Czech Technical University in Prague
Faculty of Electrical Engineering

MASTER'S THESIS

Many-valued coalgebraic logic

Prague, 2013

Author: **Matěj Dostál**
Department of Cybernetics
Open Informatics
Artificial Intelligence
Advisor: **Mgr. Marta Bílková, Ph.D.**
Charles University in Prague

České vysoké učení technické v Praze
Fakulta elektrotechnická

Katedra kybernetiky

ZADÁNÍ DIPLOMOVÉ PRÁCE

Student: Bc. Matěj D o s t á l

Studijní program: Otevřená informatika (magisterský)

Obor: Umělá inteligence

Název tématu: Vícehodnotová koalgebraická logika

Pokyny pro vypracování:

Koalgebry jsou robustním abstraktním nástrojem k popisu nekonečných datových typů a reaktivních systémů. Koalgebraická modální logika popisuje chování těchto systémů. Předmětem diplomové práce, jejíž povaha je teoretická, bude studium vícehodnotových modálních koalgebraických logik.
Matěj Dostál prostuduje existující literaturu k Mossovým koalgebraickým logikám a prozkoumá možnost zavedení vícehodnotových variant těchto logik v kategorii množin. Konkrétně popíše modální logiku vícehodnotových kripkovských rámců, logiku kripkovských rámců s vícehodnotovou sémantikou, a tato dvě zobecnění spojí do logiky vícehodnotových kripkovských rámců s vícehodnotovou sémantikou. Tato zobecnění dosud nebyla v literatuře koalgebraicky zpracována.

Seznam odborné literatury:

[1] J. Adamek: Introduction to coalgebra, Theory and Applications of Categories, Vol. 14, No. 8, 2005, pp. 157-199.
[2] P. Blackburn, M. de Rijke, Y. Venema: Modal Logic. Cambridge University Press, 2001.
[3] M. Bilkova, A. Kurz, D. Petrisan and J. Velebil: Relation Liftings on Preorders, in proccedings CALCO 2011, LNCS 6859 (2011), pp. 115-129.
[4] M. Bilkova, A. Kurz, D. Petrisan and J. Velebil: Relation Lifting, with an Application to the Many--Valued Cover Modality, submitted, 2012.
[5] P. Cintula, P. Hájek, C. Noguera (ed.): Handbook of Mathematical Fuzzy Logic. Volume 1 College Publications, 2011.
[6] L. Moss: Coalgebraic logic. Ann. Pure Appl. Logic 96, 1999.
[7] J.J.M.M. Rutten: Universal coalgebra: a theory of systems, Theoretical Computer Science 249 (2000) 3-80.

Vedoucí diplomové práce: Mgr. Marta Bílková, Ph.D.

Platnost zadání: do konce zimního semestru 2013/2014

L.S.

prof. Ing. Vladimír Mařík, DrSc.
vedoucí katedry

prof. Ing. Pavel Ripka, CSc.
děkan

V Praze dne 4. 6. 2012

Czech Technical University in Prague
Faculty of Electrical Engineering

Department of Cybernetics

DIPLOMA THESIS ASSIGNMENT

Student: Bc. Matěj D o s t á l

Study programme: Open Informatics

Specialisation: Artificial Intelligence

Title of Diploma Thesis: Many-valued Coalgebraic Logic

Guidelines:

Coalgebras are a robust abstract tool for the description of infinitary data types and reactive systems. Coalgebraic modal logic describes the behaviour of these systems. Main aim of the proposed master thesis is to study many-valued modal coelgebraic logics.
The task of Matěj Dostál is to study the existing literature on Moss' coalgebraic logics, and describe possible options to introduce many-valued variants of these logics in the category of sets. In particular, he will describe a modal logic for many-valued Kripke frames, for Kripke frames with many-valued semantics, and merge these two generalisations into a many-valued logic of many-valued Kripke frames. These generalisations have not yet been discussed coagebraically in the literature.

Bibliography/Sources:

[1] J. Adamek: Introduction to coalgebra, Theory and Applications of Categories, Vol. 14, No. 8, 2005, pp. 157-199.
[2] P. Blackburn, M. de Rijke, Y. Venema: Modal Logic. Cambridge University Press, 2001.
[3] M. Bilkova, A. Kurz, D. Petrisan and J. Velebil: Relation Liftings on Preorders, in proccedings CALCO 2011, LNCS 6859 (2011), pp. 115-129.
[4] M. Bilkova, A. Kurz, D. Petrisan and J. Velebil: Relation Lifting, with an Application to the Many--Valued Cover Modality, submitted, 2012.
[5] P. Cintula, P. Hájek, C. Noguera (ed.): Handbook of Mathematical Fuzzy Logic. Volume 1 College Publications, 2011.
[6] L. Moss: Coalgebraic logic. Ann. Pure Appl. Logic 96, 1999.
[7] J.J.M.M. Rutten: Universal coalgebra: a theory of systems, Theoretical Computer Science 249 (2000) 3-80.

Diploma Thesis Supervisor: Mgr. Marta Bílková, Ph.D.

Valid until: the end of the winter semester of academic year 2013/2014

prof. Ing. Vladimír Mařík, DrSc.
Head of Department

L.S.

prof. Ing. Pavel Ripka, CSc.
Dean

Prague, June 4, 2012

Prohlášení

Prohlašuji, že jsem předloženou práci vypracoval samostatně a že jsem uvedl veškeré použité informační zdroje v souladu s Metodickým pokynem o dodržování etických principů při přípravě vysokoškolských závěrečných prací.

V Praze dne 3.1.2012

podpis

Acknowledgements

I would like to greatly appreciate my supervisor, Marta Bílková, for her patience and guidance during the whole time I wrote the thesis. Many thanks go to Jiří Velebil for his interest in the topic and for his help with various aspects of the thesis. I would also like to thank my family for their constant support.

