposition 2.1.6, except that one must ensure that provability is respected. See any of [69, 52, 53] for more details. □

An Ω-algebra A *admits* an Ω-equation $X \vdash t = s$ if for all $\sigma : X \to A_S$, t and s evaluate to the same value in A: $\overline{\sigma}(t) = \overline{\sigma}(s)$. For an equational presentation $\mathcal{A} = (\Omega, E)$, an $\mathcal{A}$-algebra is an Ω-algebra admitting all equations in E. We obtain the category of $\mathcal{A}$-algebras, and the term algebra $T_\mathcal{A}(X)$ is free in all algebras over X (lemma 2.1.4).

We can further extend §2.1.3 to equational presentations:

Definition 2.1.12 (Internal Language) The internal language of a finitary monad $\mathsf{T} = \langle T, \eta, \mu \rangle$ on **Set** is the equational presentation

$$\mathcal{L}(\mathsf{T}) \stackrel{def}{=} (\Sigma(\mathsf{T}), \mathcal{E}(\mathsf{T}))$$

where $\mathcal{E}(\mathsf{T})$ is the set of equations admitted by the monad T, defined as

$$\mathcal{E}(\mathsf{T}) \stackrel{def}{=} \{X \vdash t = s \mid X \in \mathbf{Set}_{\mathrm{fp}}, \varepsilon_{T,X}(t) = \varepsilon_{T,X}(s)\}$$

One can then extend the adjunction from lemma 2.1.9 to an adjunction between the category of equational presentations, and finitary monads on **Set**, but we shall refrain here from doing so.

As a side remark for the interested reader, we note that Birkhoff's theorem for equational presentations, which gives sufficient and necessary conditions for a set of Ω-algebras to be the set of algebras for an equational presentation finds a categorical counterpart in Beck's theorem [52, pg. 147] (generalized in [39, Theorem 2.4]), which gives sufficient and necessary conditions for a category to be monadic over **Set**, but can be generalized to base categories other than **Set**.

2.2 Basic Principles of Enriched Monad Theory

$\boxed{!}$ Categories

As has been pointed out at the beginning of this chapter, the main point of the construction of the semantics as a monad from a categorical point of view is that it has to be *enriched* over the category $\mathcal{V}$ of sets-with-structure. In this section, we will present the basic principles of the theoretical foundations of this work: the theory of enriched monads as developed by Kelly and Power in [39]. Readers who feel put off by the very first sentence of the following paragraph should probably skip this section and continue with §2.3, which is the application of this (rather abstract) theory, and will be understandable without this section.

This general theory deals with finitary monads on a locally finitely presentable $\mathcal{V}$-category $\mathcal{A}$ (where $\mathcal{V}$ is lfp as a closed category [37], meaning that $\mathcal{V}_0$ is lfp as an ordinary category, the tensor product preserves finite presentability and the unit I is fp.) Further, $\mathcal{A}$ is *small* in the sense that there is a small set $\mathcal{N}$ (regarded as a discrete category) of objects representing isomorphism classes of fp objects of $\mathcal{A}$, with the inclusions $I : \mathcal{N} \hookrightarrow \mathcal{A}_{\mathrm{fp}}$, $J : \mathcal{A}_{\mathrm{fp}} \hookrightarrow \mathcal{A}$.

In this abstract setting, a *signature* is a functor $B : \mathcal{N} \to \mathcal{A}$, which for all $b \in \mathcal{N}$ gives the $\mathcal{A}$-object of b-ary operations. For such a signature, one constructs a monad corresponding to the term algebra as the free monad on the left Kan extension of B along JI. More explicitly, FB is given as the colimit of the following sequence of functors:

$$\begin{aligned}
FB &\stackrel{\mathit{def}}{=} \mathop{\mathit{colim}}_{n<\omega} S_n \\
S_0 &\stackrel{\mathit{def}}{=} JI \\
S_{n+1}(c) &\stackrel{\mathit{def}}{=} c + \sum_{e\in\mathcal{N}} \mathcal{A}(e, S_n c) \otimes Be
\end{aligned} \tag{2.11}$$

We can think of $S_{n+1}(X)$ as the terms of depth $n+1$, which are either variables, or constructed from operations of arity c applied to c-tuples of terms of depth n.

On the other hand, we can precompose every finitary monad with IJ to obtain a functor $\mathcal{N} \to \mathcal{A}$; and there is an (ordinary) adjunction

$$\begin{tikzcd} {[\mathcal{N}, \mathcal{A}]} \arrow[r, shift right, "U"'] \arrow[r, phantom, "\perp"] & \mathbf{Mon}_{Fin}(\mathcal{A}) \arrow[l, shift right, "F"'] \end{tikzcd} \tag{2.12}$$

The main result of [39] is that every finitary monad T on $\mathcal{A}$ can be *presented*, i.e. given as the coequalizer of σ and τ in the category of finitary monads over $\mathcal{A}$:

$$\begin{tikzcd} FE \arrow[r, shift left, "\sigma"] \arrow[r, shift right, "\tau"'] & FB \arrow[r] & T \end{tikzcd}$$

Under adjunction 2.12, this corresponds to two maps $E \overset{\sigma'}{\underset{\tau'}{\rightrightarrows}} UFB$, which for all objects $c \in \mathcal{N}$ gives a pair $(\sigma'(Ec), \tau'(Ec))$ of terms from FBc, which we can consider to be "equations" since T has to coequalize these maps.

Note how the tensor product between an object from $\mathcal{V}_0$ (on the left) and an object from $\mathcal{A}$ (on the right) in equation 2.11 generalizes bijection 2.5 (see §1.4.7 on page 34). This is the general notion of "c-tuples" (where c is an fp object): maps from finitely presentable objects, represented by the tensor. In our case, $\mathcal{A}$ will always be $\mathcal{V}$, so the tensor is just the monoidal product, but this seem more like a matter of coincidental convenience rather than purposeful planning.

Further, the c-object of operations will be a relation (preorder etc.), the objects of which are the usual term-forming operations of arity c; and the edges of which will be the rewrite rules. In order to specify rewrites between composed operations (terms) and not only between basic operations, we use the equations to specify source and target of a rewrite rule. Let $\mathcal{A} = \mathcal{V} = \mathbf{Pre}$, and consider the simple system

$$(\Omega = \{G_1, F_2\}, R = \{F(G(x), y) \to G(y)\}$$

then the corresponding signature would be given by the functor

$$\begin{aligned} B(\mathcal{I}) &\overset{def}{=} (\{G\}, \emptyset) \\ B(\mathcal{I} + \mathcal{I}) &\overset{def}{=} (\{F, s, t\}, <) \ \text{ where } \ s < t \end{aligned}$$

with the equations $F(G(x), y) = s(x, y)$ and $G(y) = t(x, y)$, given by $E(\mathcal{I} + \mathcal{I}) = (\{A, B\}, \emptyset)$ and maps $\sigma'(A) = F(G(x), y), \tau'(A) = x$ etc. We do not explicitly write down these equations; they will occur implicitly in the term constructions below. Further, the arity c above being non-discrete corresponds exactly to the variable rewrites. These should not occur for term-forming operations (like F and

G above), which corresponds to the monad T being regular (in particular, being strongly finitary as opposed to just finitary).

It is interesting to see how the difference between an n-ary operation, and a term built over a set with n variables is modelled. The former always takes precisely n arguments, and will "use" each of them once; the latter may "discard" values substituted for a variable, or "use" a variable more than once.[5] This difference is described by the left Kan extension along the inclusion $J : \mathcal{N} \to \mathcal{A}_{\text{fp}}$; for example, consider a signature with just one binary operation $\mathtt{F}$, corresponding to a functor $B(\mathbf{2}) = \{\mathtt{F}\}$, then the left Kan extension is given by

$$\mathrm{Lan}_J B(c) \stackrel{def}{=} \sum_{e \in \mathcal{N}} \mathcal{A}_{\text{fp}}(e, c) \otimes Be$$

which for the following sets gives us the following values:

$$\begin{aligned} \emptyset &\mapsto \emptyset \\ \{x\} &\mapsto \{((x,x),\mathtt{F})\} \\ \{x,y\} &\mapsto \{((x,x),\mathtt{F}),((y,y),\mathtt{F}),((x,y),\mathtt{F}),((y,x),\mathtt{F})\} \end{aligned}$$

These are more recognisable when written as terms, e.g. $\mathtt{F}(x,x)$. We will not use this in the following, but this observation allows a precise characterization of *linear terms* (terms which use each of the variables in their context exactly once): these are those for which in equation 2.11 the $\mathcal{V}_0$-object $\mathcal{A}(e, S_n c)$ is an isomorphism (and this for all $n < \omega$).

In the following section, we are going to give a particular syntactic construction for the case of $\mathcal{V} = (\mathbf{Pre}, \times, \mathcal{I})$, and later review how this can be seen as a special instance of equation 2.11.

Following [39], we can assume that $\mathcal{N}$ and $\mathcal{A}_{\text{fp}}$ have the same objects. In our particular case, the finitely presentable objects of **Rel** and **Pre** are the relations or preorders with a finite number of objects, and in **Grph**, the graphs with a finite number of vertices and edges. Finitely presentable objects in **Cat** are categories with a finite number of objects, and a finite set M of morphisms in $\mathbf{Mor}_{\mathcal{X}}$, such that every morphism is given by identifying a finite number of sequence of morphisms from M (or equivalently, all free categories of locally finitely presentable objects in **Grph** closed under finite coequalizers in **Cat**) [15, pg. 3]. This awkward definition is one of the reasons that the technical details of the monad construction for preorders are much simpler for preorders than for categories; the correct modelling of contexts by finitely presentable objects in **Cat** requires careful attention to the morphisms of the contexts– something which is not very interesting from the term rewriting point of view.

[5] This is like the weakening or contraction rules in linear logic or sequent calculi.

To sum up this section, in order to put the general theory to work, we need to find a locally finitely presentable, symmetric monoidal closed category $\mathcal{V}$ (or a symmetric monoidal closed $\mathcal{V}$ along with an lfp $\mathcal{V}$-category $\mathcal{A}$). Then equation 2.11 gives us a general recipe for constructing terms, and the main result tells us how to treat equations. To instantiate this general theory for our use, we now have to find a suitable $\mathcal{V}$ and $\mathcal{A}$.

2.3 Enriched Monads as a Semantics for Term Rewriting Systems

In §2.1, we have presented a uniform treatment of signatures and equational presentations by monads on **Set**. In the previous section, we have presented the generalization of the underlying category theory. We will now apply the previous section to term rewriting systems, obtaining a semantics for term rewriting systems. The theory of enriched monads as presented there provides the foundation of this work, a technical framework which by instantiating its parameters is adapted to our needs. It is not at all necessary to have understood (or even read) §2.2 to understand what follows; we will develop and motivate the semantics independently before showing how they are a specialization of the general theory. We will obtain a semantics which has several advantages:

- it has a solid mathematical foundation, and is easily adaptable;
- it is compositional (§2.5);
- it is useful (as will be demonstrated by the applications in Chapters 4 and 5).

The semantics in places differs from the literature definition 1.6.1: it has another way of dealing with one-step reductions, and admits a more general form of rewrite rule. In the important part, however, the two semantics coincide: a term reduces to another term in the new semantics in nil or many steps iff it does so in the old, as we will show in §2.3.5.

In order to instantiate the general theory, we first need to decide what exactly "sets with structure" should be. There is more than one possible choice here, each of which corresponds to a different aspect of term rewriting one might be interested in.

2.3.1 Sets with Structure

The usual semantics for term rewriting systems are abstract reduction systems, which are given by a set A and a family $\{\rightarrow_\alpha\}_{\alpha\in I}$ of binary relations $\rightarrow_\alpha \subseteq A\times A$ on A (see definition 1.6.1 on page 45). When combining two term rewriting systems one does not distinguish between different reductions between the same two terms in the same term rewriting system at all, the index set just being $I = \{1, 2\}$ (where $\rightarrow_1$ are the reductions in the first system, and $\rightarrow_2$ the reductions in the second system). Thus it seems also feasible to model the reduction by just one binary relation. For many-step reductions this relation has to be reflexive and transitive (i.e. a preorder).

$$
\begin{array}{rcl}
\mathtt{Nat} & \stackrel{def}{=} & (\mathcal{N}at, R_{\mathtt{Nat}}) \\
\mathcal{N}at & \stackrel{def}{=} & \{\mathtt{Z}_0, \mathtt{S}_1, \mathtt{A}_2\} \\
R_{\mathtt{Nat}} & \stackrel{def}{=} & \{\ \mathtt{A(Z,}\ \text{'}x\mathtt{)} \quad \rightarrow \quad \text{'}x, \\
 & & \quad \mathtt{A(S(}\text{'}x\mathtt{),}\ \text{'}y\mathtt{)} \quad \rightarrow \quad \mathtt{S(A(}\text{'}x, \text{'}y\mathtt{))}\}
\end{array}
$$

Table 2.1: The term rewriting system Nat

The question is really which reductions one wants to distinguish with the indexing set I. E.g. given the term rewriting system Nat in table 2.1, there are two reductions from $\mathtt{A(S(A(Z,}\ \text{'}x\mathtt{)),}\ \text{'}y\mathtt{)}$ to $\mathtt{S(A(}\text{'}x, \text{'}y\mathtt{))}$ (see diagram 2.13); and

$$
\begin{tikzcd}
 & \mathtt{A(S(A(Z,}\ \text{'}x\mathtt{)),}\ \text{'}y\mathtt{)} \arrow[dl] \arrow[dr] & \\
\mathtt{A(S(}\text{'}x\mathtt{),}\ \text{'}y\mathtt{)} \arrow[dr] & & \mathtt{S(A(A(Z,}\ \text{'}x\mathtt{),}\ \text{'}y\mathtt{))} \arrow[dl] \\
 & \mathtt{S(A(}\text{'}x, \text{'}y\mathtt{))} &
\end{tikzcd}
\tag{2.13}
$$

the question is whether we want these two reductions to "be the same", and what this equality of reductions means in the first place. A satisfactory treatment of this question leads to an "algebra of reductions". If we do not wish to distinguish between different reductions, we say we are dealing with *unnamed reductions*, otherwise we have *named reductions*.

In summary, depending on whether we want to investigate unnamed or named, and one-step or many-step reductions, our "sets with structure" can be binary relations, graphs, preorders or categories (see table 2.2).

Deciding between one-step and many-step reductions is easy, since modularity results as a rule are about combining many-step reductions, so the crucial question

	Unnamed reductions	Named reductions
One-step	Binary Relations	Graphs
Many-step	Preorders	Categories

Table 2.2: "Sets with Structure"

is whether we are interested in named or unnamed reductions. The arguments in favour of unnamed reductions are that

- the construction is considerably simpler, and does not obscure the merits of the semantics by technical details which are uninteresting from the term rewriting point of view,
- and it is closer to the literature.

where the arguments for named reductions are that

- it is more general, yet still a faithful interpretation of the literature,
- it allows us to address a variety of new questions and problems,
- and semantically, preorders are categories anyway.

We will combine the advantages of both approaches by sidestepping the decision and proceeding as follows: we will first give a term construction for unnamed reductions, and show how to model the theory of a term rewriting system by a monad on the category **Pre** of preorders.

The semantic framework in the rest of this thesis will be monads on **Cat** however[6], as from the categorical point of view the restriction to preorders does not offer many advantages. Since preorders are categories anyway the proofs for **Cat** carry over to **Pre**. The full description of how to model the named many-step theory by a monad on **Cat** has been put into appendix A, where the reader can peruse the full details.

Instantiating the General Theory

$\boxed{!}$ Categories, In order to instantiate the general theory, we need to check whether the category of sets-with-structure satisfy the criteria of the theory; in particular, whether they form a locally finitely presentable, symmetric monoidal closed $\mathcal{V}$-category. Fortunately, the category **Pre** is locally finitely presentable — the

[6] To be precise, monads enriched over **Cat** with the usual cartesian structure. There is another closed monoidal structure on **Cat** we could enrich over, but it does not seem to offer any theoretical advantages in our setting; see page 180 for further discussion.

important point here is that there is the notion of "finite objects" (the fp objects), given by finite preorders — and it also is symmetric monoidal closed — there is the cartesian product of two preorders, and a function space between two preorders, the elements of which again form a preorder (by the pointwise order).

Further, bijection 2.5 generalizes to a bijection on preorders:

$$\mathbf{Pre}(\coprod_n \mathbf{1}, X) \cong \prod_n \mathbf{Pre}(\mathbf{1}, X) \tag{2.14}$$

since $(t_1, \ldots, t_n) \leq (s_1, \ldots, s_n)$ iff for $i = 1, \ldots, n$, $t_i \leq s_i$. Further, this bijection also holds for categories (as will be needed in the appendix)

$$\mathbf{Cat}(\coprod_n \mathbf{1}, \mathcal{C}) \cong \prod_n \mathbf{Cat}(\mathbf{1}, \mathcal{C}) \tag{2.15}$$

since a functor $\mathbf{1} \to \mathcal{C}$ is just an object of $\mathcal{C}$, and giving a natural transformation on the left amounts to giving n morphisms in $\mathcal{C}$, which is exactly a morphism in the product category on the right.

2.3.2 Arities, Contexts and Rewrite Rules

The two main principles of the monad semantics can be summed up as

- terms are composed operations;
- arities are contexts, given by the finitely presentable objects of the base category.

In order to be able to compose operations in a meaningful way, we need to handle arities appropriately. In the equational case, arities are natural numbers. Here, the fp objects of **Pre** are finite preorders. This has interesting ramifications: if contexts and arities are to be finite preorders, they do not have to be discrete. In other words, we should be able to assume more about variables than merely their existence. For if a context $X = \{x, y, z\}$ of variables is just a way of modelling the assumption that there are three entities x, y and z which we can build terms with, then to model reductions there should be a way of assuming there are reductions between these. For example, in the context X above a rewrite $\alpha : x \to y$ would mean that whatever we instantiate x and y with, there has to be a rewrite between the two of them. Consequently, we arrive at the concept of a *generalized rewrite rule* $(Y \vdash l \to r)$ where Y is a finite preorder. The "usual" case — sets of variables without a reduction structure — will be retained as a special case of the generalized concept.

Definition 2.3.1 (Generalized Rewrite Rule) A *generalized rewrite rule* in a signature Ω is given by a triple (X, l, r), written as $(X \vdash l \to r)$, where $X = (X_0, \geq)$ is a preorder, and $l, r \in T_\Omega(X_0)$. X is called the context, l the left-hand side, and r the right-hand side of the rule.

If the preorder X is discrete, the rewrite rule is called *ordinary*; if the preorder X is finite, it is called *finitary*.

If $x \geq y$ in the context X, we say there is a *variable rewrite* from x to y.

It should be pointed out that this definition is a conservative extension of definition 1.6.1 in the sense that the special case of ordinary rewrite rules are precisely the rewrite rules from definition 1.6.1. In the following, we always mean generalized rewrite rule when writing rewrite rule.

With generalized rewrite rules, it is now possible to encode the requirement that two (or more variables) have a common reduct directly into the rewrite rules. For example, given the preorder $X = (X_0, \leq)$ as a context, which has objects $X_0 \stackrel{def}{=} \{x, y, z\}$, ordered as $x \leq z$, $y \leq z$, then the rewrite rule $(X \vdash \mathtt{F}(\text{'}x, \text{'}y) \to \mathtt{G}(\text{'}z))$ means that only if we can instantiate x and y with terms t_1 and t_2 which have common reduct t_3 (which is the instantiation of z), then there is a reduction from $\mathtt{F}(t_1, t_2)$ to $\mathtt{G}(t_3)$.

Further, a rewrite rule $(X \vdash l \to r)$ is called

- *variable-introducing* if $var(r) \not\subseteq var(l)$ (there are variables on the right hand side which do not occur on the left);
- *collapsing* if $\exists x \in X.\, r = \text{'}x$ (the right hand side is a variable);
- *expanding* if $\exists x \in X.\, l = \text{'}x$ (the left hand side is a variable).

We can now define term rewriting systems:

Definition 2.3.2 (Term Rewriting System) A *term rewriting system* $\Theta = (\Omega, R)$ is given by a signature Ω and a set of rewrite rules R for Ω.

A term rewriting system $\Theta = (\Omega, R)$ is finitary if Ω is finitary and all rules in R are finitary. Since we only deal with finitary operations, we restrict ourselves to finitary rewrite rules as well; hence, in the following, when we say rewrite rule we mean a finitary rewrite rule.

Normally, one restricts term rewriting systems not to contain expanding or variable-introducing rules [41]. We do not do this here, since none of the following constructions depend on it. By carefully observing where these conditions are needed in the proofs below, we shall seek to generalize as far as possible to systems not satisfying these conditions. One of these generalizations will depend on the concept of *bounded variables*, which are variables which themselves need not appear on the left-hand side of a rewrite rule, but which rewrite from something occuring there:

Definition 2.3.3 (Bounded Variables) For a rewrite rule $(X \vdash l \to r)$, a variable $y \in X$ is *bounded* if there is $x \in var(l)$ s.t. $x \geq y$.

A rewrite rule $(X \vdash l \to r)$ introduces bounded variables if if it is variable-introducing, but all variables are bounded; a rule for which there is $x \in X$ which is not bounded is said to introduce unbounded variables.

Note that a unbounded variable necessarily will not occur on the left-hand side of a rule, because for all $x \in var(l)$, $x \geq x$.

We will now construct a free or term algebra for a term rewriting system, which we will call the *term reduction algebra*. This construction will then be extended to a monad on **Pre**.

2.3.3 The Term Reduction Algebra and the Monad T_Θ

In definition 1.6.1 on page 45, the rewriting relation is obtained as a sequence of single step rewrites, which are defined to be the rewrite rules with the variables substituted, placed in a context; i.e. if $l \to r$ is a rewrite, and σ is a substitution, then $C[\sigma(l)] \to C[\sigma(r)]$ is a single rewrite step, where $\sigma(t)$ is the result of applying substitution σ to term t.

We shall here do very much the same, but without explicitly defining contexts, rather relying on an inductive definition. The term reduction algebra on a preorder $X = (X_0, \succcurlyeq)$ will be the smallest preorder on the terms $T_\Omega(X_0)$ formed by the following rules:

- All variable rewrites (rule [VAR] below) are reductions;
- Operations have to preserve reductions [PRE]. Larger contexts are built by repeated applications of this rule;
- Instantiation of a rewrite rule [INST]: if we can assign a term to each object in the context, such that for each variable rewrite in the context there is a reduction between the two corresponding terms, then the instantiated left-hand side of the rule reduces to the instantiated right-hand side of the rule.

Definition 2.3.4 (Term Reduction Algebra) Given a term rewriting system $\Theta = (\Omega, R)$ and a preorder $X = (X_0, \succcurlyeq)$, the *term reduction algebra* on X is the smallest preorder $T_\Theta(X) = (T_\Omega(X_0), \geq)$ on the terms over X_0 satisfying the implications in table 2.3.

$$[\textsc{Var}] \quad \frac{x \succcurlyeq y}{\text{'}x \geq \text{'}y}\; x, y \in X_0$$

$$[\textsc{Pre}] \quad \frac{t_1 \geq s_1, \ldots, t_n \geq s_n}{\omega(t_1, \ldots, t_n) \geq \omega(s_1, \ldots, s_n)}\; \omega \in \Omega_n$$

$$[\textsc{Inst}] \quad \frac{\begin{array}{c}\rho = ((Y_0, \succcurlyeq) \vdash l \to r) \in R, Y_0 = \{y_1, \ldots, y_n\} \\ \forall i = 1, \ldots, n\, \forall j = 1, \ldots n.\; y_i \succcurlyeq y_j \Rightarrow t_i \geq t_j\end{array}}{l[t_1, \ldots, t_n] \geq r[t_1, \ldots, t_n]}\; t_1, \ldots, t_n \in T_\Omega(X_0)$$

Table 2.3: Definition of the Term Reduction Algebra.

Note that we do not need an explicit rule for the formation of sequences, since every preorder has to be transitive anyway.

Example 2.3.5 Given the term rewriting system `Nat` and the preorder $Z = (Z_0, \leq)$ with $Z_0 \stackrel{def}{=} \{a, \ldots, z\}$, ordered lexicographically ($a \leq b$ etc.) then the term reduction algebra $T_{\mathtt{Nat}}(Z)$ has the terms $T_{\mathcal{N}at}(Z_0)$, ordered as follows:

$$\begin{array}{rcl} \mathtt{A}(\mathtt{Z}, t) & \geq & t, \\ \mathtt{A}(\mathtt{S}(t), s) & \geq & \mathtt{S}(\mathtt{A}(t, s)) \end{array}$$

So for example, we have $\mathtt{A}(\mathtt{S}(\mathtt{S}(\mathtt{Z})), \text{'}a) \geq \mathtt{S}(\mathtt{S}(\mathtt{A}(\mathtt{Z}, \text{'}a))) \geq \mathtt{S}(\mathtt{S}(\text{'}a))$, or $\mathtt{A}(\mathtt{S}(\text{'}a), \text{'}b) \geq \mathtt{S}(\mathtt{A}(\text{'}a, \text{'}b)) \geq \mathtt{S}(\mathtt{A}(\text{'}a, \text{'}c))$, or $\mathtt{S}(\mathtt{S}(\text{'}e)) \geq \mathtt{S}(\mathtt{S}(\text{'}f))$, but of course not $\mathtt{S}(\text{'}a) \geq \text{'}a$. □

Example 2.3.6 This example involves a generalized rewrite rule. Given the term rewriting system $\mathtt{Ref} = (\emptyset, R_{\mathtt{Nat}})$ which has an empty signature and only one rule

$$R_{\mathtt{Nat}} \stackrel{def}{=} \{((\{x, y\}, x \leq y) \vdash \text{'}y \to \text{'}x)\}$$

Then for a preorder $X = (X_0, \succcurlyeq)$, $T_{\mathtt{Ref}}(X)$ only has variables $\text{'}x$ for $x \in X$ as terms, but the reduction order $\geq$ is the reflexive closure of $\succcurlyeq$; and since it is also a preorder, which is transitive and reflexive, it is in fact the equivalence closure of X. □

Structural Induction

To show that a predicate P holds for all reductions in $T_\Theta(X)$, we use a structural induction, not unlike the structural induction on terms in proposition 2.1.3.

Proposition 2.3.7 *Given a preorder $X = (X_0, \geq)$ and a binary relation $P_X \subseteq T_\Theta(X) \times T_\Theta(X)$. If for all $s, t \in T_\Theta(X)$, $s \geq t$ implies $(s, t) \in P_X$ we say that P_X holds for all reductions in $T_\Theta(X)$. This is the case if P_X is a preorder and satisfies the following three implications:*

$$\frac{x \geq y}{(\text{'}x, \text{'}y) \in P_X}$$

$$\frac{\forall i = 1, \ldots, n. (s_i, t_i) \in P_X}{(\omega(t_1, \ldots, t_n), \omega(s_1, \ldots, s_n)) \in P_X}\ \omega \in \Omega_n$$

$$\frac{\forall i, j. y_i \geq y_j \Rightarrow (t_i, t_j) \in P_X}{(l[t_1, \ldots, t_n], r[t_1, \ldots, t_n]) \in P_X}\ ((Y_0, \geq) \vdash l \to r) \in R, Y_0 = \{y_1, \ldots, y_n\}$$

Proof. This is proven by reducing it to well-founded induction. We define a partial map $size : T_\Theta(X) \times T_\Theta(X) \rightharpoonup \mathbb{N}$, which for $s, t \in T_\Theta(X)$ is only defined if we can derive $s \geq t$, and then gives the number of times we have to apply one of the rules [Pre] and [Inst], starting with the variable rewrites (rule [Var]) which have size 0, or the reflexive closure (i.e. $s = t$). We define a predicate q on $\mathbb{N}$:

$$q(n) \Leftrightarrow (\forall s, t \in T_\Omega(X).\, size(s, t) \leq n \wedge s \geq t \Rightarrow (s, t) \in P_X)$$

and show that the three implications above give rise to a well-founded induction by which we can show that for all $n \in \mathbb{N}$, $q(n)$. We conclude that for all $s, t \in T_\Omega(X)$ if $s \geq t$ then $(s, t) \in P_X$. The appendix contains a more detailed proof of the general construction for categories (proposition A.1.5 on page 177). □

We are now going to define the monad T_Θ, starting with its action, the endofunctor T_Θ : **Pre** $\to$ **Pre**. Its object function will map X to $T_\Theta(X)$, and on the morphisms, it will map a preorder morphism to the lifting of the underlying function on the term algebras.

The Action of the Monad

Lemma 2.3.8 The mapping of a preorder X to the term reduction algebra $T_\Theta(X)$ extends to a functor T_Θ, by mapping a preorder morphism $f : (X_0, \geq) \to (Y_0, \geq)$ to its lifting $f^* : T_\Omega(X_0) \to T_\Omega(Y_0)$ from definition 2.1.5.

Proof. We have to show that T_Θ is indeed a functor, in particular that f^* is a preorder morphism, and that the lifting process preserves the pointwise order on morphisms (making T_Θ a **Pre**-enriched functor).

To show that f^* is a preorder morphism we have to show

$$t \geq s \Rightarrow f^*t \geq f^*s \tag{2.16}$$

This is proven by structural induction on the reduction $f^*t \geq f^*s$:

1. If $x \geq y$ in X, then $fx \geq fy$ in Y, and hence $f^*('x) \geq f^*('y)$.

2. If $(Z \vdash l \to r) \in R$, and $l[t_1, \ldots, t_n] \geq r[t_1, \ldots, t_n]$, then we can assume that $t_i \geq t_j \Rightarrow f^*t_i \geq f^*t_j$, hence $l[f^*t_1, \ldots, f^*t_n] \geq r[f^*t_1, \ldots, f^*t_n]$, and using equations 2.9:
$$\begin{aligned} f^*(l[t_1, \ldots, t_n]) &= l[f^*t_1, \ldots, f^*t_n] \\ &\geq r[f^*t_1, \ldots, f^*t_n] \\ &= f^*(r[t_1, \ldots, t_n]) \end{aligned}$$

3. Finally, for $e(t_1, \ldots, t_n) \geq e(s_1, \ldots, s_n)$ where $t_i \geq s_i$ (for $i = 1, \ldots, n$), we can assume that $f^*t_i \geq f^*s_i$, then
$$\begin{aligned} f^*(e(t_1, \ldots, t_n)) &= e(f^*t_1, \ldots, f^*t_n) \\ &\geq e(f^*s_1, \ldots, f^*s_n) \\ &= f^*(e(s_1, \ldots, s_n)) \end{aligned}$$
where the equations hold by definition of f^*.

Preorder morphisms are ordered pointwise: for $f, g : X \to Y$, $f \geq g$ iff for all $x \in X_0, fx \geq gx$. We have to show that the lifting preserves this order, and hence gives rise to a **Pre**-enriched functor:

$$\forall x \in X_0 . fx \geq gx \;\Rightarrow\; \forall t \in T_\Omega(X_0). f^*t \geq g^*t \tag{2.17}$$

This is proven by structural induction on the term t; the induction assumption is $'x \in T_\Omega(X_0)$, then $f^*('x) = {}'fx \geq {}'gx = g^*('x)$; for the induction step, let $e(t_1, \ldots, t_n) \in T_\Omega(X_0)$ and assume $f^*(t_i) \geq g^*(t_i)$, then

$$f^*(e(t_1, \ldots, t_n)) = e(f^*t_1, \ldots, f^*t_n) \geq e(g^*t_1, \ldots, g^*t_n) = g^*(e(t_1, \ldots, t_n))$$

The lifting preserves identities and composition because of the functoriality of T_Ω, and hence definition 2.1.5 is correct. $\square$

Unit and Multiplication

The unit and multiplication of the monad are given by the unit and multiplication of the monad T_Ω from proposition 2.1.6. We have to show that they are preorder morphisms (i.e. preserve the order); naturality has been shown above.

The unit is trivially a preorder morphism by rule [VAR]. That the multiplication is a preorder morphism is proven by structural induction; the crucial step is to show that $\mu_{X_0}(l[t_1, \ldots, t_n]) \geq \mu_{X_0}(r[t_1, \ldots, t_n])$ for a rule $((Z_0, \succcurlyeq) \vdash l \to r) \in R$ with $card(Z_0) = n$, and $t_i \in T_\Omega(T_\Omega(X_0))$ with $t_i \geq t_j$ if $z_i \succcurlyeq z_j$. We first show by induction on the term that for all $s \in T_\Omega(Z_0)$

$$\mu_{X_0}(s[t_1, \ldots, t_n]) = s[\mu_{X_0}(t_1), \ldots, \mu_{X_0}(t_n)] \tag{2.18}$$

Since by rule [INST], $l[t_1, \ldots, t_n] \geq r[t_1, \ldots, t_n]$, we now have to show that $\mu_{X_0}(l[t_1, \ldots, t_n]) \geq \mu_{X_0}(r[t_1, \ldots, t_n])$, which easily follows from equation 2.18 and rule [INST].

The monad laws will hold since they just carry over from the underlying morphisms. Hence, we have the main proposition of this section:

Proposition 2.3.9 *Every term rewriting system $\Theta = (\Omega, R)$ gives rise to a monad $\mathsf{T}_\Theta \stackrel{def}{=} \langle T_\Theta, \eta, \mu \rangle$ on the category* **Pre**, *enriched over* **Pre**.

We will now explore how proposition 2.3.9 works the other way around: how we can consider a monad on **Pre** as a term rewriting system by considering its *internal language*, generalizing §2.1.3. We will close the section by relating the term reduction algebra defined above to the "traditional" definition 1.6.1.

2.3.4 The Internal Language of a Monad

Given a monad $\mathsf{T} = \langle T, \eta, \mu \rangle$ on **Pre**, the object functions of its components yield an ordinary monad on **Set**, called the underlying object monad of T and denoted $\mathsf{T}_0 \stackrel{def}{=} \langle T_{Obj}, \eta_{Obj}, \mu_{Obj} \rangle$. The internal signature of this monad yields a term language about the objects of the monad T. The counit of the adjunction from lemma 2.1.9 gives us, for any preorder $X = (X_0, \succcurlyeq)$, a function $\varepsilon_{T_0, X_0} : T_{\Sigma(\mathsf{T}_0)}(X_0) \to TX_0$ which lets us evaluate terms built in the internal signature in the monad T. The internal language of T should now give us all rewrites admitted by T, much like the internal language of a monad on **Set** are all equations admitted by the monad (definition 2.1.12).

Definition 2.3.10 (Internal Language) The *internal language* of a finitary monad $\mathsf{T} = \langle T, \eta, \mu \rangle$ on **Pre** is given by

$$\mathcal{L}(\mathsf{T}) \stackrel{def}{=} (\Sigma(\mathsf{T}_0), \mathcal{R}(\mathsf{T}))$$

where $\Sigma(\mathsf{T}_0)$ is the internal signature (definition 2.1.7) of the underlying monad T_0 on **Set**, and $\mathcal{R}(\mathsf{T})$ is the set of rewrite rules admitted by the monad T defined as follows:

$$\mathcal{R}(\mathsf{T}) \stackrel{\text{def}}{=} \{(X \vdash l \to r) \mid X \in \mathbf{Pre}_{\text{fp}}, TX = (Y_0, \geq), \varepsilon_{T_0,X}(l) \geq \varepsilon_{T_0,X}(r)\}$$

Note that it is sufficient to define the internal language over the finitely presentable objects $\mathbf{Pre}_{\text{fp}}$, since all other objects are given as a directed colimit of fp objects, and finitary functors preserve directed colimits.

We will now consider two examples for the internal language, in a slightly informal manner.

Example 2.3.11 As a very simple example, consider the internal language of the identity monad $\mathbf{1_{Pre}} = \langle \mathbf{1_{Pre}}, \mathbf{1}, \mathbf{1} \rangle$ which just maps a preorder to itself.

Then $\Sigma(\mathbf{1_{Pre}})_n = \{x \mid x \in X, \mathit{card}(X) = n\}$ or slightly rephrased $\Sigma(\mathbf{1_{Pre}})_n = \{x_i \mid x_i \in \{x_1, \ldots, x_n\}\}$. The terms we can build in the internal language are e.g. $t = x_i(x_1, \ldots, x_n)$, which the evaluation takes to $\varepsilon(t) = x_i$. □

Example 2.3.12 The monad $\mathsf{B} = \langle B, \eta_B, \mu_B \rangle$ freely adjoins an initial element $\perp$ to the argument preorder. Its action is defined as

$$B(X, \geq) \stackrel{\text{def}}{=} (X + \{\perp\}, \succcurlyeq)$$

where the order is defined as $x \geq y \Rightarrow x \succcurlyeq y$, and for all $x \in X$, $\perp \succcurlyeq x$. On morphisms, $f : X \to Y$ is mapped to $f^* : B(X) \to B(Y)$ which takes $x \in X$ to $f(x) \in Y$, and $\perp$ to $\perp$.

The unit $\eta_{B,X} : X \to B(X)$ embeds X into $B(X)$, whereas $\mu_{B,X} : B(B(X)) \to B(X)$ identifies the two distinct elements $\perp$ added by the two applications of B.

The internal language of this monad is given as follows: its signature has

$$\Sigma(\mathsf{B})_n = \{\perp, x_1, \ldots, x_n\}$$

The terms are evaluated as follows

$$\begin{aligned} \varepsilon_{\mathsf{B}}(x_i(t_1, \ldots, t_n)) &= \varepsilon_{\mathsf{B}}(t_i) \\ \varepsilon_{\mathsf{B}}(\perp(t_1, \ldots, t_n)) &= \perp \end{aligned}$$

The rewrites of the internal language are essentially rewrites $\perp \to t$ for all terms t, and $l \to r$ if $l \geq r$ in X, so we can think of this monad as having, for any $n \in \mathbb{N}$, an n-ary operation $\perp$, and rewrites $\perp(x_1, \ldots, x_n) \to x_i$. The substitution here "collapses" the $\perp$, so e.g. $\mu_B(\perp(\text{'}\perp, \text{''}x)) = \perp(\text{'}x)$. □

2.3.5 Comparison with the Literature Definition 1.6.1

A natural question to ask is how the term reduction algebra $T_\Theta(X)$ is related to the many-step reduction relation $\twoheadrightarrow_R$ from definition 1.6.1 on page 45. It is fairly obvious that if X is discrete, the relation $\geq$ on the terms is the same as the many-step reduction $\twoheadrightarrow_R$. In fact, $T_\Theta(X)$ is given by $\rightarrow_R$ plus the variable rewrites given by X, closed under congruence, and formally stated this will be the main result of this section. The constructions and results of this section are only needed when relating our definitions to definition 1.6.1 (for example, to show that our definition of confluence below coincides with the one found in the literature), and this will not be the case again until §4.1.3.

In the following, consider an ordinary term rewriting system $\Theta = (\Omega, R)$, and both $T_\Theta(X) = (T_\Omega(X), \geq)$ and $\rightarrow_R \subseteq T_\Omega(X) \times T_\Omega(X)$ for an arbitrary but fixed relation $X = (X_0, <)$.[7] We are first going to define the congruence closure of $<$, followed by the main proposition of this section:

Definition 2.3.13 (Congruence Closure of Variable Rewrites) Given a preorder $X = (X_0, <)$, the congruence closure $\rightarrow_X$ of the variable rewrites in X is the smallest preorder on $T_\Omega(X)$ such that

$$\begin{aligned} \forall x, y \in X_0 . x < y \quad &\Rightarrow \quad {'x} \rightarrow_X {'y} \\ \omega \in \Omega_n, s_i \rightarrow_X t_i \text{ for } i = 1, \ldots, n \quad &\Rightarrow \quad \omega(s_1, \ldots, s_n) \rightarrow_X \omega(t_1, \ldots, t_n) \end{aligned}$$

Proposition 2.3.14 *All reductions in the reduction algebra are given as a sequence of reductions from either the one-step reduction generated by the rules, or from variable rewrites closed under congruence:*

$$\geq \; = \; (\rightarrow_R \cup \rightarrow_X)^*$$

Proof. We are going to show the equality by inclusion in both directions. For notational convenience, let $S \stackrel{def}{=} \rightarrow_R \cup \rightarrow_X$.

1. $s \geq t \;\Rightarrow\; s\, S^*\, t$

   The proof proceeds by structural induction over $s \geq t$. If $s \geq t$ by rule [VAR], then $s \rightarrow_X t$, and if $s \geq t$ by rule [INST], then $s \rightarrow_R t$. [PRE] offers a slight complication, since if there are terms $s_1, t_1, \ldots, s_n, t_n, \in T_\Omega(X)$ such that $s_i \geq t_i$, we can only assume that for all $i = 1, \ldots, n$, either $s_i \rightarrow_X t_i$ or

[7] For the latter, following definition 1.6.1 to the letter, X needs to be infinite. This seems to be a convenience in the definition, made such that one does not run out of fresh variables when one wants to rename the variables of two terms apart.

$s_i \to_R t_i$, but for different values of i, different of the two alternatives may hold, and we can not uniformly apply the induction assumption.

However, we can decompose a rewrite $\omega(s_1, \ldots, s_n) \geq \omega(t_1, \ldots, t_n)$ into n rewrites

$$\begin{array}{rcl} \omega(s_1, s_2, \ldots, s_n) & \geq & \omega(t_1, s_2, \ldots, s_n) \\ & \geq & \omega(t_1, t_2, \ldots, s_n) \\ \ldots & \geq & \omega(t_1, t_2, \ldots, t_n) \end{array} \tag{2.19}$$

Then for $i = 1, \ldots, n$, we can assume that since $s_i \geq t_i$, $s_i\ S^*\ t_i$. Since both $\to_X$ and $\to_R$ are closed under application of operations, we have $\omega(\ldots, s_i, \ldots)\ S^*\ \ \omega(\ldots, t_i, \ldots)$ for $i = 1, \ldots, n$, and by transitivity of S^* the composition of the n rewrites in equation 2.19 is in S^*.

2. $s\, S^*\, t \;\Rightarrow\; s \geq t$

   The proof proceeds by a case distinction. If $s \to_X t$, then $s \geq t$ by rules [Var] and [Pre], if $s \to_R t$, then $s \geq t$ by rules [Inst] and [Pre], and if sS^*t then $s \geq t$ by transitivity and reflexivity of $\geq$.

$\square$

This proposition has an easy corollary which will be useful later on:

Corollary 2.3.15 *If the relation X is discrete, then $\twoheadrightarrow_R = T_\Theta(X)$.*

2.4 Properties of the Monad T_Θ

Monads are a generalisation of universal algebra, in which structures are defined by operations and equations. Term rewriting systems do not have equations, hence monads are a more general concept, and not every monad on **Pre** is generated from a term rewriting system. In this section, we will try to axiomatize those properties of monads on **Pre** which are given by term rewriting systems. We try to keep these properties as weak as possible — the more general these properties are, the more generalisations we can later make.

The properties in question are finitariness (corresponding to operations of finite arity), and preservation of coequalizers (corresponding to absence of equations), which together are strong finitariness; and further a newly introduced property called *regularity* which corresponds to free term generation and essentially means that the monad's substitution is well-behaved, and that the monad does not confuse variables with other terms.

We close with a section on non-expandingness, which is not a property characterizing term rewriting systems, but a property of term rewriting systems. We introduce this property here, since the main results of chapters 4 and 5 only hold for non-expanding term rewriting systems (and monads).

2.4.1 Finitariness

In a locally finitely presentable category (as **Set** or **Pre**) every object can be given as the directed (or filtered) colimit of the finitely presentable objects. Hence, if the action of a monad preserves this particular kind of colimits, its action on any object will be determined by its action on the finitely presentable objects; such a monad is called *finitary* (see page 56 above). If moreover the monad preserves weakly filtered colimits, it is called *strongly finitary*, and these are precisely the monads corresponding to finitary term rewriting systems.

We are now going to show that the monad T_Θ arising from a finitary term rewriting system Θ is strongly finitary. We will separately show that T_Θ preserves filtered colimits and coequalizers, and use lemma 1.3.7. The rather technical proofs of the two main lemmas have been relegated into the appendix, where they are shown for the more general case of named reductions.

Lemma 2.4.1 Given a term rewriting system $\Theta = (\Omega, R)$, the monad T_Θ is strongly finitary.

Proof. By specialising lemma A.2.2 (page 187f), the monad T_Θ is finitary, and by specialising lemma A.2.3 (page 190f), the functor T_Θ preserves coequalizers. Hence, by lemma 1.3.7, T_Θ is strongly finitary. $\square$

2.4.2 Regular Monads

Regularity is concerned with a couple of additional conditions on the unit and multiplication of the monad, corresponding to conditions on the variables and the substitution. These conditions can be described as follows:

- Different objects of X should give rise to different variables;
- Lifted morphisms should not identify variables and terms: given a morphism $f : X \to Y$, and $y \in T_\Theta(Y), x \in T_\Omega(X)$ such that $f^*(x) = {}'y$, there is $x_0 \in T_\Theta(X)$ such that $x = {}'x_0$ and $y = f(x_0)$ (corresponding to the η being regular as defined below). In other words, if a lifted morphism f^* and the unit η_X agree on a term, than this term has to be variable which moreover is given by an object in the image of the unlifted morphism f.

- In the same vein, lifting should be stable under substitution: for $f : X \to Y$ as above, given $t \in T_\Theta(X), s \in T_\Theta(T_\Theta(Y))$ such that $f^*(t) = \mu_Y(s)$, there is $t_0 \in T_\Theta(T_\Theta(X))$ such that $t = \mu_X(t_0)$ and $s = f^{**}(t_0)$.

These should hold in arbitrary contexts as well. This means that the action of the monad (which gives the context-building operations) should preserve these properties.

Further, we will define the properties on the level of monads on **Cat**; as mentioned above, these will be the semantic framework in the remainder of this thesis, so it is worthwhile to give the following definitions on a more general level from the start. This in particular means that the last two conditions also have to hold for the morphisms. In fact, the two last conditions above look fairly similar, and since both the unit and the multiplication are of course natural transformations, we can formulate them in terms of a natural transformations α:

Definition 2.4.2 (Regular Natural Transformations) Given two functors $T, S : \mathbf{Cat} \to \mathbf{Cat}$, a natural transformation $\alpha : T \Rightarrow S$ is called *regular* if given the naturality square

$$\begin{tikzcd} T\mathcal{X} \arrow[r, "\alpha_\mathcal{X}"] \arrow[d, "Tf"'] & S\mathcal{X} \arrow[d, "Sf"] \\ T\mathcal{Y} \arrow[r, "\alpha_\mathcal{Y}"'] & S\mathcal{Y} \end{tikzcd}$$

the following two hold:[8]

$$\forall x \in S\mathcal{X}, y \in T\mathcal{Y}.\, Sf(x) = \alpha_\mathcal{Y}(y) \Rightarrow \exists x_0 \in T\mathcal{X}.\, x = \alpha_\mathcal{X}(x_0) \wedge y = Tf(x_0)$$

$$\forall r : x \to x' \text{ in } S\mathcal{X}, s : y \to y' \text{ in } T\mathcal{Y}.\, Sf(r) = \alpha_\mathcal{Y}(s) \Rightarrow$$
$$\exists r_0 : x_0 \to x'_0 \text{ in } T\mathcal{X}.\, r = \alpha_\mathcal{X}(r_0) \wedge s = Tf(r_0)$$

When specializing definition 2.4.2 to preorders, we of course do not have a regular transformation, we merely have two preorder morphisms f, g such that $f \leq g$ in the pointwise order. This simplifies the second condition above. This also allows us to conveniently express the requirement that the action preserves "regularity": it means that the action preserves regularity of natural transformations.

Definition 2.4.3 (Regular Monads) A monad $\mathsf{T} = \langle T, \eta, \mu \rangle$ on **Cat** is *regular* if it satisfies the following conditions:

[8] Note that this does not quite mean the naturality square is a pullback-square, since we do not require x_0 and r_0 to be unique.

(i) T is strongly finitary;

(ii) η and μ are regular in the sense of definition 2.4.2, and T preserves regularity of natural transformations;

(iii) For all categories $\mathcal{X}$, $\eta_{\mathcal{X}}$ is monic in **Cat** (i.e. injective on the objects and faithful), and T_Θ preserves monicness.

Proposition 2.4.4 *Given a term rewriting system Θ, the monad T_Θ is regular.*

Proof. In lemma 2.4.1, it was shown that T_Θ is strongly finitary. That $\eta_{\mathcal{X}}$ is monic follows from the free generation of the term reduction algebra (no terms are identified). Preservation of monicness is shown by an easy structural induction (given $F : \mathcal{X} \to \mathcal{Y}$ which is monic, show that $F^* : T_\Theta(X) \to T_\Theta(Y)$ is monic by showing that $F^*(s) = F^*(t)$ implies $s = t$). That $\eta_{\mathcal{X}}$ is regular, and that T_Θ preserves regularity follows from the definition of the lifting.

Regularity of μ is proven as follows: on the objects, given $s \in T_\Omega(T_\Omega(Y)), t \in T_\Omega(X)$ and $f : X \to Y$ s.t. $f^*(s) = \mu_X(t)$, we construct a term $t_0 \in T_\Omega(T_\Omega(X))$ by $t_0 \stackrel{def}{=} \sigma(s,t)$ where $\sigma : T_\Omega(T_\Omega(Y)) \times T_\Omega(X) \rightharpoonup T_\Omega(T_\Omega(X))$ is a partial function, defined as follows:

$$\begin{aligned} \sigma(e(s_1,\ldots,s_n), e(t_1,\ldots,t_n)) &\stackrel{def}{=} e(\sigma(s_1,t_1),\ldots,\sigma(s_n,t_n)) \\ \sigma('x,t) &\stackrel{def}{=} 't \end{aligned}$$

$\sigma(s,t)$ is undefined if $f^*(s) \neq \mu_X(t)$. On the morphisms, a similar proof applies. $\square$

We have separated strong finitariness from the other three conditions, because the construction of the coproduct of two monads in §3.1 works for any strongly finitary (not necessarily regular) monad. It is only for the modularity proofs that we need the other conditions.

Incidentally, monads in which every component of the unit is a monomorphism are called *computational monads* by Moggi [61], although the monads he considers are in general not finitary. Further, if one strengthens the requirements of definition 2.4.2 to the naturality square being a pullback square, such a natural transformation is called *cartesian*. Carboni and Johnstone [7] show that if the unit and multiplication of a monad are cartesian, and T preserves wide pullbacks[9], such monads on **Set** are exactly those given by finite operations and strongly regular equations, where an equation is strongly regular if the same variables occur in each side of the equations, in the same order and without repetition.

[9] Also called *fibre products*— pullbacks of more than two morphism with the same source.

2.4.3 Expanding and Collapsing Monads

Recall that a rewrite rule $(X \vdash l \to r)$ is expanding if $l = {}'x$ for some $x \in X$. For the monad T_Θ, this means that there are is a rewrite $s \geq t$ in $T_\Theta(X)$ where s is in the image of the unit ($s = {}'x$), and t not (otherwise, it would just be a variable rewrite). We now want to define this property for any monad T on **Pre**, and even more general, on **Cat**, with **Pre** just being a special case.

Definition 2.4.5 (Expanding Functors and Monads) A functor $F : \mathcal{X} \to \mathcal{Y}$ is *expanding (at $x \in \mathcal{X}$)*, if there is a morphism $\alpha : Fx \to y'$ in $\mathcal{Y}$ s.t. for all morphisms $\beta : x \to y$ in $\mathcal{X}$, $F\beta \neq \alpha$.

A monad $\mathsf{T} = \langle T, \eta, \mu \rangle$ on **Cat** is expanding if there is a component $\eta_\mathcal{X} : \mathcal{X} \to T\mathcal{X}$ of the unit which is expanding, and the action preserves expandingness for injective functors, i.e. if $F : \mathcal{X} \to \mathcal{Y}$ is injective and expanding, then so is TF.

Obviously, a functor is non-expanding if it is non-expanding at all $x \in \mathcal{X}$; i.e. for all $x \in \mathcal{X}$ and $\alpha : Fx \to y$, there is $\beta : x \to x'$ such that $F\beta = \alpha$. Further, a monad is non-expanding if all components of the unit are non-expanding, and the action preserves non-expanding functors. The dual notion of expanding is collapsing (although for reasons explained on page 84 below, the action of the monad is not required to preserve collapsingness or non-collapsingness respectively).

Definition 2.4.6 (Collapsing Functors and Monads) A functor $F : \mathcal{X} \to \mathcal{Y}$ is *collapsing (at $x \in \mathcal{X}$)*, if $F^{\mathrm{op}} : \mathcal{X}^{\mathrm{op}} \to \mathcal{Y}^{\mathrm{op}}$ is expanding (at x). A monad $\mathsf{T} = \langle T, \eta, \mu \rangle$ on **Cat** is collapsing if all components $\eta_\mathcal{X} : \mathcal{X} \to T\mathcal{X}$ of the unit are collapsing.

The specialization of definitions 2.4.5 and 2.4.6 to preorders is straightforward: a preorder morphism f is non-expanding if considered as a functor it is non-expanding; and a monad $\mathsf{T} = \langle T, \eta, \mu \rangle$ on **Pre** is non-expanding if all components η_X of the unit are non-expanding, and the action preserves non-expanding preorder morphisms.

It now remains to show that a non-expanding term rewriting system gives rise to a non-expanding monad. Clearly, if the term rewriting system is non-expanding, so is the unit of the monad, but slightly surprisingly the action of the monad fails to preserve non-expandingness if the term rewriting system introduces unbounded variables (see definition 2.3.3). As a counterexample, consider the monad B from example 2.3.12 from page 75. Although the unit η_B is non-expanding, the action of B does not preserve non-expandingness: consider the unique preorder morphism $! : \mathbf{0} \to \mathbf{1}$ from the empty preorder to the one-object preorder, and let $x \in \mathbf{1}$ be that object, then $B(!) : B(\mathbf{0}) \to B(\mathbf{1})$ maps the only object $\perp \in B(\mathbf{0})$ to $\perp \in B(\mathbf{1})$, but there is a morphism $\perp \to x$ in $B(\mathbf{1})$, hence $B(!)$ is expanding, whereas ! is not.

Before we show that a non-expanding system gives rise to a non-expanding monad, we need a bit of notation. A partial function from X to Y is written as $X \rightharpoonup Y$. A partial function between pre-orders is a partial function between the underlying sets which respects the order. Given such a partial function $f : X \rightharpoonup Y$, its *lifting* to a partial function $f^* : T_\Omega(X) \rightharpoonup T_\Omega(Y)$ is defined analogous to definition 2.1.5; f^* is defined at $t \in T_\Omega(X)$ iff f is defined at all $x \in \mathit{var}(t)$. With this machinery, we can prove the following technical lemma, which essentially says that the lifting of injective functions commutes with substitution— given a term l and a substitution σ such that σ applied to l lies in the image of a the lifting of an injective function f, then σ filters through the lifting of f.

Lemma 2.4.7 Given a term rewriting system $\Theta = (\Omega, R)$ and a rule $\rho = (Z \vdash l \to r)$ in R which does not introduce unbounded variables, an injective function $f : X \to Y$, a term $t \in T_\Omega(X)$ and a preorder morphism $\sigma : Z \to T_\Omega(Y)$ such that $f^*(t) = \mu_Y(\sigma^*(l))$, which is non-expanding for all $z \in Z$ (i.e. if $f^*(t) = \sigma(z)$ and $z \geq u$ in Z, then there is $s \in T_\Omega(X)$ s.t. $t \geq s$ in $T_\Theta(X)$ and $f^*(s) = \sigma(u)$).

Then there is a preorder morphism $\tau : Z \to T_\Omega(X)$ such that

$$\mu_X(\tau^*(l)) = t \tag{2.20}$$

$$\sigma = f^* \cdot \tau \tag{2.21}$$

Proof. We construct a partial map $\tau : Z \rightharpoonup T_\Omega(X)$ which satisfies equations 2.20 and 2.21, and then show it is totally defined.

The construction proceeds by induction over l. For the induction base, let $l \stackrel{def}{=} {}'z$ with $z \in Z$, then $\mu_Y(\sigma^*(l)) = \mu_Y({}'\sigma(z)) = \sigma(z)$, hence $f^*(t) = \sigma(z)$. In this case, τ has to map z to t. Now for $z_0 \in Z$ with $z \geq z_0$, then since f^* is non-expanding at z, there is $s \in T_\Omega(X)$ such that $t \geq s$ and $f^*(s) = \sigma(z_0)$. In that case, τ has to map z_0 to s. In all other cases, τ is undefined. This gives the following definition for τ:

$$\tau(z_0) \stackrel{def}{=} \begin{cases} t & \text{for } f^*(t) = \sigma(z_0) \\ \perp & \text{otherwise} \end{cases} \tag{2.22}$$

So, τ is defined for z and all variables to which z rewrites; for the latter the value of τ is given by non-expandingness of f^*, and it is well-defined by injectivity of f^*: if $f^*(t_1) = f^*(t_2)$ then $t_1 = t_2$. Equations 2.20 and 2.21 hold by $\mu_X(\tau^*(l)) = \mu_X({}'\tau(z)) = \mu_X({}'t) = t$, and $\sigma(z) = f^*(t) = f^*(\tau(z))$.

For the induction step, we have $l = \omega(l_1, \ldots, l_n)$, and further $\mu_Y(\sigma^*(l)) = \omega(\mu_Y(\sigma^*(l_1)), \ldots, \mu_Y(\sigma^*(l_n)))$. The induction assumption is that there are $t_i \in T_\Omega(X)$ with $f^*(t_i) = \mu_Y(\sigma^*(l_i))$ (for $i = 1, \ldots, n$), and that for these we have $\tau_i : Z \rightharpoonup T_\Omega(X)$ such that $\mu_X(\tau^*(l_i)) = t_i$ and $\sigma = f^* \cdot \tau_i$. Now define $\tau : Z \rightharpoonup T_\Omega(X)$ as

$$\tau(z) \stackrel{def}{=} \begin{cases} \tau_i(z) & \text{if } \tau_i \text{ is defined at } z \text{ for some } i,\ 1 \le i \le n \\ \perp & \text{otherwise} \end{cases}$$

This is well-defined: if τ_i and τ_j with $i \neq j$ are both defined at z, then by the induction assumption, $\sigma(z) = f^*(\tau_i(z))$ and $\sigma(z) = f^*(\tau_j(z))$, so $f^*(\tau_i(z)) = f^*(\tau_j(z))$, and by injectivity of f^*, $\tau_i(z) = \tau_j(z)$. This also means that equation 2.21 holds; for equation 2.20, $\mu_X(\tau^*(\omega(l_1, \ldots, l_n))) = \omega(\mu_X(\tau^*(l_1)), \ldots, \mu_X(\tau^*(l_1))) = \omega(t_1, \ldots, t_n) = t$. This concludes the induction.

It remains to show that $\tau(z)$ is defined for all $z \in Z$, and hence $\tau : Z \to T_\Omega(X)$ is a total function as required. First, for all $z \in \mathit{var}(l)$, we have some $s \in T_\Omega(X)$ such that $\sigma(z) = f^*(t)$, and hence $\tau(z) = t$. Since all variables in Z are bounded, for all $z \in Z$ there is $y \in \mathit{var}(l)$ such that $y \le z$, and $\tau(z)$ is defined by clause 2.22 above. □

We can now show the main lemma of this section.

Lemma 2.4.8 If a term rewriting system $\Theta = (\Omega, R)$ is non-expanding and does not introduce unbounded variables, then the monad $\mathsf{T}_\Theta = \langle T_\Theta, \eta, \mu \rangle$ is non-expanding.

Proof. According to definition 2.4.5, we have to show that η is non-expanding, and that T_Θ preserves non-expandingness of injective functors.

We first show that η is non-expanding: given $s, t \in T_\Omega(X)$ with $s \ge t$ in $T_\Theta(X)$, then s is in the image of η_X iff there is $x \in X$ with $s = {}'x$. Hence, s can't be of the form $\omega(s_1, \ldots, s_n)$ or $l[s_1, \ldots, s_n]$ for $(X \vdash l \to r)$ in R, since for all l, $l \neq {}'x$. So $s \ge t$ can only be derived by rule [Var], and $t = {}'y$ with $x \ge y$ in X.

To show that T_Θ preserves non-expandingness of injective pre-order morphisms, suppose $f : X \to Y$ (with $X = (X_0, \ge), Y = (Y_0, \succcurlyeq)$) is injective and non-expanding; we then have to show that f^* is non-expanding as well: for all $t \in T_\Omega(X_0)$, if there is $s' \in T_\Omega(Y_0)$ s.t $f^*(t) \succcurlyeq s'$ then there is $s \in T_\Omega(X_0)$ such that $f^*(s) = s'$ and $t \ge s$. The proof proceeds by structural induction on the reduction $f^*(t) \ge s'$ in $T_\Theta(Y)$, with the following three cases:

1. If $f^*(t) \ge s'$ by [Var], then $f^*(t) = {}'y_1$ where $y_1 = f(x_1)$ for $x_1 \in X_0$, and $s' = {}'y_2$ with $y_1, y_2 \in Y_0$ and $y_1 \succcurlyeq y_2$. Since f is non-expanding, there has to be $x_2 \in X_0$ such that $f(x_2) = y_2$ and then $s \stackrel{def}{=} {}'x_2$.

2. If $f^*(t) \geq s'$ by [PRE], then $f^*(t) = \omega(f^*(t_1), \ldots, f^*(t_n))$, and $s' = \omega(s'_1, \ldots, s'_n)$ with $f^*(t_i) \succcurlyeq s'_i$, and the induction assumption is that there are $s_i \in T_\Omega(X_0)$ such that $f^*(s_i) = s'_i$, and $t_i \geq s_i$, all for $i = 1, \ldots, n$. Then let $s \stackrel{def}{=} \omega(s_1, \ldots, s_n)$, with $f^*(s) = s'$ and $t \geq s$ as required.

3. Finally, if $f^*(t) \succcurlyeq s'$ by [INST], then there is a rewrite rule $\rho = (Z \vdash l \to r)$ in R, and a preorder morphism $\sigma : Z \to T_\Theta(Y)$ such that $s' = \mu_Y(\sigma^*(r))$ and $f^*(t) = \mu_Y(\sigma^*(l))$. The induction assumption is that f^* is non-expanding for all $z \in Z$, i.e. if $\sigma(z_1) = f^*(t)$ and $z_1 \geq z_2$, then there is $s \in T_\Omega(X)$ s.t. $t \geq s$ and $f^*(s) = \sigma(z_1)$. Then by lemma 2.4.7, there is $\tau : Z \to T_\Theta(X)$ such that $\mu_X(\tau^*(l)) = t$ and $\sigma = f^* {\cdot} \tau$. Now let $s \stackrel{def}{=} \mu_X(\tau^*(r))$, then $t \geq s$ and further

$$\begin{aligned} f^*(s) &= f^*(\mu_X(\tau^*(r))) \\ &= \mu_Y(f^{**}(\tau^*(r))) \\ &= \mu_Y((f^*\tau)^*(r)) \\ &= \mu_Y(\sigma^*(r))) = s' \end{aligned}$$

as required. This concludes the induction, and the proof.

□

The next lemma says that a non-collapsing term rewriting system gives rise to a non-collapsing monad. It also explains why a non-collapsing monad does not have to preserve non-collapsingness: by dualising the proof of lemma 2.4.8, only monads arising from term rewriting systems in which the variables of the right-hand side of every rule are contained in the left-hand side, or rewrite to a variable in there, preserve non-collapsingness, so lemma 2.4.9 would not hold.

Lemma 2.4.9 If a term rewriting system $\Theta = (\Omega, R)$ is not collapsing, then the monad $\mathsf{T}_\Theta = \langle T_\Theta, \eta, \mu \rangle$ is non-collapsing.

Proof. By simply dualising the first part of the proof of lemma 2.4.8 as follows. We have to show that η is non-collapsing: given $s, t \in T_\Omega(X)$ with $s \geq t$ in $T_\Theta(X)$, then t is in the image of η_X iff there is $y \in X$ with $t = {'y}$. Hence, t can't be of the form $\omega(t_1, \ldots, t_n)$ or $r[t_1, \ldots, t_n]$ for $(X \vdash l \to r)$ in R, since for all r, $r \neq {'x}$. So $s \geq t$ can only be derived by rule [VAR], and $s = {'x}$ with $x \geq y$ in X. □

2.4.4 Quasi-Non-Expanding Term Rewriting Systems and Monads

In some cases, non-expandingness is a too strong requirement, e.g. when considering η-expansions [32]. In these cases, it is enough that expanding rewrites can be contracted again — if there is a rewrite from a variable x to a term t, we can reduce the term to the variable x again. In the presence of variable rewrites, we actually don't need to reduce t back to x, it is sufficient to reduce t to another variable y such that there is a variable rewrite between x and y. We call these term rewriting systems quasi-non-expanding.

Definition 2.4.10 (Quasi-Non-Expanding Term Rewriting Systems) A term rewriting system $\Theta = (\Omega, R)$ is called *quasi-non-expanding* (qne), iff for all preorders $X = (X_0, \succcurlyeq)$, if there is a rewrite $'x \geq t$ in $T_\Theta(X)$, then t is not a variable ($t \neq {}'z$), and there is a rewrite $t \geq {}'y$ in $T_\Theta(X)$ such that $x \succcurlyeq y$.

We can then formulate the corresponding property for functors and monads:

Definition 2.4.11 (Quasi-non-expanding Functors and Monads) A functor $F : \mathcal{X} \to \mathcal{Y}$ is called *quasi-non-expanding at* $x \in \mathcal{X}$, if it is full and for all $\alpha : Fx \to y$ in $\mathcal{Y}$, there are $\gamma : y \to y'$ in $\mathcal{Y}$, $\beta : x \to z$ in $\mathcal{X}$ such that $\gamma \cdot \alpha = F\beta$, and it is quasi-non-expanding if it is quasi-non-expanding at all $x \in \mathcal{X}$.

A monad $T = \langle T, \eta, \mu \rangle$ on **Cat** is quasi-non-expanding if all components $\eta_\mathcal{X} : \mathcal{X} \to T\mathcal{X}$ of the unit are quasi-non-expanding, and the action preserves this for injective functors: if $F : \mathcal{X} \to \mathcal{Y}$ is quasi-non-expanding and injective, then so is TF.

The specialization of this definition to preorder morphisms and monads on **Pre** is straightforward. Then we have to prove that a qne term rewriting system gives rise to a qne monad.

Lemma 2.4.12 If the term rewriting system Θ is quasi-non-expanding and does not introduce unbounded variables, the monad T_Θ is quasi-non-expanding.

Proof. That the unit is qne follows easy: given $X = (X_0, \succcurlyeq)$, then $\eta_X(x) = {}'x$; and since Θ is qne, if $'x \geq t$, then $t \geq {}'y$ s.t. $x \succcurlyeq y$.

To show that T_Θ preserves qne requires a structural induction analogous to the proof of lemma 2.4.8. □

2.5 Compositionality

In this section, we will justify our calling the semantics defined above "compositional". We will in chapter 3 elaborate on the fact that many structuring operations, such as the disjoint union, non-disjoint union, or push-out style parametrisation can be expressed as colimits; as a simple example one may take the disjoint union of two term rewriting systems given by the coproduct.

One can either take this coproduct in the category of syntactic presentations (here, term rewriting systems), or in the category of semantic representations (here, monads). Then "compositionality of the semantics" means that the mapping from the syntax to the semantics should preserve the coproduct (or any other colimit); for this, it is sufficient that the mapping is (or extends to) a left adjoint, and this will be the main result of this section.

Of course, one also has to show that these colimits exist in both categories, and this will be the scope of chapter 3. In fact, before we have not done so we cannot fully claim that the semantics is compositional. The main result of this section will nevertheless be an important step in that direction.

We first need to define the two categories in question. The category of semantic representations is the category of strongly finitary monads on **Pre**, $\mathbf{Mon}_{Fin}(\mathbf{Pre})$ which is the subcategory of $\mathbf{Mon}(\mathbf{Pre})$ (see §1.3.6 on page 17) given by the strongly finitary monads.

The category of syntactic presentations is given by term rewriting systems and morphisms between them. A morphism between term rewriting systems is given by a signature morphism σ_S and a mapping on the rewrite rules, which maps every rule $(X \vdash l \to r)$ to one between the images of l and r under the lifted signature morphism $\widehat{\sigma}$ (see definition 2.1.8).

Definition 2.5.1 (TRS Morphism and the Category TRS) Given two term rewriting systems $\Theta = (\Omega, R)$, $\Theta' = (\Sigma, R')$, a *term rewriting system morphism* or *TRS morphism* $\sigma : \Theta \to \Theta'$ between them is given by

- a signature morphism $\sigma_S : \Omega \to \Sigma$,
- and a map $\sigma_R : R \to R'$ such that $\sigma_R(X \vdash l \to r) = (X \vdash s \to t)$ with $s = \widehat{\sigma_S}(l)$ and $t = \widehat{\sigma_S}(t)$.

The category **TRS** has term rewriting systems as objects, and TRS morphisms as morphisms.

We can now extend the mapping of a term rewriting system Θ to T_Θ to a functor, and this functor will be left adjoint to the mapping of a term rewriting system to its internal language from definition 2.3.10.

Definition 2.5.2 The functor $F : \mathbf{TRS} \to \mathbf{Mon}_{Fin}(\mathbf{Pre})$ maps a term rewriting system Θ to the monad T_Θ, and a TRS morphism $\sigma : \Theta \to \Theta'$ to its *lifting*, the monad morphism $\widehat{\sigma} : T_\Theta \Rightarrow T_{\Theta'}$, which is defined pointwise for a preorder X as a preorder morphism $\widehat{\sigma}_X : T_\Theta(X) \to T_{\Theta'}(X)$, given by the lifting of the signature morphism σ_S from definition 2.1.8.

The functor $U : \mathbf{Mon}_{Fin}(\mathbf{Pre}) \to \mathbf{TRS}$ maps a finitary monad T to its internal language $\mathcal{L}(\mathsf{T})$ (definition 2.3.10), and a monad morphism $\alpha : \mathsf{T} \Rightarrow \mathsf{S}$ to the TRS morphism $U\alpha : \mathcal{L}(\mathsf{T}) \to \mathcal{L}(\mathsf{S})$ which on the signatures is as in lemma 2.1.9, and on rules

$$(U\alpha)_R(X \vdash l \to r) \stackrel{def}{=} (X \vdash \widehat{U\alpha}_S(l) \to \widehat{U\alpha}_S(r))$$

To show that this definition is correct, we have to show that $\widehat{\sigma}$ is a preorder morphism, and that $U(\alpha)$ is a TRS morphism.

By structural induction, we can show that $\widehat{\sigma}_X$ is a preorder morphism, and that it is natural in X. That it satisfies equations 1.3 and 1.4, making it a monad morphism, follows from the fact that the lifting of σ on the object level satisfies these equations.

To show that $U\alpha$ is indeed a morphism of term rewriting systems, we have to show that $(U\alpha)_R(X \vdash l \to r)$ is a rewrite rule in the internal language of S, i.e. $(U\alpha)_R(X \vdash l \to r) \in \mathcal{R}(\mathsf{S})$. Let X_0 denote the set of objects of X; and note that since ε_{T_0} is natural in T_0 (equation 2.8), we have

$$\alpha_X \cdot \varepsilon_{T_0, X_0} = \varepsilon_{S_0, X_0} \cdot \widehat{\alpha}_S \tag{2.23}$$

With $TX = (T_0 X_0, \geq)$ and $SX = (S_0 X_0, \succcurlyeq)$, then by definition $(X \vdash l \to r) \in \mathcal{R}(\mathsf{T})$ iff $\varepsilon_{T_0,X_0}(l) \geq \varepsilon_{T_0,X_0}(r)$. Since α_X is a preorder morphism, $\alpha_{X_0}\varepsilon_{T_0,X_0}(l) \succcurlyeq \alpha_{X_0}\varepsilon_{T_0,X_0}(r)$, and by equation 2.23, $\varepsilon_{S_0,X_0}(\widehat{\alpha}_S(l)) \succcurlyeq \varepsilon_{S_0,X_0}(\widehat{\alpha}_S(r))$, hence $(X \vdash \widehat{\alpha}_S(l) \to \widehat{\alpha}_S(r))$ is in the internal language of S as required.

To show functoriality of F and U, we only need to show functoriality of the object functions, as the underlying object functions are covered by §2.1.3. The proof principles remain the same: structural induction for the functoriality of F, and unfolding the definition for functoriality of U. The latter requires the functoriality of U on the terms, and of F, to show that U maps the identity morphism id_T on a monad T to the identity TRS morphism on $\mathcal{L}(\mathsf{T})$: for id_T, $\widehat{U id_\mathsf{T}}(l) = \widehat{\mathbf{1}_{\Sigma(\mathsf{T})}}(l) = \mathbf{1}_{\mathsf{T}_{\Sigma(\mathsf{T})}}(l) = l$ (similarly for the composition). We can now state the main proposition of this section:

Proposition 2.5.3 *The two functors F and U form an adjunction $F \dashv U$: $\mathbf{TRS} \to \mathbf{Mon}_{Fin}(\mathbf{Pre})$.*

Proof. We show adjointness by constructing a unit $\upsilon : \mathbf{1}_{\mathbf{TRS}} \to UF$ which is universal from Θ to U.

Given a term rewriting system $\Theta = (\Omega, R)$, the unit at Θ is a TRS morphism $\upsilon_\Theta : \Theta \to \mathcal{L}(T_\Theta)$ defined as follows:

- On the signature, it is the unit of the adjunction from lemma 2.1.9:

$$\upsilon_{\Theta,S} \stackrel{def}{=} \upsilon_\Omega$$

- On the rules, it is defined as

$$\upsilon_{\Theta,R}(X \vdash l \to r) \stackrel{def}{=} (X \vdash \widehat{\upsilon_\Omega}(l) \to \widehat{\upsilon_\Omega}(r))$$

This is a TRS morphism by the triangle laws of the adjunction from lemma 2.1.9: these imply that $\varepsilon_{T_\Omega} \cdot \widehat{\upsilon_\Omega}(l) = l$ and $\varepsilon_{T_\Omega} \cdot \widehat{\upsilon_\Omega}(r) = r$, hence $(X \vdash \widehat{\upsilon_\Omega}(l) \to \widehat{\upsilon_\Omega}(r)) \in \mathcal{R}(\mathsf{T}_\Theta)$.

We first have to show that ν is natural in Θ, i.e. for all $\sigma : \Theta \to \Theta'$, we have $\upsilon_{\Theta'} \cdot \sigma = (UF(\sigma)) \cdot \upsilon_\Theta$. On the signature part of υ, this follows from lemma 2.1.9; on the rewrite rules, this means that for all $(X \vdash l \to r) \in R$, $UF(\sigma)(\upsilon_\Theta(X \vdash l \to r)) = \upsilon_{\Theta'}(\sigma(X \vdash l \to r))$. The proof requires the naturality on the operations and the functoriality of the lifting, and apart from that just follows by substituting the definitions.

As the final step, we show the universality of υ_Θ from Θ to U: given a monad $\mathsf{S} = \langle S, \zeta, \xi \rangle$ and a TRS morphism $\nu : \Theta \to \mathcal{L}(\mathsf{S})$, there is a unique monad morphism $!_\nu : \mathsf{T}_\Theta \Rightarrow \mathsf{S}$ such that $U!_\nu \cdot \upsilon_\Theta = \nu$. This monad morphism is given by a family of preorder morphisms $!_{\nu,X} : T_\Theta X \to SX$ for every preorder X, which are given by lemma 2.1.9.

We have to show that $!_{\nu,X}$ is a preorder morphism, i.e. with $SX = (S_0 X_0, \succcurlyeq)$, for all $s, t \in T_\Theta(X)$ such that $s \geq t$, $!_{\nu,X}(s) \succcurlyeq !_{\nu,X}(t)$. This is rather unsurprisingly shown by induction on the structure of the reduction. The base case relies on the unit ζ of S being a preorder morphism, then $'x \geq {}'y$ implies $\zeta(x) \succcurlyeq \zeta(y)$, hence $!_{\nu,X}('x) \succcurlyeq !_{\nu,X}('y)$. There are two induction steps, operations and instantiated rules. For the first, let $t = e(t_1, \ldots, t_n)$ and $s = e(s_1, \ldots, s_n)$ and assume that for $i = 1, \ldots, n$, $t_i \geq s_i$ (hence $s \geq t$), and by induction assumption $!_\nu(t_i) \succcurlyeq !_\nu(s_i)$. By bijection 2.14 (on morphisms) $[!_\nu(t_1), \ldots, !_\nu(t_n)] \succcurlyeq [!_\nu(s_1), \ldots, !_\nu(s_n)]$, and because S is a **Pre**-enriched functor, $S[!_\nu(t_1), \ldots, !_\nu(t_n)] \succcurlyeq S[!_\nu(s_1), \ldots, !_\nu(s_n)]$.

Since these are two morphisms, ordered pointwise, we have in particular for $e \in SX$, $S[!_\nu(t_1), \ldots, !_\nu(t_n)](\nu_e) \succcurlyeq S[!_\nu(s_1), \ldots, !_\nu(s_n)](\nu_e)$, and since the multiplication ζ of S is a preorder morphism, we have $\zeta(S[!_\nu(t_1), \ldots, !_\nu(t_n)](e)) \succcurlyeq \zeta(S[!_\nu(s_1), \ldots, !_\nu(s_n)](e))$, hence $!_\nu(t) \succcurlyeq !_\nu(s)$ just as required.

For the other induction step, let $(X \vdash l \to r) \in R$, and $l[t_1, \ldots, t_n] \geq r[t_1, \ldots, t_n]$, then $!_\nu(l) \succcurlyeq !_\nu(r)$, and by a similar argument $!_\nu(l[t_1, \ldots, t_n]) \succcurlyeq !_\nu(r[t_1, \ldots, t_n])$. □

Finally, note that the category of finitary monads over **Pre** is in general not a **Pre**-category.[10] Hence, the adjunction above is merely an ordinary adjunction, not an enriched one.

2.6 Conclusions and Discussion

Summary

In this chapter, we have reviewed the well-known treatment of universal algebra by finitary monads on the category **Set** of all sets. Based on the general theory of enriched finitary monads, we have generalized this to term rewriting systems, obtaining for a term rewriting system Θ a monad T_Θ on the category **Pre** of all preorders. This monad is *enriched* over **Pre**. The finite preorders play the rôle of the natural numbers as arities for operations. Non-discrete finite preorders lead to non-discrete arities, which lead us to the notion of *variable rewrites* and *generalized rewrite rules*.

The monad on **Pre** models a particular aspect of term rewriting (namely, many-step unnamed reductions), and it is feasible to find monads on other base categories to model other aspects of term rewriting (as done in the appendix).

In the other direction, we have given the internal language $\mathcal{L}(\mathsf{T})$ of a monad T as a term rewriting system, and we have shown that the mapping of a term rewriting system Θ to the monad T_Θ is left adjoint to this, making precise the claim that the semantics is compositional.

Assessment

The main novelties of the categorical semantics presented in this chapter are the following features:

- the construction of the semantics as a monad;

[10] Put more generally, the monads over $\mathcal{V}$ do not form a $\mathcal{V}$-category [39].

- the generalized rewrite rules and variable rewrites; and
- the freeness of the construction, exhibited by the adjunction in proposition 2.5.3.

The monad construction separates the process of constructing the theory of a term rewriting system, given by the monad, from the choice of the structure used to model reductions. Once the theory is constructed, we need not concern ourselves with the more subtle intricacies of enriched category theory. We can think of the term reduction algebra as a particular preorder, and the monad T_Θ just maps preorders to preorders; the additional enriched structure of the theory construction just ensures that this mapping respects the reduction structure.[11] When reasoning about this theory, we reason about preorders (or categories); contrast this with [83, 73, 81, 64], where the reductions between terms are always 2-cells in a 2-category or a Sesqui-category.

The concept of generalised rewrite rules is particularly interesting in the context of the disjoint union of term rewriting systems, since it alleviates the complications usually associated with inner and outer reductions [42]. Because the rewrites between variables specify exactly the rewrites which are possible for any instantiation, the inner reduction cannot destroy redexes, and we have a one-step completion property with respect to rewrites from one system; we will elaborate on this in chapter 4.

The crucial insights here come from the theory of enriched monads: terms are composed operations, contexts and arities are the same, and the arities of operations are the finitely presentable objects of the base category. This leads to the concept of a generalized rewrite rule: rewrite rules are the operations which build the reduction structure, and their arity is the context. Using finitely presentable preorders (or categories) as arities means that the contexts not only contain variables, but reductions (variable rewrites) between them.

Generalized rewrite rules are interesting from another point of view: they show how a gentle generalisation suggested by a semantic treatment can lead to a more satisfying theory. Here, the theory construction would not be a left adjoint if we would not allow generalized rewrite rules. Of course, one has to be careful that these generalizations are conservative and do not change the meaning of previous definitions.

[11] This is because we use the closed structure of **Pre**, which makes **Pre** itself **Pre**-category, so instead of **Pre**-categories we can think in more familiar terms of preorders and preorder morphisms.

The theory construction being left adjoint is crucial for the compositionality of the semantics: we will in the following chapter see that many important structuring operations can be expressed as colimits, which are preserved by a left adjoint.

Possible Extensions

We do not cater for many-sorted signatures. This is more in order to maintain a minimum of readability and understandability rather than because of technical difficulties. Briefly, for a set A of sorts, a signature $\Sigma = (A, \Omega)$ is modelled by a monad on the category $\mathbf{Set}^A$, and a term rewriting system to that signature by a monad on $\mathbf{Pre}^A$. All the above proofs and constructions still go through, but have to be decorated with a sorts index $S \in A$; e.g. the term reduction algebra $T_\Theta(X)$ is now given by the product of all term reduction algebras $T_{\Theta,S}(X)$ for reductions of sorts $S \in A$. [69] gives this construction for many-sorted equational presentations; the generalization to term rewriting systems based on the previous chapter is straightforward.

It is also fairly straightforward to generalize our semantics to rewriting modulo equations, given by $\Theta = (\Omega, E, R)$ where E is a set of equations over the signature Ω, and R is a set of rewrite rules. The theory construction proceeds as above, only that the objects of our preorder would be equivalence classes of terms, not terms. Of course, the resulting monad would not satisfy the properties we have shown in section 2.4 (in particular, preservation of coequalizers and regularity), so the results of the following chapters do not apply.

Extensions to conditional term rewriting or other variants of term rewriting are not quite as straightforward and will be discussed in chapter 6.

Chapter 3

Structuring operations

Structuring operations are used to build large specifications or term rewriting systems from small ones. One of the most basic structuring operations is the disjoint union, which has been the focus of most research in the term rewriting literature; more sophisticated ones can be found in [77, 12, 13, 79, 75]. As observed by Goguen and Burstall [21, 6, 22], quite a lot of structuring operations can be described by colimits, either in the category of syntactic presentations (here, term rewriting systems), or in the category of semantic representations (here, monads). Although colimits cannot express all structuring operations, they can express those which can loosely be characterized by "shared or disjoint union and quotienting"; and we posit that this class encompasses many of the important structuring operations, such that a semantics preserving this class of operations can justifiably be called compositional. Hence, we have to show that the mapping from the syntax to the semantics preserves colimits, for which it is sufficient that the mapping is (or extends to) a left adjoint. If we can then show that both categories have colimits, we can reason about structured systems in terms of colimits of their semantics; in our particular case, we can reason about the disjoint union of term rewriting systems in terms of coproducts of monads. The adjunction was shown in §2.5, so this chapter is devoted to showing the existence of the colimits, in particular in the category of strongly finitary monads.

Any colimit can be constructed using coproducts and coequalizers (see proposition 1.3.1), so we will investigate how to construct coproducts and coequalizers of monads respectively. The main emphasis will be on the coproduct of two monads, since we will concentrate on this structuring operation in the following two chapters, when we will investigate how confluence and strong normalization are preserved under the coproduct.

Structuring operations

In support of our claim that the class of structuring operations expressible by colimits comprise many of the important ones we will now exhibit some well-known structuring operations which can be expressed by colimits of particular diagrams, and some which cannot.

Since signatures are maps from $\mathbb{N}$ to **Set**, and rewrite rules form sets, we can give a naïve semantics to structuring operations on the syntactic presentations in terms of the usual operations on sets; for example, it is clear how to form the union of two signatures.[1] The reader is invited to verify that the usual definition indeed satisfies the required universal property of the colimit of the diagrams in the following.

- **Disjoint union.** The disjoint union of two term rewriting systems Θ_1,Θ_2 is given by the coproduct $\Theta_1 + \Theta_2$.

- **Non-disjoint union.** Given two term rewriting systems $\Theta_1 = (\Omega_1, R_1)$ and $\Theta_2 = (\Omega_2, R_2)$ which have a shared part $\Theta_0 = (\Omega_0, R_0)$, then the union $\Theta_1 +_{\Theta_0} \Theta_2$ is described by the push-out of the inclusions:

$$\begin{tikzcd}
(\Omega_0, R_0) \arrow[r, hook] \arrow[d, hook] & (\Omega_1, R_1) \arrow[d] \\
(\Omega_2, R_2) \arrow[r] \arrow[ur, phantom, "\ulcorner"] & \Theta_1 +_{\Theta_0} \Theta_2
\end{tikzcd}$$

- **Translation along a signature morphism.** Given a term rewriting system $\Theta = (\Omega, R)$ and a signature morphism $\sigma : \Omega \to \Sigma$, the translation of the rules along the signature morphism is given as $(\Sigma, \widehat{\sigma}(R))$. This can be described by the following push-out:

$$\begin{tikzcd}
(\Omega, \emptyset) \arrow[r, "{(\sigma, 1)}"] \arrow[d, hook] & (\Sigma, \emptyset) \arrow[d] \\
(\Omega, R) \arrow[r] \arrow[ur, phantom, "\ulcorner"] & \cdot
\end{tikzcd}$$

  If σ is the inclusion of Ω into Σ, this describes the **extension** of the signature Ω.

[1] For two signatures Ω, Σ, their union is given by $(\Omega + \Sigma)(n) \stackrel{def}{=} \Omega(n) + \Sigma(n)$.

- **Parameterization (I).** Following the approach for parameterized specifications from CLEAR [6] and ACT ONE[12], a *parameterized term rewriting system* is given by a pair (Θ_P, Θ_1) with an inclusion $i : \Theta_P \hookrightarrow \Theta_1$. An instantiation of the parameter Θ_P is given by a system Θ_A and a morphism $h : \Theta_P \to \Theta_A$, with the instantiated parameterized term rewriting system given by the push-out of h and i.

There are other ways of modelling parameterized specifications, however;[2] and this leads us to examples of structuring operations which can not be modelled with colimits:

- **Parameterization (II).** Following the approach of ASL and its successors [79, 78], a parameterized term rewriting system is now given as $\Theta_P = \lambda X : \Sigma.\Theta_B$; then Θ_P can be instantiated with any term rewriting system Ξ with signature Σ, and the instantiation is defined as $\Theta_P(\Xi) \stackrel{def}{=} \Theta_B[\Xi/X]$.

- **Hidden Operations and Reductions.** A term rewriting system with hidden operations (and rules) is given by a pair (Θ_0, Θ_1) where the operations and rules (and sorts, in the many-sorted case) Θ_0 are considered to be *visible*. This concept cannot be modelled with colimits because given a monad T all operations and rewrite rules in the internal language $\mathcal{L}(\mathsf{T})$ are visible.

- **Observational and Behavioural Equivalence.** In the context of algebraic specifications, this means that one restricts "observations" to a certain form, like terms from certain sorts [76]. Some specification languages provide operations for the closure under observational or behavioural abstraction; this cannot be expressed in terms of colimits.

A Unified Treatment

As mentioned before, we will in the remainder of the thesis consider monads on **Cat**, enriched over **Cat** as our semantic domain. The results we will obtain still apply to the previous definitions and constructions by just regarding preorders as categories with at most one morphism between any two objects.

[2] To be precise, there are different kinds of parameterization [74]: specifications of parameterized data types (like lists generic over the elements), and parameterized specifications (like sets generic over the specification of the elements).

Structure of this Chapter

The remainder of this chapter is structured as follows:

- In §3.1, we will give a detailed account of the pointwise construction of the coproduct of two finitary monads, which will be the main section of this chapter. Along with the initial object given by the identity monad, this gives all finite products.

- The coproduct monad will map a category $\mathcal{X}$ to a category $T\mathcal{X}$, given as the colimit of a diagram $D_{\mathcal{X}}$. In §3.2 we will investigate the structure of this colimit, along with a few other technical lemmas.

- We finish by briefly sketching the construction of the coequalizer of two monads in §3.3, something more of an afterthought since it will not be needed in the following.

3.1 The Coproduct of Two Strongly Finitary Monads

Given two strongly finitary monads $\mathsf{T}_1 = \langle T_1, \eta_1, \mu_1 \rangle, \mathsf{T}_2 = \langle T_2, \eta_2, \mu_2 \rangle$ on **Cat**, the *coproduct monad* $\mathsf{T}_{1+2} = \langle T, \eta, \mu \rangle$ is given by its universal property: there are two monad morphisms $\iota_1 : \mathsf{T}_1 \to \mathsf{T}_{1+2}, \iota_2 : \mathsf{T}_2 \to \mathsf{T}_{1+2}$ (called the *injections*) such that for any other monad $\mathsf{S} = \langle S, \eta_S, \mu_S \rangle$ with monad morphisms $\alpha : \mathsf{T}_1 \to \mathsf{S}$, $\beta : \mathsf{T}_2 \to \mathsf{S}$, there is a unique monad morphism $[\alpha, \beta] : \mathsf{T}_{1+2} \to \mathsf{S}$ such that $\alpha = [\alpha, \beta] \cdot \iota_1$ and $\beta = [\alpha, \beta] \cdot \iota_2$.

In the following, we give a pointwise construction of the coproduct monad. We define the action for each category $\mathcal{X} \in$ **Cat**, extend it to an endofunctor, define unit and multiplication, show it satisfies the monad laws and show the monad thus defined satisfies the universal property.

Unfortunately, it is unavoidable that the construction becomes rather technical at times. The essentials of the construction are contained in §3.1.1, whereas §3.1.2 and §3.1.3 can be omitted by hurried readers (or those disinclined to indulge in technicalities) prepared to accept that the construction indeed gives rise to the coproduct monad (proposition 3.1.9 being the result of all the technical work).

3.1.1 The Coproduct Monad as a Pointwise Colimit

Principal Subterms and Layers

The definition of the action as given below is not immediately obvious, so we will first give a motivation of its construction. For an accessible example, consider two monads on the category **Set** given by signatures Ω_1, Ω_2. Then the coproduct of T_{Ω_1} and T_{Ω_2} should map a set X to the set of all terms built from operations of $\Omega_1 + \Omega_2$. Terms from $T_{\Omega_1+\Omega_2}(X)$ can be decomposed into *layers* of terms from $T_{\Omega_1}(X)$ and $T_{\Omega_2}(X)$ (*aliens* or *principal subterms*); we here give the definition from [42], slightly paraphrased:

> For any signature Ω, a *context* is an element of $T_{\Omega\uplus\{\Box\}}(X)$, where $\Box$ is a "fresh" constant called a *hole*. For any context $C \in T_{\Omega\uplus\{\Box\}}(X)$ with n holes, and terms $t_1, \dots, t_n \in T_\Omega(X)$, $C[t_1, \dots, t_n]$ denotes the result of replacing from left to right the holes in C by $t_1, \dots, t_n$.
>
> The *root symbol* of a term $t \in T_{\Omega_1+\Omega_2}(X)$ is defined as
>
> $$root(t) \stackrel{def}{=} \begin{cases} f & \text{if } t = f(t_1, \dots, t_n) \\ x & \text{if } t = {}'x \end{cases}$$
>
> Let $t = C[t_1, \dots, t_n]$ with $C \neq \Box$, then we write $t = C[\![t_1, \dots, t_n]\!]$ if $C \in T_{\Omega_i\uplus\{\Box\}}(X)$ and $root(t_1), \dots, root(t_n) \in \Omega_j$ for $i, j \in \{1, 2\}$ and $i \neq j$. $t_1, \dots, t_n$ are *principal subterms* or *aliens*. The *rank* of a term $t \in T_{\Omega_1+\Omega_2}(X)$ is defined as
>
> $$rank(t) \stackrel{def}{=} \begin{cases} 1 & \text{if } t \in T_{\Omega_1}(X) \uplus T_{\Omega_2}(X) \\ 1 + max\{rank(t_i) \mid 1 \leq i \leq n\} & \text{if } t = C[\![t_1, \dots, t_n]\!],\ n \geq 1 \end{cases}$$

In our approach we forego the introduction of holes and principal subterms, and take the notion of a layer as primitive insofar as every application of the action T_1 or T_2 corresponds to one layer; i.e. it takes $t_1, \dots, t_n$ to $C[\![t_1, \dots, t_n]\!]$. This is because we can build term algebras on top of term algebras; for example, the elements of $T_{\Omega_1}(T_{\Omega_2}(X))$ correspond (roughly) to terms of rank two in the above definition. However, in term algebras like $T_{\Omega_2}(T_{\Omega_1}(T_{\Omega_2}(X)))$ we will have terms from $T_{\Omega_2}(X)$ treated as variables in $T_{\Omega_1}(X)$ inserted into terms of $T_{\Omega_2}(X)$, which should be equivalent to a term from $T_{\Omega_2}(X)$; for example, if $\mathtt{F} \in \Omega_2$, then `F(''F('`x`))` and `F(F('`x`))` describe the same term. This identification is called "collapsing layers".

Hence, the coproduct should be the disjoint union of all the term algebras

$$\begin{aligned} T_{\Omega_1+\Omega_2}(X) \stackrel{def}{=}\ & X + T_{\Omega_1}(X) + T_{\Omega_2}(X) + \\ & T_{\Omega_1}(T_{\Omega_2}(X)) + T_{\Omega_2}(T_{\Omega_1}(X)) + \\ & T_{\Omega_1}(T_{\Omega_2}(T_{\Omega_1}(X))) + T_{\Omega_2}(T_{\Omega_1}(T_{\Omega_2}(X))) + \dots \end{aligned}$$

quotiented by a suitable equivalence relation, which will be affected by the unit and the multiplication: the unit identifies all the variables from X in the different term algebras, and the multiplication collapses layers as described above. We arrive at a definition of the action T as the colimit of a diagram which has all the combinations of T_1 and T_2 as objects, and all morphisms which can be formed using the unit and multiplication of the two monads as morphisms.

The diagram in question will be weakly ω-filtered in the sense of definition 1.3.5. We will restrict ourselves to strongly finitary monads (the ones preserving diagrams of this kind), since they are the ones of interest and for them the construction simplifies considerably.

The Action of the Coproduct

In the following, we use the alphabet $\mathcal{L} \stackrel{def}{=} \{1, 2\}$, and let $W \stackrel{def}{=} \mathcal{L}^*$ be the words over that alphabet. We further assume that both monads T_1 and T_2 are strongly finitary.

The diagram $D_\mathcal{X}$ is defined by defining a graph $\mathcal{G}$, and a graph morphims $d_\mathcal{X} : \mathcal{G} \to U(\mathbf{Cat})$. Then $D_\mathcal{X} : \mathcal{F}(\mathcal{G}) \to \mathbf{Cat}$ is the transpose of $d_\mathcal{X}$ under the adjunction between **Grph** and **Cat** (see page 37). The graph $\mathcal{G}$ is defined as follows:

$$
\begin{array}{lrcl}
\text{Vertices:} & V(\mathcal{G}) & \stackrel{def}{=} & W \\
\text{Edges:} & E(\mathcal{G}) & \stackrel{def}{=} & \{\mathtt{e}^w_{j,v} : wv \to wjv \mid w, v \in W, j \in \mathcal{L}\} \cup \\
 & & & \{\mathtt{m}^w_{j,v} : wjjv \to wjv \mid w, v \in W, j \in \mathcal{L}\}
\end{array}
$$

The graph morphism $d_\mathcal{X} : \mathcal{G} \to U(\mathbf{Cat})$ is defined as

$$
\begin{array}{rrcl}
\text{On the vertices:} & d_\mathcal{X}(w) & \stackrel{def}{=} & T^w(\mathcal{X}) \\
\text{On the edges:} & d_\mathcal{X}(\mathtt{e}^w_{j,v}) & \stackrel{def}{=} & T^w(\eta_{j,T^v(\mathcal{X})}) \\
 & d_\mathcal{X}(\mathtt{m}^w_{j,v}) & \stackrel{def}{=} & T^w(\mu_{j,T^v(\mathcal{X})})
\end{array}
$$

where for all $w \in W$, the functor $T^w : \mathbf{Cat} \to \mathbf{Cat}$ is defined as follows:

$$
\begin{array}{rcl}
T^\varepsilon & \stackrel{def}{=} & \mathbf{1}_{\mathbf{Cat}} \\
T^{jw} & \stackrel{def}{=} & T_j T^w
\end{array}
$$

In the following, we will write $\eta^w_{j,v}$ for $T^w(\eta_{j,T^v(\mathcal{X})})$ and $\mu^w_{j,v}$ for $T^w(\mu_{j,T^v(\mathcal{X})})$: and we further drop the empty word from super- and subscripts, so e.g. $\mu_{1,21} = \mu^\varepsilon_{1,21}$.

The object function of the endofunctor T will map $\mathcal{X}$ to the colimit of $D_\mathcal{X}$ in **Cat**. For a functor $F : \mathcal{X} \to Y$ (i.e. a morphism in **Cat**), precomposition with F of the cone over $D_\mathcal{Y}$ given by the colimit $T\mathcal{Y} = \mathit{colim}\, D_\mathcal{Y}$ gives a cone over $D_\mathcal{X}$, since the following two diagrams commute for all $w, v \in W, j \in \mathcal{L}$ by naturality of

η_j and μ_j, respectively, preserved by applying T^v and hence induces a morphism $!_F : colim\, D_{\mathcal{X}} \to colim\, D_{\mathcal{Y}}$.

$$\begin{tikzcd} T^{vw}\mathcal{X} \arrow[r, "\eta^v_{j,w}"'] \arrow[d, "T^{vw}f"'] & T^{vjw}\mathcal{X} \arrow[d, "T^{vjw}f"'] & T^{vjjw}\mathcal{X} \arrow[r, "\mu^v_{j,w}"'] \arrow[d, "T^{vjjw}f"] & T^{vjw}\mathcal{X} \arrow[d, "T^{vjw}f"] \\ T^{vw}\mathcal{Y} \arrow[r, "\eta^v_{j,w}"] & T^{vjw}\mathcal{Y} & T^{vjjw}\mathcal{Y} \arrow[r, "\mu^v_{j,w}"] & T^{vjw}\mathcal{Y} \end{tikzcd} \tag{3.1}$$

Definition 3.1.1 The endofunctor $T : \mathbf{Cat} \to \mathbf{Cat}$ is defined as follows:

$$\begin{array}{rrcl} \text{On objects:} & T(\mathcal{X}) & \stackrel{def}{=} & colim\, D_{\mathcal{X}} \\ \text{On morphisms:} & T(F : \mathcal{X} \to \mathcal{Y}) & \stackrel{def}{=} & !_F : colim\, D_{\mathcal{X}} \to colim\, D_{\mathcal{Y}} \end{array}$$

In fact, we are working in an enriched setting here, so T should be a monad enriched over **Cat**. We have elided the necessary definitions to make T into a 2-monad on the 2-category **Cat**, not because this is unnecessary but in order to keep the categorical details at a manageable level. We will consider the consequences of the enrichment in detail in §3.1.4.

Before we define the unit and the multiplication of the monad, we need to show that $\mathcal{F}(\mathcal{G})$ is a weakly ω-filtered category (see definition 1.3.5 on page 23), and hence T_1 and T_2 preserve the colimit of $D_{\mathcal{X}}$. First, observe that if $d : v \to w$ ($v, w \in W$) is a morphism in $\mathcal{F}(\mathcal{G})$, there are also morphisms $d' : uv \to uw$ for all $u \in W$. Further, $\mathcal{F}(\mathcal{G})$ is *pointed*: for all $w \in W$, there is a morphism $p : \varepsilon \to w$, constructed by induction on w. For $w = \varepsilon$, voilà; for $w = jw'$ ($j \in \mathcal{L}, w' \in W$), let there be $p' : \varepsilon \to w'$; then there is $\mathsf{e}_{j,w'} : w' \to jw'$ hence $p \stackrel{def}{=} \mathsf{e}_{jw'} {\cdot} p'$. (Note that this morphism is of course not unique.) We can now show that:

Lemma 3.1.2 $\mathcal{F}(\mathcal{G})$ is a weakly ω-filtered category.