Abstrakt

Koalgebraická Mossova logika je vhodná k popisu chování různých reaktivních systémů modelovaných koalgebrami, například deterministických automatů nebo Kripkovských rámců. V této práci zavedeme pojem vícehodnotového relačního liftingu a s jeho pomocí definujeme vícehodnotovou koalgebraickou logiku jako zobecnění standardní Mossovy logiky. Dále ukážeme, že je tato logika expresivní.

Klíčová slova

Koalgebra, modální logika, vícehodnotová logika, relační lifting.

Abstract

Coalgebraic Moss' logic is useful for description of behaviour of various reactive systems modeled by coalgebras, for example deterministic automata or Kripke frames. In this thesis we introduce the notion of many-valued relation lifting and with its help we define many-valued coalgebraic logic as a generalisation of the standard Moss' logic. We show results concerning its expresiveness.

Keywords

Coalgebra, modal logic, many-valued logic, relation lifting.

Contents

Preface

In this thesis we are going to present a many-valued logic which will be useful to describe the behaviour of various reactive systems. Many examples of reactive systems can be found in the field of modal logic, automata theory, or computer science in general, because they are useful to describe processes which do not have to terminate. Examples of systems include streams and deterministic automata: the behaviour of streams gives an infinite sequence of outputs, the behaviour of an automaton is the language it accepts. We shall use the notion of coalgebra and the theory of universal coalgebra [49] as a unifying framework to model systems. A system with a set A of inner states is modeled as a single transition map — a coalgebra $c : A \to TA$ (see Definition 2.2.1), and its type is captured by a set functor T.

Finding logics that describe behaviour of systems is important: various systems can have a different internal structure and yet they can be observationally indistinguishable. Sometimes we want to abstract from the internal structure of systems. This is why we are very much interested in logics which can distinguish systems that behave differently, and which cannot distinguish systems that differ only in their internal structure and not in behaviour. The notion of behavioural equivalence (see Definition 2.6.1) and bisimulation (see Definition 2.6.2) is central in this context. This is one of the starting points for coalgebraic logic.

The field of coalgebraic logic is well-established and many results have already been proved.

There are more possible approaches how to define logics that capture the notion of behavioural equivalence in a coalgebraic level of generality, and such logics are known for a large class of systems (we give the references below). The logics are modal in nature, modalities are used to describe the dynamics of the system internally. The logics mostly extend the classical propositional logic, or its positive fragment. Classical modal logics, as for example normal modal logics, graded modal logics or probabilistic modal logics can be studied as coalgebraic logics for appropriate functors.

We concentrate on one of the possible approaches to coalgebraic logic, and generalise the logics introduced by Larry Moss [45], more precisely, their finitary variants. This approach gives us a uniform definition of logics parametric in the type T of the system. The language of the logic contains a single modality, whose arity is given by the type of the system — the functor T — and whose semantics is given by the functor as well, via the notion of relation lifting (see Section 2.4.) The language is expressive enough to distinguish systems with different behaviour.

Our goal is to present coalgebraic logics, for a large class of systems, Moss'

coalgebraic logics that are moreover many-valued, or fuzzy — they extend propositional logics given by more general algebras of truth values (such as the unit interval $[0, 1]$). Many-valued or fuzzy logics capture approximate reasoning or reasoning with vague or uncertain information, which is useful in modeling real-life situations, for example the inaccuracy of measurement. The logic we introduce will therefore have more expressive power while retaining the nice properties of standard coalgebraic logic, that is, distinguishing systems with different behaviour. This also presents a way of defining many-valued or fuzzy modal logics in a coalgebraic level of generality.

In order to achieve our goal, we are going to present some technical results which are of independent interest, namely to define semantics of many-valued Moss' logic we introduce a many-valued relation lifting.

State of the art

Coalgebraic logics The theory of coalgebras has been introduced by Peter Aczel in the late 1980s [1, 2]. The study of universal coalgebra as the general study of systems has been proposed in Jan Rutten's paper [49]. The various approaches to define coalgebraic logics include Moss' logics introduced in [45], logics with modalities arising from predicate liftings [48, 51], or equivalently from a Stone type dualities [13, 39]. Many-sorted languages have been introduced in [31], logics of coequations have been introduced in [4].

Larry Moss introduced in [45] his logic for coalgebras for set functors preserving weak pullbacks and proved its expressivity. The language of the logic he introduced was, however, infinitary. This led to the study of an expressive finitary variant of Moss' logic, and subsequently, a complete axiomatisation [33, 34] and proof theory were introduced in [10]. The applications in automata theory are given in [35, 36], and fixpoint extensions of the logic on a coalgebraic level of generality are treated in [53].

Preliminary study of many-valued Moss' coalgebraic logic stemming from many-valued relation lifting appears in [9] in the setting of enriched category theory. The aim of this thesis is to study many-valued Moss' logic for coalgebras over the category Set and to find the boundaries of this approach.

Many-valued and fuzzy logics and relations Many-valued logics are non-classical logics generalising classical logic by allowing for a set (an algebra) $\mathscr{V}$ of truth degrees other than the two-element set (boolean algebra) $\{0, 1\}$ of classical truth values. Such logics are called fuzzy if the underlying lattice of the algebra of the truth values is the unit interval, or, in a broader sense, if they are complete with respect to linearly ordered algebras — chains (see e.g. [16]). The main idea is that these are the logics of comparative truth degrees. From the large ammount of fuzzy logics we actually mention only Gödel fuzzy logic in this thesis. As a basic reference to fuzzy logics and their algebraic semantics we refer the reader to Petr Hájek's book [26], and to a more recent handbook [18].

Many-valued binary relation from A to B is a mapping $R : A \times B \to V$, where V is the underlying set of an algebra of the truth values $\mathscr{V}$ — a complete lattice in general, often a complete Heyting algebra (called also Brouwerian lattice) or a quantale (see e.g. [47]). Such a relation is called a fuzzy relation

if $\mathscr{V}$ is the unit real interval $[0;1]$ (or, in broader sense, a complete algebra for some fuzzy logics — i.e. an MV algebra or Gödel algebra).