Proof. We first show that $\mathcal{F}(\mathcal{G})$ is weakly filtered (definition 1.3.2: given $v, w \in W$, there is $u \in W$ such that there are morphisms $p : v \to u, q : w \to u$. This is shown by induction on v and w. If $v = \varepsilon$, then by pointedness of $\mathcal{F}(\mathcal{G})$, there is $p : \varepsilon \to w$, hence $u \stackrel{def}{=} w$, and the same applies if $w = \varepsilon$. Now let $w = jw'$ and $v = iv'$ (with $i, j \in \mathcal{L}$, and $w', v' \in W$, and assume that there is $u' \in W$ such that there are $p' : v' \to u'$, $q' : w' \to u'$. Now there are two cases: $i = j$, or $i \neq j$. For the first case, let $u \stackrel{def}{=} iu'$, and then there are $p : iw \to iu$ and $q : iv \to iu$. For the second case, let $u \stackrel{def}{=} iju'$; then there are $p'' : iv' \to iu'$ and $q'' : jw' \to ju'$, so let $p \stackrel{def}{=} \mathsf{e}^i_{j,u'} {\cdot} p''$ and $q \stackrel{def}{=} \mathsf{e}_{i,ju'} {\cdot} q''$ (and again, u is not unique).

To show weakly ω-filteredness, it remains to be shown that:

(i) for all $v \in W$, there at most countably infinitely many $u \in W$ such that there is $p : u \to v$. This is the case since there are only countably infinitely many objects words in W.

(ii) for all $u, v \in W$, the hom-set $\mathcal{F}(\mathcal{G})(u, v)$ is at most countably infinite. This is the case since it is generated from a finite set of edges.

□

Note that $\mathcal{F}(\mathcal{G})$ is *not* directed or filtered, because there are morphisms with the same source and target which cannot be coequalized by any other morphism in $\mathcal{F}(\mathcal{G})$. For example, there is no morphism $d : T^{121}\mathcal{X} \to T^v\mathcal{X}$ which when composed with the following two morphisms makes to the two compositions equal:

$$
\begin{tikzcd}
 & T^{21}\mathcal{X} \arrow[dr, "\eta_{1,21}"] & \\
T^{1}\mathcal{X} \arrow[ur, "\eta_{2,1}"] \arrow[dr, "\eta_2^1"'] & & T^{121}\mathcal{X} \\
 & T^{12}\mathcal{X} \arrow[ur, "\eta_1^{12}"'] &
\end{tikzcd}
$$

3.1.2 Unit, Multiplication and the Monad Laws

Having defined the action of the monad, it remains to define unit and multiplication and show that they satisfy the monad laws.

The Unit and Multiplication of the Coproduct

Recall that the colimit of a functor $D : \mathcal{J} \to \mathcal{C}$ is given by an object $C \in \mathcal{C}$ together with a cone $c : D \Rightarrow \Delta C$, which is universal amongst these cones, i.e. given any other cone $m : D \Rightarrow \Delta X$, there is a unique morphism $!_m : C \to X$, such that $m = \Delta !_m \cdot c$ (see page 14). This means that defining a morphism $F : \mathit{colim}\, D_{\mathcal{X}} \to \mathcal{Y}$ out of the colimit amounts to giving a cone $\nu : D_{\mathcal{X}} \Rightarrow \Delta\mathcal{Y}$, given by a family of functors $\nu_w : T^w\mathcal{X} \to \mathcal{Y}$ for all $w \in W$ which commute with the morphisms in $D_{\mathcal{X}}$. In the following, let $c_{\mathcal{X}} : \mathcal{D}_{\mathcal{X}} \Rightarrow \Delta(\mathit{colim}\, D_{\mathcal{X}})$ be the colimiting cone over $D_{\mathcal{X}}$, with components functors $c_{\mathcal{X},w} : T^w\mathcal{X} \to \mathit{colim}\, D_{\mathcal{X}}$; we will usually omit the index category $\mathcal{X}$ when it is clear from the context. The unit is defined as the component of the colimiting cone for the empty word ε:

Definition 3.1.3 The natural transformation $\eta : \mathbf{1}_{\mathbf{Cat}} \Rightarrow T$ is given by

$$\eta_{\mathcal{X}} \stackrel{def}{=} c_{\mathcal{X},\varepsilon}$$

The multiplication uses the fact that the two monads are strongly finitary. To define it, we need to give a morphism $\mu_{\mathcal{X}} : TT\mathcal{X} \to T\mathcal{X}$. Since $TT\mathcal{X} = \mathit{colim} D_{T\mathcal{X}}$, this morphism is given by a cone $\nu : D_{T\mathcal{X}} \to T\mathcal{X}$, that is for all $w \in W$, a morphism $\nu_w : T^w T\mathcal{X} \to T\mathcal{X}$. T^w preserves weakly ω-filtered colimits, hence $T^w T\mathcal{X} = T^w colim\, D_{T\mathcal{X}} \cong colim\, T^w D_{\mathcal{X}}$, and the morphism out of this colimit is given by another cone $\nu_{w,v} : T^w T^v \mathcal{X} \to T\mathcal{X}$ for all $v \in W$ defined as follows:

Definition 3.1.4 The multiplication $\mu_{\mathcal{X}} : TT\mathcal{X} \to T\mathcal{X}$ is given by the cone $\nu_{w,v} : T^w T^v \mathcal{X} \to T\mathcal{X}$ which for all $w, v \in W$ is defined as

$$\nu_{w,v} \stackrel{def}{=} c_{wv}$$

Naturality of the unit and multiplication follow from the fact that all components of the diagram $D_{\mathcal{X}}$ are natural transformations (diagrams 3.1). So given a functor $F : \mathcal{X} \to \mathcal{Y}$, diagram 3.2 commutes for any morphism $p_{w,\mathcal{X}} : \mathcal{X} \to T^w \mathcal{X}$

$$\begin{tikzcd} \mathcal{X} \arrow[r, "p_{w,\mathcal{X}}"] \arrow[d, "F"'] & T^w_{\mathcal{X}} \arrow[d, "T^w F"] \\ \mathcal{Y} \arrow[r, "p_{w,\mathcal{Y}}"] & T^w_{\mathcal{Y}} \end{tikzcd} \tag{3.2}$$

in the diagram, making η natural in $\mathcal{X}$.

We now have to show that the unit and multiplication as defined satisfy the monad laws, and that the monad thus defined has the universal property.

The Monad Laws

Lemma 3.1.5 For η and μ, the monad laws hold:

$$\mu_{\mathcal{X}} \eta_{T\mathcal{X}} = \mathbf{1}_{T\mathcal{X}} \tag{3.3}$$

$$\mu_{\mathcal{X}} T\eta_{\mathcal{X}} = \mathbf{1}_{T\mathcal{X}} \tag{3.4}$$

$$\mu_{\mathcal{X}} \mu_{T\mathcal{X}} = \mu_{\mathcal{X}} T\mu_{\mathcal{X}} \tag{3.5}$$

Proof. We show each of these in turn.

To show equation 3.3, we have to show that the cone inducing the composite $\mu_{\mathcal{X}} \eta_{T\mathcal{X}}$ is the same as the colimiting cone c.

The morphism $\eta_{T\mathcal{X}} : T\mathcal{X} \to TT\mathcal{X} = c_{T\mathcal{X}}$ is given by a cone $\zeta : D_{\mathcal{X}} \Rightarrow TT\mathcal{X}$, defined as $\zeta_w \stackrel{def}{=} c_{w,T\mathcal{X}} \cdot T^w(\eta_{\mathcal{X}})$:

$$T^w \mathcal{X} \xrightarrow{T^w(\eta_{\mathcal{X}})} T^w T\mathcal{X} \xrightarrow{c_{w,T\mathcal{X}}} TT\mathcal{X}$$

Since μ is the morphism induced by the cone $c_{uw} : T^u T^w \mathcal{X} \to T\mathcal{X}$, we have to show that for all morphisms $d : \mathcal{X} \to T^w \mathcal{X}$ in $D_{\mathcal{X}}$, $c_{uw} T^u d = c_u$. In other words, commutativity of the diagram on the left is shown by commutativity of the diagram on the right:

$$\begin{tikzcd} T\mathcal{X} \arrow[d, "\mathbf{1}_{\mathcal{X}}"] \arrow[dr, "\eta_{T\mathcal{X}}"] & \\ T\mathcal{X} & TT\mathcal{X} \arrow[l, "\mu_{\mathcal{X}}"] \end{tikzcd} \qquad \begin{tikzcd} T^u\mathcal{X} \arrow[d, "c_u"'] \arrow[dr, "T^u d"] & \\ T\mathcal{X} & T^uT^w\mathcal{X} \arrow[l, "c_{uw}"] \end{tikzcd}$$

The commutativity of the diagram on the right follows from the fact that if d is a morphism in $D_{\mathcal{X}}$ so is $T^u d$, and c is a cone over $D_{\mathcal{X}}$.

To show equation 3.4, $T\eta_{\mathcal{X}}$ is defined as the unique morphism making the following diagram commute

$$\begin{tikzcd} \mathcal{X} \arrow[r, "c_{\mathcal{X}}"] \arrow[d, "\eta_{\mathcal{X}}"'] & T\mathcal{X} \arrow[d, dotted, "T\eta_{\mathcal{X}}"] \\ T\mathcal{X} \arrow[r, "c_{T\mathcal{X}}"] & TT\mathcal{X} \end{tikzcd}$$

By definition $\eta_{\mathcal{X}} = c_{\mathcal{X}}$, so $T\eta_{\mathcal{X}} = c_{T\mathcal{X}} = \eta_{T\mathcal{X}}$, and then equation 3.4 reduces to equation 3.3.

For the associativity (equation 3.5), consider

$$\begin{tikzcd} T^uT^vT^w\mathcal{X} \arrow[r, "T^u c_{vw}"] \arrow[d, "c_{u,T^vT^w\mathcal{X}}"'] & T^uT\mathcal{X} \arrow[d, "c_{u,T\mathcal{X}}"] \\ TT^vT^w\mathcal{X} \arrow[r, "Tc_{vw}"'] & TT\mathcal{X} \end{tikzcd}$$

Tc_{vw} is the unique morphism on the bottom making this square commute. c_{vw} induces μ, hence the composition along the bottom and the left side induces $T\mu$; and the composition along the right side and the top induces μ_T. Hence, $\mu_T = T\mu$ and $\mu \cdot \mu_T = \mu \cdot T\mu$. $\square$

3.1.3 The Universal Property

We now have to show that the monad defined in the previous section satisfies the universal property which makes it the coproduct in the category of monads. We will first define the injections into the coproduct, and given another monad with two monad morphisms, construct the universal morphism out of the coproduct which commutes with the injections.

The Injections

Definition 3.1.6 The natural transformations $\iota_1 : T_1 \Rightarrow T$, $\iota_2 : T_2 \Rightarrow T$ are defined by

$$\begin{aligned} \iota_{1,\mathcal{X}} &\stackrel{def}{=} c_1 : T_1\mathcal{X} \to T\mathcal{X} \\ \iota_{2,\mathcal{X}} &\stackrel{def}{=} c_2 : T_2\mathcal{X} \to T\mathcal{X} \end{aligned}$$

Lemma 3.1.7 ι_1, ι_2 are monad morphisms.

Proof. We have to show that both satisfy equations 1.3 and 1.4.

The first equation holds because $\eta_{1,\mathcal{X}}$ is a morphism in the diagram $D_\mathcal{X}$, and c a cone over it: $\eta_{1,\mathcal{X}} \cdot \iota_{1,\mathcal{X}} = c_1 \cdot \eta_{1,\mathcal{X}} = c_\varepsilon = \eta_\mathcal{X}$.

For the second equation, consider diagram 3.6, where the upper right triangle commutes since μ_1 is a morphism in $D_\mathcal{X}$, and the lower left triangle by definition of μ. $\square$

$$\begin{tikzcd}
T_1T_1 \arrow[r, "\mu_1"] \arrow[d, "T_1c_1"'] \arrow[ddr, "c_{11}"] & T_1 \arrow[dd, "c_1"] \\
T_1T\mathcal{X} \arrow[d, "c_{1,T\mathcal{X}}"'] & \\
TT\mathcal{X} \arrow[r, "\mu"] & T\mathcal{X}
\end{tikzcd} \tag{3.6}$$

The Universal Property

Lemma 3.1.8 Given a monad $\mathsf{S} = \langle S, \zeta, \xi \rangle$ and two monad morphisms $\alpha : T_1 \Rightarrow S$, $\beta : T_2 \Rightarrow S$, there is a unique monad morphism $u : T \Rightarrow S$ such that $u \cdot \iota = \alpha$ and $u \cdot \iota_2 = \beta$

Proof. The proof will proceed as follows: we will construct a cone $\nu : D_\mathcal{X} \Rightarrow \Delta S\mathcal{X}$ over the diagram $D_\mathcal{X}$ which will induce a morphism $u_\mathcal{X} : T\mathcal{X} \to S\mathcal{X}$. We will then show these morphisms form a natural transformation $u : T \Rightarrow S$, and that this natural transformation is a monad morphism. Finally, we show it satisfies $u \cdot \iota_1 = \alpha$ and $u \cdot \iota_2 = \beta$.

The definition of the cone ν for an object $\mathcal{X} \in \mathbf{Cat}$ is inductive over $w \in W$. For $w = \varepsilon$, define $\nu_\varepsilon \stackrel{def}{=} \zeta_\mathcal{X}$; for $w = 1v$ (with $v \in W$), define $\nu_{T^{1v}} = \nu_{T_1T^v}$ as the composition of diagram 3.7 where square $(*)$ commutes by naturality of α, and

$$\begin{tikzcd} T_1T^v\mathcal{X} \arrow[r, "T_1\nu_v"] \arrow[d, "\alpha_{T^v\mathcal{X}}"'] \arrow[dr, phantom, "(*)"] & T_1S\mathcal{X} \arrow[d, "\alpha_{S\mathcal{X}}"] & \\ ST^v\mathcal{X} \arrow[r, "S\nu_v"'] & SS\mathcal{X} \arrow[r, "\xi_{\mathcal{X}}"] & S\mathcal{X} \end{tikzcd} \tag{3.7}$$

for $w = 2v$ correspondingly with β in place of α. Note that for $w = 1$, we have (by the above definition with $\nu_\varepsilon = \zeta$) $\nu_1 = \xi_{\mathcal{X}} \cdot S\zeta_{\mathcal{X}} \cdot \alpha_{\mathcal{X}} = \alpha_{\mathcal{X}}$ (by the unit law for S).

To show that ν is a cone, we have to show that for all morphisms $d : T^v\mathcal{X} \to T^w\mathcal{X}$ in $D_{\mathcal{X}}$, we have

$$\begin{tikzcd} T^v\mathcal{X} \arrow[rr, "d"] \arrow[dr, "\nu_v"'] & & T^w\mathcal{X} \arrow[dl, "\nu_w"] \\ & S\mathcal{X} & \end{tikzcd}$$

Since all morphisms in $D_{\mathcal{X}}$ are either of the form $\eta^r_{i,s}$ or $\mu^w_{j,v}$, or sequences of these, it is sufficient to show that for all $w \in W$ the following two hold

$$\nu_w = \nu_{1w} \cdot \eta_{1,w} \tag{3.8}$$

$$\nu_{11w} = \nu_{1w} \cdot \mu_{1,w} \tag{3.9}$$

and application of T_1 and T_2 preserve these: given an arrow $f : T^w\mathcal{X} \to T^v\mathcal{X}$ in $D_{\mathcal{X}}$ such that $\nu_w = \nu_v \cdot f$, with $j \in \mathcal{L}$, the following holds:

$$\nu_{jw} = \nu_{jv} \cdot T_j f \tag{3.10}$$

We prove the three equations in turn. For equation 3.8, consider diagram 3.11.

$$
\begin{tikzcd}
T^w\mathcal{X} \arrow[r, "\eta_{1,w}"] \arrow[d, "\nu_w"'] & T_1T^w\mathcal{X} \arrow[d, "T_1\nu_w"] \\
S\mathcal{X} \arrow[r, "\eta_{1,S\mathcal{X}}"] \arrow[dr, "\zeta_{S\mathcal{X}}"] \arrow[ddr, equal] & T_1S\mathcal{X} \arrow[d, "\alpha_{S\mathcal{X}}"] \\
 & SS\mathcal{X} \arrow[d, "\xi_\mathcal{X}"] \\
 & S\mathcal{X}
\end{tikzcd}
\tag{3.11}
$$

The square commutes by naturality of η_1, the upper of the two triangles by α being a monad morphism and the lower is the unit law for S. The composition of the arrows on the right is by definition ν_{1w}, hence the equation holds.

For equation 3.9, consider diagram 3.12. where square (1) commutes because α

$$
\begin{tikzcd}
T_1T_1T^w\mathcal{X} \arrow[rrrr, "\mu_{1,w}"] \arrow[d, "\alpha_{T_1T^w\mathcal{X}}"'] & & {} \arrow[d, phantom, "(1)"] & & T_1T^w\mathcal{X} \arrow[d, "\alpha_{T_1T^w\mathcal{X}}"] \\
ST_1T^w\mathcal{X} \arrow[rr, "S\alpha_{T^w\mathcal{X}}"] \arrow[d, "S\nu_{1w}"'] \arrow[drr, phantom, "(2)"] & & SST^w\mathcal{X} \arrow[rr, "\xi_{T^w}"] \arrow[d, "SS\nu_w"] \arrow[drr, phantom, "(3)"] & & ST^w\mathcal{X} \arrow[d, "S\nu_w"] \\
SS\mathcal{X} \arrow[drr, "\xi_\mathcal{X}"'] & & SSS\mathcal{X} \arrow[ll, "S\xi_\mathcal{X}"] \arrow[rr, "\xi_{S\mathcal{X}}"'] \arrow[d, phantom, "(4)"] & & SS\mathcal{X} \arrow[dll, "\xi_\mathcal{X}"] \\
 & & S\mathcal{X} & &
\end{tikzcd}
\tag{3.12}
$$

is a monad morphism, (2) by applying S to the definition of ν_{1w}, (3) by naturality of ξ and (4) is the associativity law of S. Then ν_{11w} is the composition of the arrows on the left side of the diagram, and ν_{1w} is the composition of the arrows on the right side of the diagram, hence equation 3.9 holds.

To show equation 3.10. assume we have $f : T^w \to T^v\mathcal{X}$ in $D_{\mathcal{X}}$ such that $\nu_w = \nu_v \cdot f$, $\nu_{1w} = \nu_{1v} \cdot T_1 f$. Applying T_1 to this we obtain the upper triangle in diagram 3.13, where the outer two triangles are the definition of ν_{1w} and ν_{1v}

$$\begin{tikzcd}
T_1T^w\mathcal{X} \arrow[rr, "T_1 f"] \arrow[dr, "T_1\nu_w"] \arrow[dddr, "\nu_{1w}"'] & & T_1T^v\mathcal{Y} \arrow[dl, "T_1\nu_v"'] \arrow[dddl, "\nu_{1v}"] \\
& T_1S\mathcal{X} \arrow[d, "\alpha_{S\mathcal{X}}"] & \\
& SS\mathcal{X} \arrow[d, "\xi_{\mathcal{X}}"] & \\
& S\mathcal{X} &
\end{tikzcd} \tag{3.13}$$

respectively. Hence equation 3.10 holds as well.

This shows that ν as defined forms a cone $\nu : D_{\mathcal{X}} \Rightarrow \Delta S\mathcal{X}$ over $D_{\mathcal{X}}$, and hence there is a unique morphism $u_{\mathcal{X}} : T\mathcal{X} \to S\mathcal{X}$ such that $u_{\mathcal{X}} \cdot c = \nu$. To be precise, we have to index the cone ν with the category $\mathcal{X}$ as well (we have not done so above to avoid unnecessarily cluttered notation), since there will be one cone ν_X for every category $\mathcal{X}$.

To show that the induced morphism $u_{\mathcal{X}} : T\mathcal{X} \to S\mathcal{X}$ is natural, we have to show that given a morphism $f : \mathcal{X} \to \mathcal{Y}$, the two cones $\nu_{\mathcal{X}}, \nu_{\mathcal{Y}}$ over the diagrams $D_{\mathcal{X}}$ and $D_{\mathcal{Y}}$ are natural in the sense that for all $u \in W$, we have to show that diagram 3.14 commutes. This proceeds by expanding the definitions of $\nu_{\mathcal{X},u}$ and

$$\begin{tikzcd}
T_u\mathcal{X} \arrow[r, "T^u f"] \arrow[d, "\nu_{\mathcal{X},u}"'] & T^u\mathcal{Y} \arrow[d, "\nu_{\mathcal{Y},u}"] \\
S\mathcal{X} \arrow[r, "Sf"] & S\mathcal{Y}
\end{tikzcd} \tag{3.14}$$

$\nu_{\mathcal{Y},u}$ in diagram 3.14, and using the fact that all of the components (α, ξ and ζ for $w = \varepsilon$) are components of natural transformations.

We now show that the natural transformation $u : T \Rightarrow S$ is a monad morphism, i.e. satisfies equations 1.3 and 1.4. For equation 1.3, we have to show that for all morphisms $d : \mathcal{X} \to T^u\mathcal{X}$ in $D_{\mathcal{X}}$, we have $\nu_u{\cdot}d = \zeta_{\mathcal{X}}$. This follows from the fact that $\zeta_{\mathcal{X}} = \nu_\varepsilon$, and the fact that ν is a cone over $D_{\mathcal{X}}$.

For equation 1.4, we have to show that for all $u, v \in W$, diagram 3.15 commutes. In order to do so, we use $u_{\mathcal{X}}{\cdot}c_{uv} = \nu_{uv}$ and proceed by induction over u.

$$\begin{tikzcd}
& T^uT^v\mathcal{X} \arrow[dl, "T^u\nu_{\mathcal{X},v}"'] \arrow[dr, "\nu_{T^v\mathcal{X},u}"] \arrow[rr, "c_{uv}"] & & T\mathcal{X} \arrow[dd, "u_{\mathcal{X}}"] \\
T^uS\mathcal{X} \arrow[dr, "\nu_{S\mathcal{X},u}"'] & & ST^v\mathcal{X} \arrow[dl, "S\nu_{\mathcal{X},v}"'] & \\
& SS\mathcal{X} \arrow[rr, "\xi"'] & & S\mathcal{X}
\end{tikzcd} \tag{3.15}$$

For $u = \varepsilon$, using the unit law of S and the definition of $\nu_\varepsilon = \zeta_{\mathcal{X}}$, the diagram simplifies to diagram 3.16. For the induction step, assume that diagram 3.15

$$\begin{tikzcd}
T^v\mathcal{X} \arrow[d, "\nu_v"'] \arrow[ddr, "\nu_v"] & \\
S\mathcal{X} \arrow[d, "\zeta_{S\mathcal{X}}"'] \arrow[dr, equal, "\mathbf{1}_{S\mathcal{X}}"'] & \\
SS\mathcal{X} \arrow[r, "\xi_{\mathcal{X}}"'] & S\mathcal{X}
\end{tikzcd} \tag{3.16}$$

commutes for $u, v \in W$, and than show it commutes for $w = 1u$ (similarly, of course for $w = 2u$). Consider diagram 3.17, where square (1) is T_1 applied to

$$\begin{tikzcd}
T_1T^uT^v\mathcal{X} \arrow[d, "T_1T^u\nu_{\mathcal{X},v}"'] \arrow[rrrd, "T_1\nu_{\mathcal{X},uv}"] \arrow[rrrr, equal] & & & & T_1T^uT^v\mathcal{X} \arrow[dddd, "\nu_{\mathcal{X},1uv}"'] \arrow[dl, phantom, "(4)"] \\
T_1T^uS\mathcal{X} \arrow[dddd, "\nu_{S\mathcal{X},1u}"'] \arrow[rrd, "T_1\nu_{S\mathcal{X},u}"] \arrow[r, phantom, "(1)"] \arrow[ddr, phantom, "(2)"] & & & & \\
 & & T_1SS\mathcal{X} \arrow[r, "T_1\xi_{\mathcal{X}}"] \arrow[d, "\alpha_{SS\mathcal{X}}"] \arrow[dr, phantom, "(3)"] & T_1S\mathcal{X} \arrow[d, "\alpha_{S\mathcal{X}}"] & \\
 & & SSS\mathcal{X} \arrow[r, "S\xi_{\mathcal{X}}"'] \arrow[dll, "\xi_{S\mathcal{X}}"'] \arrow[d, phantom, "(5)"] & SS\mathcal{X} \arrow[dr, "\xi_{\mathcal{X}}"] & \\
SS\mathcal{X} \arrow[rrrr, "\xi_{S\mathcal{X}}"'] & & & & S\mathcal{X}
\end{tikzcd} \qquad (3.17)$$

the induction assumption (using $T^{1u} = T_1T^u$), square (2) and (4) the respective definitions of $\nu_{1u,S\mathcal{X}}$ and ν_{1uv} (with $\nu_{S\mathcal{X},1} = \alpha_{S\mathcal{X}}$), square (3) the naturality of α and (5) the associativity of ξ. This concludes the proof that u is a monad morphism.

Finally, we have to show that $u{\cdot}\iota_1 = \alpha$. Since $\iota_{1,\mathcal{X}} = c_{1,\mathcal{X}}$ and $\alpha = \nu_1$, this reduces to $u{\cdot}c_1 = \nu_1$. (And similarly for $u{\cdot}\iota_2 = \beta$.) $\square$

Proposition 3.1.9 *Given two monads* $\mathsf{T}_1 = \langle T_1, \eta_1, \mu_1\rangle, \mathsf{T}_2 = \langle T_2, \eta_2, \mu_2\rangle$*, their coproduct is the monad* $\mathsf{T}_{1+2} \stackrel{def}{=} \langle T, \eta, \mu\rangle$ *with the action, unit and multiplication defined in definitions 3.1.1, 3.1.3 and 3.1.4 respectively.*

Proof. In lemma 3.1.5, T_{1+2} has been shown to be a monad, and in lemma 3.1.8 the universal property of the coproduct has been shown. $\square$

With the initial object in $\mathbf{Mon}_{Fin}(\mathbf{Cat})$ given by the identity monad $\mathsf{Id}_{\mathbf{Cat}}$ (see §1.3.6 on page 17), we have the following corollary:

Corollary 3.1.10 *The category* $\mathbf{Mon}_{Fin}(\mathbf{Cat})$ *has all finite coproducts.*

3.1.4 Consequences of the Enrichment

$\boxed{!}$ Categories We have presented a construction of the coproduct of two monads on the category **Cat**, throughout which we have neglected the enrichment– the 2-categorical properties. It should be stressed that this has been done in order to simplify the presentation, not because one can disregard the enrichment— in fact, the appropriate handling of the enrichment is the cornerstone of the theory developed in this thesis. We will now consider the various issues arising as consequences of the enriched setting.

Recall from §1.4.6 on page 33 that enriched categories in general have indexed limits; but since **Cat** has a terminal object which is the unit of the monoidal structure, one obtains the usual limits (and colimits) as a special case— so it makes indeed sense to define the diagram $D_{\mathcal{X}}$ as we have done. However, "everything is 2-everything", or slightly more precise, the monads T_1, T_2 are 2-monads, T_{1+2} has to be made into one, and all natural transformations are 2-natural transformations. Fortunately, $\mathbf{Mon}_{Fin}(\mathbf{Cat})$ is itself not a 2-category, so there is no corresponding universal property on the 2-cells (i.e. modifications between monad morphisms), which would mean all diagrams of §3.1.3 would have to bear 2-cells.

$$\begin{tikzcd}[column sep=huge, row sep=huge]
T^{vw}\mathcal{X} \arrow[r, "\eta^v_{j,w}"'] \arrow[d, shift right=4ex, "T^{vw}F"'] \arrow[d, shift left=4ex, "T^{vw}G"] \arrow[d, phantom, "\overset{T^{vw}\alpha}{\Rightarrow}"] & T^{vjw}\mathcal{X} \arrow[d, shift right=4ex, "T^{vjw}F"'] \arrow[d, shift left=4ex, "T^{vjw}G"] \arrow[d, phantom, "\overset{T^{vjw}\alpha}{\Rightarrow}"] \\
T^{vw}\mathcal{Y} \arrow[r, "\eta^v_{j,w}"] & T^{vjw}\mathcal{Y}
\end{tikzcd} \tag{3.18}$$

$$\begin{tikzcd}[column sep=huge, row sep=huge]
T^{vjjw}\mathcal{X} \arrow[r, "\mu^v_{j,w}"'] \arrow[d, shift right=4ex, "T^{vjjw}F"'] \arrow[d, shift left=4ex, "T^{vjjw}G"] \arrow[d, phantom, "\overset{T^{vjjw}\alpha}{\Rightarrow}"] & T^{vjw}\mathcal{X} \arrow[d, shift right=4ex, "T^{vjw}F"'] \arrow[d, shift left=4ex, "T^{vjw}G"] \arrow[d, phantom, "\overset{T^{vjw}\alpha}{\Rightarrow}"] \\
T^{vjjw}\mathcal{Y} \arrow[r, "\mu^v_{j,w}"] & T^{vjw}\mathcal{Y}
\end{tikzcd} \tag{3.19}$$

First, we can take the colimit of the ordinary (**Set**-enriched) functor $D_{\mathcal{X}} : \mathcal{F}(\mathcal{G}) \to \mathbf{Cat}$ in the **Set**-enriched category of all small categories as we have done; and this will be a colimit in the 2-category of all small 2-categories as well, since the diagram $D_{\mathcal{X}}$ considered as a 2-category only has identity 2-cells.[3]

We can now define the action T on 2-cells: for a natural transformation $\alpha : F \Rightarrow G$, precomposition with α induces a modification between the two cones

[3]Similarly, the coproduct of two categories in **Cat** as an ordinary category enjoys the 2-colimiting properties in **Cat** as a 2-category as well.

given by F and G respectively; that this is a modification follows from the 2-naturality of η_1, η_2 and μ_1, μ_2. We can replace the morphism F by a 2-cell $\alpha : F \Rightarrow G$ in diagrams 3.1, giving diagrams 3.18 and 3.19. Hence there is a unique a 2-cell $!_\alpha :!_F \Rightarrow !_G$, which is the action of T on 2-cells.

That η and μ are 2-natural follows from the 2-naturality of the components of $D_\mathcal{X}$, just like naturality of η and μ follows from the naturality of the components.

3.1.5 An Alternative Construction

The previous construction does have some drawbacks: it is rather involved, and it does not generalize easily to finitary (as opposed to strongly finitary) monads. A more general construction does seem desirable, and a simple, elegant construction suggesting itself would be to take the coproduct as the free monad on the functor $Q : \mathbf{Cat} \to \mathbf{Cat}$, which maps a category $\mathcal{X}$ to the following push-out in $\mathbf{Cat}$:

$$\begin{tikzcd}
\mathcal{X} \arrow[r, "\eta_1"] \arrow[d, "\eta_2"'] & T_1\mathcal{X} \arrow[d] \\
T_2\mathcal{X} \arrow[r] & Q\mathcal{X} \arrow[ul, phantom, "\ulcorner", very near start]
\end{tikzcd}$$

where the free monad FT on a finitary endofunctor $T : \mathcal{C} \to \mathcal{C}$ is given inductively by (see [35, 39])

$$S_{n+1}(d) \stackrel{def}{=} d + TS_n(d) \qquad S_0(d) \stackrel{def}{=} d \qquad FT \stackrel{def}{=} \operatorname*{colim}_{n<\omega} S_n$$

However, this simple approach does not take the multiplications into account, and hence the injections ι_1, ι_2 would not be monad morphisms, since equation 1.4 does not hold. A more sophisticated approach is to consider monads in turns as algebras, constructing the coproduct as a colimit of algebras (see §6.2.2 on page 170).

3.2 Deciding the Equality

In this section, we will investigate the structure of the category $T\mathcal{X}$. $T\mathcal{X}$ is the colimit of the diagram $D_\mathcal{X}$ in $\mathbf{Cat}$, and by proposition 1.3.1, a colimit can be expressed as a coequalizer. We know how to construct coequalizers in $\mathbf{Cat}$, so we just need to decode all this information.

The colimiting object $\mathit{colim} D_\mathcal{X}$ is a category, which has as objects equivalence classes of objects from $\coprod_{w \in W} T^w \mathcal{X}$; we will call these *composed terms*. We will give a reduction system which reduces any composed term to a unique normal form, hence deciding the equivalence. The morphisms of $colim D_{\mathcal{X}}$ are equivalence classes of paths $\langle\alpha_1, \ldots, \alpha_n\rangle$ of morphisms α_i from $\coprod_{w \in W} T^w\mathcal{X}$, and we will consider normal forms and a similar reduction system for the morphisms as well. However, we will not give a decision procedure for the equality on morphisms, since we are not really concerned with equality of the paths, as it turns out that the more relevant characteristic of paths is their length.

In the subsequent chapters, when studying modularity, these normal forms will play a pivotal rôle; essentially, to see if a property P is modular, one investigates if P is preserved in passing from a term to its normal form, and if P is preserved when combining two normal forms.

In the rest of this section we will assume that the monads $\mathsf{T}_1, \mathsf{T}_2$ are regular (see definition 2.4.3 on page 79), without explicitly requiring it every time.

3.2.1 The Coequalizer

We will now give an explicit construction of the action $T\mathcal{X}$ of the coproduct monad. By proposition 1.3.1, the colimit of $D_{\mathcal{X}}$ is given by the coequalizer of

$$\coprod_{d:u\to v \text{ in } \mathcal{F}(\mathcal{G})} T^u\mathcal{X} \underset{G}{\overset{F}{\rightrightarrows}} \coprod_{w\in W} T^w\mathcal{X} \tag{3.20}$$

diagram 3.20, where on the left side, we have for any morphism $d : T^u\mathcal{X} \to T^v\mathcal{X}$ in $D_{\mathcal{X}}$ (with $u, v \in W$) the component $T^u\mathcal{X}$ of the coproduct, and the two functors F and G are defined as $F(T^u\mathcal{X}) \stackrel{def}{=} \iota_u(T^u\mathcal{X})$ and $G(T^u\mathcal{X}) \stackrel{def}{=} \iota_v(d(T^u\mathcal{X}))$ where ι_u and ι_v are the injections into the coproduct on the right.

Let us spell this out in more detail, according to the construction given in §1.5.3 on page 44. The set of objects of $T\mathcal{X}$ is the set of objects in $\coprod_{w\in W} T^w\mathcal{X}$, quotiented by the equivalence closure $\equiv_O$ of the relation $\sim$ given by

$$x \sim \eta^w_{j,v}(x) \tag{3.21}$$

$$x \sim \mu^u_{i,t}(x) \tag{3.22}$$

for all $i, j \in \mathcal{L}$ and $t, u, v, w \in W$. We will give a normal form by orienting these equations and obtain a reduction system which we will then show to be complete. Intuitively, given any composed term t, we will collapse as many layers in t as we possibly can, and remove all quotes we can remove. For example, consider the two signatures $\Omega_1 = \{\mathtt{K}_1\}, \Omega_2 = \{\mathtt{L}_1\}$. The normal form for the composed term `K('K('`x`))` $\in T^{11}\mathcal{X}$ will be `K(K('`x`))`, for `''`$x \in T^{12}\mathcal{X}$ it will be x and for `'L(''K('`x`))` $\in T^{1221}\mathcal{X}$ it will be `L('K('`x`))`.[4]

The morphisms are equivalence classes of paths $\langle\alpha_1, \ldots, \alpha_n\rangle$ where $\alpha_i : x_i \to y_i$ in $T^w\mathcal{X}$ for $w \in W$, and $y_i \equiv_O x_{i+1}$. The paths are quotiented by the path congruence $\equiv_M$ generated by the relation $\sim$ given by

$$\begin{aligned} \langle\alpha, \beta\rangle &\sim \langle\beta\cdot\alpha\rangle \qquad \text{if } \alpha, \beta \text{ composable} && (3.23)\\ \langle\alpha\rangle &\sim \langle\eta^w_{j,v}(\alpha)\rangle && (3.24)\\ \langle\alpha\rangle &\sim \langle\mu^r_{i,s}(\alpha)\rangle && (3.25)\end{aligned}$$

By applying the same reduction system as for the objects, we are able to give a normal for paths $\langle\alpha\rangle$ of length one. For longer paths, we develop the notion of a path of *minimal length.*

3.2.2 Normal Forms for Objects

The normal form sketched above corresponds to orienting equation 3.21 right-to-left, and equation 3.22 left-to-right.

Definition 3.2.1 (The Reduction System $\to_{Ob}$) We define following reduction systems on the objects of $\coprod_{w\in W} T^w\mathcal{X}$:

$$\begin{aligned} \to_\mu &\stackrel{def}{=} \{x \to_\mu \mu^u_{j,v}(x) \mid u, v \in W, j \in \mathcal{L}, x \in T^{ujjv}\mathcal{X}\}\\ \to_\eta &\stackrel{def}{=} \{\eta^u_{j,v}(x) \to_\eta x \mid u, v \in W, j \in \mathcal{L}, x \in T^{uv}\mathcal{X}\}\\ \to_{Ob} &\stackrel{def}{=} \to_\eta \cup \to_\mu\end{aligned}$$

We now show that $\twoheadrightarrow_{Ob}$, the transitive-reflexive closure of $\to_{Ob}$, is complete, and hence obtain a decision procedure for the associated equality. We first tackle confluence. We show that $\to_{Ob}$ is confluent by showing that the two component systems $\to_\mu$ and $\to_\eta$ are confluent, and that they commute. This will be the scope of the following three lemmas.

These lemmas will make heavy use of the algebra of words over $\mathcal{L}$, and in particular the notation, definitions and lemmas of §1.7. We typically have two decompositions $w = rst$ and $w = xyz$ of a word $w \in W$, and a case distinction: if s and y are independent, then we can apply the naturality of the either μ or η (and for the latter, we additionally need regularity of η); and if they are not independent, we use

[4] Note that we have to state which component of the coproduct a composed term comes from, since the terms from different components may look the same and yet have different normal forms. For example, there are terms `K(''`x`)` $\in T^{12}\mathcal{X}$ and `K(''`x`)` $\in T^{11}\mathcal{X}$; and the former is in normal form, whereas the normal form for the second is `K('`x`)`.

- the fact that the unit is monic for confluence of $\rightarrow_\eta$,
- the associativity of the multiplication for confluence of $\rightarrow_\mu$, and
- the unit law of the monad for the commutativity of $\twoheadrightarrow_\eta$ and $\twoheadrightarrow_\mu$.

Lemma 3.2.2 $\rightarrow_\eta$ is confluent.

Proof. We have to show that given two morphisms $\eta^w_{j,v}, \eta^r_{i,s}$ in $D_\mathcal{X}$ such that $wjv = ris$, and given $y_1 \in T^{wv}\mathcal{X}, y_2 \in T^{rs}\mathcal{X}$ such that $\eta^w_{j,v}(y_1) = \eta^r_{i,s}(y_2)$, there is $u \in W, x \in T^u\mathcal{X}$, and w', v', r', s' such that $\eta^{w'}_{i,v'}(x) = y_1$, $\eta^{r'}_{i,s'}(x) = y_2$. There are two cases here: if i and j are overlapping, then $i = j$, $w = v$ and $r = s$, and we can use monicness of η.

If i and j are independent, then there is $u \in W$ such that $w_0 = wjuis$ or $w_0 = riujv$ (with $w_0 \overset{def}{=} wjv$). Suppose the former. By regularity of η_j (T^w

$$\begin{tikzcd}
T^{jus}\mathcal{X} \arrow[d, "\eta^{ju}_{i,s}"'] & T^{us}\mathcal{X} \arrow[l, "\eta_{j,us}"'] \arrow[d, "\eta^u_{i,s}"] \\
T^{juis}\mathcal{X} & T^{uis}\mathcal{X} \arrow[l, "\eta_{j,uis}"']
\end{tikzcd} \tag{3.26}$$

applied to the naturality square 3.26), there is $x_0 \in T^{wus}$ such that $\eta^w_{j,us}(x_0) = y_1$, and $\eta^{wu}_{i,s}(x_0) = y_2$. Hence, $y_1 \rightarrow_\eta x_0$ and $y_2 \rightarrow_\eta x_0$ as required. □

Lemma 3.2.3 $\rightarrow_\mu$ is confluent.

Proof. Here, we have to show that given $\mu^w_{i,v}$, $\mu^r_{j,s}$ such that $wiiv = rjjs$, then either $w = r$, $i = j$, $v = s$, or there are $w', v', r', s' \in W$ such that

$$\mu^{w'}_{j,v'} \cdot \mu^w_{i,v} = \mu^{r'}_{i,s'} \cdot \mu^r_{j,s} \tag{3.27}$$

Let $w_0 \overset{def}{=} wiiv = rjjs$. Again, we have two cases:

If ii and jj are independent, there is $u \in W$ such that $w_0 = wiiujjs$ or $w_0 = rjjuiiv$. Assume the former, then we have $w' \overset{def}{=} wiu$, $v' \overset{def}{=} s$, $r' \overset{def}{=} w$ and $s' \overset{def}{=} ujs$. To obtain equation 3.27, apply T^w to the following naturality square of μ_i (where $ujjs = v$ and $wiiu = r$):

$$\begin{tikzcd}
T^{iiujjs}\mathcal{X} \arrow[r, "\mu_{i,ujjs}"] \arrow[d, "T^{iiu}(\mu_{j,s})"'] & T^{iujjs}\mathcal{X} \arrow[d, "T^{iu}(\mu_{j,s})"] \\
T^{iiujs}\mathcal{X} \arrow[r, "\mu_{i,ujs}"'] & T^{iujs}\mathcal{X}
\end{tikzcd}$$

If $w_0 = rjjuiiv$, then there is a similar naturality square for μ_j.

If ii and jj are not independent, but (by lemma 1.7.3) overlapping, then $i = j$; and either $w = r$ and $r = s$; or $w_0 = wiiit$ or $w_0 = riiiv$. For the former, let $w' \stackrel{def}{=} w$, $v' \stackrel{def}{=} s$, $r' \stackrel{def}{=} w$ and $t' \stackrel{def}{=} t$, and apply T^w to the associativity law to show equation 3.27:

$$\begin{tikzcd}
T^{iiit}\mathcal{X} \arrow[r, "\mu_{i,T^{it}\mathcal{X}}"] \arrow[d, "T_i\mu_{i,T^t\mathcal{X}}"'] & T^{iit}\mathcal{X} \arrow[d, "\mu_{i,T^t\mathcal{X}}"] \\
T^{iit}\mathcal{X} \arrow[r, "\mu_{i,T^t\mathcal{X}}"'] & T^{it}\mathcal{X}
\end{tikzcd}$$

□

Lemma 3.2.4 $\twoheadrightarrow_\eta$ and $\twoheadrightarrow_\mu$ commute.

Proof. We in fact show something stronger: $\twoheadrightarrow_\eta$ and $\twoheadrightarrow_\mu$ subcommute (see §1.6.2 on page 46), i.e. given two morphisms $e = \eta^w_{j,v}$, $m = \mu^s_{i,t}$ in $D_\mathcal{X}$ such that $wjv = siit$ (i.e. the target of e is the source of f), then there are either $e' = \eta^{w'}_{j,v'}$, $m' = \mu^{s'}_{i,t'}$ s.t. $m{\cdot}e = e'{\cdot}m'$, or $m{\cdot}e = \mathbf{1}_{wv}$.

Let $w_0 \stackrel{def}{=} wjv = siit$. If j and ii are independent (as defined in lemma 1.7.2), then there is a naturality square similar to the one in the proof of lemma 3.2.3.

If j and ii are not independent, by lemma 1.7.3 they are overlapping, hence $i = j$ and either $w = s$ and $v = it$, or $s = iw$ and $v = t$. Then apply T^s to the unit laws (the left side for the first case, the second case for the second case)

$$\begin{tikzcd}
T^{it}\mathcal{X} \arrow[r, "\eta_{i,T^{it}\mathcal{X}}"] \arrow[dr, equal] & T^{iit}\mathcal{X} \arrow[d, "\mu_{i,T^t\mathcal{X}}"] & T^{it}\mathcal{X} \arrow[l, "T_i(\eta_{i,T^t\mathcal{X}})"'] \arrow[dl, equal] \\
& T^{iv}\mathcal{X} &
\end{tikzcd}$$

□

We can now show that $\twoheadrightarrow_{Ob}$, the transitive-reflexive closure of $\rightarrow_{Ob}$, is complete and hence obtain a decision procedure for the associated equality. To do this, we define the *rank* of a term $t \in T^w\mathcal{X}$ as $\mathrm{rank}(t) \stackrel{def}{=} |w|$ (where $|w|$ is the length of the word w).

Lemma 3.2.5 $\twoheadrightarrow_{Ob}$ is complete.

Proof. First, $\twoheadrightarrow_{Ob}$ is confluent by the Hindley-Rosen lemma 1.6.3, since $\rightarrow_\eta$ and $\rightarrow_\mu$ are confluent, and $\twoheadrightarrow_\eta$ and $\twoheadrightarrow_\mu$ commute, by the three lemmas just shown.

To show termination of $\twoheadrightarrow_{Ob}$, and hence completeness, note that in all rules of $l \rightarrow_{Ob} r$, the rank of l is strictly greater than that of r and hence there can be no infinite reduction $x_1 \rightarrow_{Ob} x_2 \rightarrow_{Ob} x_3 \ldots$. □

Since $\twoheadrightarrow_{Ob}$ is complete, every object in $t \in \coprod_{w\in W} T^w\mathcal{X}$ reduces to a unique normal form which we denote NF(t). This forms a decision procedure for the equivalence of the objects:

Lemma 3.2.6 Given $t, t' \in \coprod_{w\in W} T^w\mathcal{X}$, $Qt = Qt'$ iff $\mathrm{NF}(t) = \mathrm{NF}(t')$.

Proof. $\mathrm{NF}(t) = \mathrm{NF}(t')$ iff t and t' are related is the equational theory on $\coprod_{w\in W} T^w\mathcal{X}$ generated by $\twoheadrightarrow_{Ob}$. This theory is clearly the same as that induced by equations 3.21 and 3.22. □

3.2.3 Normal Forms for Morphisms

We now consider morphisms in the coequalizer of diagram 3.20. On the objects, the reduction system $\twoheadrightarrow_{Ob}$ gives us a normal form deciding the equivalence generated by diagram 3.20. It was obtained by orienting equations 3.21 and 3.22. Unfortunately, we can't do the same with equations 3.23, 3.24 and 3.25. Obviously, to obtain a strongly normalizing system we would have to orient 3.23 left-to-right, obtaining a reduction system $\twoheadrightarrow_{Comp}$ on the paths. This system is confluent (by associativity of the composition), but it fails to commute with $\twoheadrightarrow_\eta$ (the system generated by equation 3.24, see 3.2.7 below), as the following examples shows. Let the monad T_1 be given by a unary operation `F`, and the monad T_2 by a unary operation `G` and rules

$$R_2 = \{\mathtt{G}(\text{'}y) \rightarrow \text{'}y\}$$

and let α be the rewrite $\mathtt{F}(\text{'}\mathtt{G}(\text{'}x)) \rightarrow \mathtt{F}(\text{''}x)$ in $T^{12}\mathcal{X}$, and $\beta : x \rightarrow y$ be in $\mathcal{X}$. Then there is the path <α, ''β>, which reduces by $\twoheadrightarrow_{Comp}$ to <''$\beta\cdot\alpha$>, and by $\twoheadrightarrow_\eta$ to <α, 'β>; this span cannot be completed. This doesn't mean that one can't find a complete reduction system to produce normal forms for the paths; it just means that the straightforward approach does not work. Since we do not need to effectively decide the equality on morphisms anyway, but are more interested in the length of these paths (in order to deal with strong normalisation), we will develop the notion of a path of minimal length: a path such that no equivalent path is shorter.

We will approach the problem as follows: we will first define a normal form for morphisms in $\coprod_{w\in W} T^w\mathcal{X}$, corresponding to a normal form for paths of length one by orienting clauses 3.24 and 3.25 above. We will then define the notion of a path of minimal length, and find necessary and sufficient characterisations of paths of minimal length. For this, we will investigate pairs of two morphisms the target of the first of which is equivalent to the source of the second, but which cannot be composed; these will be called incomposable pairs and they correspond to normal forms for clause 3.23.

Normal Forms for Morphisms in $\coprod_{w\in W} T^w\mathcal{X}$

Definition 3.2.7 (The Reduction System $\rightarrow_{Mor}$) Define the reduction systems on the morphisms of $\coprod_{w\in W} T^w\mathcal{X}$:

$$\begin{aligned}
\rightarrow_\mu &\stackrel{def}{=} \{\alpha \rightarrow_\mu \mu^u_{j,v}(\alpha) \mid u, v \in W, j \in \mathcal{L}, \alpha : t \rightarrow t' \in T^{ujjv}\mathcal{X}\} \\
\rightarrow_\eta &\stackrel{def}{=} \{\eta^u_{j,v}(\alpha) \rightarrow_\eta \alpha \mid u, v \in W, j \in \mathcal{L}, \alpha : t \rightarrow t' \in T^{uv}\mathcal{X}\} \\
\rightarrow_{Mor} &\stackrel{def}{=} \rightarrow_\eta \cup \rightarrow_\mu
\end{aligned}$$

Lemma 3.2.8 $\rightarrow_{Mor}$ is complete, and every $\alpha : x \rightarrow y$ in $T^w\mathcal{X}$ reduces to a unique normal form $\mathrm{NF}(\alpha) : x' \rightarrow y'$ s.t. for all β, $Q\alpha = Q\beta$ iff $\mathrm{NF}(\alpha) = \mathrm{NF}(\beta)$.

Proof. Analogously to lemma 3.2.5 and 3.2.6. □

The mapping of terms and morphisms to their normal form can not be extended to an endofunctor on $\coprod_{w\in W} T^w\mathcal{X}$, since the presence of expanding or collapsing rewrites means that the normal form need not preserve the source and target of a morphism. For example, given the rewrite $\alpha :$ '$x \rightarrow$ `G('`x`)` in $T_1(X)$, then $\mathrm{NF}(\alpha) = \alpha$ which is in $T_1(X)$ while $\mathrm{NF}($'$x) = x$ which is in X. Such a rewrite is called "layer-expanding", since it introduces a new layer in a rewrite path. The dual notion (which is actually more important) is called "layer-collapsing".

Definition 3.2.9 (Layer-collapsing and Layer-expanding) Let $\alpha : s \rightarrow t$ be in $T^w\mathcal{X}$, and $\mathrm{NF}(\alpha) : s' \rightarrow t'$ be its normal form. Then α is called *layer-collapsing* at $y \in T^{uv}\mathcal{X}$ in T_j if t' is in the image of η_j: there are $u, v \in W, j \in \mathcal{L}$ with $t' = \eta^u_{j,v}(y)$. Dually, α is called *layer-expanding* in T_j if the above holds for s' instead of t'.

Note that a rewrite can be layer-expanding or layer-collapsing in both systems at the same time.