Fuzzy relations are a core concept of fuzzy mathematics and they play an important role in fuzzy modeling, fuzzy diagnosis, and fuzzy control. The study of fuzzy sets and relations originated in work of Lotfi Zadeh [55, 56] and Joseph Goguen [24]. There the category $\mathsf{Rel}_{\mathscr{V}}$ of $\mathscr{V}$-relations was defined, with $\mathscr{V}$ being in general a quantale (i.e. a complete lattice semigroup with the semigroup operation distributing over joins.) Fuzzy relations and their properties were consequently studied e.g. in [46], or in [15] in the framework of fuzzy class theory.

We consider the category $\mathscr{V}$-mat of sets and many-valued relations in this thesis, which has a two-dimensional structure given by a preorder on many-valued relations. We start with assumption that $\mathscr{V}$ is a quantale, later we restrict ourselves to the case when $\mathscr{V}$ is a complete Heyting algebra or a complete Gödel chain.

Many-valued modal logics Most of the many-valued modal logics appearing in the literature (see e.g. [26, 29]) are many-valued counter-parts of rather strong classical normal modal logic S5. A systematic study of a *minimal* modal logic over a given algebra $\mathscr{V}$ of truth values appeared in [21, 22] (where $\mathscr{V}$ is a finite Heyting algebra with constants in the language) and [17, 44] (where $\mathscr{V}$ is the standard Gödel chain $[0,1]$), and most consistently in [14], where $\mathscr{V}$ is a finite residuated lattice.

Particular many-valued or fuzzy logics which have appeared in the literature include e.g. fuzzy description logics (see [42] for an overview, or [27]), generalisations of belief logics to capture fuzzy events [28].

Our work relates to that on a minimal modal logic over a given lattice $\mathscr{V}$. Although the syntax of Moss' language is nonstandard, the approach gives a notion of the minimal modal logic for coalgebras of a given functor, special case of which includes variants of many-valued Kripke frames considered in [14].

Motivation

We study a many-valued variant of finitary Moss' coalgebraic logic for coalgebras for finitary set functors preserving weak pullbacks, thus providing expressive modal languages specifying behaviour of the coalgebras in a uniform way.

There are many motivating examples of such coalgebras:

- Infinite streams.
- Infinitary datatypes: trees, graphs.
- Deterministic and nondeterministic automata.
- Most variants of other automata from automata theory.
- Kripke frames from modal logic.
- Many-valued Kripke frames.

- Reactive agents from multiagent systems.

It is natural to ask whether there is a many-valued extension of the standard Moss' logic which retains most of its nice properties. We are therefore interested in finding the most general approach of defining such logics which is applicable to a large class of systems.

However, there is a (slightly unnatural) way to incorporate many-valuedness into the two-valued Moss' logic which uses the already known theory. To specify a language for T-coalgebras it is possible, to some extent, to use the existing classical Moss' logics for set coalgebras for the functor $T \times \mathscr{V}^{\mathsf{At}}$. But it is not intuitive. The atomic propositions are incorporated in the modality of the language. Moreover, the semantics of this language only allows many-valued valuation for atomic propositions, which are not an independent part of the language. All the formulas of the language have two-valued semantics.

We would like for example to vary the propositional part of the logic to coincide e.g. with a given many-valued or fuzzy logic, or intuitionistic logic. This is not possible without changing the classical setting and we found it more intuitive and natural to work with set coalgebraic models that have non-classical, many-valued valuations.

Results

The main result of the thesis is the definition of a many-valued variant of Moss' coalgebraic logic for coalgebras for finitary set functors preserving weak pullbacks.

To be able to do so, a purely technical result of independent interest (and a nontrivial one) of functorial relation lifting for many-valued relations is proved for these functors. This result moreover extends the classical lifting result of Věra Trnková in [52] in a natural way. The lifting operation satisfies the required properties:

- It commutes with the standard operations on relations (composition, opposites, restrictions).
- It is a 2-functor on the category of many-valued relations.
- For a functor T, it is an extension of T.

The algebra of truth values is restricted to be a complete Heyting algebra for the lifting to be functorial (and we give a counterexample to show that our approach does not allow for more general algebras of truth values).

We then give a notion of many-valued bisimulation and prove that the resulting notion of bisimilarity coincides with the classical notion of bisimilarity for the same structures. Therefore, this notion coincides with behavioural equivalence for the class of functors we are working with.

The resulting many-valued Moss' logic is proved to be expressive. We prove the expressivity of the closed fragment of our logic using the expressivity of the classical Moss' logic for the same class of structures. Then we show that the full language is expressive as well, under the restriction that the truth-values algebra is a complete Gödel chain. We prove this result directly by using the properties of the many-valued relation lifting.

Finally, we show examples of logics that arise from the theory. The semantics of these logics for various classes coalgebras are given and we relate them related to existing work on fuzzy modal logics.

Structure of the text

The thesis is divided into four chapters.

Chapter 1 In the first chapter we deal with mathematical prerequisites and we introduce the basic concepts which are used throughout the text. It includes an introduction to category theory, defines many-valued relations and establishes the notation. All the results in this chapter are well-established and the chapter does not contain new material.

Chapter 2 The second chapter includes a brief introduction to coalgebras and their logics. We discuss how coalgebras can model various systems in Section 2.2. Then we define relation lifting in Section 2.4 and show how it can be used to introduce expressive logics describing behaviour of coalgebras in Section 2.7. We show the logic for many-valued Kripke frames as an application of the introduced coalgebraic logic in Section 2.7 as well.

Chapter 3 We define many-valued relation lifting in order to be able to introduce many-valued logics for coalgebras in the next chapter. First we define many-valued relation lifting for polynomial functors in Section 3.3, and then we use the technique of functor presentations (introduced in Section 3.4) to extend the definition to a large class of finitary functors in Section 3.5. The definition of many-valued relation lifting using functor presentations and the proofs of its properties are novel.