Lemma 3.2.10 Layer-expanding (layer-collapsing) rewrites are those for which the normal form of the source (target) is not the same as the source (target) of the normal form:

(i) $\alpha : s \to t$ in $T^w\mathcal{X}$ is layer-expanding iff for $\mathrm{NF}(\alpha) : s' \to t'$, $s' \neq \mathrm{NF}(s)$.

(ii) $\alpha : s \to t$ in $T^w\mathcal{X}$ is layer-collapsing iff for $\mathrm{NF}(\alpha) : s' \to t'$, $t' \neq \mathrm{NF}(t)$.

Proof. We prove the first clause; the second is proven analogously.

We can apply $\to_\mu$ to a morphism $\alpha : x \to y$ if and only if we can apply $\to_\mu$ to its source x and target y. Further, whenever we can apply $\to_\eta$ to α, we can apply $\to_\eta$ to x and y. Hence, whenever $\alpha \to_{Mor} \alpha'$, the source and target of α reduce as well. The only case in which we can apply $\to_\eta$ to x (or y) but not to α is if $x = \eta^u_{j,v}(x')$, but for all $\beta : x' \to y'$, $\eta^u_{j,v}(\beta) \neq \alpha$ (and so $\alpha \not\to_\eta \beta$), which is precisely the definition of layer-expanding. $\square$

Note that the existence of an $\alpha : x \to y$ above such that for all $\beta : \mathrm{NF}(x) \to x'$, $\eta^u_{j,v}(\beta) \neq \alpha$ is exactly the definition of $\eta^u_{j,v}$ being expanding at $\mathrm{NF}(x)$, hence if both monads are non-expanding, we have $s' = \mathrm{NF}(s)$ for all rewrites $\alpha : s \to t$ and their normal form $\mathrm{NF}(\alpha) : s' \to t'$.

Normal Forms for Paths

As mentioned above, for paths $<\alpha_1, \ldots, \alpha_n>$, we do not get a complete reduction system deciding the equivalence. Yet, we merely want to reason about their length, in the light of lemma 5.2.2 below which says that the coproduct monad is not strongly normalizing if there are arbitrarily long paths in the coproduct. Hence we introduce the notion of minimal length— a path is of minimal length if there is no equivalent path which is shorter.

We need to find a sufficient and necessary characterisation for paths of minimal length. This will lead us to the notion of incomposable pairs, two morphisms α, β such that the target of α is equivalent to the source of β (hence they form a path) but α and β are incomposable, and moreover we can't find two morphisms equivalent to α and β each which are composable.

Definition 3.2.11 (Minimal Length) A path $A = <\alpha_1, \ldots, \alpha_n>$ is of *minimal length* iff

(i) All elements of the path are normal forms: for $i = 1, \ldots, n$, $\alpha_i = \mathrm{NF}(\alpha_i)$.

(ii) No equivalent path is shorter: $B \equiv A \;\Rightarrow\; |A| \leq |B|$

In other words, a path $A = \text{<}\alpha_1, \ldots, \alpha_n\text{>}$ is of minimal length if all α_i are in normal form, and α_i and α_{i+1} cannot be composed in $T^w\mathcal{X}$, or are an *incomposable pair*:

Definition 3.2.12 (Incomposable Pair) For α in $T^w\mathcal{X}$, β in $T^v\mathcal{X}$, (α, β) are an *incomposable pair* if both are in normal form and the target of α is equivalent to the source of β, but they cannot be composed, nor are they equivalent to any α', β' which can be composed: $\mathrm{NF}(\delta_t(\alpha)) = \mathrm{NF}(\delta_s(\beta))$ and $\forall \alpha', \beta'.\mathrm{NF}(\alpha') = \mathrm{NF}(\alpha) \wedge \mathrm{NF}(\beta') = \mathrm{NF}(\beta) \;\Rightarrow\; \delta_t(\mathrm{NF}(\alpha)) \neq \delta_s(\mathrm{NF}(\beta))$

A path $A = \text{<}\alpha_1, \ldots, \alpha_n\text{>}$ is of minimal length iff (α_i, α_{i+1}) for $i = 1, \ldots, n-1$ are incomposable pairs. To find a characterisation of incomposable pairs, reconsider the example on page 114 above. There, we had a pair of two morphisms α, ’β which were not composable. This is not an incomposable pair, since ’$\beta \equiv_M$ ’’β, which is composable with α. Hence we modify the example: consider the two monads given by the following two term rewriting systems (omitting the signatures):

$$\begin{aligned} R_1 &= \{\texttt{F(F('}x\texttt{))} \to \texttt{H('}x\texttt{)}\} \\ R_2 &= \{\texttt{G('}y\texttt{)} \to \texttt{'}y\} \end{aligned}$$

Then there is the incomposable pair (α_1, α_2) with

$$\begin{aligned} \alpha &: \texttt{F('G('F('}x\texttt{)))} \to \texttt{F(''F('}x\texttt{))} \\ \beta &: \texttt{F(F('}x\texttt{))} \to \texttt{H('}x\texttt{)} \end{aligned}$$

This situation is prototypical. The point here is that α is layer-collapsing, i.e. there is $y \in \mathcal{T}^{11}\mathcal{X}$ such that the target of α is $\eta^1_{2,1}(y)$. But that alone is not sufficient, since for any rewrite $\alpha' : y \to y'$, there would also be a rewrite $\eta^1_{2,1}(\alpha')$ which is composable with α. The point here is that we can apply μ_1 to y, so the source of β is in the image of μ_1, but that for all $\beta' : y \to y''$, $\mu_1(\beta') \neq \beta$. This means exactly that μ_1 is expanding at y— we say β is μ-expansive.

Of course, there is also the dual situation, in which β would be layer-expanding. The precise characterisation is given by the following lemma:

Definition 3.2.13 (μ-Expansive and μ-Contractive) $\alpha : x \to y$ is said to be *μ-expansive* at $z \in T^{ujjw}\mathcal{X}$, if for its normal form $\mathrm{NF}(\alpha) : x' \to y'$ there are $u, w \in W$, $j \in \mathcal{L}$, with $\mu^u_{j,w}(z) = x'$ s.t. $\mu^u_{j,w}$ is expanding at z: for all $\beta : z \to z'$, $\mu^u_{j,w}(\beta) \neq \mathrm{NF}(\alpha)$.

Dually, $\alpha : x \to y$ is *μ-contractive* at $z \in T^{ujjw}\mathcal{X}$ if for its normal form $\mathrm{NF}(\alpha) : x' \to y'$, we have $\mu^u_{j,w}(z) = y'$ s.t. $\mu^u_{j,w}$ is collapsing at z: for all $\beta : z' \to z$, $\mu^u_{j,w}(\beta) \neq \mathrm{NF}(\alpha)$.

Lemma 3.2.14 Given two $\alpha : x_1 \to y_1$, $\beta : x_2 \to y_2$ in normal form, such that $Q(y_1) = Q(x_2)$. Then (α, β) are an incomposable pair iff

- Either there are $r, s \in W, i, j \in \mathcal{L}, i \neq j$ and $z \in T^{riis}\mathcal{X}$ such that
  - (i) α is layer-collapsing at z, with $y_1 = \eta^{ri}_{j,is}(z)$, and
  - (ii) β is μ-expansive at z, with $x_2 = \mu^r_{i,s}(z)$, and for all $\beta' : z \to z'$, $\mu^r_{i,s}(\beta') \neq \beta$.
- or there are $r, s \in W, i, j \in \mathcal{L}, i \neq j$ and $z \in T^{riis}\mathcal{X}$ such that
  - (i) β is layer-expanding at z, with $x_2 = \eta^{ri}_{j,is}(z)$, and
  - (ii) α is μ-contractive at z, with $y_1 = \mu^r_{i,s}(z)$ andfor all $\beta' : z' \to z$, $\mu^r_{i,s}(\beta') \neq \beta$.

Proof. We first prove sufficiency. Assume the first of the two alternatives (the second is proven analogously), then we have $y_1 \to_\eta z \to_\mu x_2$. Clearly α and β cannot be composed. Suppose there were β' with $\beta' \equiv_M \beta$ which is composable with $\mathrm{NF}(\alpha)$, then β' would be in $T^{rijis}\mathcal{X}$; and since β' and β should be equivalent, and β is a normal form, $\beta' \twoheadrightarrow_{Mor} \beta$. But the only possible reduction would be $\beta' \to_\eta \beta'' \to_\mu \beta$, for some $\beta'' : z \to z''$ with $\mu^w_{j,v}(\beta') = \beta$, which is a contradiction (since β is μ-expansive). On the other hand, if there were α' with $Q\alpha = Q\alpha'$ and α' composable with $\mathrm{NF}(\beta)$, then the rank of α' would be the same as for β, which is $\mathrm{rank}(\alpha) - 2$; so α' cannot reduce to α, and neither the other way around (since α is a normal form), so there there can be no such α'.

For necessity, we know that $Q(y_1) = Q(x_2)$ but $\delta_t(\mathrm{NF}(\alpha)) \neq \delta_s(\mathrm{NF}(\beta))$. Hence $y_1 \neq \delta_t(\mathrm{NF}(\alpha))$ (then by lemma 3.2.10, α is layer-collapsing), or $x_2 \neq \delta_s(\mathrm{NF}(\beta))$ (then by lemma 3.2.10 β is layer-expanding). Assume the former (corresponding to the first clause above— the latter gives the second clause), then (by definition 3.2.9) there are $u, v \in W, i \in \mathcal{L}$ and $z \in T^{uv}\mathcal{X}$ s.t. $y_1 = \eta^u_{j,v}(z)$ (this being clause (i)). Since β is not composable with α, nor equivalent to anything composable with α, the source of β (but not β itself) has to lie in the image of μ; and this in such a way that μ and η are natural not over each other. In other words, there is x s.t. $y \to_\eta z \to_\mu x$, but $x \neq y$ and there is no z' with $y \to_\mu z' \to_\eta x$. This is only the case if $x_2 = \mu^r_{i,s}(z)$ with $i \neq j$, and i and j are not independent, i.e. $w = ri$, $v = is$; and further, β is in the image of μ: for all $\beta' : z \to z'$, $\mu^r_{i,s}(\beta') \neq \beta$. All of this constitutes clause (ii). □

3.3 The Coequalizer of Strongly Finitary Monads

The coequalizer of two strongly finitary monads is constructed as a pointwise colimit, just like the coproduct. Given two monads $\mathsf{T}_1 = \langle T_1, \eta_1, \mu_1 \rangle$, $\mathsf{T}_2 = \langle T_2, \eta_2, \mu_2 \rangle$ on **Cat**,[5] and two monad morphisms $\alpha, \beta : \mathsf{T}_1 \Rightarrow \mathsf{T}_2$. Then for a category $\mathcal{X}$, there is the coequalizer

$$\begin{tikzcd} T_1\mathcal{X} \arrow[r, shift left, "\alpha_{\mathcal{X}}"] \arrow[r, shift right, "\beta\mathcal{X}"'] & T_2\mathcal{X} \arrow[r, "q"] & Q \end{tikzcd}$$

Mapping every $\mathcal{X}$ to the coequalizer Q above would give an endofunctor on **Cat**, but it would fail to be a monad, since one cannot construct a multiplication. To get around this, we take the same approach as for the coproduct: we build a large diagram from all possible combinations of T_1 and T_2 which is weakly ω-filtered, and closed under application of T_1 and T_2. Then applying T_1 or T_2 to the colimit C of the diagram will give rise to a cone over the diagram, and hence a morphism out of T_1C. This way, we can construct a morphism out of the diagram built at the colimit of another diagram— i.e. the constructed endofunctor applied twice— and this will give us the multiplication of the monad.

Clearly, the vertices of the diagram should be same as those for the coproduct diagram— iterated combinations of T_1 and T_2. For the edges, we will need α and β, in any context, under application of T_1 and T_2, and the multiplication of the two monads. Surprisingly, we do not need the units of the two monads.

Formally, the diagram is given by a graph $\mathcal{Q}$, which has

$$\begin{array}{lcl} \text{Edges:} \quad V(\mathcal{Q}) & \stackrel{def}{=} & \mathcal{L}^+ \\ \text{Vertices:} \quad E(\mathcal{Q}) & \stackrel{def}{=} & \{\mathtt{m}^w_{j,v} : wjjv \to wjv \mid w, v \in W, j \in \mathcal{L}\} \cup \\ & & \{\mathtt{a}^w_v : w1v \to w2v \mid w, v \in W\} \cup \\ & & \{\mathtt{b}^w_v : w1v \to w2v \mid w, v \in W\} \end{array}$$

(Recall that $\mathcal{L}^+$ is the set of non-empty words over the alphabet $\mathcal{L}$.) Then, for a category $\mathcal{X} \in$ **Cat**, we define the functor $Q_{\mathcal{X}} : \mathcal{F}(\mathcal{Q}) \to$ **Cat** as the transpose of the graph morphism $q_{\mathcal{X}} : \mathcal{G} \to U(\mathbf{Cat})$, which is given as follows:

$$\begin{array}{rrcl} \text{On the vertices:} & q_{\mathcal{X}}(w) & \stackrel{def}{=} & T^w(\mathcal{X}) \\ \text{On the edges:} & q_{\mathcal{X}}(\mathtt{m}^w_{j,v}) & \stackrel{def}{=} & T^w(\mu_{j,T^v(\mathcal{X})}) \\ & q_{\mathcal{X}}(\mathtt{a}^w_v) & \stackrel{def}{=} & T^w(\alpha_{T^v(\mathcal{X})}) \\ & q_{\mathcal{X}}(\mathtt{b}^w_v) & \stackrel{def}{=} & T^w(\beta_{T^v(\mathcal{X})}) \end{array}$$

[5] Again, these monads are actually enriched, but in this section, we will altogether pass over this fact.

The endofunctor $T : \mathbf{Cat} \to \mathbf{Cat}$, which is the action of the coequalizer monad, maps $\mathcal{X}$ to $colim\, Q_{\mathcal{X}}$, and a functor $F : \mathcal{X} \to \mathcal{Y}$ to the morphism $colim\, Q_{\mathcal{X}} \to colim\, Q_{\mathcal{Y}}$, induced by precomposing the colimiting cone over $Q_{\mathcal{Y}}$ with F; since all components of the diagram are natural transformations, this yields a cone over $Q_{\mathcal{X}}$, and hence a morphism $!_F : colim\, Q_{\mathcal{X}} \to colim\, Q_Y$.

Let $c : Q_{\mathcal{X}} \to colim Q_{\mathcal{X}}$ be the colimiting cone, then the unit η of the monad is given by $\mathcal{X} \xrightarrow{\eta_2} T_2\mathcal{X} \xrightarrow{c_2} colim\, Q_{\mathcal{X}}$. The multiplication μ, as mentioned above, is given by the fact that $Q_{\mathcal{X}}$ is a weakly ω-filtered diagram (which is easy to check — from any vertex, there is a morphism into $T_2\mathcal{X}$ which is sufficient for weak filteredness, and for ω-filteredness, $card(\mathcal{F}(\mathcal{Q})) \leq \aleph_0$). Then $TT\mathcal{X} = colim\, Q_{colim\, Q_{\mathcal{X}}}$, and T_1 preserves weakly ω-filtered colimits: $T_1 colim\, Q_{\mathcal{X}} \cong colim T_1(Q_{\mathcal{X}})$. $T_1(Q_{\mathcal{X}})$ is again in $Q_{\mathcal{X}}$, so the colimiting cone over $Q_{\mathcal{X}}$ gives a colimiting cone over $T_1 Q_{\mathcal{X}}$, and ultimately over $Q_{colim\, Q_{\mathcal{X}}}$ as well, which induces a morphism $colim Q_{colim\, Q_{\mathcal{X}}} \to colim Q_{\mathcal{X}}$, which is the multiplication. The proof of the monad laws then proceeds along the same lines as for the coproduct.

This gives the monad $\mathsf{T} \stackrel{def}{=} \langle T, \eta, \mu \rangle$, and the monad morphism $q : T_2\mathcal{X} \to T\mathcal{X}$ (given by the colimiting cone c at $T_2\mathcal{X}$), both of which form the coequalizer of the two monad morphisms α, β.

It remains to show that it is indeed the coequalizer, which is shown by the universal property. Given any other monad $\mathsf{S} = \langle S, \zeta, \xi \rangle$, and a monad morphism $\gamma : \mathsf{T}_2 \to \mathsf{S}$ such that $\gamma\alpha = \gamma\beta$, there has to be a unique $u : \mathsf{T} \to \mathsf{S}$ such that $uq = \gamma$. This unique u is constructed by giving by a cone $\nu_{\mathcal{X}} : Q_{\mathcal{X}} \to S\mathcal{X}$ (for all categories $\mathcal{X}$), which is defined inductively. The base case is $\nu_1 \stackrel{def}{=} \gamma_{\mathcal{X}}\alpha_{\mathcal{X}} = \gamma_{\mathcal{X}}\beta_{\mathcal{X}}$, and $\nu_2 \stackrel{def}{=} \gamma$. If $\nu_w : T^w\mathcal{X} \to S\mathcal{X}$ is given, we obtain ν_{1w} and ν_{2w} by the following diagram:

$$\begin{tikzcd}
T_1 T^w \mathcal{X} \arrow[r, "T_1\nu_w"] & T_1 S\mathcal{X} \arrow[d, shift right, "\alpha_{S\mathcal{X}}"'] \arrow[d, shift left, "\beta_{S\mathcal{X}}"] & & \\
T_2 T^w \mathcal{X} \arrow[r, "T_2\nu_w"] & T_2 S\mathcal{X} \arrow[r, "\gamma_{S\mathcal{X}}"] & SS\mathcal{X} \arrow[r, "\xi_{\mathcal{X}}"] & S\mathcal{X}
\end{tikzcd}$$

where $\gamma_{S\mathcal{X}} {\cdot} \alpha_{S\mathcal{X}} = \gamma_{S\mathcal{X}} {\cdot} \beta_{S\mathcal{X}}$ because γ coequalizes α and β. Hence, we have constructed the coequalizer of α and β:

Proposition 3.3.1 *Given two strongly finitary monads $\mathsf{T}_1, \mathsf{T}_2$, and two monad morphisms $\alpha, \beta : \mathsf{T}_1 \to \mathsf{T}_2$, the monad $\mathsf{T} = \langle T, \eta, \mu \rangle$ as defined above is their coequalizer in $\mathbf{Mon}_{Fin}(\mathbf{Cat})$.*

And with proposition 1.3.1 (colimits are given by coproducts and coequalizers) and corollary 3.1.10 ($\mathbf{Mon}_{Fin}(\mathbf{Cat})$ has all coproducts), we have the cocompleteness of $\mathbf{Mon}_{Fin}(\mathbf{Cat})$:

Corollary 3.3.2 *The category* $\mathbf{Mon}_{Fin}(\mathbf{Cat})$ *has all small finite colimits.*

3.4 Summary and Conclusion

This chapter started with the claim that "structuring operations are colimits". While not all structuring operations can be expressed with colimits, many important ones can be; and while this may not be the *ultima ratio* it may at least serve as a useful working hypothesis, in particular since we are going to concentrate on one particular operation, the one which hitherto has received most attention in the term rewriting literature: the disjoint union of two term rewriting systems.

The working hypothesis has two ramifications. Firstly, it means we have to look at the colimits in the category of semantic presentations, the category of finitary monads on $\mathbf{Cat}$. To this end, we have given a detailed construction of the coproduct of two monads, and we have sketched the construction of the coequalizer. Together, this gives the existence of all colimits. Secondly, compositionality can now be given a precise meaning: a semantics is compositional if the mapping from syntactic presentations to semantics representations preserves colimits. Here, this is shown by the existence of a right adjoint for the mapping from the category of term rewriting systems to the category of monads, and it means in particular that the coproduct of two monads arising from term rewriting systems Θ_1, Θ_2 is isomorphic to the theory of their disjoint union:

$$\mathsf{T}_{\Theta_1+\Theta_2} \cong \mathsf{T}_{\Theta_1} + \mathsf{T}_{\Theta_2}$$

This result is important because results about the coproduct monad now pertain to the disjoint union — e.g. if we now show that the coproduct is confluent, then this will mean that the disjoint union of two term rewriting systems is confluent as well.

Apart from constructing the coproduct, we have developed a theory about "equality of objects and morphisms in the coproduct". The coproduct essentially consists of *composed terms*, which are elements of $\coprod_{w\in W} T^w\mathcal{X}$, quotiented by an equivalence relation which identifies variables and collapses layers. We have given a decision procedure for this equality, by giving a normal form for the equivalence. This will aid us in reasoning about the coproduct.

Morphisms in the coproduct are equivalence classes of paths of morphisms from $\coprod_{w\in W} T^w\mathcal{X}$; here, our main attention has been the length of these paths, since when considering modularity of strong normalization we will show that the coproduct is strongly normalizing if there is an upper bound on the length of the paths in the coproduct. Instead of developing a normal form for the morphisms, we have developed the notion of *minimal length* for paths — a path being of minimal length if there is no equivalent path which is shorter.

In conclusion, this chapter has completed the demonstration of the compositionality of the semantics, and has set up the technical machinery needed to reason about the modularity of confluence and strong normalization, as we will do in the next two chapters.

Chapter 4

Modularity of Confluence

In this chapter, we will prove a categorical version of Toyama's theorem [88]: the disjoint union of two term rewriting systems is confluent iff the two term rewriting systems are. In the light of the results of the previous chapter, this means that the coproduct of two regular monads is confluent iff the two monads are.

The coproduct monad has been defined as the colimit of the diagram $D_{\mathcal{X}}$. Proposition 1.3.1 from page 16 describes how to construct the colimit of a diagram from coproducts and coequalizers, so much of the proof that the coproduct is confluent can be done in a more abstract setting, investigating the preservation of confluence under the formation of coproducts and colimits in **Cat**. But even before we embark on this, we have to define what it means for a category and a monad to be confluent. This turns out to be a simple extension of the definition of confluence for relations, but it requires a little more effort to show that this new, semantic definition is equivalent to the old, syntactic one.

It should be stressed that the parts of this chapter dealing with the preservation of confluence are solely concerned with regular monads, and nowhere depend on the fact that these monads arise from term rewriting systems (even if we use term rewriting systems and signatures as examples.) This allows us to prove a slightly more general theorem than the original [88].

Structure of the Chapter

- In §4.1, we will define confluence for categories and monads on **Cat**. We call this the *semantic* notion of confluence. We will show that the semantic notion coincides with the syntactic notion, i.e. the monad T_Θ is confluent according to our definition iff the term rewriting system Θ is.

- In §4.2, we will consider the preservation of confluence under formation of coequalizers in **Cat**. (Preservation of confluence for the coproduct of two categories is trivial.) We will introduce the notion of a *witness* of a term, and show that the coequalizer preserves confluence if any two equivalent objects have a common witness.

- We will then apply this technique in §4.3 to the coequalizer which gives us the colimit of the diagram $D_{\mathcal{X}}$. We show that the normal form of a term t with respect to the induced equivalence, as defined in §3.2.2, is a witness of t. We then conclude that the coequalizer of $D_{\mathcal{X}}$ preserves confluence, and hence the coproduct of two regular non-expanding confluent monads is confluent, the main result of this chapter.

- In §4.4, we will drop the non-expanding requirement and investigate quasi-non-expanding monads (see §2.4.4 on page 85). Extending the main result of §4.3, we will show that the coproduct of two regular quasi-non-expanding confluent monads is confluent.

4.1 A Semantic Definition of Confluence

We will first extend the well-known definition of confluence for preorders to confluence of categories and monads. We then show that this semantic definition coincides with the usual, syntactic definition which talks about derivations of terms — i.e. that the monad given by a term rewriting system Θ is confluent according to our new, semantic definition if and only if the system Θ is confluent according to the old, syntactic definition. This will form proposition 4.1.7, the main result of this section.

4.1.1 Confluent Monads

Recall from §1.6.2 on page 46 that a preorder $X = (X_0, \geq)$ is *confluent* if for all elements $x, y, z \in X_0$ such that $x \geq y$ and $x \geq z$ (called *spans* or *co-initial cells*), there is a completion: an element $u \in X_0$ such that $y \geq u$, $z \geq u$.

When extending this definition to categories, one has to decide whether this completion is required to commute; i.e. given α, β in diagram 4.1, we will always require γ, δ to exist, but do we require diagram 4.1 to commute? We will in

$$\begin{tikzcd} & x \arrow[dl, "\alpha"'] \arrow[dr, "\beta"] & \\ y \arrow[dr, "\gamma"'] & & z \arrow[dl, "\delta"] \\ & u & \end{tikzcd} \tag{4.1}$$

general require it not to commute, since this is closer to what is happening with term rewriting systems; if it always commutes, we will call the category commuting confluent (and we can show modularity for both confluence and commuting confluence). Confluence of a category $\mathcal{X}$ coincides with the confluence of the associated preorder $J(\mathcal{X})$, obtained by identifying all morphisms with the same source and target (see page 43).

Further, a term rewriting system $\Theta = (\Omega, R)$ is confluent if all spans in $\twoheadrightarrow_R$ can be completed; but here we have to assume that all spans of variable rewrites can be completed by variable rewrites, which is equivalent to the preorder X being confluent. Hence, a monad T is confluent if TX is confluent whenever X is.

Definition 4.1.1 (Confluence) A category $\mathcal{C}$ is *confluent* if for any two morphisms $\alpha : x \to y, \beta : x \to z$ there are morphisms $\gamma : y \to u, \delta : z \to u$. $\mathcal{C}$ is *commuting confluent* if additionally $\gamma \cdot \alpha = \delta \cdot \beta$.

A monad $\mathsf{T} = \langle T, \eta, \mu \rangle$ on **Cat** is *(commuting) confluent* if for all (commuting) confluent categories $\mathcal{X}$, $T\mathcal{X}$ is (commuting) confluent.

Our definition coincides with Stell's [82, pg. 105], but is different from Hilken [29], where confluence is what is here called commuting confluence, and from Jay [31], where a confluent category is a category enriched over confluent orders.

The reason for introducing two different concepts here (confluence *vs.* commuting confluence) is that commuting confluence is strictly stronger than confluence. For example, the term rewriting system $\mathcal{G} = (\Sigma_{\mathcal{G}}, R_{\mathcal{G}})$ in figure 4.1 is confluent,[1] but not commuting confluent: the span in diagram 4.2 (obtained by applying

$$
\begin{array}{rcll}
\Sigma_{\mathcal{G}} & = & \{\ \mathtt{e}_0, \mathtt{I}_1, \mathtt{o}_2\ \} & \\
R_{\mathcal{G}} & = & \{\ \mathtt{o}(\mathtt{e}, \text{'}x) \to \text{'}x, & \mathtt{o}(\mathtt{I}(\text{'}x), \text{'}x) \to \text{'}x, \\
 & & \quad \mathtt{o}(\mathtt{o}(\text{'}x)\text{'}y, \text{'}z) \to \mathtt{o}(\text{'}x, \mathtt{o}(\text{'}y, \text{'}z)), & \mathtt{o}(\mathtt{I}(\text{'}x), \mathtt{o}(\text{'}x, \text{'}z)) \to \text{'}z, \\
 & & \quad \mathtt{o}(\text{'}y, \mathtt{e}) \to \text{'}y, & \mathtt{I}(\mathtt{I}(\text{'}x)) \to \text{'}x, \\
 & & \quad \mathtt{I}(\mathtt{e}) \to \mathtt{e}, & \mathtt{o}(\text{'}x, \mathtt{I}(\text{'}x)) \to \mathtt{e}, \\
 & & \quad \mathtt{I}(\mathtt{o}(\text{'}x, \text{'}y)) \to \mathtt{o}(\mathtt{I}(\text{'}x), \mathtt{I}(\text{'}y)), & \mathtt{o}(\text{'}y, \mathtt{o}(\mathtt{I}(\text{'}y), \text{'}x)) \to \text{'}x\}
\end{array}
$$

Figure 4.1: A Confluent Term Rewriting System for Group Theory

[1] This may not be evident on first glance, if one has not seen this term rewriting system before, but it is in fact a rather famous confluent system, obtained by orienting the equations of group theory as term rewrite rules, and making the system confluent by adding in additional rules, the so-called Knuth-Bendix completion [43, 41].

the second and third rule respectively) can be completed again (by the first and fourth rule, respectively), but this diagram does not commute in the monad $\mathsf{T}_{\mathcal{G}}$ on

$$\begin{tikzcd}[column sep=small] & \texttt{o(o(I('}x\texttt{),'}x\texttt{),'}z\texttt{)} \arrow[dl] \arrow[dr] & \\ \texttt{o(e,'}z\texttt{)} \arrow[dr] & & \texttt{o(I('}x\texttt{),o('}x\texttt{,'}z\texttt{))} \arrow[dl] \\ & \texttt{'}z & \end{tikzcd} \tag{4.2}$$

Cat given by the term named-reduction algebra (see definition A.1.8 on page 183 below).

On the other hand, commuting confluence is a stronger property; a term rewriting system is commuting confluent if it is confluent without overlapping rules (hence no spans like in diagram 4.2 above), and if we can show modularity of commuting confluence, this is a different result than modularity of confluence; hence, we keep both notions, and show modularity of both. Essentially, in our proofs if we only make sure there is at least one completion of any span, we can show modularity of confluence, and for modularity of commuting confluence, we need to do some more work and show the completions actually commute.

4.1.2 Confluent Term Rewriting Systems

The above definition of confluent monad was preceded by a sentence arguing why this definition is the one "making sense". If we want to be able to relate results about confluent monads to confluent term rewriting systems, we have to formally show that these two notions coincide by showing that an ordinary TRS Θ is confluent by the syntactic definition from §1.6.2 if and only if the monad T_Θ is confluent by the semantic definition 4.1.1.

Recall from definition 1.6.1 that $\to_R$ is the one-step reduction relation induced by a term rewriting system, $\twoheadrightarrow_R$ the many-step reduction relation, and from §1.6.2 on page 46 that an (ordinary) term rewriting system $\Theta = (\Omega, R)$ is confluent if $\twoheadrightarrow_R$ is confluent. Further, recall from definition 2.3.13 that $\to_X$ the congruence closure of the variable rewrites for a preorder X, and from proposition 2.3.14 that all reductions in $T_\Theta(X)$ are given as sequences of either one-step reductions, or variable rewrites:

$$T_\Theta(X) \;=\; (\to_R \cup \to_X)^*$$

It is easy to see that if T_Θ is confluent, then Θ is confluent (note that a discrete preorder is confluent, and use corollary 2.3.15: $T_\Theta(X) = \twoheadrightarrow_R$ if X is discrete). In the other direction, we use proposition 2.3.14. By assumption, both $\to_X$ and $\to_R$ are confluent. Unfortunately, they do not commute, that is we do not have a completion property like the diagram 4.3 otherwise we could use the Hindley-Rosen

$$\begin{tikzcd} & x \arrow[dl, "X"'] \arrow[dr, "R"] & \\ y \arrow[dr, "R"'] & & z \arrow[dl, "X"] \\ & u & \end{tikzcd} \tag{4.3}$$

lemma 1.6.3. Let us first see why they do not commute: given the term rewriting system $D = (\{\mathtt{G}, \mathtt{F}\}, \mathtt{F}(\texttt{'}x, \texttt{'}x) \to \mathtt{G}(\texttt{'}x))$, and the preorder $X = (\{a, b, c, d\}, \geq)$, ordered as $a \geq b, a \geq c, b \geq d, c \geq d, a \geq d$ (both of which are obviously confluent), then there is the span in (4.4) which cannot be completed as above. (In

$$\begin{tikzcd} & \mathtt{F}(\texttt{'}a, \texttt{'}a) \arrow[dl, "X"'] \arrow[dr, "R"] & \\ \mathtt{F}(\texttt{'}b, \texttt{'}c) & & \mathtt{G}(\texttt{'}a) \end{tikzcd} \tag{4.4}$$

the term $\mathtt{F}(\texttt{'}x, \texttt{'}x)$, $\texttt{'}x$ is sometimes called a δ-redex; and in the span above, the rewrite on the left is said to destroy δ-redexes). We can complete spans like this by first adding another variable rewrite to the rewrite on the left.

This means that $\to_R$ is *strongly extendable* over $\to_X$ as defined by Kahrs [34, pg 19ff]. He shows that if two relations are confluent, and one is strongly extendable over the other, which additionally is strongly normalizing, then their union is confluent as well. Here, we can not assume that $\to_X$ is strongly normalizing, but we can show that the additional rewrites do not destroy any more δ-redexes and that they (but only they) commute with $\to_R$. Somewhat lost for words, as one tends to be when christening new exotic and rather abstract concepts, we say that $\to_X$ is *coherently extendable* over $\to_R$, and we can prove a variation of Kahrs' lemma that the union of two confluent relations, in which one is coherently extendable over the other, is confluent.

Definition 4.1.2 (Coherent Extendability) A relation $\to_R \subseteq X \times X$ is *coherently extendable* over $\to_S \subseteq X \times X$, if

(i) there is a subrelation $\to_{S_0} \subseteq \to_S$ (called the *commuting subrelation*) that commutes with $\to_R$

(ii) and for all $x, y, z \in X$ such that $x \twoheadrightarrow_R y$, $x \twoheadrightarrow_S z$, there are $u, v \in X$ such that

$$\begin{tikzcd}
x \arrow[r, ->>, "S"'] \arrow[dd, ->>, "R"'] & z \arrow[d, ->>, "S_0"] \\
 & u \arrow[d, ->>, "R"] \\
y \arrow[r, ->>, "S"'] & v
\end{tikzcd}$$

Lemma 4.1.3 Given two confluent relations $\rightarrow_R, \rightarrow_S \subseteq X \times X$ such that $\rightarrow_R$ is coherently extendable over $\rightarrow_S$ and the commuting subrelation $\rightarrow_{S_0}$ is confluent, then $\rightarrow_R \cup \rightarrow_S$ is confluent as well.

Proof. Recall that by equation 1.24 from §1.5.2 we have

$$(\rightarrow_R \cup \rightarrow_S)^* = (\twoheadrightarrow_S; \twoheadrightarrow_R)^* \tag{4.5}$$

Diagram 4.6 shows that $\twoheadrightarrow_S; \twoheadrightarrow_R$ commutes: the squares (2) are the coherent extendability of $\rightarrow_R$ over $\rightarrow_S$, square (1) is confluence of $\rightarrow_S$, square (3) is the

$$\begin{tikzcd}
\cdot \arrow[r, ->>, "S"] \arrow[d, ->>, "S"'] \arrow[dr, phantom, "(1)"] & \cdot \arrow[rr, ->>, "R"] \arrow[d, ->>, "S"'] \arrow[drr, phantom, "(2)"] & & \cdot \arrow[d, ->>, "S"] \\
\cdot \arrow[r, ->>, "S"] \arrow[dd, ->>, "R"'] \arrow[ddr, phantom, "(2)"] & \cdot \arrow[r, ->>, "S_0"] \arrow[d, ->>, "S_0"'] \arrow[dr, phantom, "(3)"] & \cdot \arrow[r, ->>, "R"] \arrow[d, ->>, "S_0"'] \arrow[dr, phantom, "(4)"] & \cdot \arrow[d, ->>, "S"] \\
 & \cdot \arrow[r, ->>, "S_0"] \arrow[d, ->>, "R"'] \arrow[dr, phantom, "(4)"] & \cdot \arrow[r, ->>, "R"] \arrow[d, ->>, "R"'] \arrow[dr, phantom, "(5)"] & \cdot \arrow[d, ->>, "R"] \\
\cdot \arrow[r, ->>, "S"'] & \cdot \arrow[r, ->>, "S"'] & \cdot \arrow[r, ->>, "R"'] & \cdot
\end{tikzcd} \tag{4.6}$$

confluence of the commuting subrelation $\rightarrow_{S_0}$, square (4) is $\rightarrow_R$ commuting with $\rightarrow_{S_0}$, and (5) is confluence of $\rightarrow_R$.

Hence $(\twoheadrightarrow_S; \twoheadrightarrow_R)^*$ is confluent, and any relation P such that $P^* = (\twoheadrightarrow_S; \twoheadrightarrow_R)^*$, so by equation 4.5, $\rightarrow_R \cup \rightarrow_S$ is confluent. ☐

4.1.3 Equivalence of the Two Notions

We can now use lemma 4.1.3 to show that the semantic definition 4.1.1 of confluence coincides with the syntactic definition from §1.6.2. We will first exhibit the commuting subrelation of $\to_X$, and then show that $\to_R$ is coherently extendable over $\to_X$. First, note that $\to_X$ is confluent if X is; this is shown by structural induction. Then the commuting subrelation will consist of the congruence closure of those variable rewrites which do not destroy any δ-redexes, or in other words do no rewrite the same variable to different variables:

Definition 4.1.4 Let $\to_X^C$ be the smallest relation on $T_\Omega(X)$ such that $t \to_X^C s$ iff $s = t[y_1/x_1, \ldots, y_n/x_n]$ with $y_i, x_i \in X$ and $x_i \geq y_i$ for $i = 1, \ldots, n$.

That $\to_X^C$ is confluent is fairly easy to see, since any span can be completed pointwise in the variables of the substitution. To show that it commutes with $\to_R$, assume that there is a term $t \in T_\Omega(X)$ such that $t = C[\sigma(l)]$ for a rule $(Y \vdash l \to r) \in R$ and a substitution $\sigma : Y \to T_\Omega(X)$, and that $x_i \geq y_i$ for $i = 1, \ldots, n$. Let $s \stackrel{def}{=} C[\sigma(r)]$ and we have a span $t \to_R s$, $t \to_X^C t[y_i/x_i]_{i=1,\ldots,n}$. Then there is a completion $s \to_X^C s[y_i/x_i]_{i=1,\ldots,n}$ and $C[\sigma(l)][y_i/x_i] \to_R C[\sigma(r)][y_i/x_i]$. From the following lemma we obtain the coherent extendability of $\to_R$ over $\to_X$. It shows that any rewrite in $\to_X$ can be extended by one from $\to_X^C$ into one from $\to_X^C$ by pointwise completing all destroyed δ-redexes with rewrites which do not destroy δ-redexes themselves.

Lemma 4.1.5 For all $s, t \in T_\Omega(X)$ such that $t \to_X s$, there is $u \in T_\Omega(X)$ such that $s \to_X^C u$ and $t \to_X^C u$.

Proof. Let $X = \{x_1, \ldots, x_n\}$. We are going to construct a sequence $s'_0, \ldots, s'_n \in T_\Omega(X)$ of terms, starting with $s'_0 = s$, and ending with $s'_n = u$. For $1 \geq i \geq n$, let $R(x_i) \stackrel{def}{=} \{y \in X \mid x_i \geq y\}$ (this can be thought of as the set of reducts of x_i). Then by confluence of X, there is $z_i \in X$ such that $\forall y \in R(x_i).\, y \geq z_i$ (i.e. we can complete any spans created by the different reducts of x_i). Let $s'_i = s'_{i-1}[z/y_1, \ldots, z/y_m]$ with $R(x_i) = \{y_1, \ldots, y_m\}$, then $s'_{i-1} \to_X^C s'_i$.

Let $u \stackrel{def}{=} s'_n$, then $s \to_X^C s'_1 \to_X^C s'_2 \ldots s'_n = u$; and further, $t \to_X^C u$, since $u = t[z_i/x_i]_{i=1,\ldots,n}$ and $x_i \geq y_i$. □

Lemma 4.1.6 The term rewriting system $\Theta = (\Omega, R)$ is confluent iff if the monad T_Θ on **Pre** is confluent.

Proof. We have to show that the term rewriting system $\Theta = (\Omega, R)$ is confluent iff the term reduction algebra $T_\Theta(X)$ is confluent whenever the preorder X is confluent. Right-to-left, by corollary 2.3.15 a discrete preorder X is always confluent, hence $T_\Theta(X)$ will be confluent, and so will be $\twoheadrightarrow_R$.

Left-to-right, assume that $\twoheadrightarrow_R$ is confluent and that X is confluent. From lemma 4.1.5 it follows that $\rightarrow_R$ is coherently extendable over $\rightarrow_X$, with $\rightarrow_X^C$ being the commuting subrelation. Since $\rightarrow_X^C$ is confluent and commutes with $\rightarrow_R$, by lemma 4.1.3 $(\rightarrow_R \cup \rightarrow_X)^*$ is confluent, and by proposition 2.3.14 $(\rightarrow_R \cup \rightarrow_X)^* = T_\Theta(X)$, hence $T_\Theta(X)$ is confluent. $\square$

The previous lemma talks about the monad T_Θ on **Pre** given by the term reduction algebra; it remains to make the final link to the monad on **Cat**, which is given by the *named reduction algebra* (definition A.1.8), and show that confluence of that monad coincides with confluence of Θ.

Proposition 4.1.7 *The term rewriting system $\Theta = (\Omega, R)$ is confluent iff the monad T_Θ on* **Cat** *is confluent.*

Proof. We show that the monad T_Θ on **Cat** is confluent iff the monad T_Θ on **Pre** is confluent, and use lemma 4.1.6.

Left-to-right, given a confluent preorder X, we have to show the term reduction algebra $T_\Theta(X)$ is confluent. By equation 1.25 we have $J(I(X)) = X$ (where $I : \mathbf{Pre} \rightarrow \mathbf{Cat}$ is the inclusion of preorders into categories, see page 42), hence if X is confluent, so is $I(X)$ and hence because the monad T_Θ on **Cat** is confluent, $T_\Theta(I(X))$ is confluent, which means $J(T_\Theta(I(X)))$ is confluent, and by equations A.15 and 1.25, $J(T_\Theta(I(X))) = T_\Theta(J(I(X))) = T_\Theta(X)$, hence $T_\Theta(X)$ is confluent as required.

Right-to-left, given a confluent category $\mathcal{X}$, we know $J(\mathcal{X})$ is confluent, and hence $T_\Theta(J(\mathcal{X}))$, and by equation A.15, $T_\Theta(J(\mathcal{X})) = J(T_\Theta(\mathcal{X}))$, hence $T_\Theta(\mathcal{X})$ is confluent, making the monad T_Θ confluent. $\square$

That the implication "Θ confluent $\Rightarrow$ T_Θ confluent" is harder to prove than the other way is an indication that we have managed to incorporate more properties into the definition without actually strengthening it. The definition has become less axiomatic, but easier to work with — something one would like from a semantic definition.

4.2 Preservation of Confluence for Coequalizers

We have postulated above that most structuring operations on term rewriting systems are constructed as colimits. In our category of semantic representations, these are colimits of monads, which in turn are constructed pointwise as colimits of diagrams in **Cat**. Reasoning about preservation of confluence for any structuring operation then amounts to reasoning about preservation of confluence for colimits of certain diagrams like $D_{\mathcal{X}}$ for the coproduct. The colimit of these diagrams can be calculated using coproducts and coequalizers in **Cat** (proposition 1.3.1). Breaking down the reasoning further, we have to reason about preservation of confluence for coproducts and coequalizers. That coproducts preserve confluence is nearly trivial, since the coproduct does not identify any objects or morphisms. This leaves us with the preservation of confluence under coequalizers, which will be the scope of this section. In the next section, we will apply the techniques developed in this section to prove the modularity of confluence for the coproduct of two monads.

We will attempt to give an abstract characterisation (of the categories and functors involved) which makes the coequalizer of two functors F, G confluent:

$$\mathcal{X} \underset{G}{\overset{F}{\rightrightarrows}} \mathcal{Y} \xrightarrow{Q} \mathcal{Z}$$

Recall from §1.5.3 on page 44 that the coequalizer $\mathcal{Z}$ of the two functors F, G is a category which has as objects equivalence classes of objects from $\mathcal{Y}$, and as morphisms (equivalence classes of) paths in a graph $\mathcal{Z}_0$, the edges of which are morphisms from $\mathcal{Y}$. We first show a lemma which states that $\mathcal{Z}$ will be confluent if $\mathcal{Y}$ is, and we can complete any span of two morphisms in $\mathcal{Y}$ the sources of which are equivalent in the equivalence relation induced on the objects: this property will be referred to as the *diamond property*, since it corresponds to a "one-step completion property modulo F". We then introduce a technique which can be used to show that the coequalizer (or more precisely, the category $\mathcal{Y}$ with respect to the coequalizer) has said diamond property.

Further, we can systematically strengthen the obtained properties to be sufficient for the preservation of commuting confluence. This will allow us to prove modularity of both confluence and commuting confluence in the same framework.

4.2.1 The Diamond Property

Definition 4.2.1 (Diamond Property) Given a functor $F : \mathcal{Y} \to \mathcal{Z}$, the category $\mathcal{Y}$ has the *diamond property with respect to* F, written $\mathcal{Y} \models_F \Diamond$, if for all morphisms $\alpha : x \to x'$, $\beta : y \to y'$ in $\mathcal{Y}$ such that $Fx = Fy$ there are morphisms $\gamma : v \to v', \delta : w \to w'$ in $\mathcal{Y}$ such that $Fx' = Fv$, $Fy' = Fw$ and $Fv' = Fw$.

If additionally $F\gamma{\cdot}F\alpha = F\delta{\cdot}F\beta$, then we say $\mathcal{Y}$ has the *strong diamond property with respect to* F, written $\mathcal{Y} \models_F \blacklozenge$.

The idea of the following lemma is that to complete a span of two paths $p = \langle\alpha_1, \ldots, \alpha_n\rangle$ and $q = \langle\beta_1, \ldots, \beta_m\rangle$ with equivalent sources it is sufficient to be able to "tile" the squares in figure 4.2 (except that sources and targets of the squares are merely equivalent, not equal). This tiling is equivalent to two inductions over both the length of p and q. For commuting confluence, figure 4.2 additionally has to be commute up to equivalence.

$$
\begin{tikzcd}
{} \arrow[r, "\alpha_1"] \arrow[d, "\beta_1"'] & {} \arrow[r, "\alpha_2"] \arrow[d, "\beta'_1"] & {} \arrow[d, "\beta''{}_1"] \arrow[r, phantom, "\cdots"] & {} \arrow[r, "\alpha_n"] \arrow[d, "\beta_1^{(n-1)}"] & {} \arrow[d, "\beta_1^{(n)}"] \\
{} \arrow[r, "\alpha'_1"'] \arrow[d, "\beta_2"'] & {} \arrow[r, "\alpha'_2"'] \arrow[d, "\beta'_2"] & {} \arrow[d, "\beta''_2"] \arrow[r, phantom, "\cdots"] & {} \arrow[r, "\alpha'_n"'] \arrow[d, "\beta_2^{(n-1)}"] & {} \arrow[d, "\beta_2^{(n)}"] \\
{} \arrow[r, "\alpha''_1"'] \arrow[d, phantom, "\vdots"] & {} \arrow[r, "\alpha''_2"'] \arrow[d, phantom, "\vdots"] & {} \arrow[d, phantom, "\vdots"] \arrow[r, phantom, "\cdots"] & {} \arrow[r, "\alpha''_n"'] \arrow[d, phantom, "\vdots"] & {} \arrow[d, phantom, "\vdots"] \\
{} \arrow[r, "\alpha_1^{(m-1)}"] \arrow[d, "\beta_m"'] & {} \arrow[r, "\alpha_2^{(m-1)}"] \arrow[d, "\beta'_m"] & {} \arrow[d, "\beta''_m"] \arrow[r, phantom, "\cdots"] & {} \arrow[r, "\alpha_n^{(m-1)}"] \arrow[d, "\beta_m^{(n-1)}"] & {} \arrow[d, "\beta_m^{(n)}"] \\
{} \arrow[r, "\alpha_1^{(m)}"'] & {} \arrow[r, "\alpha_2^{(m)}"'] & {} \arrow[r, phantom, "\cdots"] & {} \arrow[r, "\alpha_n^{(m)}"'] & {}
\end{tikzcd}
$$

Figure 4.2: Completing sequences of morphisms by "tiling"

Lemma 4.2.2 (The Tiling Lemma) Let $Q : \mathcal{Y} \to \mathcal{Z}$ be the coequalizer of two functors $F, G : \mathcal{X} \to \mathcal{Y}$ in **Cat** as given in §1.5.3. If $\mathcal{Y}$ is confluent and $\mathcal{Y} \models_Q \lozenge$, then $\mathcal{Z}$ is confluent; if $\mathcal{Y}$ is commuting confluent and $\mathcal{Y} \models_Q \blacklozenge$, then $\mathcal{Z}$ is commuting confluent.

Proof. To show confluence, we show that given two paths p, q such that $\mathtt{s}(p) = \mathtt{s}(q)$, there are paths r, s such that $\mathtt{t}(p) = \mathtt{s}(r)$, $\mathtt{t}(q) = \mathtt{s}(s)$ and $\mathtt{t}(s) = \mathtt{t}(t)$. For commuting confluence, we show that additionally $p{::}r \equiv_M q{::}s$ (where $\equiv_M$ is the equivalence on morphisms induced by Q, see §1.5.3). Both are proven by induction over the lengths of the two paths p and q.

We first show that for all $p = \langle \alpha_1, \ldots, \alpha_n \rangle$ and $\beta \in \mathcal{Y}(x, y)$ such that $[x] = \mathtt{s}(p)$, there is a morphism β' and a path $\langle \alpha'_1, \ldots, \alpha'_n \rangle$ such that $p' \stackrel{def}{=} \langle \alpha_1, \ldots, \alpha_n, \beta' \rangle$ and $q' \stackrel{def}{=} \langle \beta, \alpha'_1, \ldots, \alpha'_n \rangle$ are both paths with the same target: $\mathtt{t}(p') = \mathtt{t}(q')$; and under the additional assumption $\mathcal{Y} \models_Q \blacklozenge$, that $p' \equiv_M q'$.

This is shown by induction on n. The induction base is $n = 0$, in which case $\beta' \stackrel{def}{=} \beta$. For the induction step, given a path $p = \langle \alpha_1, \ldots, \alpha_n \rangle$ and $\beta \in \mathcal{Y}(x, y)$ with $[x] = \delta_s(\alpha_1)$, since $\mathcal{Y} \models_Q \lozenge$ there are morphisms γ, δ in $\mathcal{Y}$ s.t. $Q(\delta_t(\alpha_1)) = Q(\delta_s(\gamma))$, $Q(y) = Q(\delta_s(\delta))$ and $Q(\delta_t(\gamma)) = Q(\delta_t(\delta))$, i.e. $\langle \alpha_1, \gamma \rangle$ and $\langle \beta, \delta \rangle$ are paths with the same target. We can apply the induction assumption to γ and $\langle \alpha_2, \ldots, \alpha_n \rangle$ (since the latter is only of length $n-1$), and obtain morphisms $\gamma', \alpha'_2, \ldots, \alpha'_n$ such that $p' \stackrel{def}{=} \langle \alpha_2, \ldots, \alpha_n, \gamma' \rangle$ and $q' \stackrel{def}{=} \langle \gamma, \alpha'_2, \ldots, \alpha'_n \rangle$ are paths with the same target. Then the desired completions are γ' and $\langle \delta, \alpha'_2, \ldots, \alpha'_n \rangle$, with $\mathtt{t}(\langle \beta, \delta, \alpha'_2, \ldots, \alpha'_n \rangle) = Q(\delta_t(\alpha'_n)) = Q(\delta_t(\gamma')) = \mathtt{t}(\langle \alpha_1, \ldots, \alpha_n, \gamma' \rangle)$ as required.