Chapter 4 The notion of many-valued relation lifting gives rise to an interesting logic for coalgebras. We describe its syntax and semantics in Section 4.1 and show examples of logics for specific functors. Then we study the notion of bisimilarity induced by many-valued relation lifting and show some of its properties (Section 4.3). Finally, we give expressivity results for the introduced logic in Section 4.4. The results contained in this chapter are new.

Chapter 1

Mathematical prerequisites

Throughout this thesis we are going to actively use notions from various branches of mathematics and computer science. In this chapter we will present some of the core concepts and examples from category theory, show some basic set-theoretical constructions and we will look at some of the algebraic structures used for capturing "fuzzy truth values". One of the core notions of the thesis, namely *coalgebra*, will be thoroughly treated in the following chapter.

The concepts introduced in this chapter can be considered as a standard material. For each topic we will therefore provide the reader with references for introductory texts and for further reading.

Some of the more specialised definitions that are not considered standard are going to be introduced in later chapters within the flow of the text.

1.1 Category theory

We will very often use the language of category theory, for it will allow us to formulate arguments of vast generality very succintly. Unfortunately, this section cannot be used as a complete introduction to the subject, so we are referring the curious reader to literature in case of need. A great elementary introduction to category theory can be found in [41], the textbook [8] is meant for those interested in applications of category theory in computer science. For those more mathematically inclined, we point to [6] and to the standard reference [43].

Categories and constructions

Definition 1.1.1 (Category)**.** A *category* $\mathscr{X}$ is a two-sorted structure consisting of a collection of objects and a collection of morphisms which are interconnected as follows:

- Each morphism f has a unique domain object A and a unique codomain object B, denoted by $f : A \to B$.
- For each pair of morphisms $f : A \to B$, $g : B \to C$ we have a unique morphism $g \circ f : A \to C$ called the composition of f and g.

The composition has to satisfy the following axioms:

- Identity axiom: For every object A there is a morphism $\mathsf{id}_A : A \to A$ such that for every morphism $f : A \to B$ it holds that

$$\mathsf{id}_B \circ f = f = f \circ \mathsf{id}_A.$$

- Associativity axiom: For each triple of morphisms $f : A \to B$, $g : B \to C$, $h : C \to D$, their composition is associative, that is,

$$(h \circ g) \circ f = h \circ (g \circ f).$$

If for each pair A and B of objects from $\mathscr{X}$ the collection of morphisms from A to B constitutes a set, the category $\mathscr{X}$ is said to be *locally small*. We then call this set of morphisms a *hom-set* and denote it by $\mathscr{X}(A, B)$.

Example 1.1.2 (Sets and functions)**.** The collection of all sets and the collection of functions between sets together form a category. Observe that for every set there is an identity function operating on that set and that composition of functions is associative. We denote this category by Set.

Moreover, this example shows us why we are talking about "collections" of objects and morphisms and not about sets: the collection of all sets is a proper class and not a set.

Note 1.1.3 (Sets and their elements)**.** Usually we are going to denote sets by capital letters A,B,C; typical element of the set A is a. In case we need to talk about more elements from B, we use subscript notation b_1, b_2, b_3, or more generally b_i going over an indexing set I.

Example 1.1.4 (Natural numbers)**.** We denote the set of natural numbers by $\mathbb{N}$ and include 0 as a natural number for reasons of convenience.

Moreover, n can stand for a set $\{0, 1, \ldots, n-1\}$ with n elements. The set 0 is therefore the empty set $\emptyset$.

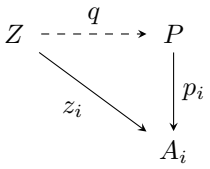
Definition 1.1.5 (Opposite category)**.** Given a category $\mathscr{X}$, we get its *opposite category* $\mathscr{X}^{op}$ by reversing the direction of morphisms. More precisely, the collection of objects of $\mathscr{X}^{op}$ is the same as the collection of objects of $\mathscr{X}$, and the hom-set $\mathscr{X}^{op}(A, B)$ is the hom-set $\mathscr{X}(B, A)$. Composition in $\mathscr{X}^{op}$ is derived from the composition in $\mathscr{X}$ in the obvious way:

$$f \circ^{op} g = (g \circ f)^{op},$$

where $f : A \to B$ and $g : B \to C$ are morphisms in $\mathscr{X}$. It is easy to see that this construction indeed gives a category.

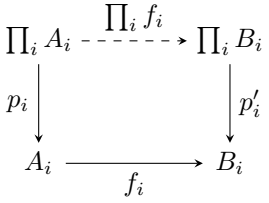
Now we are going to introduce categorical constructions which have a well-known meaning in the category of sets.

Definition 1.1.6 (Products)**.** Given a collection A_i of objects indexed by a set I, their *product* is an object P together with a collection $p_i : P \to A_i$ of morphisms indexed by I satisfying the *universal property*: for any object Z and a collection $z_i : Z \to A_i$ there is a unique morphism $q : Z \to P$ making the following diagram commute



for all $i \in I$. The morphism q is usually called *mediating*. We shall often denote P by $\prod_i A_i$.

Note 1.1.7 (Product of functions)**.** Having a collection $f_i : A_i \to B_i$ of morphisms (indexed by I), we can form their product with the use of the universal property of products. We define the morphism $\prod_i f_i : \prod_i A_i \to \prod_i B_i$ as the mediating morphism making the following diagram



commute for every $i \in I$.

Note 1.1.8 (Binary products)**.** In the case that the collection contains just two objects, say A and B, their product is denoted by $A \times B$, analogously we write $f \times g$ as the product of two morphisms f and g.

Example 1.1.9 (Binary products in Set)**.** In the category Set, the binary product $A \times B$ is the set of all ordered pairs $\{(a, b) \mid a \in A, b \in B\}$.

The product of two functions $f : A \to X$ and $g : B \to Y$ is a function $f \times g : A \times B \to X \times Y$ and is defined by the following formula:

$$(f \times g)(a, b) = (f(a), g(b)).$$

Definition 1.1.10 (Coproducts)**.** Given a collection A_i of objectsfrom category $\mathscr{X}$ indexed by a set I, their *coproduct* is the dual notion to that of a product. That is, coproducts in the category $\mathscr{X}$ are products in the category $\mathscr{X}^{op}$.