Under the stronger assumption $\mathcal{Y} \models_Q \blacklozenge$, we have that above

$$Q\gamma \cdot Q\alpha_1 = Q\delta \cdot Q\beta \Leftrightarrow \langle \alpha_1, \gamma \rangle \equiv_M \langle \beta, \delta \rangle$$

The induction assumption is strengthened to

$$\langle \alpha_2, \ldots, \alpha_n, \gamma' \rangle \equiv_M \langle \gamma, \alpha'_2, \ldots, \alpha'_n \rangle$$

and the desired completions are equivalent as follows:

$$\begin{aligned} \langle \alpha_1, \alpha_2, \ldots, \alpha_n, \gamma' \rangle &\equiv_M \langle \alpha_1, \gamma, \alpha'_2, \ldots, \alpha'_n \rangle \\ &\equiv_M \langle \beta, \delta, \alpha'_2, \ldots, \alpha'_n \rangle \end{aligned}$$

Now we can do an induction on the length m of the second sequence. The induction base is $m = 0$, which is trivial. For $m > 0$, since $\mathtt{s}(p) = \mathtt{s}(q)$, we can use the previous result to obtain β' and a path $\alpha'_1, \ldots, \alpha'_n$ such that $\langle \alpha_1, \ldots, \alpha_n, \beta' \rangle$ and $\langle \beta_1, \alpha'_1, \ldots, \alpha'_n \rangle$ are paths with the same target. The induction assumption is that for $p' \stackrel{def}{=} \langle \beta_2, \ldots, \beta_m \rangle$ and $q' \stackrel{def}{=} \langle \alpha'_1, \ldots, \alpha'_n \rangle$ there are paths p'', q'' such that $\mathtt{t}(p') = \mathtt{s}(p'')$, $\mathtt{t}(q') = \mathtt{s}(q'')$ and $\mathtt{t}(p'') = \mathtt{t}(q'')$. Then the desired completions are $r \stackrel{def}{=} \langle \beta' \rangle :: q''$ and $s \stackrel{def}{=} p''$.

This concludes that proof that $\mathcal{Z}$ is confluent if $\mathcal{Y} \models_Q \lozenge$. Under the stronger assumption that $\mathcal{Y} \models_Q \blacklozenge$, we can (in the induction step) use the previous result to obtain

$$\langle \alpha_1, \ldots, \alpha_n, \beta' \rangle \equiv_M \langle \beta_1, \alpha'_1, \ldots, \alpha'_n \rangle$$

Then the induction assumption is that $p' :: p'' \equiv_M q' :: q''$, and hence we have

$$\begin{aligned} p :: r &\equiv_M <\alpha_1, \ldots, \alpha_n, \beta'> :: q'' \\ &\equiv_M <\beta_1, \alpha'_1, \ldots, \alpha'_n> :: q'' \\ &\equiv_M <\beta_1> :: q' :: q'' \\ &\equiv_M <\beta_1> :: p' :: p'' \\ &\equiv_M <\beta_1, \beta_2, \ldots, \beta_m> :: p'' \\ &\equiv_M q :: s \end{aligned}$$

This means that the completion commutes, and hence $\mathcal{Z}$ is commuting confluent.

□

4.2.2 Witnesses

In this section, we introduce the notion of a witness, and show how it can be used to assert the diamond property as defined above. The idea is that any span in $\mathcal{Z}$ comes from two morphisms $\alpha : x \to x'$, $\beta : y \to y'$ in $\mathcal{Y}$ such that $Qx = Qy$; we call this a *span under* Q. Being able to complete any such span under Q is equivalent to $\mathcal{Y}$ having the diamond property w.r.t. Q.

In order to find a completion of α and β, we find a span in $\mathcal{Y}$ of two morphisms $\alpha' : z \to z', \beta' : z \to z''$ which are equivalent to α and β respectively, then use the confluence of $\mathcal{Y}$ to complete the span in $\mathcal{Y}$, and Q will preserve the completion. The object z is constructed by introducing a *witness*. We say an object z witnesses an object x if for any rewrite α out of x there is a rewrite α' out of z which is equivalent (i.e. $Q\alpha = Q\beta$). If we can prove that any two equivalent objects have the same witness, then we will be able to complete any spans under Q, and obtain confluence; if we can even prove that the completion commutes, we will obtain commuting confluence.

Definition 4.2.3 (Witness) Given a functor $F : \mathcal{X} \to \mathcal{Y}$, and two objects $x, y \in \mathcal{X}$, x is a *witness of* y *with respect to* F, written $x \,\mathrm{wit}_F\, y$ if $Fx = Fy$ and for all morphisms $\beta : y \to y'$ in $\mathcal{X}$, there is a morphism $\alpha : x \to x'$ in $\mathcal{X}$ such that $F\alpha = F\beta$.

Lemma 4.2.4 Given the coequalizer $Q : \mathcal{Y} \to \mathcal{Z}$ of two functors $F, G : \mathcal{X} \to \mathcal{Y}$, if $\mathcal{Y}$ is (commuting) confluent and for all $x, y \in \mathcal{Y}$ s.t. $Qx = Qy$ there is a common witness $z \in \mathcal{Y}$ such that $z \,\mathrm{wit}_Q\, x$ and $z \,\mathrm{wit}_Q\, y$ then $\mathcal{Z}$ is (commuting) confluent.

Proof. We show that $\mathcal{Y} \models_Q \lozenge$ if $\mathcal{Y}$ is confluent, and $\mathcal{Y} \models_Q \blacklozenge$ if $\mathcal{Y}$ is commuting confluent, and then use lemma 4.2.2.

Given $\alpha : x \to x'$, $\beta : y \to y'$ such that $Qx = Qy$. Then since x and y have a common witness $z \in \mathcal{Y}$ with $Qz = Qy = Qx$, there are $\alpha' : z \to z'$, $\beta' : z \to z''$ such that $Q\alpha' = Q\alpha, Q\beta' = Q\beta$. α' and β' form a span in $\mathcal{Y}$, which by confluence of $\mathcal{Y}$ has a completion $\gamma : z' \to u, \delta : z'' \to u$ with $Qz = Qx'$ and $Qz' = Qy'$ as required; hence $\mathcal{Y} \models_Q \lozenge$. If further $\mathcal{Y}$ is commuting confluent, we have $\gamma \cdot \alpha' = \delta \cdot \beta'$, hence (since Q preserves composition) $Q\gamma \cdot Q\alpha' = Q\delta \cdot Q\beta'$, hence $Q\gamma \cdot Q\alpha = Q\delta \cdot Q\beta$, so $\mathcal{Y} \models_Q \blacklozenge$. □

We will now put this machinery to work with the diagram $D_{\mathcal{X}}$ to show modularity of confluence and commuting confluence for the coproduct of two regular, non-expanding monads.

4.3 Toyama's Theorem

We are now going to prove modularity of confluence: given two regular, non-expanding monads $\mathsf{T}_1 = \langle T_1, \eta_1, \mu_1 \rangle$ and $\mathsf{T}_2 = \langle T_2, \eta_2, \mu_2 \rangle$, we want to show that the coproduct monad $\mathsf{T}_{1+2} = \langle T, \eta, \mu \rangle$ is confluent iff the two monads T_1 and T_2 are. One direction of the implication is easy: if T_{1+2} is confluent, T_1 and T_2 must be confluent as well, since T_1 and T_2 are faithfully embedded into T by the two injections. To show the other direction of the implication, we have to show that under the assumption that T_1 and T_2 are confluent, for all categories $\mathcal{X}$ which are confluent, $T\mathcal{X}$ is confluent.

By definition 3.1.1, the coproduct $T\mathcal{X}$ is the colimit of the diagram $D_{\mathcal{X}}$ in **Cat**. By proposition 1.3.1, this colimit is given by the coequalizer of diagram 3.20 reproduced here:

$$\coprod_{d:u\to v \text{ in } \mathcal{F}(\mathcal{G})} T^u\mathcal{X} \underset{G}{\overset{F}{\rightrightarrows}} \coprod_{w\in W} T^w\mathcal{X}$$

In order to be able to use lemma 4.2.4, we first need to show that the target of the two arrows to be coequalized is (commuting) confluent:

Lemma 4.3.1 If T_1 and T_2 are (commuting) confluent, then the coproduct $\coprod_{w\in W} T^w\mathcal{X}$ is (commuting) confluent whenever $\mathcal{X}$ is.

Proof. It is sufficient to show that for all $w \in W$, $T^w\mathcal{X}$ is (commuting) confluent. This is shown by a simple induction on w: for $w = \varepsilon$, $T^w\mathcal{X} = \mathcal{X}$ which is (commuting) confluent by assumption; and if $T^w\mathcal{X}$ is (commuting) confluent, so is $T^{jw}\mathcal{X} = T_j T^w \mathcal{X}$ (for all $j \in \mathcal{L}$), since both T_1 and T_2 preserve (commuting) confluence. $\square$

We now need to find a common witness for any two equivalent objects $s, t \in \coprod_{w \in W} T^w \mathcal{X}$. Recall from lemma 3.2.6 that for every object t there is a normal form $\mathrm{NF}(t)$ (with respect to the reduction relation $\twoheadrightarrow_{Ob}$ from definition 3.2.1) which decides the equivalence on the objects induced by F and G, hence $Qt = Qs$ iff $\mathrm{NF}(s) = \mathrm{NF}(t)$. This normal form will also be the common witness of s and t, thus ensuring in one fell swoop that any two equivalent objects have the same witness and that every object has one. All that remains to be shown is that the normal form $\mathrm{NF}(x)$ of an object x witnesses x. This is shown by a rather simple induction on the derivation, but we have to assume that the monads T_1, T_2 are non-expanding in the sense of definition 2.4.5. This is essential, since in the presence of expanding rewrite rules Toyama's theorem doesn't hold, as the following counterexample shows. Consider the two confluent term rewriting systems (where R_1 is expanding)

$$\begin{aligned} R_1 &\stackrel{def}{=} \{\texttt{'}x \to \mathtt{A}(\texttt{'}x)\} \\ R_2 &\stackrel{def}{=} \{\mathtt{B}(\mathtt{C}(\texttt{'}x)) \to \mathtt{C}(\mathtt{B}(\texttt{'}x))\} \end{aligned}$$

then in the disjoint union $R_1 + R_2$ there is the uncompletable span

$$\begin{tikzcd} & \mathtt{B}(\mathtt{C}(\texttt{'}x)) \arrow[dl, "R_2"'] \arrow[dr, "R_1"] & \\ \mathtt{C}(\mathtt{B}(\texttt{'}x)) & & \mathtt{B}(\mathtt{A}(\mathtt{C}(\texttt{'}x))) \end{tikzcd}$$

However, instead of requiring every rewrite to be non-expanding, we can slightly relax the requirement to every rewrite being able to be made into one which is non-expanding; this is the quasi-non-expanding property from definition 2.4.11 on page 85, and this generalisation will be the subject of §4.4 below.

Lemma 4.3.2 If T_1 and T_2 are non-expanding, then the normal form of an object x witnesses x:

$$\mathrm{NF}(x) \text{ wit } x$$

Proof. Given any $\alpha : x \to x'$, then by lemma 3.2.10, if $\mathsf{T}_1, \mathsf{T}_2$ are non-expanding, there is $\mathrm{NF}(\alpha) : \mathrm{NF}(x) \to x'$. Since $Q(\mathrm{NF}(\alpha)) = Q(\alpha)$, $\mathrm{NF}(x)$ witnesses x. $\square$

Theorem 4.3.3 (Modularity of Confluence) *The coproduct of two regular, (commuting) confluent, non-expanding monads on* **Cat** *is (commuting) confluent.*

Proof. If $\mathcal{X}$ is a (commuting) confluent category, then by lemma 4.3.1, the coproduct $\coprod_{w \in W} T^w \mathcal{X}$ is (commuting) confluent. Given $x, y \in \coprod_{w \in W} T^w \mathcal{X}$ s.t. $Qx = Qy$, then by lemma 3.2.6 $\mathrm{NF}(x) = \mathrm{NF}(y)$, and by lemma 4.3.2, $\mathrm{NF}(x)$ wit x and $\mathrm{NF}(y)$ wit y, so x and y have a common witness $\mathrm{NF}(x) = \mathrm{NF}(y)$. Hence by lemma 4.2.4, the colimit *colim* $D_{\mathcal{X}} = T\mathcal{X}$ is (commuting) confluent, making the monad $\mathsf{T}_1 + \mathsf{T}_2$ (commuting) confluent. □

The original theorem is a corollary of theorem 4.3.3:

Corollary 4.3.4 (Toyama) *The disjoint union of two confluent, non-expanding term rewriting systems which do not introduce unbounded variables is confluent.*

Proof. Given two confluent, non-expanding term rewriting systems Θ_1, Θ_2 which do not introduce unbounded variables, the monads T_{Θ_1} and T_{Θ_2} are non-expanding by lemma 2.4.8, regular by proposition 2.4.4 and confluent by proposition 4.1.7. Then by theorem 4.3.3, $\mathsf{T}_{\Theta_1} + \mathsf{T}_{\Theta_2}$ is confluent, and by proposition 2.5.3, $\mathsf{T}_{\Theta_1} + \mathsf{T}_{\Theta_2} \cong \mathsf{T}_{\Theta_1 + \Theta_2}$. Hence (again by proposition 4.1.7), $\Theta_1 + \Theta_2$ is confluent. □

Note that the proof nowhere used the fact that the monads $\mathsf{T}_1, \mathsf{T}_2$ arise from term rewriting systems (apart from them being regular). In particular, the proof of theorem 4.3.3 does not depend on the fact the rules of the term rewriting systems do not introduce variables, as long as the resulting monad is non-expanding. Hence by lemma 2.4.8, corollary 4.3.4 holds for term rewriting systems which do introduce bounded variables on the right as well, slightly generalising the original theorem [88, 42].

4.4 Modularity of Confluence for Quasi-Non-Expanding Monads

In this section we will extend Toyama's theorem (theorem 4.3.3) to systems which are quasi-non-expanding (qne, see definition 2.4.11). For the proof of theorem 4.3.3, lemma 3.2.10 is essential, since it ascertains that for any rewrite $\alpha : s \to t$ which is not layer-expanding, there is a rewrite $\mathrm{NF}(\alpha) : \mathrm{NF}(s) \to t$, making the normal form a witness and thus allowing to complete every span under Q. If the monads $\mathsf{T}_1, \mathsf{T}_2$ are expanding, this will not hold any more, but as it turns out it is sufficient that we are able to compose any rewrite with one which is not layer-expanding such that the composition is not layer-expanding either.

We will give a more abstract characterisation of this situation: if we have a confluent subcategory of $\coprod_{w \in W} T^w \mathcal{X}$ (like the non-layer-expanding morphisms) for which the diamond property wrt Q holds, we call this a *completable* subcategory; and if we can compose any morphism with one from the subcategory to obtain a morphism in the subcategory we call this subcategory *co-initial.* Then the existence of a completable co-initial subcategory is sufficient to show confluence. Strengthening our assumption to commuting confluence and the strong diamond property, we obtain commuting confluence.

4.4.1 Completable and Co-Initial Subcategories

Definition 4.4.1 (Completable Subcategory) Given a functor $F : \mathcal{X} \to \mathcal{Y}$, a *(commuting) completable subcategory* w.r.t. F is a subcategory $J : \mathcal{C} \hookrightarrow X$ such that $\mathcal{C}$ is (commuting) confluent and satisfies the (strong) diamond property w.r.t. FJ: $\mathcal{C} \models_{FJ} \lozenge$ ($\mathcal{C} \models_{FJ} \blacklozenge$)

We will in the following omit the embedding functor J, and say that an object $x \in \mathcal{X}$, or a morphism $\alpha : x \to x'$ in $\mathcal{X}$ is in $\mathcal{C}$ if it is the image of J. In this notation, the definition of a completable subcategory spells out as follows: for all morphisms $\alpha : x \to x'$, $\beta : y \to y'$ in $\mathcal{C}$ such that $F(x) = F(y)$ there are morphisms $\gamma : x'' \to u$, $\delta : y'' \to u'$ in $\mathcal{C}$ (and if $\mathcal{C}$ is commuting completable, $F\gamma \cdot F\alpha = F\delta \cdot F\beta$). We can now complete any paths which consist only of morphisms in $\mathcal{C}$, obtaining a relaxed version of the tiling lemma 4.2.2.

We also need some more notation for paths: for two paths A, B we say $A \sim B$ iff $\mathtt{s}(A) = \mathtt{s}(B)$ and $\mathtt{t}(A) = \mathtt{t}(B)$ — i.e. A and B are paths with the same source and target.

Lemma 4.4.2 Let $Q : \mathcal{Y} \to \mathcal{Z}$ be the coequalizer of two functors $F, G : \mathcal{X} \to \mathcal{Y}$, and let $J : \mathcal{C} \hookrightarrow \mathcal{Y}$ be a completable subcategory of $\mathcal{Y}$ w.r.t. Q. We say that a path $A = \texttt{<}\alpha_1, \ldots, \alpha_n\texttt{>}$ is in $\mathcal{C}$ if all α_i are in $\mathcal{C}$.

Then for all paths A, B which are in $\mathcal{C}$, and for which $\mathtt{s}(A) = \mathtt{s}(B)$, there are paths C, D in $\mathcal{C}$ such that $\mathtt{t}(A) = \mathtt{s}(C)$, $\mathtt{t}(B) = \mathtt{s}(D)$ and $A :: C \sim B :: D$.

If $\mathcal{C}$ is a commuting confluent subcategory, we furthermore have $Q(A :: C) = Q(B :: D)$.

Proof. By "tiling", i.e. induction on the length of A and B (like lemma 4.2.2). The crucial point here is that the completion of two morphisms $\alpha : x \to x'$, $\beta : y \to y'$ in $\mathcal{C}$ is in $\mathcal{C}$ again, allowing the induction to go through. □

We say the subcategory $\mathcal{C}$ is co-initial in $\mathcal{X}$ if for every morphism α in $\mathcal{X}$, there is a morphism β in $\mathcal{C}$ such that their composition $\beta \cdot \alpha$ is in $\mathcal{C}$ as well.

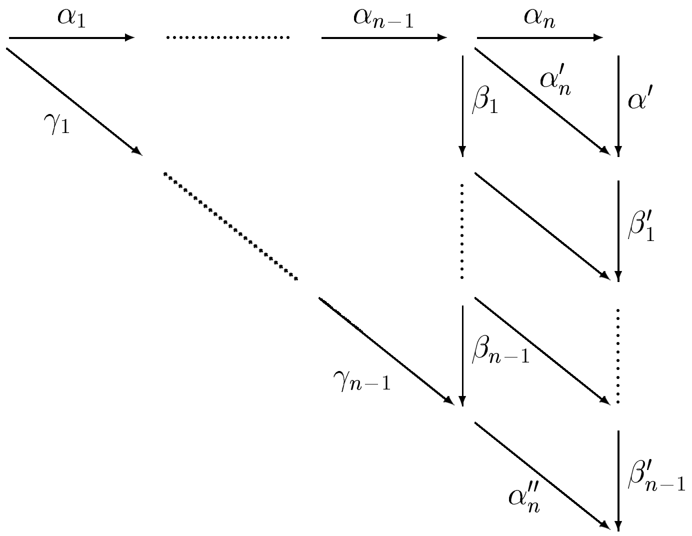
Definition 4.4.3 (Co-initial Subcategory) Given a functor $F : \mathcal{X} \to \mathcal{Y}$, a subcategory $J : \mathcal{C} \hookrightarrow \mathcal{X}$ is called *co-initial*, if for all morphisms $f : x \to y$ in $\mathcal{X}$ there is a morphism $g : y \to z$ in $\mathcal{C}$ such that $g \cdot f$ is in $\mathcal{C}$.

Note that a necessary condition for $\mathcal{C}$ to be a co-initial subcategory is that $\mathcal{C}$ contains all the objects of $\mathcal{X}$. Now, if there is a (commuting) completable subcategory with respect to the coequalizer of two functors which is co-initial as well, the coequalizing category will be (commuting) confluent:

Lemma 4.4.4 Let $Q : \mathcal{Y} \to \mathcal{Z}$ be the coequalizer of two functors $F, G : \mathcal{X} \to \mathcal{Y}$. If there is a co-initial (commuting) completable subcategory w.r.t. Q, and $\mathcal{Y}$ is (commuting) confluent, then $\mathcal{Z}$ is (commuting) confluent.

Proof. We will first show that for all paths $A = \mathtt{<}\alpha_1, \dots, \alpha_n\mathtt{>}$ there are paths $B = \mathtt{<}\beta_1, \dots, \beta_n\mathtt{>}$ and $C = \mathtt{<}\gamma_1, \dots, \gamma_n\mathtt{>}$ in $\mathcal{C}$ such that $\mathtt{t}(A) = \mathtt{s}(B)$, and $A :: B \sim C$; and under the stronger assumption that $\mathcal{C}$ is a commuting completable subcategory, we have $A :: B \equiv_M C$. We can then use lemma 4.4.2 to conclude confluence of $\mathcal{Z}$.

The proof proceeds by induction on the length of A. The induction assumption is $n = 0$, which is trivial. For the induction step, given $A = \mathtt{<}\alpha_1, \dots, \alpha_n\mathtt{>}$, we can apply the induction assumption to $A' = \mathtt{<}\alpha_1, \dots, \alpha_{n-1}\mathtt{>}$ and obtain $B' = \mathtt{<}\beta_1, \dots, \beta_{n-1}\mathtt{>}, C' = \mathtt{<}\gamma_1, \dots, \gamma_{n-1}\mathtt{>}$ in $\mathcal{C}$ such that $A' :: B' \sim C'$. Now since $\mathcal{C}$ is a completable subcategory, for α_n there is α' in $\mathcal{C}$ s.t. $\alpha'_n \stackrel{def}{=} \alpha' \cdot \alpha_n$ is in $\mathcal{C}$. Since α'_n and $\beta_1, \dots, \beta_{n-1}$ are in $\mathcal{C}$, there are α''_n and $\mathtt{<}\beta'_1, \dots, \beta'_{n-1}\mathtt{>}$ (by tiling as in the proof of lemma 4.2.2) in $\mathcal{C}$ with $\mathtt{<}\beta_1, \dots, \beta_{n-1}, \alpha''_n\mathtt{>} \sim \mathtt{<}\alpha'_n, \beta'_1, \dots, \beta'_{n-1}\mathtt{>}$. Then let $B \stackrel{def}{=} \mathtt{<}\alpha'_n, \beta'_1, \dots, \beta'_{n-1}\mathtt{>}$ and $C \stackrel{def}{=} \mathtt{<}\gamma_1, \dots, \gamma_{n-1}, \alpha''_n\mathtt{>}$, both of which are in $\mathcal{C}$, and $\mathtt{t}(A) = \mathtt{s}(B)$ and further $A :: B \sim C$ as required. See also diagram 4.7, where



(4.7)

the large triangle on the left is the induction assumption, the small triangle on the top right corner is given by the fact that $\mathcal{C}$ is a co-initial subcategory, and the parallelograms down the right side are constructed from the top (by tiling) since $\beta_1, \ldots, \beta_{n-1}$ and α'_n are in $\mathcal{C}$. Note the large triangle, and the parallelograms do not commute, unless we can assume commuting completability as follows.

If there is a co-initial commuting completable subcategory, we can show that $A::B \equiv_M C$. The induction assumption strengthens to $A' \cdot B' \equiv_M C'$, and for $\beta'_1, \ldots, \beta'_n$ and α'' we have that $<\beta_1, \ldots, \beta_{n-1}, \alpha''_n> \equiv_M <\alpha'_n, \beta'_1, \ldots, \beta'_{n-1}>$. Then we can show that

$$\begin{aligned} A::B &= & &<\alpha_1, \ldots, \alpha_n, \alpha', \beta'_1, \ldots, \beta'_{n-1}> \\ &\equiv_M & &<\alpha_1, \ldots, \alpha_{n-1}, \alpha'_n \cdot \alpha_n, \beta'_1, \ldots, \beta'_{n-1}> \\ &\equiv_M & &<\alpha_1, \ldots, \alpha_{n-1}, \beta_1, \ldots, \beta_{n-1}, \alpha''_n> \\ &\equiv_M & &A'::B'::<\alpha''_n> \equiv_M C'::<\alpha''_n> = C \end{aligned}$$

This concludes the induction.

We can now show confluence. Given two paths A, B such that $\mathtt{s}(A) = \mathtt{s}(B)$, we have just shown there are paths B', A', C_1, C_2 in $\mathcal{C}$ such that $A::A' \sim C_1$ and $B::B' \sim C_2$. By lemma 4.4.2, there are D_1, D_2 such that $C_1::D_1 \sim C_2::D_2$. Hence, for A and B we have $A'::D_1$ and $B'::D_2$ such that $A::(A'::D_1) \sim C_1::D_1 \sim C_2::D_2 \sim B::(B'::D_2)$, so we have a completion for A and $\mathcal{Z}$ is confluent.

Under the stronger assumption of $\mathcal{C}$ being a commuting completable subcategory, we have shown that $A::A' \equiv_M C_1$ and $B::B' \equiv_M C_2$; and by lemma 4.4.2, there are D_1, D_2 such that $C_1::D_1 \equiv_M C_2::D_2$. By the same reasoning as above, it follows that $A::(A'::D_1) \equiv_M C_1::D_1 \equiv_M C_2::D_2 \equiv_M B::(B'::D_2)$, hence we have a commuting completion for A and B and $\mathcal{Z}$ is commuting confluent. $\square$

4.4.2 Finding a Completable Co-initial Subcategory

In this case, the completable co-initial subcategory will be given by the non-layer-expanding rewrites. That a span of two non-layer-expanding rewrites can be completed has been shown above, so the main work will be to show that for every rewrite α, we can find a rewrite β such that $\beta \cdot \alpha$ is not layer-expanding, and that this β is not layer-expanding itself. β will exists if both monads $\mathsf{T}_1, \mathsf{T}_2$ are qne, so any expanding rewrite can be contracted again. However, α can be expanding in several layers at the same time, so we need to contract all these layers. For example, consider the term rewriting system $\mathcal{E} = (\mathtt{E}_1, \{\,'x \to \mathtt{E}('x), \mathtt{E}('x) \to 'x\})$

as the first system, and just the signature with one unary operation `F` and no rewrites as the second system; then we have the rewrite $\mathtt{F('' F('' }x\mathtt{))} \twoheadrightarrow \mathtt{F('E('F('E('}x\mathtt{))))}$ in $T_2T_1T_2T_1\mathcal{X}$ which is expanding in two layers at the same time. However, both can be contracted again, and moreover, the rewrite contracting a rewrite in one layer is not expanding another layer.

Lemma 4.4.5 If $\mathsf{T}_1, \mathsf{T}_2$ are quasi-non-expanding, the subcategory of $\coprod_{w\in W} T^w\mathcal{X}$ given by the non-layer-expanding morphisms is co-initial.

Proof. We have to show that for all α in $T^w\mathcal{X}$, there is α' in $T^w\mathcal{X}$ which is not layer-expanding such that $\alpha'\cdot\alpha$ is not layer-expanding. Let $\alpha : s \to t$ be layer-expanding, with $\mathrm{NF}(\alpha) : s_0 \to t_0$. Then $s_0 \neq \mathrm{NF}(s)$, with $s_0 \to_\eta s_1 \to_\eta \ldots \mathrm{NF}(s_0)$ (there can be no reduction steps from $\to_\mu$ here, since $\alpha \to_\mu \alpha'$ iff $\delta_s(\alpha) \to_\mu \delta_s(\alpha)$). To be more precise, we have $s_0, \ldots, s_n$ with $s_i = \eta^{u_i}_{j_i,v_i}(s_{i+1})$ for $i = 0, \ldots, n-1$ ($u_i, v_i \in W, j_i \in \mathcal{L}$) and $s_n = \mathrm{NF}(s_0)$.

The proof proceeds by induction on n. The induction base is $n = 0$, which is trivial. For the induction step, we assume the lemma for $s_1 \twoheadrightarrow_\eta \mathrm{NF}(s_0)$, and let $s_0 = \eta^u_{j,v}(s_1)$. Now for $\mathrm{NF}(\alpha) : s_0 \to t_0$, since $\eta^u_{j,v}$ is qne there are $\beta : t_0 \to t'_0, \gamma : s_1 \to t_1$ such that $\eta^u_{j,v}(\gamma) = \beta\cdot\mathrm{NF}(\alpha)$. Since t_0 is not in the image of $\eta^u_{j,v}$ (otherwise $\eta^u_{j,v}$ would not be full, and T_j not qne), β is not expanding w.r.t. $\eta^u_{j,v}$ and hence not layer-expanding.

However, γ will be expanding w.r.t. $\eta^{u_1}_{j_1,v_1}$, i.e. for all $\delta : s_2 \to t_2$ we will have $\eta^{u_1}_{j_1,v_1}(\delta) \neq \gamma$ (but $\eta^{u_1}_{j_1,v_1}(s_2) = s_1$). We can apply the induction assumption to s_1 and γ: there is $\gamma' : t_1 \to t'_1$ which is not layer-expanding, and further $\gamma'\cdot\gamma$ is not layer-expanding either. With $t'_0 = \eta^u_{j,v}(t_1)$, we can compose β above and $\eta^u_{j,v}(\gamma')$, obtaining $\alpha' \stackrel{def}{=} \eta^u_{j,v}(\gamma')\cdot\beta$. Since both β and $\eta^u_{j,v}(\gamma')$ are not layer-expanding, α' is neither, and since $\beta\cdot\mathrm{NF}(\alpha)$ is not layer-expanding, $\alpha'\cdot\mathrm{NF}(\alpha) = \eta^u_{j,v}(\gamma')\cdot\beta\cdot\mathrm{NF}(\alpha)$ will not be. $\square$

Depending on whether the monads $\mathsf{T}_1, \mathsf{T}_2$ are confluent or commuting confluent, the non-layer-expanding morphisms will form a completable or commuting completable subcategory of $\coprod_{w\in W} T^w\mathcal{X}$.

Lemma 4.4.6 If $\mathsf{T}_1, \mathsf{T}_2$ are (commuting) confluent and quasi-non-expanding, the non-layer-expanding morphisms form a (commuting) completable subcategory of $\coprod_{w\in W} T^w\mathcal{X}$ w.r.t. Q.

Proof. Given two non-layer-expanding morphisms $\alpha : s \to s', \beta : t \to t'$ in $\coprod_{w\in W} T^w\mathcal{X}$ such that $Q(s) = Q(t)$. By lemma 3.2.10, we have $\mathrm{NF}(\alpha) : \mathrm{NF}(s) \to$

s'' and $\mathrm{NF}(\beta) : \mathrm{NF}(t) \to t''$, and by lemma 3.2.6 $\mathrm{NF}(s) = \mathrm{NF}(t)$ iff $Q(s) = Q(t)$. By confluence of $T^w\mathcal{X}$, the span formed by $\mathrm{NF}(\alpha)$ and $\mathrm{NF}(\beta)$ can then be completed, i.e. there are $\gamma : s'' \to u$ and $\delta : t'' \to u$ in $T^w\mathcal{X}$.

If further $T^w\mathcal{X}$ is commuting confluent, we have $\gamma{\cdot}\mathrm{NF}(\alpha) = \delta{\cdot}\mathrm{NF}(\beta)$, hence $Q(\gamma){\cdot}Q(\alpha) = Q(\gamma){\cdot}Q(\mathrm{NF}(\alpha)) = Q(\delta){\cdot}Q(\mathrm{NF}(\beta)) = Q(\delta){\cdot}Q(\beta)$. □

We can now combine the previous lemmas into the main theorem:

Theorem 4.4.7 (Modularity of Confluence for Quasi-Non-Expanding Monads) *The coproduct of two (commuting) confluent, regular, quasi-non-expanding monads is (commuting) confluent.*

Proof. If T_1 and T_2 are two confluent, quasi-non-expanding monads, and if $\mathcal{X}$ is a (commuting) confluent category, then by lemma 4.3.1, the coproduct $\coprod_{w\in W} T^w\mathcal{X}$ is (commuting) confluent. By lemmas 4.4.6 and 4.4.5, the non-layer-expanding rewrites form a co-initial (commuting) completable subcategory in $\coprod_{w\in W} T^w\mathcal{X}$, so by lemma 4.4.4 the colimit $\mathit{colim}\, D_{\mathcal{X}} = T\mathcal{X}$ is (commuting) confluent, making the monad $\mathsf{T}_1 + \mathsf{T}_2$ (commuting) confluent. □

Again, the corollary of this is that the disjoint union of two quasi-non-expanding confluent rewrite systems is confluent:

Corollary 4.4.8 *The disjoint union of two confluent quasi-non-expanding term rewriting systems which do not introduce unbounded variables is confluent.*

Proof. Substitute theorem 4.3.3 with theorem 4.4.7, and lemma 2.4.8 with lemma 2.4.12 in the proof of corollary 4.3.4. □

Of course, §4.3 is a special case of this section, but we felt developing it separately would lead to a clearer exposition. The aim of this section is not so much to prove a more general result, but rather to demonstrate the flexibility and adaptability of our modularity proof.

4.5 Summary and Conclusion

In this chapter, we have proven the modularity of confluence for the disjoint union. We first defined a *semantic* notion of confluence for categories and monads, and showed that it is equivalent to the usual, syntactic definition in the sense that the monad T_Θ is confluent according to the new, semantic definition if and only if the term rewriting system Θ is confluent according the old, syntactic definition.

Building on the observation that the colimit of any diagram can be constructed as a coequalizer of coproducts, we then developed techniques to reason about preservation of confluence for coequalizers in **Cat**. We gave a property under which confluence is preserved by a coequalizer, the so-called *diamond property*, and we further introduced the notion of a *witness*.

The main result was to show that the coproduct of two confluent, non-expanding regular monads is confluent (Toyama's theorem), using the machinery set up before. Since the proof solely depends on the properties of the two monads, our proof slightly generalizes the original theorem to systems which introduce bounded variables. We then extended this proof to quasi-non-expanding monads, showing that the coproduct of two confluent quasi-non-expanding regular monads is confluent.

To close this section, we will compare our proof with [42], a simplified and clearer version of the original proof [88]. For the notation and notions used in the following, the reader should recall the quote on page 96. The authors start their proof by observing that

> the main obstacle for giving a 'straightforward' proof of the modularity of confluence is the fact that the black and white layer structure of a term need not be preserved under reduction. That is, by a destructive rewrite step a e.g. black layer may disappear, thus allowing two originally distinct white layers to coalesce.

They proceed to define the notion of a *preserved* term — a term from which no reduction sequence with a destructive rewrite starts, and of *inner preserved term*, a term all the principal subterms of which are preserved. It is shown that

- inner preserved terms are confluent (meaning any span starting from the terms can be completed), and that
- every term reduces to a *witness*. Given a term s, the witness $\dot{s}$ is an inner preserved term such that if all principal subterms of s are confluent, then for all other terms t, if $s \twoheadrightarrow t$, then $\dot{s} \downarrow \dot{t}$.

The final step is to show that every term t is confluent by induction on the rank of t. From the induction assumption, it follows that all principal subterms of t are confluent; so given any reduction $t \twoheadrightarrow s$, the previous lemma about witnesses can be applied to $\dot{t}$ and $\dot{s}$, hence t and s are joinable, making t confluent.

Comparing the two proofs, our notion of a witness (definition 4.2.3) is similar only insofar as it represents reductions as well. Whereas witnesses in [42] are merely joinable, our witnesses represent all reductions in an "algebraic" sense (i.e. if there is a reduction starting from t, then there is an equivalent reduction starting from the witness of t as well).

In [42], the main obstacle of the proof — the destructive reductions — is handled by reducing all terms until no further destructive reductions can occur. Hence, the completion of a span is not very constructive. In contrast, in our proof we take the reductions in the theory of the two systems (the many-step reductions, as given by the monads) as primitive, and show that for any span of these, we can find ones which are *equivalent* under the equivalence generated by combining the two systems, described by the diagram $D_{\mathcal{X}}$ and given by its colimit, and can be joined. This gives our proof an algebraic, and rather constructive, flavour, since it involves neither preservation nor induction on the ranks of the terms involved.

Further, the presence of the variable rewrites allows us to abolish the distinction between inner and outer reductions. Since the variable rewrites specify the rewrites between the terms which are instantiated for the variables, instantiation cannot destroy inner redexes. Hence, in a confluent monad every span of rewrites can be completed in one step (taking many-step reductions in one system as basic steps). For example, consider the two systems

$$\begin{aligned} R_1 &\stackrel{def}{=} \{\mathtt{F}(\text{'}x, \text{'}x) \to \mathtt{G}(\text{'}x)\} \\ R_2 &\stackrel{def}{=} \{\mathtt{K}(\text{'}x) \to \mathtt{L}(\text{'}x), \mathtt{K}(\text{'}x) \to \mathtt{H}(\text{'}x), \mathtt{L}(\text{'}x) \to \mathtt{H}(\text{'}x)\} \end{aligned}$$

Then there is the following span

$$\begin{tikzcd} & \mathtt{F}(\mathtt{K}(\text{'}x),\mathtt{K}(\text{'}x)) \arrow[dl, "R_2"'] \arrow[dr, "R_1"] & \\ \mathtt{F}(\mathtt{L}(\text{'}x),\mathtt{H}(\text{'}x)) & & \mathtt{G}(\mathtt{K}(\text{'}x)) \end{tikzcd}$$

In order to complete this span, we first need some more inner R_2-steps on the left. In the monad semantics, the existence of these R_2-steps precisely corresponds to variable rewrites, which by definition of confluence we can assume to be confluent; and by rule [VAR], these R_2-steps are R_1-steps as well, so the outer system (R_1) is completable in one step as well. Nevertheless, confluence for monads is not a strictly stronger concept than confluence for term rewriting systems, as was shown in proposition 4.1.7. In the proof of that proposition, exactly the above situation showed up (diagram 4.4), which shows that our definition handles this situation adequately. The handling of substitution as a natural transformation, which has to preserve the reduction structure, together with the variable rewrites, is the key element here.

The algebraic nature of the proof extends it to systems with bounded variables on the right, and further allows the generalisation of §4.4 to quasi-non-expanding systems. (Both of these allow rules that can arbitrarily increase the rank of a reduction, and hence break the original proof.) In closing, I think one can say that the proof method introduced here is substantially new, not merely a rephrasing of the old proof in a categorical framework.

Chapter 5

Modularity of Strong Normalization

A term rewriting system is *strongly normalizing* if there are only finite reduction sequences: reduction can't go on for ever. Strongly normalizing systems are also called terminating, and for variety, we shall use the two words synonymously.

Termination is important, because it means that in an implementation of the term rewriting system, every reduction eventually ends. A system which is confluent as well as terminating is called *complete*. A complete term rewriting system gives a decision procedure for the equational theory the term rewriting system describes[1] (if the term rewriting system has finitely many rules): every term reduces to exactly one *normal form*, and two terms are equal in the equational theory iff their normal forms are the same.

Known Results about Modularity of Termination

Unlike confluence, strong normalization is in general not modular. The simplest, earliest and best-known counterexample [87] is given by the following two terminating systems:

$$\begin{aligned}\mathcal{R} &= (\{\mathtt{A}_0, \mathtt{B}_0, \mathtt{F}_3\}, \{\mathtt{F(A,B,}\,\text{'}x\mathtt{)} \to \mathtt{F(}\text{'}x, \text{'}x, \text{'}x\mathtt{)}\}) \\ \mathcal{S} &= (\{\mathtt{G}_2\}, \{\mathtt{G(}\text{'}x, \text{'}y\mathtt{)} \to \text{'}x, \mathtt{G(}\text{'}x, \text{'}y\mathtt{)} \to \text{'}y\})\end{aligned}$$

Then in the coproduct, there is an infinite reduction sequence starting

$$\mathtt{F(G(A,B),G(A,B),G(A,B))} \twoheadrightarrow_{\mathcal{S}} \mathtt{F(A,B,G(A,B))} \to_{\mathcal{R}} \mathtt{F(G(A,B),G(A,B),G(A,B))} \ldots$$

One can also show [87] that completeness is not modular, i.e. even if both systems are confluent as well as terminating, their disjoint union will in general not be

[1] Actually, to decide the equality weak normalization [41] together with confluence suffices.

terminating (although of course it will be confluent). There is a wide variety of conditions, however, under which strong normalization is modular:

- Both systems are not collapsing [72];
- Both systems are not duplicating [72];
- One of the systems is neither duplicating nor collapsing [58];
- Both systems are simply terminating, and finite [44].

  This is defined as follows: a *simplification ordering* for a term rewriting system $\Theta = (\Omega, R)$ is a partial order $(T_\Omega(V), >)$ on the terms[2] which

  – is monotonic: $s > t$ implies $\mathtt{F(}\ldots, s, \ldots\mathtt{)} > \mathtt{F(}\ldots, t, \ldots\mathtt{)}$;

  – possesses the subterm property: $\mathtt{F(}\ldots, t, \ldots\mathtt{)} > t$

  Then Θ is *simply terminating* if the reduction order can be embedded into a simplification ordering $>$, i.e. $\twoheadrightarrow_R \subseteq >$.

- One system is left-linear, the other is right-linear and there is no overlap between the left sides of the first, and the right sides of the second system [16];
- Both systems are confluent, and left-linear [89].
- One system is not collapsing, and the other one preserves strong normalization under non-deterministic collapses [23, 62], which means the disjoint of that system with the system $\mathcal{C}_\mathcal{E}$ is strongly normalizing, where $\mathcal{C}_\mathcal{E}$ is the term rewriting system

$$\mathcal{C}_\mathcal{E} \stackrel{def}{=} \{\mathtt{G(}{'x}, {'y}\mathtt{)} \to {'x}, \mathtt{G(}{'x}, {'y}\mathtt{)} \to {'y}\}$$

Some of these conditions are strictly more general than others, for example the last subsumes all but the second last. We will in this chapter give a technical framework to prove modularity results for strong normalization, and as an application show modularity of strong normalization for non-collapsing systems.

[2] Recall from definition 1.6.1 that the set V of variables is supposed to be part of the signature.

Structure of this Chapter

- We fill first (in §5.1) define the notion of strong normalization for categories and monads on **Cat**. Just like we have done for confluence, we will show the equivalence of this *semantic* definition of strong normalization and the syntactic definition 1.6.4 from page 47.

- In §5.2, we will give a simple necessary and sufficient condition for the coproduct of two strongly normalizing monads to be strongly normalizing: that we cannot form paths of arbitrary length (infinite paths, in a manner of speaking).

- In §5.3, we will put the technical framework developed in the previous section to use and prove modularity of strong normalization for non-collapsing monads.

5.1 A Semantic Definition of Strong Normalization

5.1.1 Termination for Categories and Monads

For termination, we really need to consider categories rather than just preorders. The reason is that we have to be able to tell apart the identity on an object x from some other endomorphism on x, such as the cyclic reduction on `F(G(A,B),G(A,B),G(A,B))` at the beginning of this chapter which makes a term rewriting system non-terminating. The other possibility would be to use transitive relations (i.e. modelling a term rewriting system by a monad over the category **TRel** of transitive relations). This requires a different semantics than the one presented in chapter 2, whereas the named reduction semantics from appendix A is a proper generalization of it.

Roughly speaking, a category is strongly normalizing if the underlying relation minus the reflexive relation given by the identities is strongly normalizing. We call this the underlying non-identity relation:

Definition 5.1.1 (Strong Normalization for Categories and Monads) The *underlying non-identity relation* of a category $\mathcal{C}$, $R^{-}(\mathcal{C})$ is defined as

$$R^{-}(\mathcal{C}) \stackrel{def}{=} (|\mathcal{C}|, \{x > y \mid \exists \alpha : x \to y \wedge \alpha \neq \mathbf{1}_x\})$$

A category $\mathcal{C}$ is *strongly normalizing* (or *terminating*), written $\mathcal{C} \models \mathrm{SN}$, iff its underlying non-identity relation is strongly normalizing:

$$\mathcal{C} \models \mathrm{SN} \Leftrightarrow R^{-}(\mathcal{C}) \models \mathrm{SN}$$

A monad T on **Cat** is strongly normalizing if it preserves strong normalization:

$$\mathcal{X} \models \mathrm{SN} \Rightarrow T\mathcal{X} \models \mathrm{SN}$$

In a strongly normalizing category, the only endomorphisms are identities. Further, there can be no pair of morphisms $f : x \to y, g : y \to x$: even if $gf = \mathbf{1}_x$ and $fg = \mathbf{1}_y$, we would have an infinite sequence in the underlying irreflexive relation $R^{-}(\mathcal{C})$, given by $x > y > x > y \ldots$; hence, the only isomorphisms in a strongly normalizing category are identities. Such a category (with the latter property concerning isomorphisms) is called *skeletal*.

The definition above differs substantially from Hilken's [29], where strong normalization is defined in terms of separating cocones for filtered diagrams. The precise relation between the two definitions still needs to be clarified.

5.1.2 Equivalence of Notions

Just as for confluence, we have to make sure that the our notion of strong normalization coincides with the one found in the literature for term rewriting systems. Recall from definition 1.6.4 on page 47 that a term rewriting system Θ is terminating if there is no infinite sequence $x_1 \to_\Theta x_2 \to_\Theta x_3 \to_\Theta \ldots$. We now want to show that Θ is terminating by the above definition if and only if T_Θ is by definition 5.1.1. Unfortunately, we cannot directly use proposition 2.3.14, since as pointed out above in a preorder we cannot distinguish identities from cyclic rewrites, so we first prove a variation of that proposition, using the term named-reduction algebra from definition A.1.8.

Lemma 5.1.2 Given a category $\mathcal{X}$, the underlying non-identity relation of the term named-reduction algebra on $\mathcal{X}$ is equal to the transitive closure of the union of $\to_R$ and $\to_X$, where $X \stackrel{def}{=} R^{-}(\mathcal{X})$:

$$R^{-}(T_\Theta(\mathcal{X})) = (\to_R \cup \to_X)^+$$

Proof. The proof of this lemma is a variation of the proof of proposition 2.3.14. The equality is proven by proving inclusion in both directions. Right-to-left is rather easy, since any reduction in $\to_R$ or $\to_X$ corresponds to a non-identity morphism in $T_\Theta(\mathcal{X})$.

Left-to-right we again need to decompose the arguments for the [PRE] rule; and further, there is a non-identity rewrite $s \to t$ only if there is a non-identity variable rewrite (by rule [VAR]) or an instantiated rewrite rule ([INST]); hence we have the transitive closure (not the reflexive-transitive closure) on the right.

□

Now the main step in the proof will be to show that $\to_X \cup \to_R$ is terminating if $\to_R$ is. We can show that for any reduction in the union of $\to_X$ and $\to_R$, we get one in $\to_R$ which has the same number of $\to_R$ steps; hence if there were an infinite reduction in the union, there would have to be one in $\to_R$ as well, which leads to a contradiction.

The key idea, due to Neil Ghani, is to assume the relation X to be confluent as well as strongly normalizing. If it isn't, we can complete it to a relation $\hat{X}$ which is confluent, and we show strong normalization of one relation by embedding it into one which is strongly normalizing. Under this additional assumption, we can show that any reduction in $\twoheadrightarrow_S$ gives rise to one in $\twoheadrightarrow_R$ of at least the same length. This is done by reducing every variable x to its normal form. The normal form is uniquely determined, and hence this reduction does not destroy any redexes. This is the content of the following lemma:

Lemma 5.1.3 Given a complete binary relation $(X, <)$, and $t_1, t_2 \in T_\Omega(X)$ s.t. $t_1 \to_R t_2$, then

$$\begin{tikzcd} t_1 \arrow[r, "R"] \arrow[d, two heads, "X"'] & t_2 \arrow[d, two heads, "X"'] \\ \mathrm{NF}_X(t_1) \arrow[r, "R"] & \mathrm{NF}_X(t_2) \end{tikzcd}$$

where $\mathrm{NF}_X(t_1)$ is the normal form of t_1 in $\to_X$.

Proof. $t_1 \to_R t_2$ iff there is a context C and a substitution σ s.t. $t_1 = C[\sigma(l)]$ and $t_2 = C[\sigma(r)]$, and $l \to r \in R$. Define the substitution $\tau(x) \stackrel{def}{=} \mathrm{NF}_X(x)$ (which is well-defined since X is complete), then $\mathrm{NF}_X(t_1) = \tau(t_1) = C[\tau(\sigma(l))]$ (since substitutions commute with operations and contexts) and similarly $\mathrm{NF}_X(t_2) = C[\tau(\sigma(r))]$. Hence, with the substitution $\sigma' = \tau \cdot \sigma$, $\mathrm{NF}_X(t_1) \twoheadrightarrow_R \mathrm{NF}_X(t_2)$ as required. □

Now given a sequence $t_1 \to_S t_2 \to_S \ldots t_n$, we can now apply lemma 5.1.3, and get a normal form with respect to $\twoheadrightarrow_X$. We obtain a shorter sequence containing only reductions in R.

Corollary 5.1.4 *Let $S \stackrel{def}{=} \to_R \cup \to_X$ and $(X, <)$ be complete. For any sequence $P = t_1 \to_S t_2 \to_S \ldots t_n$ in $\to_S$ with m $(m \leq n)$ reductions in $\to_R$, there is a reduction $s_1 \to_R s_2 \to_R \ldots s_m$, where all s_i are normal forms with respect to $\to_X$, $t_1 \twoheadrightarrow_X s_1$ and $t_n \twoheadrightarrow_X s_m$.*

Proof. The corollary is shown by an induction over length n of the sequence P, and applying lemma 5.1.3. □

We are now in a position to show the main result of this section:

Proposition 5.1.5 *A term rewriting system Θ is strongly normalizing iff T_Θ is a strongly normalizing monad.*

Proof. Let $\mathcal{X}$ be a strongly normalizing category, and $X \stackrel{def}{=} R^-(\mathcal{X})$. Then by lemma 5.1.2, $R^-(T_\Theta(\mathcal{X})) = (\to_R \cup \to_X)^+$, hence $T_\Theta(\mathcal{X}) \models \mathrm{SN}$ iff $(\to_X \cup \to_R)^+ \models \mathrm{SN}$ iff $\to_X \cup \to_R \models \mathrm{SN}$. As an abbreviation, let $S \stackrel{def}{=} \to_R \cup \to_X$. Then to show the lemma, we have to show that $\to_R \models \mathrm{SN}$ iff $S \models \mathrm{SN}$.