We denote the coproduct of objects A_i by $\coprod_i A_i$.

Note 1.1.11 (Coproduct of functions)**.** The coproduct of a collection $f_i : A_i \to B_i$ of morphisms is the morphism $\coprod_i f_i : \coprod_i A_i \to \coprod_i B_i$ defined again with the use of the mediating morphism.

Note 1.1.12 (Binary coproducts)**.** If the collection again contains just two objects A and B, their coproduct is denoted by $A + B$, and we write $f + g$ as the coproduct of two morphisms f and g.

Example 1.1.13 (Binary coproducts in Set)**.** The coproduct (or sum) $A + B$ of two sets A,B can be encoded as the set of ordered pairs

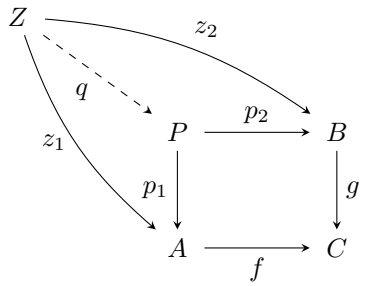
$$\{(a, 1) \mid a \in A\} \cup \{(b, 2) \mid b \in B\}.$$

The sum of two functions $f : A \to X$ and $g : B \to Y$ is a function $f + g : A + B \to X + Y$ and is defined by the following formula:

$$(f+g)(s) = \begin{cases} (f(a), 1) & \text{if } s = (a, 1) \\ (g(b), 2) & \text{if } s = (b, 2) \end{cases}.$$

Below we are going to introduce a common generalisation of the set intersection and inverse image operation.

Definition 1.1.14 (Pullbacks)**.** A *pullback* of two morphisms $f : A \to C$ and $g : B \to C$ is a pair of morphisms $p_1 : P \to A$ and $p_2 : P \to B$ such that the square in the following diagram commutes



and for any pair of morphisms z_1 and z_2 making the outline commute, there is a unique mediating morphism $q : Z \to P$ filling the diagram in.

Example 1.1.15 (Pullbacks in Set)**.** In the category Set, the pullback P of the functions $f : A \to C$ and $g : B \to C$ is the set

$$\{(a, b) \mid f(a) = g(b)\}$$

together with the obvious projection morphisms p_1 and p_2.

Example 1.1.16 (Intersections and inverse images)**.** Suppose that in the preceding diagram $A \subseteq C$ and $B \subseteq C$, and the functions f,g are the inclusion maps. Then P is the intersection $A \cap B$ and the maps p_1 and p_2 are inclusion maps $A \cap B \hookrightarrow A$ and $A \cap B \hookrightarrow B$.

In the more general case when we know just that $B \subseteq C$ and that g is the inclusion map, the set P is the inverse image of B under f:

$$P = \{a \in A \mid f(a) \in B\}.$$

In this case p_1 is the inclusion map $P \hookrightarrow A$.

Definition 1.1.17 (Weak pullbacks)**.** The definition of a *weak pullback* is the same as that of a pullback with the distinction that the mediating morphism does not have to be unique.

To the contrary of pullbacks, weak pullbacks need not be determined uniquely up to an isomorphism. For example, in the category Set, any superset of a nonempty pullback is a weak pullback.

Functors

A functor can be thought of as a homomorphism of categories. We can give it another meaning from the computer science point of view: a functor is a parametric datatype [25].

Definition 1.1.18 (Functor)**.** Let $\mathscr{X}$ and $\mathscr{Y}$ be categories. A *functor* $T : \mathscr{X} \to \mathscr{Y}$ is given by two actions:
Action on objects:

$$A \mapsto TA$$

is given for every object A of $\mathscr{X}$, and the image TA is an object of $\mathscr{Y}$.
Action on morphisms:

$$(f : A \to B) \mapsto (Tf : TA \to TB)$$

for every morphism $f \in \mathscr{X}$.
The action on morphisms must satisfy two laws, the identity law

$$T(\mathsf{id}_A) = \mathsf{id}_{TA}$$

and the composition law

$$T(g \circ f) = T(g) \circ T(f).$$

If the categories $\mathscr{X}$ and $\mathscr{Y}$ are the same, we say that T is an *endofunctor of* $\mathscr{X}$.

Definition 1.1.19 (Functor composition)**.** Two functors $T : \mathscr{X} \to \mathscr{Y}$ and $U : \mathscr{Y} \to \mathscr{Z}$ can be composed to give a functor $U \circ T : \mathscr{X} \to \mathscr{Z}$, defined by the action on objects (for any object A from $\mathscr{X}$)

$$A \mapsto UTA,$$

and by the action on morphisms (for any morphism $f : A \to B$ from $\mathscr{X}$)

$$(f : A \to B) \mapsto (UTf : UTA \to UTB).$$

Remark 1.1.20 (Contravariant functors)**.** Sometimes we find ourselves in a situation where we "nearly" find a functor from $\mathscr{X}$ to $\mathscr{Y}$, but the action on morphisms reverses the direction of morphisms. That is, the action on morphisms is the following

$$(f : A \to B) \mapsto (Tf : TB \to TA)$$

and it holds that

$$T(g \circ f) = T(f) \circ T(g).$$

This just means that we have a functor $T : \mathscr{X}^{op} \to \mathscr{Y}$. Some authors explicitly call this kind of functors *contravariant* ones.

In the following text, we are going to work almost exclusively with endofunctors on Set, that is, with functors $T : \mathsf{Set} \to \mathsf{Set}$.

Functors as parametric datatypes One very helpful way to think of functors is look at them from the computer science point of view. Having an endofunctor $T : \mathscr{X} \to \mathscr{X}$, we can think of $\mathscr{X}$ as being the category of types of a certain programming language. Then T can take a type $A \in \mathscr{X}$ and construct the type TA, which can for example be all arrays with elements of type A. Moreover, given a function $f : A \to B$, we automatically get a function $Tf : TA \to TB$ from arrays of A to arrays of B. This view is proposed in [25] and the following examples are inspired by the mentioned text.