One direction is trivial, since $\to_R \subseteq S$. For the other direction, assume that X is confluent as well as strongly normalizing, and proceed by contradiction as follows: assume there were an infinite sequence $P = t_1 \to_S t_2 \to_S \ldots t_n \to_S \ldots$ in S^+, then this sequence would have to contain infinitely many steps from both $\to_R$ and $\to_X$ (since both are SN). With the additional assumption of X being complete, we can apply corollary 5.1.4; so there would be an infinite sequence P' in $\to_R$. But $\to_R$ is terminating by assumption, hence there cannot be such a sequence in $\to_S$.

If X is not confluent, then it can be completed to a relation $\hat{X}$ which is. The previous paragraph applies to $\hat{X}$, hence $\to_R \cup \to_{\hat{X}}$ is strongly normalizing; but then $(\to_R \cup \to_X) \subseteq (\to_R \cup \to_{\hat{X}})$, and any subrelation of a strongly normalizing relation is strongly normalizing. □

5.2 Strong Normalization for the Coproduct Monad

Unfortunately, there is no equivalent of the tiling lemma 4.2.2 for strong normalization. There does not seem to be a useful characterization of preservation of strong normalization for the coequalizer of two functors.

Hence, we have to bite the bullet and directly reason about the coproduct of the colimit of $D_{\mathcal{X}}$. Fortunately, we can find a good characterization when this colimit is strongly normalizing. This characterization is based on the fact that for every path in the colimit of $D_\mathcal{X}$, there is an equivalent one of minimal length (§3.2.3 on page 114). These paths of minimal length give rise to sequences in the underlying non-identity relation of *colim* $D_\mathcal{X}$, and if roughly speaking there are no infinite paths of minimal length, then there will be no infinite sequences in $R^{-}(colim\, D_\mathcal{X})$, and hence $colim\, D_\mathcal{X} \models$ SN. In other words, if we cannot form arbitrarily long paths of morphisms from $\coprod_{w \in W} T^w \mathcal{X}$ in the coequalizer, then the coequalizer will be strongly normalizing.

Of course, every path on its own is finite (otherwise it would not have a target), so we have to be more precise here. We do not have one infinite path, but a sequence

$$\texttt{<}\alpha_1\texttt{>}, \texttt{<}\alpha_1, \alpha_2\texttt{>}, \texttt{<}\alpha_1, \alpha_2, \alpha_3\texttt{>}, \texttt{<}\alpha_1, \alpha_2, \alpha_3, \alpha_4\texttt{>}, \texttt{<}\alpha_1, \alpha_2, \alpha_3, \alpha_4, \alpha_5\texttt{>}, \ldots$$

of paths of minimal length which keeps growing indefinitely. We call this an infinite sequence in *colim* $D_\mathcal{X}$:

Definition 5.2.1 (Infinite Sequence) An *infinite sequence* $(\alpha_i)_{i \in \mathbb{N}^+}$ in *colim* $D_\mathcal{X}$ is given by morphisms $\alpha_i : x_i \to y_i$ in $\coprod_{w \in W} T^w \mathcal{X}$ such that for all $k \in \mathbb{N}^+$, $\texttt{<}\alpha_1, \ldots, \alpha_k\texttt{>}$ is a path of minimal length.

If we can find such an infinite sequence in $colimD_\mathcal{X}$ then $\mathsf{T}_1 + \mathsf{T}_2$ is not strongly normalizing.

Lemma 5.2.2 Let $\mathsf{T}_1, \mathsf{T}_2 \models$ SN. If there is an infinite sequence $(\alpha_i)_{i \in \mathbb{N}^+}$ in *colim* $D_\mathcal{X}$ for some strongly normalizing category $\mathcal{X}$, then $\mathsf{T}_1 + \mathsf{T}_2 \not\models$ SN.

Proof. If there is an infinite sequence $(\alpha_i)_{i \in \mathbb{N}^+}$ in $colimD_\mathcal{X}$, we construct an infinite sequence

$$z_1 < z_2 < z_3 < \ldots$$

in $R^{-}(colim\, D_\mathcal{X})$ by letting $z_i \stackrel{def}{=} [\mathtt{s}(\alpha_i)]$, so $colim\, D_\mathcal{X} \not\models$ SN and hence $\mathsf{T}_1 + \mathsf{T}_2 \not\models$ SN. Since for all $k \in \mathbb{N}^+$, $A = \texttt{<}\alpha_1, \ldots, \alpha_k\texttt{>}$ is of minimal length, $\alpha_i \neq \mathbf{1}$ for all $i \in \mathbb{N}^+$ (otherwise A would not be of minimal length), hence $[\mathtt{s}(\alpha_i)] < [\mathtt{t}(\alpha_i)]$ in $R^{-}(colim\, D_\mathcal{X})$, and with $[\mathtt{t}(\alpha_i)] = [\mathtt{s}(\alpha_{i+1})] = z_{i+1}$, $z_i < z_{i+1}$. □

The converse of lemma 5.2.2 does not hold in general, because non-termination of the coproduct can also be caused by an unbounded increase in rank; see the counterexample on page 155 below. However, if we restrict ourselves to non-expanding monads, the converse of lemma 5.2.2 holds.

Lemma 5.2.3 Let $\mathsf{T}_1, \mathsf{T}_2 \models$ SN. If $\mathsf{T}_1, \mathsf{T}_2$ are non-expanding and $\mathsf{T}_1 + \mathsf{T}_2 \not\models$ SN, then there is an infinite sequence $(\alpha_i)_{i \in \mathbb{N}^+}$ in *colim* $D_\mathcal{X}$ for some strongly normalizing category $\mathcal{X}$.

Proof. If $\mathsf{T}_1 + \mathsf{T}_2 \not\models \mathrm{SN}$, then $colim\, D_{\mathcal{X}} \not\models \mathrm{SN}$ for some strongly normalizing category $\mathcal{X}$, which is the case if there is an infinite sequence

$$S = z_1 < z_2 < z_3 < \dots$$

in $R^{-}(colimD_{\mathcal{X}})$. S is given by morphisms $f_i : z_i \to z_{i+1}$ in $colimD_{\mathcal{X}}$ (for $i \in \mathbb{N}^+$), with $f_i \neq \mathbf{1}$. Morphisms in $colimD_{\mathcal{X}}$ are equivalence classes of paths, represented by paths of minimal length. Without loss of generality, we can assume that these paths of minimal length are of length one, i.e. $f_i = \langle\beta_i\rangle$ with $\beta_i : x_i \to y_i$ in $\coprod_{w\in W} T^w\mathcal{X}$ and in normal form. (If they are not, we can decompose them into paths of length one, obtaining a longer, still infinite sequence S above.)

Hence, S is given by $\beta_i : x_i \to y_i$ in $\coprod_{w\in W} T^w\mathcal{X}$ with $[x_i] = z_i$ and $[y_i] = z_{i+1}$ for $i \in \mathbb{N}^+$. We now have to construct an infinite sequence in $colim\, D_{\mathcal{X}}$ from S. Intuitively, this is done by composing as many β_i, β_{i+1} as possible; then the remaining ones are incomposable and will hence form a path of minimal length. Since both T_1 and T_2 are strongly normalizing, $\coprod_{w\in W} T^w\mathcal{X}$ is strongly normalizing as well, and so we can only compose finitely many β_i and β_{i+1}, so the remaining sequence will be infinite.

To this end, we define two functions $\sigma, \upsilon : \mathbb{N}^+ \to \mathbb{N}^+$, and a sequence $(\gamma_{\beta,k})_{k\in\mathbb{N}^+}$ which for $k \in \mathbb{N}^+$ gives a pair $\sigma(k), \upsilon(\sigma(k))$ such that $\sigma(k) \leq \upsilon(\sigma(k))$ and $\langle\gamma_{\beta,\upsilon(\sigma(k))}\rangle \equiv_M \langle\beta_{\sigma(k)}, \dots, \beta_{\upsilon(\sigma(k))}\rangle$. $\sigma(k)$ can be thought of as the first morphism in the sequence S to make up of the k-th morphism in the infinite sequence, $\upsilon(k)$ as the last morphism in S still "composable" with β_k in the sense that there are β_k, β_{k+1} which are not an incomposable pair,[3] and $\gamma_{\beta,\upsilon(\sigma(k))}$ gives the k-th morphism in the infinite sequence. σ, υ and γ_β are defined inductively as follows:

$$\begin{aligned}
\sigma(1) &\stackrel{def}{=} 1 \\
\sigma(k+1) &\stackrel{def}{=} \upsilon(\sigma(k)) + 1 \\
\gamma_{\beta,1} &\stackrel{def}{=} \beta_1 \\
\gamma_{\beta,k+1} &\stackrel{def}{=} \begin{cases} \beta_{k+1} & \text{if } (\gamma_{\beta,k}, \beta_{k+1}) \text{ are an incomposable pair} \\ \alpha & \text{where } \langle\alpha\rangle \equiv_M \langle\gamma_{\beta,k}, \beta_{k+1}\rangle \end{cases} \\
\upsilon(k) &= \begin{cases} \upsilon(k+1) & \text{otherwise} \\ k & \text{if } (\gamma_{\beta,k}, \beta_{k+1}) \text{ are an incomposable pair} \end{cases}
\end{aligned}$$

We first show that the above is well-defined. To do so, it is sufficient to show that

$$\forall k \in \mathbb{N}^+ \; \exists l \in \mathbb{N}^+ \,.\, k \leq l \wedge (\gamma_{\beta,l}, \beta_{l+1}) \text{ are an incomposable pair} \tag{5.1}$$

[3]This means that we may not be able to compose β_k, β_{k+1} directly but there will be two morphisms equivalent to β_k, β_{k+1} which are composable. The result of this composition for $\beta_{\sigma(k)}, \dots, \beta_{\upsilon(\sigma(k))}$ is accumulated in $\gamma_{\beta,k}$. Actually, σ and υ depend on β as well, and hence should be indexed with β, but to maintain readability we only do this with γ.

Since $\mathsf{T}_1, \mathsf{T}_2$ are non-expanding, by lemma 4.3.2, $\mathrm{NF}(x)$ is a witness of x, i.e. for all $\alpha : x \to x'$, there is $\beta : \mathrm{NF}(x) \to x''$ s.t. $<\alpha> \equiv_M <\beta>$. Furthermore, for all $\alpha : y \to y'$ with $y \equiv_O x$, there is $\beta : \mathrm{NF}(x) \to x'$ with $<\alpha> \equiv_M <\beta>$, since $\mathrm{NF}(y) = \mathrm{NF}(x)$. To show 5.1, let $t \stackrel{def}{=} \delta_s(\beta_k)$ with $t \in T^w\mathcal{X}$, and let $i = k, k+1, k+2, \ldots$. If $(\gamma_{\beta,i}, \beta_{i+1})$ are an incomposable pair, $l \stackrel{def}{=} i$ and we are done; if not, there is α s.t. $<\alpha> \equiv_M <\gamma_{\beta,i}, \beta_{i+1}>$ and since $t \equiv_M \delta_s(\alpha)$, there is $\alpha' : t \to t'$ such that $<\alpha'> = <\alpha>$. So wolg. we may assume that $\delta_s(\alpha)$ is actually t, and hence all $\gamma_{\beta,i}$ are morphisms in $T^w\mathcal{X}$. Since for all $j \in \mathbb{N}^+$, $\beta_j \neq \mathbf{1}$, this gives rise to a sequence $z_0 < z_1 < z_2 < \ldots$ in $R^-(T^w\mathcal{X})$, with $z_0 \stackrel{def}{=} t$, $z_i \stackrel{def}{=} \delta_t(\gamma_{\beta,i})$. But $T^w\mathcal{X} \models \mathrm{SN}$, so this sequence cannot be infinite, hence there has to be an l such that $(\gamma_{\beta,l}, \beta_{l+1})$ are an incomposable pair.

Unfortunately, we are not quite done yet. We would like to define an infinite sequence the k-th element of which is $\gamma_{\beta,v(\sigma(k))}$; but from the definition, it merely follows that $\gamma_{\beta,v(\sigma(k))}$ and $\beta_{v(\sigma(k+1))} = \beta_{\sigma(k+1)}$ are an incomposable pair, not $\gamma_{\beta,v(\sigma(k))}$ and $\gamma_{\beta,v(\sigma(k+1))}$, as would be required. Hence, we iterate our construction by again composing as many morphisms as can be composed; and again appeal to strong normalization of the two monads $\mathsf{T}_1, \mathsf{T}_2$ for termination.

In detail, we define sequences $(\alpha^i_k)_{k\in\mathbb{N}^+}$ for $i \in \mathbb{N}$ as follows:

$$\begin{aligned} \alpha^0_k &\stackrel{def}{=} \gamma_{\beta,v(\sigma(k))} \\ \alpha^{i+1}_k &\stackrel{def}{=} \gamma_{\alpha^i,v(\sigma(k))} \end{aligned}$$

It remains to show that this iteration terminates, i.e. for all $k \in \mathbb{N}^+$, there is $m \in \mathbb{N}^+$ such that $\alpha^m_k = \alpha^{m+1}_k$, or equivalently $(\alpha^m_k, \alpha^m_{k+1})$ are an incomposable pair. This is shown by contradiction: assume there would be $k \in \mathbb{N}^+$ such that for all $m \in \mathbb{N}$, $\alpha^m_k \neq \alpha^{m+1}_k$. Without loss of generality, we can assume that this is the smallest such k, i.e. there is an $n \in \mathbb{N}$ s.t. $n \leq m$ and for all $l \leq k$, $\alpha^n_l = \alpha^{n+1}_l$. Then the source of all α^m_k with $n \leq m$ would be the same. Call this object t where $t \in T^w\mathcal{X}$. With $\mathsf{T}_1, \mathsf{T}_2$ non-expanding, $\mathrm{NF}(t)$ witnesses all α^m_k, and since $\alpha^m_k \neq \alpha^{m+1}_k$, there is an infinite sequence $z_0 < z_1 < z_2 < \ldots$ in $R^-(T^w\mathcal{X})$, with $z_0 \stackrel{def}{=} \mathrm{NF}(t)$, $z_i \stackrel{def}{=} \delta_s(\alpha^{m+i}_k)$. Since $T^w\mathcal{X} \models \mathrm{SN}$, there cannot be such a sequence, so for all $k \in \mathbb{N}^+$, there is $m \in \mathbb{N}^+$ such that $\alpha^m_k = \alpha^{m+1}_k$, or equivalently $(\alpha^m_k, \alpha^m_{k+1})$ are an incomposable pair. We then define the sequence $(\nu_k)_{k\in\mathbb{N}^+}$ in $\mathit{colim}\, D_{\mathcal{X}}$ as

$$\nu_k \stackrel{def}{=} \alpha^i_k \quad \text{where } \alpha^i_k = \alpha^i_{k+1}$$

for which by construction, (ν_k, ν_{k+1}) are an incomposable pair as required, hence $(\nu_k)_{k\in\mathbb{N}^+}$ is in infinite sequence in $\mathit{colim}\, D_{\mathcal{X}}$. □

The negation of lemma 5.2.3 gives us the following useful corollary:

Corollary 5.2.4 *Let* $\mathsf{T}_1, \mathsf{T}_2 \models SN$. *If* $\mathsf{T}_1, \mathsf{T}_2$ *are non-expanding and for every strongly normalizing category* $\mathcal{X}$, *there is no infinite sequence* $(\alpha_i)_{i \in \mathbb{N}^+}$ *in* $colim\, D_{\mathcal{X}}$, *then* $\mathsf{T}_1 + \mathsf{T}_2 \models SN$.

We now give a counterexample, due to Paul-André Melliès, showing why lemma 5.2.3 requires the monads to be non-expanding. Recall the monad $\mathsf{B} = \langle B, \eta_B, \mu_B \rangle$ from example 2.3.12 on page 75. It was there given on **Pre**; on **Cat**, it adds an object $\perp$ and for every object $x \in \mathcal{X}$ a morphism $\perp \to x$ to its argument category $\mathcal{X}$. It is not non-expanding, or more precisely does not preserve non-expandingness (see page 81), and by this property gives us a counterexample to the converse of lemma 5.2.3. Let $\Sigma \stackrel{def}{=} \{\mathtt{F}_1\}$ be the signature with just one unary operation, then in $T_\Sigma + B(\mathcal{X})$ (for any non-empty $\mathcal{X}$), there is the sequence

$$\begin{tikzcd}
\perp \arrow[r, "\beta_1"'] & \mathtt{'F('\perp)} \arrow[r, "\beta_2"'] & \mathtt{'F(''F('\perp))} \arrow[d, equal, "\equiv"] & & \\
& & \mathtt{'F(F('\perp))} \arrow[r, "\beta_3"'] & \mathtt{'F(F(''F('\perp)))} \arrow[d, equal, "\equiv"] & \\
& & & \mathtt{'F(F(F('\perp)))}\ \ldots &
\end{tikzcd}$$

Here, β_1 lives in $B(T_\Sigma(B(\mathcal{X})))$, and β_2 lives in $B(T_\Sigma(B(T_\Sigma(B(\mathcal{X})))))$; their composition (or more precisely, the morphism γ equivalent to the path $<\beta_1, \beta_2>$) lives in the latter as well. Similarly, β_3 cannot be directly composed with γ, but $\beta_3' : \mathtt{'F(''F('\perp))} \to \mathtt{'F(''F(''F('\perp)))}$ (which is equivalent) can. This way, we construct a sequence $\alpha_1 \stackrel{def}{=} \beta_1, \alpha_2 \stackrel{def}{=} \gamma_1, \alpha_3 \stackrel{def}{=} \beta_3' \cdot \gamma_1, \ldots$ in which the rank strictly increases ($\mathrm{rank}(\alpha_i) < \mathrm{rank}(\alpha_{i+1})$), but which is not an infinite sequence in the sense of definition 5.2.1. This is only possible because B does not preserve non-expandingness, in particular $B(T_\Sigma(B(T_\Sigma(\eta_B))))$ maps $\mathtt{'}\perp \in B(T_\Sigma(B(T_\Sigma(\mathcal{X}))))$ to $\mathtt{'}\perp \in B(T_\Sigma(B(T_\Sigma(B(\mathcal{X})))))$, and there is β_2 as above which does not come from any map in $B(T_\Sigma(B(T_\Sigma(\mathcal{X}))))$.

5.3 Modularity of Strong Normalization for Non-Collapsing Monads

As an application of our methodology, we will in this section prove the modularity of strong normalization for non-collapsing monads, and term rewriting systems. Our main tools will be lemma 3.2.14 and corollary 5.2.4. The argument will be that if there are no collapsing rewrite rules, then by lemma 3.2.14 there are no incomposable pairs (which correspond to new rewrites being created by collapsing of layers), and hence all sequences in $colimD_{\mathcal{X}}$ (for a strongly normalizing category

$\mathcal{X}$) will be of length one. Then by corollary 5.2.4 the coproduct monad will be strongly normalizing.

Lemma 3.2.14 says that an incomposable pair (α, β) is given by two morphisms such that α is layer-collapsing, and β μ-expansive. The most straightforward argument would be that if a monad is non-collapsing, which would mean that its unit is non-collapsing and the action preserves non-collapsingness, then there could be no layer-collapsing morphisms. Unfortunately, this argument is too restrictive, since only monads arising from term rewriting systems in which the variables of the right-hand side of every rule are contained in the left-hand side preserve non-collapsingness (this the dual of lemma 2.4.8); a counterexample is obtained by dualising the counterexample given on page 81.

So in order to show the result for all collapsing term rewriting systems, we can only assume that the units of the monads are non-collapsing. We will now first give a sketch of the proof for two monads arising from term rewriting systems, before generalizing the proof to arbitrary monads; when doing so, we will need to develop certain notions from term rewriting like "variables occuring in a term" in the monad setting.

5.3.1 Modularity of Termination for Non-Collapsing Term Rewriting Systems

Given two non-collapsing term rewriting systems $\Theta = (\Omega, R)$ and $\Psi = (\Sigma, S)$. The main lemma will be to show that for any strongly normalizing category $\mathcal{X}$ there is no incomposable pair in *colim* $D_{\mathcal{X}}$. By lemma 3.2.14, it is sufficient to show that if there is $\alpha : t' \to t$ in $T_\Theta(T_\Psi(T_\Theta(\mathcal{X})))$ which is in normal form (for all α' in $T_\Theta(T_\Theta(\mathcal{X}))$, $T_\Theta \eta_{\Psi, T_\Theta(\mathcal{X})}(\alpha') \neq \alpha$), but there is $s \in T_\Theta(T_\Theta(\mathcal{X}))$ such that $t = T_\Theta \eta_{\Psi, T_\Theta(\mathcal{X})}(s)$ (i.e. α is layer-collapsing), then for all $\beta : \mu_{\Theta, \mathcal{X}}(s) \to s''$ in $T_\Theta(\mathcal{X})$ there is $\beta' : s \to s'$ such that $\beta' = \mu_{\Theta, \mathcal{X}}(\beta)$; in other words, if there is a layer-collapsing rewrite, it cannot be followed by a μ-expansive one.

For the proof, we decompose the rewrite $\alpha : t' \to t$ into its "top layer" α_0 and the rest, as follows. For a term t, we have the set $var(t)$ of variables of t; here, since $t \in T_\Theta(T_\Psi(T_\Theta(\mathcal{X})))$, the variables are a subset of the objects of $T_\Psi(T_\Theta(\mathcal{X}))$. Let σ be the inclusion $\sigma : var(t) \hookrightarrow T_\Psi(T_\Theta(\mathcal{X}))$, then t can be given as $t = T_\Theta \sigma(t_0)$ where $t_0 \in T_\Theta(var(t))$ is the "top layer". Similarly, for α there is a rewrite α_0 in $T_\Theta(\mathcal{Z})$ where $\mathcal{Z}$ is the subcategory of $T_\Psi(T_\Theta(\mathcal{X}))$ given by those terms and rewrites between them occuring in α; we can define this by structural induction (just like the variables of a term on page 54). α is then given as $\alpha = T_\Theta \sigma(\alpha_0)$ where $\sigma : \mathcal{Z} \hookrightarrow T_\Psi(T_\Theta(\mathcal{X}))$ is the inclusion of $\mathcal{Z}$ into $T_\Psi(T_\Theta(\mathcal{X}))$.

Since α is a normal form, no rewrite γ in $\mathcal{Z}$ is in the image of $\eta_{\Psi,T_\Theta(\mathcal{X})}$, i.e. no rewrite is a variable, and because $\eta_{\Psi,T_\Theta(\mathcal{X})}$ is non-collapsing, no target of γ is in the image of $\eta_{\Psi,T_\Theta(\mathcal{X})}$:

$$\forall x \in var(t)\, \forall y \in T_\Psi(T_\Theta(\mathcal{X}))\,.\, x \neq \eta_{\Psi,T_\Theta(\mathcal{X})}(y) \tag{5.2}$$

Now furthermore assume that α is collapsing, i.e. there is $s \in T_\Theta(T_\Theta(\mathcal{X}))$ such that $T_\Theta\eta_{\Psi,T_\Theta(\mathcal{X})}(s) = t$. We now want to show that for all $\beta : \mu_{\mathcal{X}}(s) \to s'$ in $T_\Theta(X)$ there is $\beta' : s \to s''$ in $T_\Theta(T_\Theta(X))$ s.t. $\beta = \mu_{\mathcal{X}}(\beta')$.

We prove this by showing that under the assumptions just given t as above is a constant, i.e. $var(t) = \emptyset$. The proof proceeds by structural induction on t. Let $Z \stackrel{def}{=} var(t)$ be the set of variables of the whole term t. For the induction step, if $t = e(t_1, \ldots, t_n)$ then $T_\Theta\eta_{\Psi,T_\Theta(X)}(s) = e(t_1, \ldots, t_n)$ only if $s = e(s_1, \ldots, s_n)$ and $T_\Theta\eta_{\Psi,T_\Theta(X)}(s_i) = t_i$— that is the induction assumption. For the induction base, $t = {}'x$ (with $x \in Z$), then $T_\Theta\eta_{\Psi,T_\Theta(X)}(s) = {}'x$ only if $s = {}'y$, $\eta_{\Psi,T_\Theta(X)}(y) = x$. This can however be excluded by 5.2 above; hence there can only be an s such that $T_\Theta\eta_{\Psi,T_\Theta(X)}(s) = t$ if there are no variables in t.

Now if t is a constant, s is a constant as well, as will be $\mu_{\mathcal{X}}(s)$, and if we assume non-expandingness of the two term rewriting systems, a constant in $T_\Theta(X)$ only rewrites to another constant s' in $T_\Theta(X)$, which in turn can be given as a constant s'' in $T_\Theta(T_\Theta(X))$, giving rise to a rewrite $s \to s''$ in $T_\Theta(T_\Theta(X))$ as required.

5.3.2 The Monad Proof

The proof just sketched used a couple of concepts for terms which we have not yet developed in the monad setting:

- A term (or rewrite) from $T^w\mathcal{X}$ was decomposed into a top layer from $T_1\mathcal{X}$, and the rest from $T^w\mathcal{X}$;
- For a term $t \in T\mathcal{X}$, variables $x \in \mathcal{X}$ may or may not occur in t;
- From this, constants were defined (terms in which no variables occur), along with some properties.

For a monad $\mathsf{T} = \langle T, \eta, \mu \rangle$ on **Cat**, a term is an object of $T\mathcal{X}$ and a rewrite is a morphism in $T\mathcal{X}$, for some category $\mathcal{X}$ (see §2.3.4 on page 74); the category $\mathcal{X}$ is the context, its objects are the variables, and its morphisms the variable rewrites. But a term $t \in T_\Omega(X)$ need not contain all variables $x \in X$ (i.e. there are t s.t $var(t) \neq |\mathcal{X}|$), so what does it mean for an object $x \in \mathcal{X}$ to actually occur in $t \in T\mathcal{X}$? It seems easy to characterize when x does *not* occur in t; namely, if we can map x to different values while t remains the same. In other words, if there are two maps (i.e. functors) $\sigma_1, \sigma_2 : \mathcal{X} \to \mathcal{Y}$ such that $\sigma_1(x) \neq \sigma_2(x)$ but $T\sigma_1(t) = T\sigma_2(t)$. Then x occurs in t if $T\sigma_1(t) = T\sigma_2(t)$ implies $\sigma_1(x) = \sigma_2(x)$. This allows us to define the set, or rather the category, of variables of $t \in T\mathcal{X}$ as a subcategory of the context $\mathcal{X}$. This notion easily carries over to a morphism $\alpha : s \to t$ in $T\mathcal{X}$.

Definition 5.3.1 (Category of Variables) Given a monad $\mathsf{T} = \langle T, \eta, \mu \rangle$ on **Cat**, then for $t \in T\mathcal{X}$, for some $\mathcal{X}$, we say that for $x \in \mathcal{X}$,

$$x \text{ occurs in } t \Leftrightarrow \forall \sigma_1, \sigma_2 : \mathcal{X} \to \mathcal{Y} \,.\, T\sigma_1(t) = T\sigma_2(t) \Rightarrow \sigma_1(x) = \sigma_2(x)$$

We similarly define when a variable rewrite $f : x \to y$ in $\mathcal{X}$ occurs in t, and when x or f occur in $\alpha : s \to t$ in $T\mathcal{X}$.

The *category of variables* $\mathbf{Var}(t)$ of t (and $\mathbf{Var}(\alpha)$ of α) is defined as the smallest subcategory of $\mathcal{X}$ given by all objects and morphisms occuring in t or α, respectively.

Note that for an arbitrary monad we can have non-identity variable rewrites $f : x \to y$ occuring in a term $t \in T\mathcal{X}$; this will not occur with monads given by term rewriting systems, since it corresponds to the presence of an operation with a non-discrete arity, e.g. an operation F which takes two arguments such that the first argument has to rewrite to the second one.

It is easy to show that for a monad T_Θ given by term rewriting system Θ, the objects of $\mathbf{Var}(\mathcal{X})$ are isomorphic to the set $var(t)$ of variables of t for all $t \in T_\Theta(X)$; this is done by a structural induction on t, showing that if $x \in var(t)$, x occurs in t as defined above, and on the other hand if x occurs in t than it has to be an element of $var(t)$. Further, if η is monic we can show that x occurs in $\eta(y)$ iff $x = y$, using naturality if η: $T\sigma_1(\eta_\mathcal{X}(x)) = T\sigma_2(\eta_\mathcal{X}(x))$ iff $\eta_\mathcal{Y}(\sigma_1(x)) = \eta_\mathcal{Y}(\sigma_2(x))$ iff $\sigma_1(x) = \sigma_2(x)$.

A fact that is actually needed below is that the variables occuring in the source and target of a rewrite will occur in the variables of the rewrite itself as well:

Lemma 5.3.2 For a morphism $\alpha : s \to t$ in $T\mathcal{X}$, $\mathbf{Var}(s)$ and $\mathbf{Var}(t)$ are subcategories of $\mathbf{Var}(\alpha)$: $\mathbf{Var}(s) \hookrightarrow \mathbf{Var}(\alpha)$ and $\mathbf{Var}(t) \hookrightarrow \mathbf{Var}(\alpha)$

Proof. Given $x \in \mathcal{X}$, we want to show that if x occurs in s, then x occurs in α as well. Since $T\sigma_1, T\sigma_2$ are functors, we have first implication below, and if x occurs in s we have the second implication:

$$\begin{aligned} T\sigma_1(\alpha) = T\sigma_2(\alpha) &\Rightarrow T\sigma_1(s) = T\sigma_2(s) \\ T\sigma_1(s) = T\sigma_2(s) &\Rightarrow \sigma_1(x) = \sigma_2(x) \end{aligned}$$

so taking the two together, we can conclude that x occurs in α.

A similar argument holds for the variable rewrites $f : x \to y$ occuring in s and hence α, making $\mathbf{Var}(s)$ a subcategory of $\mathbf{Var}(\alpha)$; and similarly, we can show that $\mathbf{Var}(t)$ is a subcategory of $\mathbf{Var}(\alpha)$. $\square$

We can now decompose any object t or morphism $\alpha : u \to v$ in $T_i T^w(\mathcal{X})$ (with $i \in \mathcal{L}, w \in W$) into $t_0 \in T_i(\mathbf{Var}(t))$, $\sigma : \mathbf{Var}(t) \hookrightarrow T^w(\mathcal{X})$ with $t = T\sigma(t_0)$, or $\alpha_0 : u_0 \to v_0$ in $T_i(\mathbf{Var}(\alpha))$, $\tau : \mathbf{Var}(\alpha) \hookrightarrow T^w(\mathcal{X})$ (where τ, σ are the obvious embeddings).

Constants are objects t or morphisms α in $T\mathcal{X}$ the category of variables of which is the empty category, $\mathbf{0}$. The empty category is initial in $\mathbf{Cat}$, i.e. for any other category $\mathcal{Y}$ there is a unique functor $!_{\mathcal{Y}} : \mathbf{0} \to \mathcal{Y}$. Then the fact that constants remain constants under substitution of variables, and that in a non-expanding system constants can only rewrite to constants follow categorically: for any monad $\mathsf{T} = \langle T, \eta, \mu \rangle$ on $\mathbf{Cat}$ and for all categories $\mathcal{Y}$,

$$\mu_{\mathcal{Y}}(T!_{T\mathcal{Y}}) = T!_{\mathcal{Y}} \tag{5.3}$$

which is proven by diagram 5.4, where the left triangle is T applied to $!_{T\mathcal{Y}} = \eta_{\mathcal{Y}} \cdot !_{\mathcal{Y}}$,

$$\begin{tikzcd}
T\mathbf{0} \arrow[r, "T!_{\mathcal{Y}}"] \arrow[dr, "T!_{T\mathcal{Y}}"'] & T\mathcal{Y} \arrow[d, "T\eta_{\mathcal{Y}}"] \arrow[dr, equal] & \\
& TT\mathcal{Y} \arrow[r, "\mu_{\mathcal{Y}}"'] & T\mathcal{Y}
\end{tikzcd} \tag{5.4}$$

the uniqueness of $!$, and the right triangle is one of the monad laws.

Further, if T is non-expanding, then $T(!_{\mathcal{Y}})$ is non-expanding, i.e. for all $t \in T\mathbf{0}$, if there is $\alpha : T!_{\mathcal{Y}}(t) \to s$, then there is $\alpha'(t) \to s'$ in $T\mathbf{0}$ s.t. $T!_{\mathcal{Y}}(\alpha') = \alpha$. This follows because $!_{\mathcal{Y}} : \mathbf{0} \to \mathcal{Y}$ is non-expanding (simply because there is no $x \in \mathbf{0}$ for which there could be $f : !_{\mathcal{Y}}(x) \to y$ in $\mathcal{Y}$), and if T is non-expanding, it preserves this, making $T!_{\mathcal{Y}}$ non-expanding.

We have now defined and proven the main notions and properties to show the main technical lemma:

Lemma 5.3.3 Let $\mathsf{T}_1, \mathsf{T}_2$ be strongly normalizing, regular, non-expanding and non-collapsing monads on $\mathbf{Cat}$, and $\mathcal{X}$ be a strongly normalizing category. Let $\alpha' : r' \to t'$ be a morphism in $T_1 T_2 T_1 \mathcal{X}$ with $\alpha : r \to t$ its normal form $\alpha = \mathrm{NF}(\alpha')$ such that α is layer-collapsing, i.e. there is $s \in T_1 T_1 \mathcal{X}$ s.t. $t = T_1(\eta_{2,T_1\mathcal{X}}(t_0))$. Then

$\mu_{1,\mathcal{X}}$ is not expansive at s, i.e. for all $\beta : \mu_{1,\mathcal{X}}(s) \to s'$ in $T_1\mathcal{X}$, there is $\beta' : s \to s''$ such that $\mu_{1,\mathcal{X}}(\beta') = \beta$.

Proof. We decompose α into α_0 in $T_1(\mathbf{Var}(\alpha))$, and $\sigma : \mathbf{Var}(\alpha) \to T_2T_1\mathcal{X}$ such that $\alpha = T_1\sigma(\alpha_0)$; and similarly, t into $t_0 \in T_1(\mathbf{Var}(t))$, $\tau : \mathbf{Var}(t) \to T_2T_1\mathcal{X}$ such that $t = T_1\tau(t_0)$. Since α is a normal form, for all κ in $\mathbf{Var}(\alpha)$ there is no λ in $T_1\mathcal{X}$ such that $\eta_{2,T_1\mathcal{X}}(\lambda) = \sigma(\kappa)$; and then since η_2 is non-collapsing, for all $x \in \mathbf{Var}(\alpha)$ there is no $v \in T_1\mathcal{X}$ such that $\eta_{2,T_1\mathcal{X}}(v) = \sigma(x)$. By lemma 5.3.2, $\mathbf{Var}(t)$ is a subcategory of $\mathbf{Var}(\alpha)$, hence this holds in particular for all $x \in \mathbf{Var}(t)$: there is no $v \in T_1\mathcal{X}$ such that $\eta_{2,T_1\mathcal{X}}(v) = \tau(x)$. On the other hand, $t = T_1\eta_{2,T_1\mathcal{X}}(t_0)$, hence for all $x \in \mathbf{Var}(t)$, there is $u \in T_1\mathcal{X}$ such that $\eta_{2,T_1\mathcal{X}}(u) = \tau(x)$.

Since no $x \in \mathbf{Var}(t)$ can satisfy both of these properties, we conclude $\mathbf{Var}(t)$ has no objects (or morphisms): $\mathbf{Var}(t) \cong \mathbf{0}$. Then $\tau : \mathbf{Var}(t) \to T_2T_1\mathcal{X}$ is the unique morphism $!_{T_2T_1\mathcal{X}} : \mathbf{0} \to T_2T_1\mathcal{X}$. By T_1 applied to uniqueness of $!$, $T_1!_{T_2T_1\mathcal{X}} = T_1\eta_{2,T_1\mathcal{X}} \cdot T_1!_{T_1\mathcal{X}}$, hence $t = T_1\eta_{2,T_1\mathcal{X}}(s) = T_1\eta_{2,T_1\mathcal{X}} \cdot T_1!_{T_1\mathcal{X}}(t_0)$, and by injectivity of $T_1\eta_2$ (given by regularity of T_1 and T_2) $s = T_1!_{T_1\mathcal{X}}(t_0)$. This means that s is a constant as well, as it can be decomposed into $t_0 \in T_1\mathbf{0}$ and $!_{T_1\mathcal{X}} : \mathbf{0} \to T_1\mathcal{X}$.

By equation 5.3, $\mu_{1,\mathcal{X}}(s) = \mu_{1,\mathcal{X}}(T_1!_{T_1\mathcal{X}}(t_0)) = T_1!_{\mathcal{X}}(t_0)$. Since $T_1!_{\mathcal{X}}$ is non-expanding, for all $\beta : \mu_{1,\mathcal{X}}(s) \to s'$, i.e. $\beta : T_1!_{\mathcal{X}}(t_0) \to s'$, there is $\gamma : t_0 \to s_0$ such that $\beta = T_1!_{\mathcal{X}}(\gamma) = \mu_{1,\mathcal{X}}(T_1!_{T_1\mathcal{X}}(\gamma))$, hence we have $\beta' \stackrel{def}{=} T_1!_{T_1\mathcal{X}}(\gamma)$ with $\mu_{1,\mathcal{X}}(\beta') = \beta$ as required. $\square$

Theorem 5.3.4 *The coproduct of two strongly normalizing, non-collapsing, regular, and non-expanding monads $\mathsf{T}_1, \mathsf{T}_2$ on* **Cat** *is strongly normalizing.*

Proof. Given a strongly normalizing category $\mathcal{X}$, we have to show that the coproduct $\mathsf{T}_1 + \mathsf{T}_2$ at $\mathcal{X}$, given by $colim\, D_{\mathcal{X}}$, is strongly normalizing.

By corollary 5.2.4, it is sufficient to show that for all strongly normalizing categories $\mathcal{X}$, there are no infinite sequences $(\alpha_i)_{i\in\mathbb{N}^+}$ in $colim\ D_{\mathcal{X}}$. Such an infinite sequence consists of morphisms α_i in $\coprod_{w\in W} T^w\mathcal{X}$ such that for all $k \in \mathbb{N}^+$, $<\alpha_1, \ldots, \alpha_k>$ is a path of minimal length; then by definition 3.2.11 all (α_i, α_{i+1}) have to be incomposable pairs.

We will show that there are no incomposable pairs; we can then conclude that there is no infinite sequence. By lemma 3.2.14, an incomposable pair (α, β) is given by two morphisms $\alpha : x_1 \to y_1$, $\beta : x_2 \to y_2$ in normal form such that there are $r, s \in W, i, j \in \mathcal{L}, i \neq j$ and $z \in T^{riis}\mathcal{X}$ where α is layer-collapsing at z with $y_1 = \eta^{ri}_{j,is}(z)$, and β is μ-expansive at z: $x_2 = \mu^r_{i,s}(z)$, and for all $\beta' : z \to z'$,

$\mu^r_{i,s}(\beta') \neq \beta$. (We can exclude the second case from lemma 3.2.14 here, because both monads are non-expanding.)

Any layer-collapsing morphism $\alpha : s \to t$ in $T^{rijis}\mathcal{X}$ gives rise to a layer-collapsing morphism $\alpha' : s' \to t'$ in $T^{ijis}\mathcal{X}$, for which by lemma 5.3.3 there are no μ-expansive morphisms in $T^{iis}\mathcal{X}$ (set $\mathcal{X}$ in lemma 5.3.3 to $T^s\mathcal{X}$, which is a strongly normalizing category.) Since T_1 and T_2 are non-expanding, they will preserve the non-expandingness of $\mu_{i,\mathcal{X}}$, hence there is no μ-expansive β in T^{ris} and hence there cannot be an incomposable pair. $\square$

Modularity of strong normalization for non-collapsing term rewriting systems then follows as a corollary of this theorem:

Corollary 5.3.5 *The disjoint union of two strongly normalizing, non-collapsing, term rewriting systems which do not introduce unbounded variables is strongly normalizing.*

Proof. Given two strongly normalizing, non-collapsing, non-expanding term rewriting systems Θ_1, Θ_2 which do not introduce unbounded variables, the monads T_{Θ_1} and T_{Θ_2} are strongly normalizing by proposition 5.1.5, non-collapsing by lemma 2.4.9, regular by proposition 2.4.4 and non-expanding by lemma 2.4.8. Then by theorem 5.3.4, $\mathsf{T}_{\Theta_1} + \mathsf{T}_{\Theta_2}$ is strongly normalizing, and by proposition 2.5.3, $\mathsf{T}_{\Theta_1} + \mathsf{T}_{\Theta_2} \cong \mathsf{T}_{\Theta_1 + \Theta_2}$. Hence (again by proposition 5.1.5), $\Theta_1 + \Theta_2$ is strongly normalizing. $\square$

5.4 Summary and Conclusion

In this chapter, we have investigated modularity of strong normalization (termination) for monads.

We have defined a semantic notion of termination for categories and monads, and showed it to be equivalent to the usual, syntactic definition found in the literature, in the sense that the monad T_Θ is terminating according to the semantic definition if and only if the term rewriting system Θ is terminating according the syntactic definition.

We have given a characterization of termination of the coproduct monad of two terminating monads by infinite sequences of morphisms from the components $T^w\mathcal{X}$ of $\coprod_{w \in W} T^w\mathcal{X}$, i.e. the coproduct is terminating if and only if the process of forming paths terminates. In one direction, this characterization only holds for non-expanding monads. This technical lemma is an important tool in proving the strong normalization of the coproduct monad.

Unlike confluence, termination is not modular in general, it only holds under certain assumptions on the term rewriting systems in questions. Usually these formulations are formulated rather syntactically. Here we have proven modularity of strong normalization for non-collapsing monads on a semantic level, using purely categorical reasoning without recourse to the syntax. In order to be able to this, we had to develop categorical counterparts of certain syntactic properties, e.g. we had to define a notion of a variable (an object $x \in \mathcal{X}$) occuring in a term (an object $t \in T\mathcal{X}$) for an arbitrary monad T. The proof demonstrates that our approach is also suitable for modularity proofs for strong normalization.

All in all, strong normalization with the inherent notion of "counting steps" seems less amenable to categorical analysis than confluence, which after all is just completing diagrams. Nevertheless we were able to show some modularity results, hopefully setting the scene for some more in the future. A worthwhile target would be a categorical version of the results by Gramlich [23] and Ohlebusch [62] that strong normalization is modular for a system which is strongly normalizing under non-deterministic collapses, or an appropriate categorical version thereof. In the long run, exploring a more abstract definition of strong normalization (perhaps along the lines of Hilken [29] mentioned above) may be potentially fruitful avenue of further research.

Chapter 6

Conclusions and Further Work

The aim of this thesis was to give a semantics to term rewriting systems which can express the key concepts such as layers and substitutions, but abstracts from unnecessary syntactic details (such as "contexts"), to demonstrate the advantages of the semantics by applying it to modularity problems, and to show the usefulness of (enriched) category theory in the process.

Summary

Our starting point was the modelling of equational presentations by monads on the category of sets.

We have then generalized this to the modelling of term rewriting systems by monads on the category of sets-with-structure. Depending on what aspect of term rewriting one is interested in (one-step *vs.* many-step, named *vs.* unnamed reductions), sets-with-structure can be relations, preorders, categories etc, and we can instantiate the general theory to any of these. The theoretical basis is enriched category theory, in particular the theory of finitary enriched monads; the crucial point here is that the construction has to be properly enriched. One of the consequences of this is that the finitely presentable objects of the category of sets-with-structure are the arities of the operations, leading to generalized rewrite rules in which rewrites between variables are allowed.

We have shown that the mapping of term rewriting systems to monads has a right adjoint, given by the internal language of a monad, justifying our calling the semantics compositional; the working assumption here (following Goguen and Burstall) is that many important structuring operations can be expressed as colimits. We have shown in detail how to construct the coproduct of two finitary monads (corresponding to the disjoint union of two term rewriting systems), and sketched the construction of the coequalizer; from these, any other colimit can be constructed.

As an application of our semantics, we have turned our attention to modularity problems, in particular the modularity of confluence and strong normalization under the disjoint union. We have proven modularity of confluence (Toyama's theorem), and the modularity of strong normalization for non-collapsing term rewriting systems. By proving these results in our more general, abstract, categorical framework we were able to extend Toyama's theorem to systems which introduce so-called bounded variables on the right, and to quasi-non-expanding systems.

In the rest of this concluding chapter, we will review other approaches to categorical term rewriting, and compare them with our work. We will close this chapter by indicating directions of future research.

6.1 Related Work

Categorical term rewriting — using category theory to give models of term rewriting systems — is by no means new. We will now give a (necessarily subjective and incomplete) review of the literature in this field, in order to assess the significance of the work presented in this thesis.

As far as the category theory is concerned, the reader is referred to [69] for an account of the work on monads. Suffice it to say here that the modelling of equational presentations started with Linton [48], and that its generalization, to base categories other than **Set** and to enriched categories, is due, apart from Kelly and Power [39], to Dubuc and Kelly [11], and Burroni [5].

6.1.1 Early Work

The idea of representing term rewriting systems by 2-categories (or other enriched categories), in which the morphisms represent the terms and the 2-cells the reductions between them, goes back to two seminal papers by Seely [81] and Rydeheard and Stell [73].

Seely [81] argues that "2-categories occur naturally as structures in computer science", and as an example presents a 2-category **LAMBDA** which has the types of the simply typed λ-calculus as objects, the terms as morphisms and $\beta\eta$-reductions between the terms as 2-cells. The motivation for putting 2-cells in a perfectly ordinary category lie in categorical logic: both proofs in intuitionistic first-order logic and terms of the simply typed λ-calculus can be presented by a cartesian closed category, in which the types are objects and the terms are morphisms (see [45]). Then reductions between proofs, as studied in proof theory [67], lead to 2-cells between morphisms and 2-categories as models of these reductions (called hyperdoctrines [80]), and since reductions between proofs correspond to reductions in the λ-calculus, one arrives at the 2-category sketched above.

The foundations for this work have been laid by the "Australian school" around the Sydney Category Seminar and their work on 2-categories and enriched categories; in particular, Street [86] proposes the notion of a *computad* which is essentially a Semi-Thue or string rewriting system, but without making the relation to term rewriting explicit; this has been left to Power [64].

Rydeheard and Stell [73] start from a completely different (and more familiar) angle: the categorical treatment of universal algebra. They consider the Kleisli-category $\mathbf{Set}_{\mathsf{T}_\Omega}$ of the monad T_Ω given by a signature Ω: this has sets of variables as objects, and substitutions of variables as morphisms (i.e. morphisms in $\mathbf{Set}_{\mathsf{T}_\Omega}$ are of the form $\sigma : X \to T_\Omega(Y)$, where X and Y are sets, which we can think of as assignment of variables from X to terms in $T_\Omega(Y)$), and proceed to introduce reductions on the substitutions as 2-cells. This work has been developed by Stell in his thesis [82] and the subsequent [83], in which he advocates the use of Sesqui-categories instead of 2-categories, because they have a notion of length which 2-categories lack.

6.1.2 Later and Contemporary Work

The work of Seely [81] was further developed by Jay and Ghani [31, 32, 17, 18]. They investigate structures modelling reductions for a specific term rewriting systems, namely the typed λ-calculus and extensions of it. The category theory is used to arrive at "well-behaved" rewrite rules; in particular [17] proposes the idea that the introduction and elimination rules for a particular type should be adjoint to each other. This for example leads to η-expansions rather than η-contractions, and was successful in solving a long standing problem about the confluence of the simply typed λ-calculus with a coproduct type. [17] lead to a series of papers, systematically applying the theory of η-expansions to increasingly complicated type theories, from System F [19] to the Calculus of Constructions [20].

The 2-category $\mathbf{T}_\Omega$ constructed in [73] can be obtained essentially as the Kleisli-category of the monad T_Θ of proposition 2.3.9, except that $\mathbf{T}_\Omega$ has sets rather than categories as objects. Stell [82, 83] uses Sesqui-categories rather than 2-categories, but the construction remains the same. All of this is more concerned with investigating the structure of the reductions given by the term rewriting systems, rather than finding a compositional semantics as is our aim; for example, a typical theorem [82, Theorem 5.4.2] shows that a specific kind of spans[1] in the Sesqui-category corresponds precisely to the critical pairs of the term rewriting system. Hilken [29] investigates the algebraic structure given by the 2-cells in the categorical models of the typed λ-calculus in order to develop a proof theory for the reductions in the typed λ-calculus.

Gray, in a series of papers, proposed "enriched sketches" to model typed λ-calculi [28] and algebraic datatypes [26, 25, 27]. This work uses the formalism of sketches to investigate properties of the datatypes and term rewriting systems specified (to great effect: [28] shows that the typed λ-calculus can be given as an initial algebra for a finite limit sketch, and derives from it an implementation in the rewriting tool *Mathematica*), and is not concerned with modularity.

Johnson [33], on the other hand, goes a step further and gives a 3-categorical model of (linear) term rewriting systems. This allows to consider linear term rewriting systems as Semi-Thue systems with an extra dimension, but it is not clear that the additional complications of 3-categories outweigh this simplification.

A rather different approach is put forward by Melliès, which he calls *axiomatic rewriting* [55, 56]. He defines an *Axiomatic Rewrite System* as a graph with a reduction (or derivation) structure on the paths, giving rise to a 2-category, which is then shown to satisfy certain standardization properties. The key point here is the correlation of the Church-Rosser property to the existence of push-outs, and the standardization to factorization systems as in [14].

6.1.3 Other Work

Meseguer [57] uses a generalized form of term rewriting systems called rewrite theories to model concurrent systems. Briefly, a rewrite theory is a a signature with equations, and conditional term rewrite rules of the form $[s] \twoheadrightarrow [t]$ **if** $[s_1] \twoheadrightarrow [t_1], \ldots, [s_n] \twoheadrightarrow [t_n]$ meaning the equivalence class $[s]$ rewrites to $[t]$ if all $[s_i]$ rewrite to $[t_i]$. The construction of the syntactic model $T_{\mathcal{R}}(X)$ for a rewrite theory $\mathcal{R}$ is somewhat similar to the construction of the term reduction algebra $T_{\Theta}(\mathcal{X})$ (similar rules to generate derivations, similar equations on them), but it is far more general in scope (providing a unified model of concurrency). The common feature here, also occurring in Milner's action structures [60] is the use of enriched categories with a monoidal structure, the morphisms of which are terms, and the enrichment models the reduction. [9] investigates the relation between Meseguer's rewriting logic and Stell's modelling of term rewriting further, though not necessarily more deeply.

[1] Two 2-cells with the same source.

Power [65] considers a categorical formulation of Hoare's data refinement [30] in terms of enriched monads over **Pos** as an instantiation of the enriched monad theory in [39], but although the categorical framework is essentially the same as used in this thesis, no explicit connection to term rewriting is made.

Reichel [68] and Stokkermans [84] both consider critical-pair completion procedures in a 2-categorical framework. This allows an abstract notion of critical pair completion, subsuming such apparently diverse problems as Knuth-Bendix completion [43], construction of Gröbner bases or resolution [70]. Although an elegant abstract notion, it does not lend itself to new results.

Although the phrase "free"[2] is liberally used to describe categorical models of term rewriting systems [83, 57], only [86] explicitly exhibits the adjunction between syntax (computads) and semantics (the category of 2-categories) (and in particular, the right adjoint).

6.1.4 Discussion

As mentioned above, the main novelties of the semantics presented in thesis are:

- the construction of the semantics as a monad;
- the generalized rewrite rules and variable rewrites;
- the freeness of the construction, exhibited by the adjunction in proposition 2.5.3;
- the modularity results obtained from it.

The monad gives the abstract way of building the semantics, and the theory of enriched monads tells us how the terms are constructed, what arities are, what algebras are etc. Separate from this, the different choices of enrichment correspond to different models of reduction, so we can model all of ordered categories, Sesqui-categories or 2-categories in this framework. This is precisely what makes the semantics amenable to generalizations in various respects (see §6.2 below): the "modularization" of the semantics allows to change parts of it without having to rebuild it from scratch.

When the work mentioned in §6.1.1– §6.1.3 lead to new results (e.g. [18]), these are mostly proven syntactically, on the level of terms; that term rewriting results (like the generalization of Toyama's theorem) are obtained using purely

[2] Sometimes, "free" is used to describe a left adjoint (e.g. [12]). This wording is unfortunate, since the more common understanding in calling a functors $F : \mathcal{C} \to \mathcal{D}$ free is that $\mathcal{D}$ is monadic over $\mathcal{C}$.

categorical reasoning is another distinguishing feature of this work. In fact, none of the work mentioned above tackled modularity, probably because there is no canonical way to combine arbitrary 2-categories. And while I would not claim that the construction of the coproduct monad is particularly simple, it is far simpler than the corresponding combination of 2-categories.

6.2 Future Work

The work in this thesis hopefully serves as a starting point for future research.The amenability to generalization has been pointed out as one of the advantages of the approach, so possibilities for future work exist at least in three directions:

- generalizing the term rewriting;
- improvements of the theory;
- or generalizing the structuring operations.

6.2.1 Generalizing the Term Rewriting

This thesis dealt with unconditional, single-sorted, first-order term rewriting systems. The generalization to many-sorted term rewriting systems is straightforward (see page 91), but one can envisage other more challenging generalizations:

- Conditional term rewriting systems.

  A conditional term rewriting system is given by a signature Ω, and rules $l \to r \Leftarrow s_1 \approx t_1 \dots s_n \approx t_n$. The $\approx$ can be interpreted in at least three different ways, giving rise to different variations of conditional rewrite systems (see e.g. [3]).

  One way to model a conditional term rewriting system C in this framework would be to construct an endofunctor S_C on the category Ω-**Alg** of Ω-algebras in **Pre**, taking any algebra A to the smallest algebra A' s.t. $A' \models l \to r$ if $A \models s_i \approx t_i$ for all $i = 1, \dots, n$ and all conditional rules $l \to r \Leftarrow s_1 \approx t_1 \dots s_n \approx t_n$, or taking A to A if this is not the case. Then let T_C be the free monad on S_C, and by precomposition with the adjunction between **Pre** and Ω-**Alg** to obtain a monad on **Pre**. Another way to model conditional systems would be to encode the conditions in the arities (using non-discrete arities); this would mean that the arities could be non-finite, and would lead to non-finitary monads.

- Variable rewrites.

  Variable rewrites allow us to write down more things than before, for example the equivalence closure of a term rewriting systems as a term rewriting system (see example 2.3.6 on page 71). In effect, this is a (albeit rather limited) form of conditional term rewriting, and it would be interesting to investigate the limits of the additional expressivity gained.

- Hidden sorts and operations.

  There should be a way to model hidden operations (and sorts, for the many-sorted case), and observational equivalence, perhaps by using final semantics rather than initial semantics [91]. The monad framework might even allow to combine initial and final semantics [90].

- λ-calculi.

  It is well-known how to model typed λ-calculi in cartesian closed categories [45], and how closed cartesian categories can be obtained as algebras for a 2-monad on **Cat**; so it should be possible to generalize this to the reduction structure given by the η-expansion and β-reduction. It would be particularly attractive to be able to use the modularity results of chapters 4 and 5 to show confluence and strong normalization of different variations of the typed λ-calculus.

- Orthogonal term rewriting systems.

  A term rewriting system is *orthogonal* if it is left-linear, and the rules contain no overlapping left sides. This is actually a specialization rather than a generalization, but orthogonal systems enjoy a lot of desirable properties (e.g. they are confluent), and it might be worthwhile to arrive at an abstract notion of orthogonality on the level of monads, perhaps along the lines of [63]. Related to this, one might want to consider critical pairs and completion procedures in this setting, as in [84] and [68].

- Rewriting modulo equations.

  The generalization of the monad construction to a rewriting system with equations (aka. a rewriting logic [57]) is straightforward and has already been sketched above (page 91). However, the resulting monad does not preserve coequalizers and will hence not be strongly finitary (only finitary), so neither the coproduct construction nor the modularity results apply. Hence, the treatment of rewriting modulo equations, in particular modularity results for these systems, will require a more general construction for the coproduct.

- Connection with axiomatic rewriting.

  The connection of the approach presented here with Melliès' axiomatic rewriting [55, 56] is far from clear, and deserves further clarification. For example, can we derive similar modularity results as in chapter 4 if we replace our definition of confluence with Melliès' pushout property?

- Other properties.

  Although confluence and strong normalization are the most important properties of term rewriting systems, one might want to prove to investigate modularity results for other properties such as weak normalization (which is modular, see e.g. [59], where further modularity results for other properties can be found as well). In particular, a categorical proof of the modularity of completeness for left-linear systems [89] would be an interesting challenge ([54] presents a simplified proof of this result).

6.2.2 Improving the Theory

The coproduct construction as given in chapter 3 is rather ad hoc: it assumes that the two monads in question are strongly finitary, a quite strong restriction. A more general and systematic approach (suggested by John Power) would start from the observation that the category of monads on $\mathcal{A}$ is itself monadic over the category of finitary endofunctors on $\mathcal{A}$, and then apply the construction of colimits of algebras from [2, Chapter 9.3] and [49].

Another area worthy of improvement are the regularity conditions on the monads in §2.4.2. The present definitions are more a convenient technical shortstop than a good attempt at an axiomatization, and it would be worthwhile to investigate whether they can be replaced by a set of conditions which would be easier to verify, less technical in nature, or hopefully both. In particular, it should be investigated wether one can replace definition 2.4.2 by requiring the unit and multiplication to be cartesian natural transformations (see also [7], and the discussion on page 80).

6.2.3 Generalizing the Structuring Operations

We have only considered coproducts of monads above, so the obvious generalization here would be to consider modularity for coequalizers of monads. Unfortunately, in this generality we even lose modularity of confluence (coequalizers correspond to arbitrary quotienting, and it is easy to see that this destroys confluence). Perhaps attention should be restricted to particular colimits such as push-outs modelling unions with shared constructors. This would correspond to a pushout diagram like 6.1 in which the inclusions i_1, i_2 are non-expanding.

$$\begin{tikzcd}
\Theta_0 \arrow[r, hook, "i_1"] \arrow[d, hook, "i_2"'] & \Theta_2 \arrow[d] \\
\Theta_2 \arrow[r] & \Theta_1 +_{\Theta_0} \Theta_2
\end{tikzcd} \tag{6.1}$$

6.3 Concluding Remarks

The aim of this thesis was to present a semantics for term rewriting systems which is on the one hand abstract enough to be able to forget all about the details of the syntax, and on the other hand still expressive enough to be able to prove the properties arising from the syntax.

I hope the reader will agree that the proofs given in chapter 4 and 5 justify these claims. All in all, this work should just be a start — the real test will be to arrive at really new term rewriting results, along the lines of the future research mentioned above. A semantics is after all only useful if it gives us new insights into the area we study.

Appendix A

Using Monads to Model Named Reductions

In this appendix, we will elaborate on the remarks on page 67 and show how to model named reductions by a monad on the category of small categories. The construction contains three main steps, just like the corresponding construction for unnamed reductions in chapter 2.

Apart from showing how the generic framework of enriched monad theory adumbrated in §2.2 can be instantiated to a different framework then preorders, named reductions are important in their own right for two reasons:

- When considering strongly normalizing systems, preorders are actually not precise enough, because the reflexive closure destroys strong normalization for any term reduction algebra: the term reduction algebra from definition 2.3.4 will never be strongly normalizing, neither as a relation (by definition 1.6.4) nor as a category (by definition 5.1.1). In order to make them strongly normalizing by definition 5.1.1, we need to be able to tell apart the identity rewrite $\mathbf{1}_t : t \to t$ on any term t from other, cyclic rewrites from t to itself.

- When considering quasi-non-expanding systems, we actually want equations on the reductions. We would like to be able to express that a reduction $\alpha : t \to s$ has a *retract* $\alpha^{-1} : s \to t$, which when composed with α yields the identity on t: $\alpha^{-1} \cdot \alpha = \mathbf{1}_t$. In order to formulate equations on reductions, they have to be named.

The structure of this chapter mirrors the structure of chapter 2: we first define a monad T_Θ on the category **Cat** in §A.1 (as in §2.3)), then show that the monad is regular (§A.2, corresponding to §2.4), and finally show that there is an adjunction between the category of term rewriting systems and the category of monads on **Cat** (§A.3, corresponding to §2.5).

A.1 A Term Construction for Named Reductions

In chapter 2, we have seen how term rewriting systems can be modelled by monads on the category of sets-with-structure, generalizing the treatment of equational presentations. The choice of the particular set-with-structure depends on which aspect of term rewriting one is interested in, and in §2.3, we have seen how a monad on the category **Pre** of preorders models unnamed many-step reductions. In this section of the appendix, we will see how to model *named* many-step reductions by a monad on the category **Cat** of all small categories.

The basic construction will be the same— closing the rules under substitution, application of operations, reflexivity (identities) and transitivity (composition), but the named reductions offer us some technical complications. Since we now want to distinguish different reductions between the same two terms, we have to introduce a term structure on the reductions. This in turn means that where in the unnamed case it was enough to show that the definitions on the objects preserve the order structure, we now have to define everything in the morphisms explicitly.

We will, for a category $\mathcal{X}$, construct the *term named-reduction algebra* $T_\Theta(\mathcal{X})$, a category which has as objects the terms built over the objects of $\mathcal{X}$, and as morphisms named reductions between the terms. These reductions will be freely generated, and then quotiented in order to make the mapping of $\mathcal{X}$ to the term reduction algebra a suitably enriched functor, and to make the unit and multiplication natural transformations. These equations can also be motivated from a term rewriting point of view, and it is instructive to see how the instantiation of the general theory makes sense in a particular setting.

The freely generated term structure, called the prereductions, will be defined in §A.1.1, together with a substitution function, and a notion of lifting functors and natural transformations. In §A.1.2, we will define the necessary equations on the prereductions, obtaining the term named-reduction algebra $T_\Theta(X)$ on a category $\mathcal{X}$, which has terms as objects and named reductions between terms as morphisms. Finally, in §A.1.3, we will show that the mapping $\mathcal{X} \mapsto T_\Theta(X)$ extends to a monad T_Θ on **Cat**. We will close the section by investigating the relationship to the "unnamed" term reduction algebra defined in §2.3.3.

The monad T_Θ is of course enriched, and as has been mentioned in §1.5.3 on page 43, there are two different closed monoidal structures on **Cat**, the familiar cartesian one, and a slightly more exotic one called Sesqui (as in Sesqui-category). It is possible to enrich over either of them, but instead of picking one now, we will first develop that part of the theory which is common to both enrichments, and on page 180, we will discuss the differences within this framework; this allows us a uniform treatment of both enrichments and a discussion of their difference. As it turns out, the differences are fairly small, amounting to just one additional equation being necessary for the cartesian closed structure.

A.1.1 Prereductions

Since the contexts of the rewrite rules are given by the finitely presentable objects of the base category, they will now be given by fp categories (i.e. finitely presentable objects in **Cat**, see §1.3.8 on page 19), and we need to readjust the definition of generalized rewrite rules accordingly. This is merely a rephrasing of definition 2.3.1 on page 69, which it supersedes for the rest of this section:

Definition A.1.1 (Generalized Rewrite Rule) A *generalized rewrite rule* in a signature Ω is given by a triple $(\mathcal{X}, l, r)$, written as $(\mathcal{X} \vdash l \to r)$, where $\mathcal{X}$ is a category, and $l, r \in T_\Omega(|\mathcal{X}|)$. $\mathcal{X}$ is called the context, l the left-hand side, and r the right-hand side of the rule. If the category $\mathcal{X}$ is discrete, the rewrite rule is called *ordinary*; if the category $\mathcal{X}$ is fp, it is called *finitary*. If there is a morphism $f : x \to y$ in $\mathcal{X}$, we say there is a *variable rewrite* from x to y.

When instantiating a finitary rule $(\mathcal{X} \vdash l \to r)$, it is no more sufficient to give the instantiations of the terms, we have to explicitly supply a morphism $\mathit{Inst}(\alpha)$ for all morphisms $\alpha : x \to y$ in $\mathcal{X}$, which means an instantiation of a rule $(\mathcal{X} \vdash l \to r)$ in $\mathcal{Z}$ is a functor $\mathit{Inst} : \mathcal{X} \to \mathcal{Z}$.

The construction of prereductions now follows very much definition 2.3.4 — the prereductions are given by variable rewrites (rule [Var] below), application of operations (rule [Pre]) and instantiated rewrite rules (rule [Inst]). Note that in the latter case, although a finitely presentable category $\mathcal{X}$ may have infinitely many morphisms, a functor $F : \mathcal{X} \to \mathcal{Y}$ is determined by its action on a finite subset of the morphisms in $\mathcal{X}$.

Definition A.1.2 (Prereductions) For a category $\mathcal{X}$ (called the *context*), the *prereduction graph* $\mathcal{Q}_\Theta(\mathcal{X})$ on $\mathcal{X}$ is the graph with

- vertices, the set of terms built over the objects of $\mathcal{X}$: $V(\mathcal{Q}_\Theta(\mathcal{X})) \stackrel{def}{=} T_\Omega(|\mathcal{X}|)$

- edges, the smallest set satisfying the implications in table A.1.

$$[\textsc{Var}]\ \frac{\kappa \in \mathcal{X}(x,y)}{\text{'}\kappa : \text{'}x \to \text{'}y \in E(\mathcal{Q}_\Theta(\mathcal{X}))}$$

$$[\textsc{Pre}]\ \frac{\text{for } i = 1, \ldots, m.\ \beta_i : s_i \to t_i \text{ in } \mathcal{F}(\mathcal{Q}_\Theta(\mathcal{X}))}{\omega(\beta_1, \ldots, \beta_m) : \omega(s_1, \ldots, s_m) \to \omega(t_1, \ldots, t_m) \in E(\mathcal{Q}_\Theta(\mathcal{X}))}\ \omega \in \Omega_m$$

$$[\textsc{Inst}]\ \frac{\mathit{Inst} : \mathcal{Y} \to \mathcal{F}(\mathcal{Q}_\Theta(\mathcal{X})) \text{ is a functor}}{\begin{array}{l}\rho[\mathit{Inst}] : s \to t \in E(\mathcal{Q}_\Theta(\mathcal{X})) \\ \quad \text{where } s \stackrel{def}{=} \mu_{|\mathcal{X}|}(\mathit{Inst}_{\mathit{Obj}}{}^*(l)), t \stackrel{def}{=} \mu_{|\mathcal{X}|}(\mathit{Inst}_{\mathit{Obj}}{}^*(r))\end{array}}\ \rho = (\mathcal{Y} \vdash l \to r) \in R$$

Table A.1: Definition of prereduction graph.

The *prereductions* on $\mathcal{X}$ are given by the free category on the prereduction graph $\mathcal{Q}_\Theta(\mathcal{X})$:

$$P_\Theta(\mathcal{X}) \stackrel{def}{=} \mathcal{F}(\mathcal{Q}_\Theta(\mathcal{X}))$$

For rule [Pre], recall from page 37 the notation $\mathtt{s}(s)$ and $\mathtt{t}(s)$ for the source and target of a path t. Further, for rule [Inst] $\mathit{Inst}_{\mathit{Obj}}{}^*$ is the lifting of the object function $\mathit{Inst}_{\mathit{Obj}}$ of the functor *Inst*, as from definition 2.1.5, and $\mu_{|\mathcal{X}|}$ is the multiplication for terms of the monad T_Ω from proposition 2.1.6. Then $\mu_{|\mathcal{X}|}(\mathit{Inst}_{\mathit{Obj}}{}^*(l))$ is the substitution of the variables in l, which in the notation from page 58 can also be written $l[t_1, \ldots, t_n]$ with $t_i \stackrel{def}{=} \mathit{Inst}(y_i)$.

Under the adjunction between **Grph** and **Cat**, in order to define a functor $F : P_\Theta(\mathcal{X}) \to \mathcal{Y}$ we have to give a graph morphism $f : \mathcal{Q}_\Theta(\mathcal{X}) \to U\mathcal{Y}$, and this means we have to define a vertex function $f_V : T_\Omega(|\mathcal{X}|) \to |\mathcal{Y}|$ (the object function for F), and a map $f_E : \mathcal{Q}_\Theta(\mathcal{X}) \to U\mathbf{Mor}_\mathcal{Y}$, mapping edges in $T_\Theta(\mathcal{X})$ to morphisms in $\mathcal{Y}$. This way we can define the process of lifting functors and transformations, which will later become the action of the monad modelling the term rewriting system on the morphisms and 2-cells of **Cat**, respectively.

Note that we define the lifting for the slightly more general notion of *transformations* rather than natural transformations— this means that we can use the same definition for both different enrichments over **Cat**.[1]

[1] Of course, we will have to show that the lifting of a natural transformation is again natural — or rather, we will need some equations to enforce this since it is not the case with the definition as it stands.

Definition A.1.3 (Lifting of Functors and Transformations) For a functor $F : \mathcal{X} \to \mathcal{Y}$ its *lifting* is given by the functor $F^* : P_\Theta(\mathcal{X}) \to P_\Theta(\mathcal{Y})$, which on objects is given by ${F_{Obj}}^* : T_\Omega(|\mathcal{X}|) \to T_\Omega(|\mathcal{Y}|)$ from definition 2.1.5, and on the paths is defined by the following mapping on the edges:

$$\begin{array}{rcll}
F^*(\omega\,(\alpha_1,\ldots,\alpha_n)) & \stackrel{def}{=} & \mathtt{<}\omega\,(F^*\alpha_1,\ldots,F^*\alpha_n)\mathtt{>} & \text{where } \omega \in \Omega_n \\
F^*(\rho\,\mathtt{[}Inst\mathtt{]}) & \stackrel{def}{=} & \mathtt{<}\rho\,\mathtt{[}F^*\cdot Inst\mathtt{]>} & \text{where } \rho \in R \\
F^*(\text{'}\kappa) & \stackrel{def}{=} & \mathtt{<}\text{'}(F\kappa)\mathtt{>} &
\end{array}$$

Given a transformation $\alpha : F \Rightarrow G : \mathcal{X} \to \mathcal{Y}$, its *lifting* is a transformation $\alpha^* : F^* \Rightarrow G^* : P_\Theta(\mathcal{X}) \to P_\Theta(\mathcal{Y})$, given by a family of prereductions in $P_\Theta(\mathcal{Y})$, indexed by terms $t \in T_\Omega(|\mathcal{X}|)$, defined inductively on t as follows:

$$\begin{array}{rcll}
\alpha^*_{\omega(t_1,\ldots,t_n)} & \stackrel{def}{=} & \mathtt{<}\omega\,(\alpha^*_{F^*(t_1)},\ldots,\alpha^*_{F^*(t_n)})\mathtt{>} & \text{for } t_1,\ldots,t_n \in T_\Omega(|\mathcal{X}|) \\
\alpha^*_{\text{'}x} & \stackrel{def}{=} & \mathtt{<}\text{'}\alpha_x\mathtt{>} & \text{for } x \in |\mathcal{X}|
\end{array}$$

Having defined the lifting, we give the transformation which models the the substitution of the nascent term calculus. We leave aside the variables (i.e. the unit) just now, since we do not need them for the time being.

Definition A.1.4 (Substitution) For a category $\mathcal{X}$, the object function of the functor $\mu_\mathcal{X} : P_\Theta(P_\Theta(\mathcal{X})) \to P_\Theta(\mathcal{X})$ is given by the natural transformation $\mu_{|\mathcal{X}|} : T_\Omega(T_\Omega(|\mathcal{X}|)) \to T_\Omega(|\mathcal{X}|)$ from proposition 2.1.6, and the morphisms function as follows:

$$\begin{array}{rcl}
\mu_\mathcal{X}(e\,(\alpha_1,\ldots,\alpha_m)) & \stackrel{def}{=} & \mathtt{<}e\,(\mu_\mathcal{X}(\alpha_1),\ldots,\mu_\mathcal{X}(\alpha_m))\mathtt{>} \\
\mu_\mathcal{X}(\rho\,\mathtt{[}Inst\mathtt{]}) & \stackrel{def}{=} & \mathtt{<}\rho\,\mathtt{[}\mu_\mathcal{X}\cdot Inst\mathtt{]>} \\
\mu_\mathcal{X}(\text{'}\kappa) & \stackrel{def}{=} & \kappa
\end{array}$$

The process of lifting is unfortunately not functorial. In particular, the lifting id^*_F of the identity transformation id_F on a functor $F : \mathcal{X} \to \mathcal{Y}$ is not the identity transformation id_{F^*} on the lifted functor, as can be easily seen: the latter is given by the empty path $\mathtt{id}_{F^*(t)}$ for any $t \in P_\Theta(\mathcal{X})$, whereas lifted transformations are by definition not empty. (Recall from page 37 that the empty path from t to itself is written as $\mathtt{id}_t$.) Hence, we will need to enforce some equations on the prereductions in order to make it into one. Apart from a category theorist's idle worries, there are actually quite good reasons for these equations from the term rewriting point of view, which we will explain below.

But first, we would like to prove that $\mu_\mathcal{X}$ is natural in $\mathcal{X}$. For this, we introduce our main proof principle — a structural induction scheme like proposition 2.3.7.

Structural Induction

We need some notation for fp categories first. Let $\mathcal{Y}$ be an fp category, then $\mathit{FinMor}_{\mathcal{Y}}$ is the finite set of morphisms generating all other morphisms by composition and identifying. Those morphisms determine the image of any functor $F : \mathcal{Y} \to \mathcal{X}$, since any functor has to preserve composition, and any morphism in $\mathcal{Y}$ can be given as a composition of morphisms from $\mathit{FinMor}_{\mathcal{Y}}$. The reason for introducing this notation is the computation of the size of a prereduction below, which would not go through if we would form a sum over an infinite set.

Proposition A.1.5 *Given a predicate $Z \subseteq P_\Theta(\mathcal{X})$, we write $\alpha \models Z$ (Z holds for α) if $\alpha \in Z$ for $\alpha \in P_\Theta(\mathcal{X})$. Then $\forall \alpha \in P_\Theta(\mathcal{X}).\ \alpha \models Z$ (Z holds for all prereductions), if all implications in table A.2 hold.*

$$[\textsc{IndBase}]\quad \frac{}{\texttt{<}\text{'}\kappa\texttt{>} \models Z}\ \kappa : x \to y \text{ in } \mathcal{X}$$

$$[\textsc{IndInst}]\quad \frac{\forall \alpha \in \mathit{FinMor}_{\mathcal{Y}}.\ \mathit{Inst}(\alpha) \models Z}{\texttt{<}\rho\texttt{[}\mathit{Inst}\texttt{]>} \models Z}\ \begin{array}{l}\rho = (\mathcal{Y} \vdash l \to r) \in R,\\ \mathit{Inst} : \mathcal{Y} \to P_\Theta(\mathcal{X})\end{array}$$

$$[\textsc{IndPre}]\quad \frac{\beta_1 \models Z, \ldots, \beta_n \models Z}{\texttt{<}e(\beta_1, \ldots, \beta_n)\texttt{>} \models Z}\ e \in \Omega_n,\ \beta_i \in P_\Theta(\mathcal{X}) \text{ for } i = 1, \ldots, n$$

$$[\textsc{IndSeq}]\quad \frac{\texttt{<}\alpha_1\texttt{>} \models Z, \ldots, \texttt{<}\alpha_n\texttt{>} \models Z}{\texttt{<}\alpha_1, \ldots, \alpha_n\texttt{>} \models Z}\ \alpha_i \in \mathcal{Q}_\Theta(\mathcal{X}) \text{ for } i = 1, \ldots, n$$

Table A.2: Structural Induction for Prereductions.

Proof. We prove the validity of structural induction by reducing it to natural induction. To this end, we define the size of a prereduction (intuitively, the number of times one of the prereduction forming rules [Pre] and [Inst] is applied) as follows:

$$\begin{aligned} \mathit{size}(\text{'}\kappa) &\stackrel{\mathit{def}}{=} 0 \\ \mathit{size}(e(\alpha_1, \ldots, \alpha_n)) &\stackrel{\mathit{def}}{=} 1 + \sum_{i=1}^{n} \mathit{size}(\alpha_i) \end{aligned} \qquad (A.1)$$

$$size(\texttt{<}\rho\texttt{[}Inst\texttt{]>}) \stackrel{def}{=} 1 + \sum_{\alpha \in FinMor_{\mathcal{Y}}} size(Inst(\alpha)) \tag{A.2}$$

$$size(\texttt{<}\alpha_1, \ldots, \alpha_n\texttt{>}) \stackrel{def}{=} \sum_{i=1}^{n} size(\alpha_i)$$

Given a predicate Z on $P_\Theta(\mathcal{X})$ as above, we define the predicate q on $\mathbb{N}$:

$$q(n) \Leftrightarrow (\forall \alpha \in P_\Theta(\mathcal{X}).\, size(\alpha) \leq n \Rightarrow \alpha \models Z)$$

If under the assumption that the four implications in table A.2 hold we can show that $q(n)$ holds for all $n \in \mathbb{N}$, we can then by definition of q conclude that Z holds for all prereductions α.

We now show that $q(n)$ holds for all $n \in \mathbb{N}$ by well-founded induction on n. Generally, for any path $\texttt{<}\alpha_1, \ldots, \alpha_m\texttt{>} \in P_\Theta(\mathcal{X})$ if we can establish $\texttt{<}\alpha_i\texttt{>} \models Z$ for $i = 1, \ldots, m$, we can use [IndSeq] to conclude $\texttt{<}\alpha_1, \ldots, \alpha_m\texttt{>} \models Z$.

The induction base is $q(0)$. Then for a path $\texttt{<}\alpha_1, \ldots, \alpha_m\texttt{>} \in P_\Theta(\mathcal{X})$ with $size(\texttt{<}\alpha_1, \ldots, \alpha_m\texttt{>}) = 0$, we have that for $j = 1, \ldots, m$, α_j must be a variable, so $\alpha_j = \text{'}\kappa$; and by applying [IndBase], $\texttt{<'}\alpha_j\texttt{>} \models Z$, hence $\texttt{<}\alpha_1, \ldots, \alpha_n\texttt{>} \models Z$. Further, note that by [IndSeq], $\texttt{id}_t \models Z$ for all t, i.e. Z holds for all empty paths.

For the induction step, we have to show $q(n)$, assuming $q(m)$ for all $m < n$; i.e. assume that for all $\beta \in P_\Theta(\mathcal{X})$ with $size(\beta) < n$, $\beta \models Z$, and then show (using the implications) that for all $\texttt{<}\alpha_1, \ldots, \alpha_l\texttt{>} \in P_\Theta(\mathcal{X})$ with $size(\texttt{<}\alpha_1, \ldots, \alpha_l\texttt{>}) = n$, $\texttt{<}\alpha_1, \ldots, \alpha_l\texttt{>} \models Z$. For $i = 1, \ldots, l$, α_i is of size $size(\alpha_i) \leq n$, and we distinguish three cases:

1. $\alpha_i = \text{'}\kappa$, then by [IndBase], $\texttt{<'}\kappa\texttt{>} \models Z$.

2. $\alpha_i = e\texttt{(}\beta_1, \ldots, \beta_k\texttt{)}$, then for all $j = 1, \ldots, k$, by clause A.1 above $size(\beta_j) < n$, and by the induction assumption $\beta_j \models Z$; using [IndPre], we conclude $\texttt{<}e\texttt{(}\alpha_1, \ldots, \alpha_k\texttt{)>} \models Z$.

3. $\alpha_i = \rho\texttt{[}Inst\texttt{]}$, then similarly by clause A.2 we can apply the induction assumption to all $\alpha \in FinMor_{\mathcal{Y}}$, and use [IndInst] to obtain $\texttt{<}\rho\texttt{[}Inst\texttt{]>} \models Z$.

Hence we can conclude that $\texttt{<}\alpha_1, \ldots, \alpha_m\texttt{>} \models Z$ and that $q(n)$ holds. □

As an example of structural induction, we show that the substitution $\mu_{\mathcal{X}} : P_\Theta(P_\Theta(\mathcal{X})) \to P_\Theta(\mathcal{X})$ is natural in $\mathcal{X}$, i.e. for all functors $F : \mathcal{X} \to \mathcal{Y}$, we have

$$F^* \mu_{\mathcal{X}} = \mu_{\mathcal{Y}} F^{**} \tag{A.3}$$

The equation is shown extensionally, i.e. we show that $F^*\mu_{\mathcal{X}}(\alpha) = \mu_{\mathcal{Y}} F^{**}(\alpha)$ for all prereductions α in $P_\Theta(P_\Theta(\mathcal{X}))$ by structural induction. The induction base is $\alpha = \texttt{<'}\kappa\texttt{>}$, then $F^*(\mu_{\mathcal{X}}(\texttt{<'}\kappa\texttt{>})) = F^*\kappa = \mu_{\mathcal{Y}}\texttt{<'}F^*\kappa\texttt{>} = \mu_{\mathcal{Y}} F^{**}\texttt{<'}\kappa\texttt{>}$. The induction steps proceed by unfolding the definitions, applying the induction assumption, and folding the definitions again, e.g.

$$\begin{aligned} F^*\mu_{\mathcal{X}}\texttt{<}e\texttt{(}\beta_1,\ldots,\beta_n\texttt{)>} &= \texttt{<}e\texttt{(}F^*\mu_{\mathcal{X}}\beta_1,\ldots,F^*\mu_{\mathcal{X}}\beta_n\texttt{)>} \\ &= \texttt{<}e\texttt{(}\mu_{\mathcal{Y}}F^{**}\beta_1,\ldots,\mu_{\mathcal{Y}}F^{**}\beta_n\texttt{)>} \\ &= \mu_{\mathcal{Y}}F^{**}\texttt{<}e\texttt{(}\beta_1,\ldots,\beta_n\texttt{)>} \end{aligned}$$

A.1.2 Named Reductions

We are now going to describe the particular set of equations which we are going to enforce on the prereductions. These equations can be motivated twofold: from a categorical point of view, these equations make the whole construction into a monad; in particular, they make the lifting process functorial. This may not be very interesting from a term rewriting point of view, but one might want to bear in mind that in category theory, "everything important happens in the morphisms"[2], so having accepted its basic tenets we should let this principle guide us to a "good" construction. Moreover, we can also motivate these equations from a term rewriting point of view; here, they mean that the composition of rewrites is compatible with putting rewrites into a context as well as with taking morphisms from $\mathcal{X}$ to rewrites in $P_\Theta(\mathcal{X})$, and that identity variable rewrites are identity rewrites. In an informal notation,

- if C is a context, and α_1, α_2 prereductions, then

$$C[\alpha_1] :: C[\alpha_2] = C[\alpha_1 :: \alpha_2]$$

- if κ and λ are composable variable rewrites, then

$$\texttt{<'}\kappa\texttt{>} :: \texttt{<'}\lambda\texttt{>} = \texttt{<'}\lambda\cdot\kappa\texttt{>}$$

- and for $t \in T_\Omega(|\mathcal{X}|)$,

$$\texttt{id}_t = \texttt{<'}\mathbf{1}_t\texttt{>}$$

One could further want that putting a rewrite into context commutes with instantiating rules; this will be the equation that distinguishes Sesqui-enrichment from cartesian enrichment.

[2] Or rather hom-objects, since we are talking enriched categories here.

Different Enrichments

At this juncture we need to have another look at the two possible enrichments over the two monoidal structures on **Cat**. The above equations are enough — as we will presently see — to define a monad on $\mathbf{Cat}_S$, the Sesqui-category of all small categories. But this monad will not be enriched over $\mathbf{Cat}_C$, the 2-category of all small categories, since the lifting of a natural transformation is not necessarily natural.

To explain this, consider the simple system with two unary operations $\mathtt{F}, \mathtt{G}$ and the rule $\rho = (\mathcal{Z} \vdash \mathtt{F('}z\mathtt{)} \to \mathtt{G('}z\mathtt{)})$, where $\mathcal{Z}$ is the category with one object z. Let $\mathcal{X}$ be the category with two objects x, y and a non-identity morphism $f : x \to y$. By mapping z to x and y, respectively, we have two functors $\mathtt{I}, \mathtt{J} : \mathcal{Z} \to \mathcal{X}$, and the morphism f in $\mathcal{X}$ gives a natural transformation $\alpha : \mathtt{I} \Rightarrow \mathtt{J}$. By definition A.1.3, we can lift $\mathtt{I}, \mathtt{J}$ and α to $\alpha^* : \mathtt{I}^* \Rightarrow \mathtt{J}^* : P_\Theta(\mathcal{X}) \to P_\Theta(\mathcal{Y})$. Consider the prereduction $s \overset{def}{=} \mathtt{<}r\mathtt{[}\mathbf{1}_z\mathtt{]>}$ in $P_\Theta(\mathcal{Z})$, then $\mathtt{I}^*(s) = \mathtt{<}\rho\mathtt{[}\mathbf{1}_x\mathtt{]>}$ and $\mathtt{J}^*(s) = \mathtt{<}\rho\mathtt{[}\mathbf{1}_y\mathtt{]>}$. If α^* were natural, then the following diagram in $P_\Theta(\mathcal{X})$ would commute

$$\begin{tikzcd} \mathtt{F('}x\mathtt{)} \arrow[r, "\alpha^*_{\mathtt{F('}z\mathtt{)}}"] \arrow[d, "\mathtt{I}^*(s)"'] & \mathtt{F('}y\mathtt{)} \arrow[d, "\mathtt{J}^*(s)"] \\ \mathtt{G('}x\mathtt{)} \arrow[r, "\alpha^*_{\mathtt{G('}z\mathtt{)}}"] & \mathtt{G('}y\mathtt{)} \end{tikzcd}$$

but this does not follow from the equations above. Another way to look at this is that the rewrite $\mathtt{'}f : \mathtt{'}x \to \mathtt{'}y$ put into the contexts given by the source and target of the rewrite rule ρ should commute with the instantiated rewrite rule.

From a term rewriting point of view, this question is equivalent to asking that for any rule $\rho : l \to r$, and any rewrite $\alpha : s \to t$, we have

$$\mathtt{<}l[\alpha], \rho\mathtt{[}t\mathtt{]>} = \mathtt{<}\rho\mathtt{[}s\mathtt{]}, r[\alpha]\mathtt{>}$$

and it is debatable whether this question should hold or not. One may think of the equation as describing that rewrites can take place in parallel, concurrently [57]. From a categorical point of view, cartesian enrichment seems favourable, since the main advantage of Sesqui-categories is that they have a categorical notion of length [83], which we do not need for our treatment of strong normalization, and all in all cartesian enrichment seems more natural.

The language of lifted functors, transformations and the substitution $\mu_\mathcal{X}$ from definition A.1.4 lets us make rather elusive informal notation such as $l[\alpha]$ precise, as we will be doing now in order to define the necessary equations.

Horizontal Composition

Given a natural transformation $\alpha : F \Rightarrow G : \mathcal{X} \to \mathcal{Y}$, and a functor $H : \mathcal{Y} \to \mathcal{Z}$, the *horizontal composition* of α with H is a natural transformation $H\alpha : HF \Rightarrow HG : \mathcal{X} \to \mathcal{Z}$, defined by $(H\alpha)_x \stackrel{def}{=} H(\alpha_x)$. Similarly, the horizontal composition of α with a functor $M : \mathcal{W} \to \mathcal{X}$ is defined as $\alpha M : FM \Rightarrow GM : \mathcal{W} \to \mathcal{Y}$, $(\alpha M)_x \stackrel{def}{=} \alpha_{Mx}$.[3]

Here, the functors roughly correspond to terms or contexts, and natural transformations correspond to rewrites, so the horizontal composition of α with a functor H on the left corresponds to putting the rewrite into context, whereas composition of α with M on the right corresponds to instantiating the variables in α. To illustrate this principle, we will now prove two equations, which say that substitution commutes with rewriting (equation A.4), and that translation (along a lifted functor) commutes with the substitution of variables (equation A.5). The equations will be needed later on.

For the first equation, given a natural transformation $\alpha^* : F^* \Rightarrow G^* : P_\Theta(\mathcal{X}) \to P_\Theta(\mathcal{Y})$, we can compose α^* with the functor $\mu_\mathcal{X} : P_\Theta(P_\Theta(\mathcal{X})) \to P_\Theta(\mathcal{X})$ on the right, or we can compose the functor $\mu_\mathcal{Y} : P_\Theta(P_\Theta(\mathcal{Y})) \to P_\Theta(\mathcal{Y})$ with the lifting of α^*, $\alpha^{**} : F^{**} \Rightarrow G^{**}$, on the right. Both of these should be equal:

$$\alpha^* \mu_\mathcal{X} = \mu_\mathcal{Y} \alpha^{**} \tag{A.4}$$

This is also called the 2-naturality of μ, and it is proven by showing that for all $t \in P_\Theta(P_\Theta(\mathcal{X}))$, $\alpha^*_{\mu_\mathcal{X}(t)} = \mu_\mathcal{Y} \alpha^{**}_t$. The proof proceeds by structural induction on the term t. The induction base is given by

$$\alpha^*_{\mu_\mathcal{X}(\text{'}s)} = \alpha^*_s = \mu_\mathcal{Y}\texttt{<'}\alpha^*_s\texttt{>} = \mu_\mathcal{Y}(\alpha^{**}_s)$$

and the induction step is similar (if even more straightforward).

For the second equation, given a functor $F : \mathcal{X} \to \mathcal{Y}$, and a natural transformation $\alpha : I \Rightarrow J : \mathcal{Z} \to P_\Theta(\mathcal{X})$, we can lift the horizontal composition $F^*\alpha$, obtaining $(F^*\alpha)^* : F^{**}I^* \Rightarrow F^{**}J^*$, with $(F^*\alpha)^*_t = F^{**}(\alpha^*_t)$. Together with the naturality of μ (equation A.3), we obtain

$$F^*(\mu_\mathcal{X}(\alpha^*_t)) = \mu_\mathcal{Y}(F^{**}(\alpha^*_t)) = \mu_\mathcal{Y}((F^*\alpha)^*_t) \tag{A.5}$$

Equations on prereductions

An equation on prereductions is a pair of prereductions in the same context $\mathcal{X}$ with the same source and target, i.e. a triple $(\mathcal{X}, l, r)$ where $\mathcal{X}$ is a fp category,

[3]This is special case of the tadpole composition introduced on page 31.

and $l, r \in P_\Theta(\mathcal{X})$ s.t. $\mathtt{s}(l) = \mathtt{s}(r)$, $\mathtt{t}(l) = \mathtt{t}(r)$. An equation $(\mathcal{X}, l, r)$ is denoted as $\mathcal{X} \vdash l = r$. Recall from page 37 that a path congruence is an equivalence relation on paths with the same source and target (implication 1.22), compatible with the composition (implication 1.23). A congruence relation on prereductions is a path congruence additionally compatible with the application of operations and instantiation of rules.

Definition A.1.6 (Prereduction Congruence) For a category $\mathcal{X}$, a *prereduction congruence* is given by a path congruence $\equiv \subseteq P_\Theta(\mathcal{X}) \times P_\Theta(\mathcal{X})$ such that for all $\omega \in \Omega_n$ and $\alpha_1, \beta_1, \ldots, \alpha_n, \beta_n \in P_\Theta(\mathcal{X})$,

$$\alpha_1 \equiv \beta_1, \ldots, \alpha_n \equiv \beta_n \Rightarrow \mathtt{<}\omega\mathtt{(}\alpha_1, \ldots, \alpha_n\mathtt{)>} \equiv \mathtt{<}\omega\mathtt{(}\beta_1, \ldots, \beta_n\mathtt{)>} \tag{A.6}$$

and for all $(\mathcal{Z} \vdash l \to r) \in R$, $\mathit{Inst}_1, \mathit{Inst}_2 : \mathcal{Z} \to P_\Theta(\mathcal{X})$

$$(\forall \alpha \text{ in } \mathcal{Z} \,.\, \mathit{Inst}_1(\alpha) \equiv \mathit{Inst}_2(\alpha)) \;\Rightarrow \mathtt{<}\rho\mathtt{[}\mathit{Inst}_1\mathtt{]>} \equiv \mathtt{<}\rho\mathtt{[}\mathit{Inst}_2\mathtt{]>} \tag{A.7}$$

Given a set of equations on prereductions, we can generate a prereduction congruence for categories $\mathcal{X}$ by closing under equivalence (i.e. reflexivity, transitivity and symmetry) and implications 1.23, A.6 and A.7.

Reductions

Putting the last two sections together, we can now make the informal notation of the section before that precise.

Definition A.1.7 (Equations on the Prereductions) For an fp category $\mathcal{X}$, the set of equations $\mathbf{E_0}$ is defined as the smallest set of equations such that:

- For all operations $\omega \in \Omega_n$, and for $i = 1, \ldots, n$, $\alpha_i \in P_\Theta(\mathcal{X}), \beta_i \in P_\Theta(\mathcal{X})$ s.t. $\mathtt{t}(\alpha_i) = \mathtt{s}(\beta_i)$

$$\mathcal{X} \vdash \mathtt{<}e\mathtt{(}\alpha_1, \ldots, \alpha_n\mathtt{)}, e\mathtt{(}\beta_1, \ldots, \beta_n\mathtt{)>} = \mathtt{<}e\mathtt{(}\alpha_1 :: \beta_1, \ldots, \alpha_n :: \beta_n\mathtt{)>} \;\in \mathbf{E_0} \tag{A.8}$$

- For all $\kappa : x \to y, \lambda : y \to z$ in $\mathcal{X}$:

$$\mathcal{X} \vdash \mathtt{<'}\kappa, \mathtt{'}\lambda\mathtt{>} = \mathtt{<'}(\lambda \cdot \kappa)\mathtt{>} \;\in \mathbf{E_0} \tag{A.9}$$

- For all $x \in \mathcal{X}$,

$$\mathcal{X} \vdash \mathtt{<'}\mathbf{1}_x\mathtt{>} = \mathtt{id}_{\mathtt{'}x} \;\in \mathbf{E_0} \tag{A.10}$$

The set $\mathbf{E_1}$ is defined as $\mathbf{E_0}$, augmented by the following additional equations: for all rules $\rho = (\mathcal{Z} \vdash l \to r) \in R$, functors $Inst_1, Inst_2 : \mathcal{Z} \to P_\Theta(\mathcal{X})$ and natural transformations $\alpha : Inst_1 \Rightarrow Inst_2$,

$$\mathcal{X} \vdash \texttt{<}\rho\texttt{[}Inst_1\texttt{]>} :: \mu_\mathcal{X}(\alpha_r^*) = \mu_\mathcal{X}(\alpha_l^*) :: \texttt{<}\rho\texttt{[}Inst_2\texttt{]>} \in \mathbf{E_1} \tag{A.11}$$

Definition A.1.8 (Term Named-Reduction Algebra) Given a term rewriting system $\Theta = (\Omega, R)$. Let $\equiv_0$ be the prereduction congruence generated from the set of equations $\mathbf{E_0}$, and $\equiv_1$ be the prereduction congruence generated from $\mathbf{E_1}$. Then for a category $\mathcal{X}$, the *term named-reduction algebra* $T_\Theta(\mathcal{X})$ on $\mathcal{X}$ is the category with terms $T_\Omega(|\mathcal{X}|)$ as objects and the *reductions* as morphisms:

$$T_\Theta(\mathcal{X}) \stackrel{def}{=} P_\Theta(\mathcal{X})/{\equiv_1}$$

with the obvious sources and targets, the identity on t being given by $\mathbf{1}_t \stackrel{def}{=} [\mathtt{id}_t]$, and the composition by the composition of (equivalence classes of) paths.

This definition is suited for cartesian enrichment; for the Sesqui variant, we need to quotient the reductions by the equivalence $\mathbf{E_0}$. Instead of using equivalence classes we will continue to work with prereductions, using the congruences $\equiv_0$ and $\equiv_1$ as equality. This also means we can still use structural induction as defined above. When defining functions (or predicates) in this manner, we have to ascertain that they are well-defined, i.e. they preserve the congruence. This means in particular they have to respect equations A.8 to A.11. We will stress the cartesian enrichment in the following, since we consider it more important; but we will use $\mathbf{E_0}$ as equality where it is sufficient to do so, making it clear for which results precisely the additional equation in $\mathbf{E_1}$ is used. It turns out that it is only needed to show the naturality of lifted natural transformations.

A.1.3 The Monad T_Θ

We are now going to extend the assignment $\mathcal{X} \mapsto T_\Theta(\mathcal{X})$ to a functor on the category $\mathbf{Cat}_C$ of all small categories, enriched over the cartesian product. We have defined above (definition A.1.3) the lifting of functors and natural transformations on prereductions; we now have to show that these definitions are well-defined w.r.t. to the congruences $\equiv_0$ and $\equiv_1$, respectively; and further, that this lifting is functorial in the sense that it preserves compositions and identities.

We first treat the lifting of functors. Given a functor $F : \mathcal{X} \to \mathcal{Y}$, we have to show that equations A.8 to A.11 are preserved, i.e. if $\mathcal{Z} \vdash s = t \in \mathbf{E_0}$ then $F^*(l) \equiv_0 F^*(r)$. For equation A.8, we use the fact that F^* preserves composition of paths, for equation A.9 the fact that F is a functor and preserves composition in $\mathcal{X}$, and for equation A.10, the fact that F preserves identities (hence $\texttt{<'}\mathbf{1}_{Fx}\texttt{>} = \texttt{<'}F\mathbf{1}_x\texttt{>}$). For equation A.11, we use equation A.5, then

$$\begin{array}{rcl} F^*(\texttt{<}\rho\texttt{[}Inst_1\texttt{]>}::\mu_{\mathcal{X}}(\alpha_r^*)) & = & \texttt{<}\rho\texttt{[}F^*Inst_1\texttt{]>}::\mu_{\mathcal{Y}}((F^*\alpha)_r^*) \\ & \equiv_1 & \mu_{\mathcal{Y}}(F^*\alpha)_l^*::\texttt{<}\rho\texttt{[}F^*Inst_2\texttt{]>} \\ & = & F^*(\mu_{\mathcal{X}}(\alpha_l^*)::\texttt{<}\rho\texttt{[}Inst_2\texttt{]>}) \end{array}$$

This ends the proof that F^* is well-defined. We now turn to natural transformations. Given $\nu : F \Rightarrow G : \mathcal{X} \to \mathcal{Y}$, ν^* is clearly well-defined. To prove that ν^* is natural, we have to show that for all $\alpha : s \to t$ in $P_\Theta(\mathcal{X})$,

$$F^*(\alpha)::\nu_t^* \equiv_1 \nu_s^*::G^*(\alpha)$$

The proof proceeds by structural induction on α. The induction base uses the naturality of ν: let $\kappa : x \to y$ in $\mathcal{X}$, then

$$F^*\texttt{<'}\kappa\texttt{>}::\nu_{\texttt{'}y}^* = \texttt{<'}F\kappa\texttt{>}::\texttt{<'}\nu_y\texttt{>} \equiv_0 \texttt{<'}\nu_y\cdot F\kappa\texttt{>} = \texttt{<'}G\kappa\cdot\nu_x\texttt{>} \equiv_0 \nu_{\texttt{'}x::G^*\texttt{<'}\kappa\texttt{>}}^*$$

The induction steps [IndPre] and [IndSeq] are just a matter of applying the definitions, and the induction assumption. The case [IndInst] is different. We first apply the definition of F^*, followed by equations A.4 and A.11:

$$\begin{array}{rcl} F^*(\texttt{<}\rho\texttt{[}Inst\texttt{]>})::\nu_{\mu_{\mathcal{X}}\,Inst^*(r)}^* & = & \texttt{<}\rho\texttt{[}F^*Inst\texttt{]>}::\mu_{\mathcal{X}}\nu_{Inst^*(r)}^{**} \\ & = & \texttt{<}\rho\texttt{[}F^*Inst\texttt{]>}::\mu_{\mathcal{X}}(\nu_{Inst}^*)_r^* \\ & \equiv_1 & \mu_{\mathcal{Y}}(\nu_{Inst}^*)_l^*::\texttt{<}\rho\texttt{[}G^*Inst\texttt{]>} \\ & = & \nu_{\mu_{\mathcal{Y}}\,Inst^*(l)}::G^*\texttt{<}\rho\texttt{[}Inst\texttt{]>} \end{array}$$

This ends the proof of well-definedness for ν^*. We can now define the action of the monad.

Definition A.1.9 (Action of the Monad) Given a term rewriting system $\Theta = (\Omega, R)$, the functor $T_\Theta : \mathbf{Cat} \to \mathbf{Cat}$ maps a category $\mathcal{X}$ to the term reduction algebra $T_\Theta(\mathcal{X})$ on $\mathcal{X}$, a functor $F : \mathcal{X} \to \mathcal{Y}$ to its lifting $F^* : T_\Theta(\mathcal{X}) \to T_\Theta(\mathcal{Y})$, and a natural transformation $\nu : F \Rightarrow G$ to its lifting $\nu^* : F^* \Rightarrow G^*$.

We have to show that T_Θ is indeed a functor (a 2-functor to be precise), meaning it preserves composition and identities for functors natural transformations.