Example 1.1.21 (List)**.** Given a set A, the set $\mathsf{List}(A)$ is the set of all finite lists (or tuples) that can be constructed using the elements of A. If we have a set $A = \{a_1, a_2, a_3\}$, some examples of lists are (a_1, a_2), (a_3, a_3, a_3) or even an empty list $()$. From another point of view, we can see the set A as an (possibly infinite) alphabet and the elements of $\mathsf{List}(A)$ as words over the alphabet A.

To be more precise,

$$\mathsf{List}(A) = \{(a_i)_{i<n} \mid n \in \mathbb{N},\ \forall i : a_i \in A\},$$

where $(a_i)_{i<n}$ is a shortcut for the list $(a_0, a_1, \ldots, a_{n-1})$.

Now we are half done with the definition of our List functor. To define it completely, we have to describe its action on morphisms as well. This action is well known to every functional programmer: Suppose we have a function $f : A \to B$. Then the function $\mathsf{List}(f) : \mathsf{List}(A) \to \mathsf{List}(B)$ is the map function:

$$(a_i)_{i<n} \mapsto (f(a_i))_{i<n}.$$

Now it is easy to see what we mean when we say that functors describe parametric datatypes. We can view any set A as a type and $\mathsf{List}(A)$ constructs the list datatype known from many programming languages. That this datatype is parametric means that we can construct it for any type A. Moreover, if we have a "change of type" morphism $f : A \to B$, we can map lists of type A by $\mathsf{List}(f)$ to lists of type B.

Example 1.1.22 (Powerset)**.** The covariant powerset functor $\mathcal{P}$ maps every set to its powerset

$$\mathcal{P}A = \{Z \mid Z \subseteq A\},$$

and functions are mapped to their corresponding direct image counterparts:

$$\mathcal{P}f(Z) = f[Z],$$

where $f[Z] = \{f(z) \mid z \in Z\}$.

Example 1.1.23 (Finitary powerset)**.** The finitary powerset functor $\mathcal{P}_\omega$ is a modification of the powerset functor. It maps every set to its finitary powerset. That is, a set A is mapped to the set $\{Z \mid Z \subseteq A,\ Z \text{ finite}\}$.

Building the polynomial functors Now we have enough intuition about functors to introduce a very important class of functors: polynomial functors on the category Set. These functors are inductively built from a few basic building blocks and constructions. With them, we will be able to give another description of the List functor and we will see their importance once again in the Section 2.2. They will serve to describe "interfaces" of various kinds of systems, for example automata as we know them from automata theory.

First we define the two building blocks of polynomial functors: constant functor and identity functor.

Example 1.1.24 (Constant functor)**.** Given a set C, the constant functor $C : \mathsf{Set} \to \mathsf{Set}$ maps all sets to C and all maps to the identity map on C.
Action on objects:

$$X \mapsto C$$

Action on morphisms:

$$f \mapsto \mathsf{id}_C$$

Despite its triviality, this functor allows for modeling outputs of different kinds of systems.

Example 1.1.25 (Identity functor)**.** The identity functor $\mathsf{Id} : \mathsf{Set} \to \mathsf{Set}$ acts as an identity map on both objects and morphisms.
Action on objects:

$$A \mapsto A$$

Action on morphisms:

$$f \mapsto f$$

This seemingly trivial functor will allow us to define internal states of systems or automata.

The following examples describe operations on functors, which serve as a means to construct new functors out of simpler ones.

Example 1.1.26 (Binary product of functors)**.** Having two functors T and T', we can form their product $T \times T'$ as well.
Action on objects:

$$A \mapsto TA \times T'A$$

Action on morphisms:

$$f \mapsto Tf \times T'f.$$

Having two datatypes T and T', forming their product can be thought of as forming a struct or record datatype known from C or Pascal, containing datatypes T and T'.

Example 1.1.27 (Arbitrary sum of functors)**.** Given a collection $T_i : \mathsf{Set} \to \mathsf{Set}$, where $i \in I$, we denote by $\coprod_i T_i : \mathsf{Set} \to \mathsf{Set}$ the functor defined as follows:

$$\left(\coprod_i T_i\right)(A) = \coprod_i T_i A, \quad \left(\coprod_i T_i\right)(f) = \coprod_i T_i f.$$

That we have indeed defined a functor follows immediately from the universal properties of coproducts.

Example 1.1.28. Using the datatype perspective of functors, the summing operation describes combination of two datatypes: the functor $\mathsf{List} + \mathcal{P}$ applied to a set A creates a datatype consisting of lists over the set A or subsets of A, and the mapping $(\mathsf{List} + \mathcal{P})(f)$ operates as a map function or as a direct image function depending on whether the input is a list or a set.

Later, this operation will for example allow us to talk about systems which can terminate.

Now we can finally define the class of polynomial functors using the building blocks and operations introduced above.

Definition 1.1.29 (Polynomial functors)**.** We say that a functor T is *polynomial* if it can be constructed inductively according to the following Backus-Naur form:

$$T ::= C \mid \mathsf{Id} \mid T \times T \mid \coprod_i T_i.$$

Remark 1.1.30. Some authors introduce another operation in their definition of polynomial functors. This operation is called *exponentiation*, denoted T^I and for a set A it constructs the set $(TA)^I$. This is a useful operation, because it allows modeling of inputs for various systems. It is, however, equivalent to the product operation.

For some examples, we are going to use the notation T^n when n is a finite set, because this construction is equivalent to the n-fold product of T:

$$T^0 = 1; \quad T^{n+1} = T \times T^n.$$

The formal ommision of the exponentiation operation in the definition of polynomial functors will allow us to simplify the inductive proofs involving polynomial functors.