The former means that for two functors $F : \mathcal{X} \to \mathcal{Y}$ and $G : \mathcal{Y} \to \mathcal{Z}$, $(GF)^* = G^*F^*$, and for any category $\mathcal{X}$, $(\mathbf{1}_{\mathcal{X}})^* = \mathbf{1}_{T_\Theta(\mathcal{X})}$, both of which are proven pointwise (i.e. for any category $\mathcal{X}$) by routine structural induction (on the reductions in $T_\Theta(\mathcal{X})$). The latter means that for any two natural transformations $\nu : F \Rightarrow G : \mathcal{X} \to \mathcal{Y}$ and $\iota : G \Rightarrow H : \mathcal{X} \to \mathcal{Y}$, $(\iota{\cdot}\nu)^* = \iota^*{\cdot}\nu^*$. This is also proven pointwise: for any category $\mathcal{X}$, and any object $t \in T_\Theta\mathcal{X}$ (i.e. term $t \in T_\Omega(|\mathcal{X}|)$), show that $(\iota{\cdot}\nu)^*_t \equiv_0 \iota^*_t{\cdot}\nu^*_t$ by structural induction on t.

The identity on a functor $F : \mathcal{X} \to \mathcal{Y}$ is given the identity transformation $id_F : F \Rightarrow F$ (see page 14). Preservation of this means that the lifting of the identity is the same as the identity on the lifting, i.e. $id_{F^*} = id^*_F$. In the same vein as before, the proof proceeds pointwise, showing for any category $\mathcal{X}$, $\mathtt{id}_{F^*(t)} \equiv_1 id^*_{F(t)}$ for all objects $t \in T_\Omega(|\mathcal{X}|)$ by structural induction on t. We are now in a position to state the main result:

Proposition A.1.10 *Every term rewriting system $\Theta = (\Omega, R)$ gives rise to a monad $\mathsf{T}_\Theta = \langle T_\Theta, \eta, \mu\rangle$ on* **Cat**.

Proof. The action of the monad has been defined in definition A.1.9, and the multiplication in definition A.1.4. Equations A.3 and A.4 show naturality and 2-naturality of μ. It remains to define the unit, and show the monad laws.

The unit is given by a natural transformation $\eta : \mathbf{1}_{\mathbf{Cat}} \Rightarrow T_\Theta$, which is a family of functors

$$\eta_\mathcal{X} : \mathcal{X} \to T_\Theta(\mathcal{X})$$

indexed by categories $\mathcal{X}$. On objects, it is defined by the natural transformation $\eta_{|\mathcal{X}|} : \mathbf{1}_{|\mathcal{X}|} \Rightarrow T_\Omega$ from lemma 2.1.4, and on morphisms as follows:

$$\eta_\mathcal{X}(\kappa) \stackrel{def}{=} \mathtt{<'}\kappa\mathtt{>}$$

Showing that every $\eta_\mathcal{X}$ is a functor, and that η forms a natural transformation is a matter of routine induction.

It remains to prove the monad laws. In particular these are

$$\mu_\mathcal{X}{\cdot}\eta_{T_\Theta(\mathcal{X})} = \mathbf{1}_\mathcal{X} \quad \text{(A.12)}$$

$$\mu_\mathcal{X}{\cdot}\eta^*_\mathcal{X} = \mathbf{1}_\mathcal{X} \quad \text{(A.13)}$$

$$\mu_\mathcal{X}{\cdot}\mu^*_\mathcal{X} = \mu_\mathcal{X}{\cdot}\mu_{T_\Theta(\mathcal{X})} \quad \text{(A.14)}$$

All of these equations are proven pointwise on reductions $\alpha \in T_\Theta(\mathcal{X})$. Equation A.12 follows straightforward from the definition of η and μ. Equations A.13 and A.14 are proven by routine structural induction on α. □

A.1.4 Named *vs.* Unnamed Reductions

In §2.3.3 on page 70, the term reduction algebra was defined (which we will call term unnamed-reduction algebra in the following to distinguish it from the term named-reduction algebra), and a natural question to ask is the precise relation between that and the term named-reduction algebra which we have just defined. The answer is that the preorder given by the term named-reduction algebra over a category $\mathcal{X}$ is equal to the term unnamed-reduction algebra over the preorder given by $\mathcal{X}$. For the following, recall that $J : \mathbf{Cat} \to \mathbf{Pre}$ is the functor taking a category to a preorder by identifying all morphisms between the same objects in the category (see page 42).

Lemma A.1.11 For a term rewriting system $\Theta = (\Omega, R)$, and an fp category $\mathcal{X}$,

$$J(T_\Theta(\mathcal{X})) = T_\Theta(J(\mathcal{X})) \tag{A.15}$$

where on the left we have the term named-reduction algebra, and on the left the term unnamed-reduction algebra built over the preorder $J(\mathcal{X})$.

Proof. Obviously, the objects of $J(T_\Theta(\mathcal{X}))$ and $T_\Theta(J(\mathcal{X}))$ are the same (namely, the terms $T_\Omega(|\mathcal{X}|)$); it remains to show that there is a reduction $s \geq t$ in $T_\Theta(J(\mathcal{X}))$ iff there is a reduction $\alpha : s \to t$ in $T_\Theta(\mathcal{X})$ (for $s, t \in T_\Omega(|\mathcal{X}|)$). We show that $s \geq t$ in $T_\Theta(J(\mathcal{X}))$ iff there is a prereduction $\alpha : s \to t$ by a routine structural induction in two directions, observing that for all rules in table A.1, definition A.1.2, there is a corresponding rule in table 2.3, definition 2.3.4, and *vice versa.* Then iff there is at least one prereduction, there will be at least one reduction. □

However, the term unnamed-reduction algebra considered as a category over a preorder X is not isomorphic to the term named-reduction algebra over X considered as a category, because the former (but not the latter) will identify any two reductions with the same source and target.

A.2 Properties of the Monad T_Θ

We are now going to show that the monad T_Θ is regular in the sense of definition 2.4.3. We first show that T_Θ is strongly finitary. For this, we separately show that T_Θ preserves filtered colimits, and coequalizers, and then use lemma 1.3.7. To prove preservation of filtered colimits, we need the following lemma, which roughly says that a colimit does not contain more objects and morphisms than necessary:

Lemma A.2.1 Given a filtered functor $F : \mathcal{J} \to \mathbf{Cat}$, and a colimiting cone $c : F \Rightarrow \Delta\mathcal{C}$, then all objects and morphisms in $\mathcal{C}$ are in the image of c in the following sense:

$$\forall x \in \mathcal{C}\, \exists j \in \mathcal{J}, y \in Fj.\, x = c_j(y) \tag{A.16}$$

$$\forall \alpha \in \mathbf{Mor}_{\mathcal{C}}\, \exists j \in \mathcal{J}, \beta \in \mathbf{Mor}_{Fj}.\, c_j(\beta) = \alpha \tag{A.17}$$

Proof. Recall from proposition 1.3.1 that every colimiting cone can be constructed as a coequalizer, and that every coequalizer is epi. From section 1.5.3 we know that epis in **Cat** are functors which are surjective on objects (that is (A.16) above), and surjective under closure on morphisms:

$$\forall \alpha \in \mathcal{C}\, \exists j \in \mathcal{J}, \beta_1, \ldots, \beta_n \in \mathbf{Mor}_{Fj} \,.\, c_j(\beta_n) \cdot \ldots \cdot c_j(\beta_1) = \alpha \tag{A.18}$$

For filtered colimits, the hom-sets of $\mathcal{C}$ can be constructed "pointwise" without needing to close under composition since F does not identify objects (cf. [4, 5.2.2.f]), then (A.18) simplifies to (A.17) above. $\square$

Lemma A.2.2 Given a term rewriting system $\Theta = (\Omega, R)$, the monad T_Θ is finitary.

Proof. Let $F : \mathcal{J} \to \mathbf{Cat}$ be a functor with $\mathcal{J}$ filtered. Let $c : F \Rightarrow \Delta\mathcal{X}$ be a colimiting cone for F. We now want to show that the lifting $c^* : T_\Theta F \Rightarrow \Delta T_\Theta(\mathcal{X})$ of c is colimiting for $T_\Theta F$. We do this by establishing the universal property: given any other cone $\nu : T_\Theta F \Rightarrow \Delta\mathcal{Y}$ over $T_\Theta F$, there is a unique functor $!_\nu : \mathcal{X} \to \mathcal{Y}$ such that

$$\begin{tikzcd} T_\Theta F \arrow[r, Rightarrow, "c^*"] \arrow[dr, Rightarrow, "\nu"'] & \Delta T_\Theta(\mathcal{X}) \arrow[d, Rightarrow, "\Delta !_\nu"] \\ & \Delta\mathcal{Y} \end{tikzcd}$$

We are now going to define the functor $!_\nu$. The idea of the proof is as follows: we show that any object in $T_\Theta(\mathcal{X})$ is in the image of c^* (A.19 below), and then its value under $!_\nu$ is the value of this object under ν. This gives a mapping on the objects of $T_\Theta(\mathcal{X})$, which we then extend to a functor.

To define $!_\nu$, we first prove that

$$\forall t \in T_\Omega(|\mathcal{X}|)\, \exists j \in \mathcal{J}, y \in T_\Omega(|Fj|) \,.\, c_j^*(y) = t \tag{A.19}$$

and then define

$$!_\nu(t) \stackrel{def}{=} \nu_j(y) \tag{A.20}$$

and we furthermore have to show that this definition is well-defined (in particular, independent of the choice of j in A.19).

Both the proof of A.19 and of the well-definedness proceed by structural induction on t. For the induction base, let $'x \in T_\Omega(|\mathcal{X}|)$, then $x \in \mathcal{X}$, and by lemma A.2.1, there is $j \in \mathcal{J}, y \in Fj$ such that $x = c_j(y)$, hence $c^*('y) = {'x}$.

For the induction step, let $\omega(t_1, \ldots, t_n) \in T_\Omega(|\mathcal{X}|)$, and assume that for $i = 1, \ldots, n$, we have $k_i \in \mathcal{J}$ and $t'_i \in Fk_i$ s.t. $c^*_{k_i}(t'_i) = t_i$. Since $\mathcal{J}$ is filtered, there is an object $j \in \mathcal{J}$ and morphisms $p_i : k_i \to j$, and with c^* being a cone over $T_\Theta F$, we have

$$c^*_{k_i} = c^*_j (Fp_i)^*$$

and then y is defined as

$$y \stackrel{def}{=} \omega((Fp_1)^*(t'_1), \ldots, (Fp_n)^*(t'_n))$$

and as required

$$\begin{aligned} c^*_j(y) &= c^*_j(\omega((Fp_1)^*(t'_1), \ldots, (Fp_n)^*(t'_n))) \\ &= \omega(c^*_j(Fp_1)^*(t'_1), \ldots, c^*_j(Fp_n)^*(t'_n)) \\ &= \omega(c^*_{k_1}(t'_1), \ldots, c^*_{k_n}(t'_n)) \\ &= \omega(t_1, \ldots, t_n) \end{aligned}$$

Now show that equation A.20 is well-defined: suppose we have another object $l \in |\mathcal{J}|$ with morphisms $q_i : k_i \to l$; then this defines

$$z \stackrel{def}{=} \omega((Fq_1)^*(t'_1), \ldots, (Fq_n)^*(t'_n))$$

and we have to show that $\nu_l(z) = \nu_j(y)$. By filteredness of $\mathcal{J}$, for $i = 1, \ldots, n$ there is an object $m \in \mathcal{J}$ and morphisms $s : l \to m$ and $r : j \to m$ such that $s \cdot q_i = r \cdot p_i$; and since ν is a cone over $T_\Theta F$, diagram A.21 commutes and we have

$$\begin{tikzcd}
 & T_\Theta Fl \arrow[dr, "(Fs)^*"] \arrow[drr, "\nu_l"] & & \\
T_\Theta Fk_i \arrow[ur, "(Fq_i)^*"] \arrow[dr, "(Fp_i)^*"'] & & T_\Theta Fm \arrow[r, "\nu_m"] & |\mathcal{X}| \\
 & T_\Theta Fj \arrow[ur, "(Fr)^*"] \arrow[urr, "\nu_j"'] & &
\end{tikzcd} \tag{A.21}$$

the following:

$$
\begin{aligned}
\nu_j(y) &\stackrel{def}{=} \nu_j(\omega((Fp_1)^*(t'_1), \ldots, (Fp_n)^*(t'_n))) \\
&= \nu_m(\omega((Fr)^*(Fp_1)^*(t'_1), \ldots, (Fr)^*(Fp_n)^*(t'_n))) \\
&= \nu_m(\omega((Fs)^*(Fq_1)^*(t'_1), \ldots, (Fs)^*(Fq_n)^*(t'_n))) \\
&= \nu_l(\omega((Fq_1)^*(t'_1), \ldots, (Fq_n)^*(t'_n))) \\
&= \nu_l(z)
\end{aligned}
$$

This concludes the structural induction and the construction of the object function of $!_\nu$. We now have to construct $!_\nu$ on the morphisms. We have to show that

$$\forall \alpha \in T_\Theta(\mathcal{X})\ \exists j \in \mathcal{J} \beta \in T_\Theta(Fj)\,.\, c_j^*(\beta) = \alpha \tag{A.22}$$

and then define

$$!_\nu(\alpha) \stackrel{def}{=} \nu_j(\beta) \tag{A.23}$$

Again, this is proven by structural induction, this time on the reduction α.

The base case is $\mathtt{<'}\kappa\mathtt{>} \in T_\Theta(\mathcal{X})$, then $\kappa \in \mathbf{Mor}_\mathcal{X}$ and by lemma A.2.1 there is $j \in \mathcal{J}$, $\lambda \in \mathbf{Mor}_{Fj}$ such that $c_j(\lambda) = \kappa$, hence $c^*(\mathtt{<'}\lambda\mathtt{>}) = \mathtt{<'}\kappa\mathtt{>}$. The case of $\mathtt{<}\omega(\alpha_1, \ldots, \alpha_n)\mathtt{>}$ is very similar to the induction step of the object case above (use induction assumption for the α_i, and filteredness of $\mathcal{J}$).

For the case of $\mathtt{<}\rho\mathtt{[}Inst\mathtt{]>}$, with $\rho = (\mathcal{Z} \vdash l \to r)$, the induction assumption is that for all $\alpha : z \to z'$ in $\mathcal{Z}$ there is $l_\alpha \in \mathcal{J}, \beta_\alpha : s \to t$ in $T_\Theta(Fj)$ s.t. $c_{l_\alpha}^*(\beta_\alpha) = Inst(\alpha)$. Again, by filteredness there is $j \in \mathcal{J}$ and morphisms $p_\alpha : l_\alpha \to j$ s.t. $c_{l_\alpha}^* = c_j^*(Fp_\alpha)^*$. We define the instantiation

$$Inst'(\alpha) = Fp_\alpha(\beta_\alpha)$$

then $c_j^*(Inst'(\alpha)) = c_j^*(Fp_\alpha(\beta_\alpha)) = c_{l_\alpha}^*(\beta_\alpha) = Inst(\alpha)$ and have $y \stackrel{def}{=} \mathtt{<}\rho\mathtt{[}Inst'\mathtt{]>}$, with

$$
\begin{aligned}
c_j^*\mathtt{<}\rho\mathtt{[}Inst'\mathtt{]>} &= \mathtt{<}\rho\mathtt{[}c_j^* Inst\mathtt{]>} \\
&= \mathtt{<}\rho\mathtt{[}Inst\mathtt{]>}
\end{aligned}
$$

Uniqueness is again proven like before: if there is any other object $l \in \mathcal{J}$ satisfying A.22, then there is another object $m \in |\mathcal{J}|$ and morphisms $r : j \to m$, $s : l \to m$, and both ν_l and ν_j filter through ν_m, hence $!_\nu(x)$ is well-defined. This ends the structural induction on the reductions, and the proof.

□

Lemma A.2.3 Given a term rewriting system Θ, the functor T_Θ preserves coequalizers.

Proof. Given two functors $F, G : \mathcal{X} \to \mathcal{Y}$. Let $Q : \mathcal{Y} \to \mathcal{Z}$ be the coequalizer of F and G, and $P : T_\Theta(\mathcal{X}) \to \mathcal{C}$ be the coequalizer of F^* and G^*. To show the lemma, we show that $\mathcal{C}$ and $T_\Theta(\mathcal{Z})$ are isomorphic.

Since all coequalizers are epi (see lemma A.2.1), all objects in $\mathcal{C}$ lie in image of P (which in the following will just be denoted by the representative $[x] = P(x)$); and for the same reason, all objects in $\mathcal{Z}$ lie in the image of Q, hence all objects in $T_\Theta(\mathcal{Z})$ lie in the image of Q^*. We show the isomorphism by showing that, for all objects $s, t \in T_\Omega(|\mathcal{Y}|)$

$$Q^*(s) = Q^*(t) \Leftrightarrow [s] = [t] \tag{A.24}$$

and for all reductions α, β in $T_\Theta(\mathcal{Y})$

$$Q^*(\alpha) = Q^*(\beta) \Leftrightarrow [\alpha] = [\beta] \tag{A.25}$$

In other words, we prove that the equivalence generated by F^* and G^* is a congruence on the terms; and this follows from the fact that F^* and G^* by definition respect the structure of the terms.

We will first prove equation A.24. To this end, some notation: let $\sim$ be the relation generating the equivalence $\equiv$ on $T_\Omega(|\mathcal{Y}|)$, defined as

$$x \sim y \Leftrightarrow \exists z \in T_\Omega(|\mathcal{X}|). f^*z = x, g^*z = y$$

then $\equiv$ is the equivalence closure of $\sim$. The proof proceeds by structural induction on both s (the outer induction) and t (the inner induction).q We distinguish four case (corresponding to the inner and outer induction bases and steps, respectively):

1. $s = {}'x, t = {}'y$

   For all $x \in \mathcal{X}$, $Q^*({}'x) = {}'Qx = {}'[x] = [{}'x]$; hence $Q^*({}'x) = Q^*({}'y)$ iff $[x] = [y]$ iff $[{}'x] = [{}'y]$.

2. $s = {}'x, t = \omega(t_1, \ldots, t_m)$

   Since $Q^*({}'x) \neq Q^*(\omega(t_1, \ldots, t_m))$, we show that $[{}'x] \neq [\omega(t_1, \ldots, t_m)]$. In this case $s \not\sim t$, since there is no $z \in T_\Omega(|\mathcal{X}|)$ such that $F^*z = {}'x$ and $G^*z = \omega(t_1, \ldots, t_m)$ (because $F^*z = {}'x$ only if $z = {}'u$, and then $G^*z \neq \omega(t_1, \ldots, t_m)$, and vice versa). This is preserved by the equivalence closure: reflexive and symmetric closure are trivial, and for the transitive closure, $G^*z_1 = F^*z_2$ only if both $z_1 = {}'u_1$ and $z_2 = {}'u_2$, or $z_1 = e(u_1, \ldots, u_n)$ and $z_2 = e(v_1, \ldots, v_n)$ with $F^*u_i = G^*v_i$.

3. $s = e(s_1, \ldots, s_n), t = {}'y$

   This case is symmetric to the one before.

4. $s = e(s_1, \ldots, s_n), t = \omega(t_1, \ldots, t_m)$

   $Q^*(e(s_1, \ldots, s_n)) = Q^*(\omega(t_1, \ldots, t_m))$ iff $e = \omega$ and $Q^* s_i = Q^* t_i$ for $i = 1, \ldots, n$. The induction assumption here is that $Q^*(s_i) = Q^*(t_i)$ iff $[t_i] = [s_i]$; and it remains to show that $[e(s_1, \ldots, s_n)] = [\omega(t_1, \ldots, t_m)]$ iff $e = \omega$ and $[s_i] = [t_i]$ for all $i = 1, \ldots, n$. The direction "only if" is easy to see; to show the other direction, note that $F^* z = e(s_1, \ldots, s_n)$ implies $z = e(z_1, \ldots, z_n)$ (similarly for G^*); with this, we can show that $e(s_1, \ldots, s_n) \sim \omega(t_1, \ldots, t_m)$ if $e = \omega$ and $s_i \sim t_i$ for $i = 1, \ldots, m$. This is preserved under the equivalence closure, hence $[e(s_1, \ldots, s_n)] = [\omega(t_1, \ldots, t_m)]$ iff $e = \omega$ and $[s_i] = [t_i]$ for all $i = 1, \ldots, n$ as desired.

This ends the proof of equivalence A.24.

The proof is completed by showing the equivalence between the morphisms of $\mathcal{C}$ and $T_\Theta(\mathcal{Z})$ (equation A.25). This proof proceeds by a similar structural induction on the two reductions α, β in equation A.25. The basic proof scheme is the same, but for reductions we have four implications in the structural induction, as opposed to two for terms, leading to sixteen cases. Reductions generated from different rules can never be equivalent (cases 3 and 4 above), nor can lifted functors identify them; rules generated with the same rule are identified if they are generated with the same parameter (the operations e and ω above, here either two operations, or two rules), which is exactly the case in which the lifted functors can identify objects. □

We can now state the main result of this section:

Proposition A.2.4 *Given a term rewriting system Θ, the monad T_Θ on* **Cat** *is regular.*

Proof. By lemma A.2.2, T_Θ is finitary, by lemma A.2.3 preserves coequalizers, hence by lemma 1.3.7, T_Θ is strongly finitary.

That η is monic follows straightforward from the definition, as well as η being regular. That μ is regular is proven similar to proposition 2.4.4; we have to extend the partial function σ from the objects to the morphisms, by defining the morphism function of $\sigma : T_\Theta(T_\Theta(Y)) \times T_\Theta(X) \rightharpoonup T_\Theta(T_\Theta(X))$ as follows:

$$\begin{aligned}
\sigma(e(\alpha_1, \ldots, \alpha_n), e(\beta_1, \ldots, \beta_n)) &\stackrel{def}{=} e(\sigma(\alpha_1, \beta_1), \ldots, \sigma(\alpha_n, \beta_n]) \\
\sigma(\rho[\mathit{Inst}], \rho[\mathit{Inst}']) &\stackrel{def}{=} \rho[\sigma\langle \mathit{Inst}, \mathit{Inst}'\rangle] \\
\sigma(\text{'}\kappa, \lambda) &\stackrel{def}{=} \texttt{<}\text{'}\lambda\texttt{>}
\end{aligned}$$

One now shows that this is well-defined, i.e. respects the equations A.8 to A.11 by a structural induction on the first argument. □

A.3 Compositionality

This section extends §2.5 to the named reduction case. There it was shown that there is an adjunction

$$\mathbf{TRS} \underset{\longleftarrow}{\overset{\longrightarrow}{\perp}} \mathbf{Mon}_{Fin}(\mathbf{Pre})$$

between the category of term rewriting systems, and the category of strongly finitary monads on **Pre**, and it was argued that this adjunction justified our calling the semantics compositional. Here, we are going to show that there is an adjunction

$$\mathbf{TRS} \underset{\longleftarrow}{\overset{\longrightarrow}{\perp}} \mathbf{Mon}_{Fin}(\mathbf{Cat})$$

between the category of term rewriting systems, and the category of strongly finitary monads on **Cat**, making the named reduction semantics compositional as well.

We first define the internal language of a finitary monad on **Cat**. This internal language is a more elaborate version of the internal language for a monad on **Pre** (see §2.3.4 on page 74), where we can now tell the structure of a reduction — one can think of this as the reason, or proof, why one term rewrites to another.

For the following, recall the adjunction from lemma 2.1.9, given by the monad generated from a signature, and the internal signature of a monad; in particular its counit, consisting of natural transformations $\varepsilon_T : T_{\Sigma(\mathsf{T})} \Rightarrow \mathsf{T}$ for every monad T on **Set**, which allows us to evaluate terms built in the internal signature in T.

Definition A.3.1 (The Internal Language) The *internal language* of a finitary monad $\mathsf{T} = \langle T, \eta, \mu \rangle$ on **Cat** is given by

$$\mathcal{L}(\mathsf{T}) \stackrel{def}{=} (\Sigma(\mathsf{T}_0), \mathcal{R}(\mathsf{T}))$$

where $\Sigma(\mathsf{T}_0)$ is the internal signature (definition 2.1.7) of the underlying object monad T_0 on **Set** (see §2.3.4 on page 74), and $\mathcal{R}(\mathsf{T})$ is the set of rewrite rules admitted by T defined as follows:

$$\mathcal{R}(\mathsf{T}) \stackrel{def}{=} \{r(\alpha) : (\mathcal{X} \vdash l \to r) \mid \begin{array}{l} \exists \mathcal{X} \in \mathbf{Cat}_{\mathrm{fp}}, \\ \alpha \in T\mathcal{X}(\varepsilon_{T_0,|\mathcal{X}|}(l), \varepsilon_{T_0,|\mathcal{X}|}(r))\} \end{array}$$

where $r(\alpha)$ can be thought of as the rewrite rule given by the the morphism α.

We are now going to extend the mapping of a term rewriting system Θ to the monad T_Θ from proposition A.1.10, and on the other hand a monad to its internal language, to a pair of functors between the categories **TRS** and $\mathbf{Mon}_{Fin}(\mathbf{Cat})$.

Definition A.3.2 The functor $F : \mathbf{TRS} \to \mathbf{Mon}_{Fin}(\mathbf{Cat})$ maps a term rewriting system Θ to the monad T_Θ, and a TRS morphism $\sigma : \Theta \to \Theta'$ to its *lifting*, the monad morphism $\widehat{\sigma} : \mathsf{T}_\Theta \Rightarrow \mathsf{T}_{\Theta'}$, defined pointwise for every category $\mathcal{X}$ as a functor $\widehat{\sigma}_\mathcal{X} : T_\Theta(\mathcal{X}) \to T_{\Theta'}(\mathcal{X})$. The object function is given by the lifting of the signature morphism σ_S in definition 2.1.8, and the morphism function as follows:

$$\begin{aligned}
\widehat{\sigma}_\mathcal{X}(e(\alpha_1, \ldots, \alpha_n)) &\stackrel{def}{=} \mathtt{<}(\sigma_S e)(\widehat{\sigma}_\mathcal{X}(\alpha_1), \ldots, \widehat{\sigma}_\mathcal{X}(\alpha_n))\mathtt{>} \\
\widehat{\sigma}_\mathcal{X}(\rho[\mathit{Inst}]) &\stackrel{def}{=} \mathtt{<}(\sigma_R)[\widehat{\sigma}_\mathcal{X}\cdot \mathit{Inst}]\mathtt{>} \\
\widehat{\sigma_\mathcal{X}}(\text{'}\kappa) &\stackrel{def}{=} \mathtt{<}\text{'}\kappa\mathtt{>}
\end{aligned}$$

The functor $U : \mathbf{Mon}_{Fin}(\mathbf{Cat}) \to \mathbf{TRS}$ maps a finitary monad T to its internal language $\mathcal{L}(\mathsf{T})$, and a monad morphism $\sigma : \mathsf{T} \Rightarrow \mathsf{S}$ to the TRS morphism $U\sigma : \mathcal{L}(\mathsf{T}) \to \mathcal{L}(\mathsf{S})$ which on the signatures is as in definition 2.1.7, and on rules

$$U\sigma_R(r(\alpha) : (\mathcal{X} \vdash l \to r)) \stackrel{def}{=} r(\sigma_\mathcal{X}(\alpha)) : (\mathcal{X} \vdash \widehat{(U\sigma)}_S(l) \to \widehat{(U\sigma)}_S(r))$$

That the lifting $\widehat{\sigma}$ is natural in $\mathcal{X}$ is shown by structural induction; that it satisfies equation 1.3 is fairly obvious, that it satisfies 1.4 requires another easy induction. Hence $\widehat{\sigma}$ is a monad morphism. Two more structural inductions show that the lifting is functorial, i.e. $\widehat{\tau\cdot\sigma} = \widehat{\tau}\cdot\widehat{\sigma}$, and $\widehat{\mathbf{1}_\Theta} = id_{T_\Theta}$.

To show that $U\sigma$ as above is a morphism of term rewriting systems, we have to show that $U\sigma_R(r(\alpha))$ is a rewrite rule in the internal language of S, i.e. $U\sigma_R(r(\alpha)) \in \mathcal{R}(\mathsf{S})$. Since ε_{T_0} is natural in T_0 (equation 2.8), we have

$$\sigma_{|\mathcal{X}|}\cdot\varepsilon_{T_0,|\mathcal{X}|} = \varepsilon_{S_0,|\mathcal{X}|}\cdot\widehat{\sigma}_S$$

and hence the following chain of implications:

$$\begin{array}{cc} & \alpha \in T\mathcal{X}(\varepsilon_{T_0,|\mathcal{X}|}(l), \varepsilon_{T_0,|\mathcal{X}|}(r)) \\ \Rightarrow & \sigma_{\mathcal{X}}\alpha \in S\mathcal{X}(\sigma_{|\mathcal{X}|}\varepsilon_{T_0,|\mathcal{X}|}(l), \sigma_{|\mathcal{X}|}\varepsilon_{T_0,|\mathcal{X}|}(r)) \\ \Rightarrow & \sigma_{\mathcal{X}}\alpha \in S\mathcal{X}(\varepsilon_{S_0,|\mathcal{X}|}\cdot\widehat{\sigma}_S(l), \varepsilon_{S_0,|\mathcal{X}|}\cdot\widehat{\sigma}_S(r)) \\ \Rightarrow & r(\sigma_{\mathcal{X}}(\alpha)) \in \mathcal{R}(\mathsf{S}) \end{array}$$

We can now show the main result of this section, that the two functors F and U are adjoint.

Proposition A.3.3 *The two functors F and U form an adjunction $F \dashv U$: $\mathbf{TRS} \to \mathbf{Mon}_{Fin}(\mathbf{Cat})$.*

Proof. Given a term rewriting system $\Theta = (\Omega, R)$, the unit of the adjunction is a TRS morphism $\upsilon_\Theta : \Theta \to \mathcal{L}(\mathsf{T}_\Theta)$ defined as follows:

- On the signatures, it is the unit of the adjunction from lemma 2.1.9:

$$\upsilon_{\Theta,S} \stackrel{def}{=} \upsilon_\Omega$$

- On the rules, it is defined as

$$\begin{array}{rl} \upsilon_{\Theta,R}(\rho) \stackrel{def}{=} & r(\texttt{<}\rho\texttt{[}\mathbf{1}_{\mathcal{X}}\texttt{]>}) : (\mathcal{X} \vdash \widehat{\upsilon_\Omega}(l) \to \widehat{\upsilon_\Omega}(r)) \\ & \text{where } \ \rho = (\mathcal{X} \vdash l \to r) \end{array}$$

This is a TRS morphism by the triangle laws of the adjunction from lemma 2.1.9: these imply that $\varepsilon_{T_\Omega}\cdot\widehat{\upsilon_\Omega}(l) = l$ and $\varepsilon_{T_\Omega}\cdot\widehat{\upsilon_\Omega}(r) = r$, hence $r(\texttt{<}\rho\texttt{[}\mathbf{1}_{\mathcal{X}}\texttt{]>}) \in \mathcal{R}(\mathsf{T}_\Theta)$.

To show adjointness, we show the universality of υ_Θ from Θ to U: given a monad $\mathsf{S} = \langle S, \zeta, \xi \rangle$ and a TRS morphism $\nu : \Theta \to \mathcal{L}(\mathsf{S})$, there is a unique monad morphism $!_\nu : \mathsf{T}_\Theta \Rightarrow \mathsf{S}$ such that $U!_\nu\cdot\upsilon_\Theta = \nu$. This monad morphism is a natural transformation given by a family of functors $!_{\nu,\mathcal{X}} : T_\Theta\mathcal{X} \to S\mathcal{X}$ for every category $\mathcal{X}$, which on the objects are given by $!_{S_0,|\mathcal{X}|} : T_\Omega(|\mathcal{X}|) \to S_0(|\mathcal{X}|)$ from lemma 2.1.9, and on the morphisms is defined as follows:

$$\begin{array}{rcl} !_{\nu,\mathcal{X}}(e\texttt{(}\alpha_1, \ldots, \alpha_1\texttt{)}) & \stackrel{def}{=} & \nu(e)[!_{\nu,\mathcal{X}}(\alpha_1), \ldots, !_{\nu,\mathcal{X}}(\alpha_n)] \\ !_{\nu,\mathcal{X}}(\rho\texttt{[}Inst\texttt{]}) & \stackrel{def}{=} & \xi_{\mathcal{X}}(S(!_{\nu,\mathcal{X}}\cdot Inst)(\phi)) \\ & & \text{where } \ r(\phi) = \nu(\rho) \\ !_{\nu,\mathcal{X}}(\texttt{'}\kappa) & \stackrel{def}{=} & \zeta_{\mathcal{X}}(\kappa) \end{array}$$

which is readily shown to be natural in $\mathcal{X}$, a monad morphism and the only one satisfying the equation above. □

We finish this section by again drawing attention to the fact that the adjunction above is ordinary, not enriched (see the footnote on page 89).

A.4 Summary and Conclusion

In this appendix, we have shown how to instantiate the general theory of enriched monads in a slightly different and more elaborate way, obtaining a semantics for named reductions by monads over the category **Cat** of all small categories. The construction of the semantics is mainly an extension of chapter 2 to a suitable term structure on the morphisms. The semantics is compositional as well, i.e. there is an adjunction between the category of term rewriting systems and the category of finitary monads over **Cat**.

This semantics goes beyond normal term rewriting in that we can distinguish different reductions between two terms. One could use this setting to e.g. reason about reduction strategies. It is also useful when reasoning about strong normalization, where one needs to be able to tell the identity on a object apart from other endomorphisms (since the latter always lead to non-termination).

Bibliography

[1] J. Adamek and J. Rosický. *Locally Presentable and Accessible Categories.* Number 189 in London Mathematical Society Lecture Note Series. Cambridge University Press, 1994.

[2] M. Barr and C. Wells. *Toposes, Triples and Theories.* Number 278 in Grundlehren der mathematischen Wissenschaften. Springer Verlag, 1985.

[3] J. A. Bergstra and J. W. Klop. Conditional rewrite rules: Confluence and termination. *Journal of Computer and System Sciences*, 32:323– 362, 1986.

[4] F. Borceux. *Handbook of Categorical Algebra 2: Categories and Structures.* Number 51 in Encyclopedia of Mathematics and its Applications. Cambridge University Press, 1994.

[5] A. Burroni. Algèbres graphiques. *Cahiers de Topologie et Géométrie Différentielle*, 23:249– 265, 1981.

[6] R. M. Burstall and J. A. Goguen. The semantics of CLEAR, a specification language. In *Proc. Advanced Course in Abstract Software Specification*, number 86 in Lecture Notes in Computer Science, pages 292– 332, Copenhagen, 1980. Springer Verlag.

[7] A. Carboni and P. T. Johnstone. Connected limits, familial representability and Artin glueing. *Mathematical Structures in Computer Science*, 5:441– 449, 1995.

[8] P. M. Cohn. *Universal Algbra.* Harper and Row, 1965.

[9] A. Corradini, F. Gadducci, and U. Montanari. Relating two categorical models of term rewriting. In J. Hsiang, editor, *Rewriting Techniques and Applications,* 6^{th} *International Conference*, number 914 in Lecture Notes in Computer Science, pages 225– 240, Kaiserslautern, Apr. 1995. Springer Verlag.

[10] N. Dershowitz and J. P. Jouannaud. Rewrite systems. In S. Abramsky, D. M. Gabbay, and T. S. E. Maibaum, editors, *Handbook of Theoretical Computer Science*, volume B (Formal Methods and Semantics), chapter 6, pages 243–320. The MIT Press, 1990.

[11] E. J. Dubuc and G. M. Kelly. A presentation of topoi as algebraic relative to categories or graphs. *Journal for Algebra*, 81:420–433, 1983.

[12] H. Ehrig and B. Mahr. *Fundamentals of Algebraic Specification 1: Equations and Initial Semantics*, volume 6 of *EATCS Monographs on Theoretical Computer Science*. Springer Verlag, 1985.

[13] H. Ehrig and B. Mahr. *Fundamentals of Algebraic Specification 2: Module Specifications and Constraints*, volume 21 of *EATCS Monographs on Theoretical Computer Science*. Springer Verlag, 1990.

[14] P. J. Freyd and G. M. Kelly. Categories of continuous functors I. *Journal for Pure and Applied Algebra*, 2:169–191, 1972.

[15] P. Gabriel and F. Ulmer. *Lokal präsentierbare Kategorien*. Number 221 in Lecture Notes in Mathematics. Springer Verlag, 1971.

[16] H. Ganzinger and R. Giegerich. A note on termination in combinations of heterogeneous term rewriting systems. *Bulletin of the EATCS*, 31, Feb. 1987.

[17] N. Ghani. *Adjoint Rewriting*. PhD thesis, University of Edinburgh, 1995.

[18] N. Ghani. $\beta\eta$-equality for coproducts. In *Second Conference on Typed Lambda Calculus and its Applications*, Edinburgh, Apr. 1995.

[19] N. Ghani. Eta-expansions in system F. Technical Report LIENS-96-10, LIENS-DMI, École Normale Supérieure, 1996.

[20] N. Ghani. Eta-expansions in dependent type theory— the calculus of constructions. In *Typed Lambda-Calculus and Applications*, number 1210 in Lecture Notes in Computer Science, pages 164– 180. Springer Verlag, 1997.

[21] J. A. Goguen. A categorical manifesto. Technical Report PRG-72, Oxford University Computing Laboratory, Programming Research Group, Oxford, England, Mar. 1989.

[22] J. A. Goguen and R. Burstall. Institutions: Abstract model theory for specification and programming. *Journal of the ACM*, 39:95– 146, 1992.

[23] B. Gramlich. Generalized sufficient conditions for modular termination of rewriting. In *Proceedings of the Third International Conference on Algebraic and Logic Programming*, number 632 in Lecture Notes in Computer Science, pages 53–68. Springer Verlag, 1992.

[24] J. W. Gray. *Formal Category Theory: Adjointness for 2-Categories.* Number 391 in Lecture Notes in Mathematics. Springer Verlag, 1974.

[25] J. W. Gray. Categorical aspects of data type constructors. *Theoretical Computer Science*, 50:103–135, 1987.

[26] J. W. Gray. The category of sketches as a model for algebraic semantics. In J. W. Gray and A. Scedrov, editors, *Categories in Computer Science and Logic*, number 92 in Contemporary Mathematics, pages 109–135. American Mathematical Society, 1989.

[27] J. W. Gray. The integration of logical and algebraic types. In H. Ehrig, editor, *Categorical Methods in Computer Science (with Aspects from Topology)*, number 393 in Lecture Notes in Computer Science, pages 16–35, Berlin, 1989. Springer Verlag.

[28] J. W. Gray. Order-enriched sketches for typed lambda calculi. In Carboni, Pedicchio, and Rosolini, editors, *Category Theory*, number 1488 in Lecture Notes in Mathematics, pages 105–130, Como, 1990. Springer Verlag.

[29] B. Hilken. Towards a proof theory of rewriting: the simply typed 2λ-calculus. *Theoretical Computer Science*, 170:407– 444, 1996.

[30] C. A. R. Hoare and J. He. Data refinement in a categorical setting, Feb. 1988.

[31] C. B. Jay. Modelling reductions in confluent categories. In *Proceedings of the Durham Symposium on Applications of Categories in Computer Science*, 1990.

[32] C. B. Jay and N. Ghani. The virtues of η-expansion. *Journal for Functional Programming*, 5(2):135– 154, Apr. 1995.

[33] M. Johnson. Linear term rewriting systems are higher dimensional string rewriting systems. In C. M. I. Rattray and R. G. Clark, editors, *The Unified Computation Laboratory*, pages 103– 112. Oxford University Press, 1991.

[34] S. Kahrs. *λ-rewriting.* PhD thesis, Universität Bremen, Jan. 1991.

[35] G. M. Kelly. A unified treatment of transfinite constructions for free algebras, free monoids, colimits, associated sheaves and so on. *Bulletins of the Australian Mathematical Society*, 22:1– 83, 1980.

[36] G. M. Kelly. *Basic Concepts of Enriched Category Theory*, volume 64 of *London Mathematical Society Lecture Note Series*. Cambridge University Press, 1982.

[37] G. M. Kelly. Structures defined by finite limits in the enriched context I. *Cahiers de Topologie et Géométrie Différentielle*, XXIII(1):3–40, 1982.

[38] G. M. Kelly. Elementary observations on 2-categorical limits. *Bulletins of the Australian Mathematical Society*, 39:301–317, 1989.

[39] G. M. Kelly and A. J. Power. Adjunctions whose counits are coequalizers, and presentations of finitary monads. *Journal for Pure and Applied Algebra*, 89:163– 179, 1993.

[40] G. M. Kelly and R. Street. Review of the elements of 2-categories. In *Category Seminar Sydney 1972/73*, number 420 in Lecture Notes in Mathematics, pages 75–103. Springer Verlag, 1974.

[41] J. W. Klop. Term rewriting systems. In S. Abramsky, D. M. Gabbay, and T. S. E. Maibaum, editors, *Handbook of Logic in Computer Science*, volume 2 (Background: Computational Structures), pages 1–116. Oxford University Press, 1992.

[42] J. W. Klop, A. Middeldorp, Y. Toyama, and R. de Vrijer. A simplified proof of Toyama's theorem. *Information Processing Letters*, 49:101–109, 1994.

[43] D. Knuth and P. Bendix. Simple word problems in universal algebra. In J. Leech, editor, *Computational Problems in Universal Algebras*, pages 263–297. Pergamon Press, 1970.

[44] M. Kurihara and A. Ohuchi. Modularity of simple termination of term rewriting systems with shared constructors. *Theoretical Computer Science*, 103:273– 282, 1992.

[45] J. Lambek and P. J. Scott. *Introduction to Higher Order Categorical Logic*, volume 7 of *Cambridge studies in advanced mathematics*. Cambridge University Press, 1986.

[46] F. W. Lawvere. Metric spaces, generalized logic, and closed categories. In *Rend. del Sem. Mat. e Fis. di Milano*, volume 43, pages 135–166, 1973.

[47] F. W. Levi. On semigroups. *Bulletin of the Calcutta Mathematical Society*, 36:142–146, 1944.

[48] F. E. J. Linton. Some aspects of equational categories. In S. Eilenberg, D. K. Harrison, S. MacLane, and H. Röhrl, editors, *Proceedings of the Conference on Categorical Algebra*, pages 84–94, La Jolla, 1965. Springer Verlag.

[49] F. E. J. Linton. Coequalizers in categories of algebras. In B. Eckmann, editor, *Seminar on Triples and Categorical Homology Theory*, number 80 in Lecture Notes in Mathematics, pages 75– 90. Springer Verlag, 1969.

[50] C. Lüth. Compositional term rewriting: An algebraic proof of Toyama's theorem. In H. Ganzinger, editor, *Rewriting Techniques and Applications, 7th International Conference*, number 1103 in Lecture Notes in Computer Science, pages 261– 275, New Brunswick, USA, July 1996. Springer Verlag.

[51] C. Lüth and N. Ghani. Monads and modular term rewriting. In *Category Theory in Computer Science CTCS'97*, number 1290 in Lecture Notes in Computer Science, pages 69– 86, Santa Margherita, Italy, Sept. 1997. Springer Verlag.

[52] S. Mac Lane. *Categories for the Working Mathematician*, volume 5 of *Graduate Texts in Mathematics*. Springer Verlag, 1971.

[53] E. G. Manes. *Algebraic Theories*, volume 26 of *Graduate Texts in Mathematics*. Springer Verlag, 1976.

[54] M. Marchiori. Modularity of completeness revisited. In J. Hsiang, editor, *Proceedings of the 6th International Conference on Rewriting Techniques and Applications*, number 914 in Lecture Notes in Computer Science, pages 2– 10, Kaiserslautern, Apr. 1995. Springer Verlag.

[55] P.-A. Melliès. A factorisation theorem in rewriting theory. In *Category Theory in Computer Science CTCS'97*, number 1290 in Lecture Notes in Computer Science, pages 49– 68, Santa Margherita, Italy, Sept. 1997. Springer Verlag.

[56] P.-A. Melliès. A stability theorem in rewriting theory. In *14th Annual Symposium on Logic in Computer Science*, Indianapolis, USA, 1998. IEEE, Computer Society Press.

[57] J. Meseguer. Conditional rewriting logic as a unified model of concurrency. *Theoretical Computer Science*, 96:73–155, 1992.

[58] A. Middeldorp. A sufficient condition for the termination of the direct sum of term rewriting systems. In *Fourth Annual Symposium on Logic in Computer Science*, pages 396–401. IEEE, Computer Society Press, June 1989.

[59] A. Middeldorp. *Modular Properties of Term Rewriting Systems*. PhD thesis, Vrije Universiteit te Amsterdam, 1990.

[60] R. Milner. Action structures. Technical Report ECS-LFCS-92-249, LFCS, Dec 1992.

[61] E. Moggi. Computational lambda-calculus and monads. In *Fourth Annual Symposium on Logic in Computer Science*. IEEE, Computer Society Press, June 1989.

[62] E. Ohlebusch. On the modularity of termination of term rewriting systems. *Theoretical Computer Science*, 136:333– 360, 1994.

[63] V. v. Oostrom. *Confluence for Abstract and Higher-Order Rewriting*. PhD thesis, Vrije Universiteit te Amsterdam, 1994.

[64] A. J. Power. An abstract formulation for rewrite systems. In D. H. Pitt, D. E. Rydeheard, P. Dybjer, A. M. Pitts, and A. Poigné, editors, *Category Theory and Computer Science*, number 389 in Lecture Notes in Computer Science, pages 300–312, Manchester, Sept. 1989. Springer Verlag.

[65] A. J. Power. An algebraic formulation for data refinement. In M. Main, A. Meton, M. Mislove, and D. Schmidt, editors, *Mathematical Foundations of Programming Semantics.* 5^{th} *International Conference*, number 442 in Lecture Notes in Computer Science, pages 390–401. Springer Verlag, Mar/Apr 1989.

[66] A. J. Power. A 2-categorical pasting theorem. *Journal of Algebra*, 2(129), Mar 1990.

[67] D. Prawitz. Ideas and results in proof theory. In J. E. Fenstad, editor, *Proceedings of the* 2^{nd} *Scandinavian Logic Symposium*, pages 235–307. North Holland, 1971.

[68] H. Reichel. A 2-category approach to critical pair completion. In *Recent Trends in Data Type Specification*, number 534 in Lecture Notes in Computer Science, pages 266–273. Springer Verlag, 1991.

[69] E. Robinson. Variations on algebra: monadicity and generalisations of equational theories. Technical Report 6/94, Sussex Computer Science Technical Report, 1994.

[70] J. A. Robinson. A machine-oriented logic based on the resolution principle. *Journal of the ACM*, 12:23– 41, 1965.

[71] R. K. Rosen. Tree-manipulating systems and Church-Rosser theorems. *Journal of the ACM*, 20:160– 187, Nov 1973.

[72] M. Rusinowitch. On the termination of the direct sum of term-rewriting systems. *Information Processing Letters*, 26(2):65–70, 1987.

[73] D. E. Rydeheard and J. G. Stell. Foundations of equational deduction: A categorical treatment of equational proofs and unification algorithms. In *Category Theory and Computer Science*, number 283 in Lecture Notes in Computer Science, pages 114– 139. Springer Verlag, 1987.

[74] D. Sannella, S. Sokołowski, and A. Tarlecki. Toward formal development of programs from algebraic specifications: Parameterisation revisited. *Acta Informatica*, 29(8):689–736, 1992.

[75] D. T. Sannella and R. M. Burstall. Structured theories in LCF. In *8th Colloqium on Trees in Algebra and Programming*, number 159 in Lecture Notes in Computer Science, pages 377– 391, L'Aquila, Italy, 1983. Springer Verlag.

[76] D. T. Sannella and A. Tarlecki. On observational equivalence and algebraic specification. *Journal of Computer and System Sciences*, 34(2/3):150– 178, April/June 1987.

[77] D. T. Sannella and A. Tarlecki. Specifications in an arbitrary institution. *Information and Computation*, 76(2/3):165–210, Feb/Mar 1988.

[78] D. T. Sannella and A. Tarlecki. A kernel specification formalism with higher-order parameterisation. LFCS Report Series ECS-LFCS-91-139, LFCS, Feb. 1991.

[79] D. T. Sannella and M. Wirsing. A kernel language for algebraic specification and implementation. In *International Conference on Foundations of Computation Theory*, number 158 in Lecture Notes in Computer Science, pages 413– 427, Borgholm, Sweden, 1983. Springer Verlag.

[80] R. A. G. Seely. *Hyperdoctrines and Natural Deduction.* PhD thesis, University of Cambridge, Jun 1977.

[81] R. A. G. Seely. Modelling computations: A 2-categorical framework. In *Proceedings of the Second Annual Symposium on Logic in Computer Science*, pages 65–71, 1987.

[82] J. G. Stell. *Categorical Aspects of Unification and Rewriting.* PhD thesis, Unversity of Manchester, 1992.

[83] J. G. Stell. Modelling term rewriting systems by Sesqui-categories. Technical Report TR94-02, Keele Unversity, Jan. 1994.

[84] K. Stokkermans. A categorical formulation for critical-pair/completion procedures. In M. Rusinowitch and J. L. Rémy, editors, *Third International Workshop on Conditional Term Rewriting Systems*, number 656 in Lecture Notes in Computer Science, pages 171–175, Pout-à-Mousson, 1992.

[85] R. Street. The formal theory of monads. *Journal for Pure and Applied Algebra*, 2:149–168, 1972.

[86] R. Street. Limits indexed by category-valued 2-functors. *Journal for Pure and Applied Algebra*, 8:149–181, 1976.

[87] Y. Toyama. Counterexamples to termination for the direct sum of term rewriting systems. *Information Processing Letters*, 25(3):141–143, 1987.

[88] Y. Toyama. On the Church-Rosser property for the direct sum of term rewriting systems. *Journal of the ACM*, 34(1):128–143, 1987.

[89] Y. Toyama, J. W. Klop, and H. P. Barendregt. Termination for the direct sums of left-linear complete term rewriting systems. *Journal of the ACM*, 42(6):1275– 1304, Nov 1995.

[90] D. Turi and G. Plotkin. Towards a mathematical operational semantics. In *Twelfth Annual Symposium on Logic in Computer Science.* IEEE, Computer Society Press, 1997.

[91] D. Turi and J. Rutten. On the foundations of final co-algebra semantics: non-well-founded sets, partial orders, metric spaces. *Mathematical Structures in Computer Science*, To appear, 1998.

[92] H. P. Yap. *Some Topics in Graph Theory*, volume 108 of *London Mathematical Society Lecture Note Series*. Cambridge University Press, 1986.

Index