Example 1.1.31 (List functor as a polynomial functor)**.** We have seen in Example 1.1.21 that the List functor applied to a set A forms a set of lists of finite length. Since a list of length n can be thought of as an element of A^n, we can see the definition of List as equivalent to the following polynomial functor:

$$\mathsf{List} = \coprod_{n \in \mathbb{N}} \mathsf{Id}^n,$$

which has object action

$$X \mapsto \coprod_{n \in \mathbb{N}} X^n$$

and the morphism action precisely copies the morphism action of the List functor, as the reader can easily check.

Definition 1.1.32. A functor T is said to preserve weak pullbacks if it maps weak pullback squares to weak pullback squares. Concretely, for every weak pullback, as seen in the following diagram,

$$\begin{array}{ccc} P & \xrightarrow{p_2} & B \\ {\scriptstyle p_1}\downarrow & & \downarrow{\scriptstyle g} \\ A & \xrightarrow[f]{} & C \end{array}$$

the following diagram has to form a weak pullback as well:

$$\begin{array}{ccc} TP & \xrightarrow{Tp_2} & TB \\ {\scriptstyle Tp_1}\downarrow & & \downarrow{\scriptstyle Tg} \\ TA & \xrightarrow[Tf]{} & TC. \end{array}$$

So far all of the endofunctors of Set we introduced do preserve weak pullbacks. This technical property will be important for us in the following text.

Natural transformations

Natural transformations can be seen as a collection of functions which transforms one datatype into another in a "natural" way. Naturality in this context means that the functions in the collection behave well with regard to substitution of the base type upon which the datatypes are built.

Definition 1.1.33 (Natural transformation)**.** Given two functors $T : \mathscr{X} \to \mathscr{Y}$ and $T' : \mathscr{X} \to \mathscr{Y}$, a *natural transformation* ε from $T : \mathscr{X} \to \mathscr{Y}$ to $T' : \mathscr{X} \to \mathscr{Y}$ is a collection of morphisms $\varepsilon_A : TA \to T'A$ for every object A from $\mathscr{X}$ making the following diagram commute for every morphism $f : A \to B$:

$$\begin{array}{ccc} TA & \xrightarrow{\varepsilon_A} & T'A \\ {\scriptstyle Tf}\downarrow & & \downarrow{\scriptstyle T'f} \\ TB & \xrightarrow[\varepsilon_B]{} & T'B \end{array}$$

We use the notation $\varepsilon : T \to T'$ for natural transformations.

Example 1.1.34 (List to powerset)**.** We can make a set out of any list: for any set A, there is a function $\epsilon_A : \mathsf{List}(A) \to \mathcal{P}A$ working as follows:

$$(a_i)_{i<n} \mapsto \{a_i \mid i < n\}.$$

This collection of functions gives a natural transformation $\epsilon : \mathsf{List} \to \mathcal{P}$.

Moreover, for any A, the function ϵ_A is a surjection. We say that such collection constitutes a *natural epimorphism*.

1.2 Many-valued structures and relations

Relations and their many-valued generalisations are going to play a central role through the whole text. It will be very useful to regard relations as a special kind of two-valued matrices. We are going to generalise the two-valued matrices by allowing them to have entries from a given set V of values.

We start with the classical concepts first.

Definition 1.2.1 (Relation)**.** A (binary) *relation* $R : A \nrightarrow B$ is a subset of the product $A \times B$. We say that the pair $(a, b) \in A \times B$ is in relation R, denoted by $a\ R\ b$, if and only if $(a, b) \in R$.

Definition 1.2.2 (Opposite relation)**.** Given a relation $R : A \nrightarrow B$, its opposite relation is $R^{op} : B \nrightarrow A$, and for elements $b \in B$, $a \in A$ it holds $b\ R^{op}\ a$ if and only if $a\ R\ b$.

Matrices We are going to view relations as matrices. Therefore, instead of denoting $R \subseteq A \times B$, we regard the subset R as its characteristic function $R : A \times B \to 2$, and $R(a, b) = 1$ if and only if $a\ R\ b$ holds.

Definition 1.2.3 (Relation composition)**.** Two relations $R : A \nrightarrow B$ and $S : B \nrightarrow C$ can be composed to give a relation $S \circ R : A \nrightarrow C$ defined as follows: $(S \circ R)(a, c) = 1$ iff there is a $b \in B$ such that $a\ R\ b$ and $b\ S\ c$.

If we view relations as matrices, their composition is simply a generalised multiplication of the matrices that represent them. In a formula,

$$(S \circ R)(a, c) = \bigvee_{b \in B} (R(a, b) \wedge S(b, c)).$$

Here we use the two-element Boolean algebra to interpret the formula.

Definition 1.2.4 (Restriction of a relation)**.** Given a relation $R : A \nrightarrow B$ and two subsets $A' \subseteq A$, $B' \subseteq B$, we define the *restricted relation* $R \restriction_{A' \nrightarrow B'} : A' \nrightarrow B'$ as follows: for two elements $a' \in A'$ and $b' \in B'$ it holds that $R \restriction_{A' \nrightarrow B'} (a', b')$ if and only if $R(a', b')$ holds.

Example 1.2.5 (The category of relations)**.** The collection of all sets as objects and the collection of all relations between sets as morphisms form a category. Composition of relations is associative and the identity function on each set can be viewed as a relation which fulfills the needed identity axiom. We denote the category of sets and relations by Rel.

The reason we are interested in relations described as matrices is that this representation allows us to generalise relations to many-valued relations in a very natural way. Informally, we are just going to replace the two-element set 2 in the characteristic function of a relation by a larger set V, containing as elements the degrees of relatedness.

In order to introduce many-valued relations, we need to introduce a set of "truth values" together with algebraic operations which will allow us to compute the composition of these relations. In the case of standard relations, introduced in the previous section, this algebraic structure was a two-element Boolean algebra.

Definition 1.2.6 (Commutative quantale)**.** A *commutative quantale* $\mathscr{V}$ is a complete lattice $(V, \bigwedge, \bigvee)$ with the structure of a commutative monoid $(V, \otimes, \mathsf{e})$ such that the tensor is monotone and distributes over arbitrary joins.

More in detail, $\mathscr{V}$ being a complete lattice means that V is partially ordered by $\leq$ and every subset $V' \subseteq V$ has an infimum (or meet) and a supremum (or join), denoted by $\bigwedge V'$ and $\bigvee V'$ respectively. From this it follows that V contains the greatest element $\top$ and the lowest element $\bot$. The fact that $(V, \otimes, \mathsf{e})$ is a commutative monoid means that $\otimes$ is commutative, associative, and e is the identity element:

$$v \otimes \mathsf{e} = v = \mathsf{e} \otimes v.$$

The monotonicity of the tensor requires that

$$v \otimes w \leq v' \otimes w$$

holds for $v \leq v'$, and distributivity of tensor over arbitrary joins means that the following equality

$$\left(\bigvee_i x_i\right) \otimes y = \bigvee_i (x_i \otimes y)$$

is satisfied.

The structure of commutative quantales is the same as that of *complete residuated lattices*, and residuated lattices have been intensively studied, see for example [23]. These two structures differ in their notions of homomorphism. We are, however, not going to use homomorphisms of commutative quantales, and therefore these two structures coincide for our purposes.

Definition 1.2.7 (Complete Heyting algebra)**.** A *complete Heyting algebra* $\mathscr{V}$ is a commutative quantale where $\otimes = \wedge$ and $\mathsf{e} = \top$. In other words, it is a complete lattice $(V, \bigwedge, \bigvee)$ where the meet operation distributes over arbitrary joins:

$$\left(\bigvee_i x_i\right) \wedge y = \bigvee_i (x_i \wedge y).$$

Definition 1.2.8 (Gödel chain)**.** We say that a complete Heyting algebra $\mathscr{V}$ is a *Gödel chain* if the ordering relation $\leq$ of the underlying lattice of $\mathscr{V}$ is a linear order, that is, for two elements $v \neq v'$ it either holds that $v \leq v'$ or $v' \leq v$.

Remark 1.2.9 (Residuation)**.** We shall use the *residuation* operation of Heyting algebras and Gödel chains in the Chapter 4. For two elements v and w from V we define the residuation operation $v \to w$ as follows:

$$v \to w = \bigvee \{u \in V \mid v \wedge u \leq w\}.$$

In the case of $\mathscr{V}$ being the two-valued Boolean algebra, residuation coincides with the notion of implication.

We can use quantales, Heyting algebras and Gödel chains to define many-valued relations. The definition will be a straightforward generalisation of the matrix approach to standard relations.

Definition 1.2.10 (Many-valued relation)**.** For a given quantale $\mathscr{V}$, a *many-valued relation* $R : A \nrightarrow B$ is a function $R : A \times B \to V$. We view this function as a $\mathscr{V}$-valued matrix.

We compose two relations $R : A \nrightarrow B$ and $S : B \nrightarrow C$ to get a relation $S \circ R : A \nrightarrow C$ such that

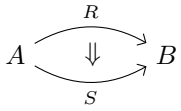
$$(S \circ R)(a, c) = \bigvee_{b \in B} (R(a, b) \otimes S(b, c))$$

holds in $\mathscr{V}$. See that in the case $\mathscr{V}$ is a complete Heyting algebra, the preceding equation is syntactically exactly the same as the equation for standard relations.

Relation ordering Let us have two relations $R : A \nrightarrow B$ and $S : A \nrightarrow B$. We say that $R \leq S$ if and only if for every $a \in A$ and $b \in B$ the following inequality holds:

$$R(a, b) \leq S(a, b).$$

We also sometimes denote this inequality diagramatically as follows:



Definition 1.2.11 (2-functor)**.** We say that a functor $T : \mathsf{Rel} \to \mathsf{Rel}$ is a *2-functor* if it is *locally monotone*: for relations two relations R and S, if $R \leq S$ it holds that

$$TR \leq TS.$$

Notation Sometimes we can use $a\ R\ b$ to denote the value $R(a, b)$, mimicking the notation for standard relations. To be more succint, we say that R is a $\mathscr{V}$-relation instead of calling it a "many-valued relation for a quantale $\mathscr{V}$".

Definition 1.2.12 (Opposite $\mathscr{V}$-relation)**.** Given a $\mathscr{V}$-relation $R : A \nrightarrow B$, its opposite $\mathscr{V}$-relation is $R^{op} : B \nrightarrow A$, and for elements $b \in B$, $a \in A$ it holds

$$R^{op}(b, a) = R(a, b).$$

The opposite of a relation is an example of an operation on matrices. In the following chapters, we will show another interesting operation, namely relation "lifting", which will be of crucial importance.

Definition 1.2.13 (Restriction of a $\mathscr{V}$-relation)**.** Given a $\mathscr{V}$-relation $R : A \nrightarrow B$ and two subsets $A' \subseteq A$, $B' \subseteq B$, we define the *restricted $\mathscr{V}$-relation* $R \restriction_{A' \nrightarrow B'}$: $A' \nrightarrow B'$ in the same manner as we defined restrictions for two-valued relations: for two elements $a' \in A'$ and $b' \in B'$ we define

$$R \restriction_{A' \nrightarrow B'} (a', b') = R(a', b').$$

Example 1.2.14 (The category of $\mathscr{V}$-relations)**.** The collection of all sets and of $\mathscr{V}$-relations between sets is another example of a category. There is an identity $\mathscr{V}$-relation id_A for every set A:

$$\mathsf{id}_A(a, a') = \begin{cases} \mathsf{e} & \text{if } a = a' \\ \bot & \text{otherwise.} \end{cases}$$

An easy computation yields that $\mathscr{V}$-relation composition is associative. We therefore have a category, which we denote as $\mathscr{V}$-mat (meaning $\mathscr{V}$-matrices).

The following are examples of functors which are not endofunctors of Set.

Example 1.2.15 (Graph functor)**.** Functions are often defined as special relations. Because of this, we can construct a functor $\mathsf{gr} : \mathsf{Set} \to \mathsf{Rel}$, which is identity on objects, and it sends a function $f : A \to B$ to its graph:

$$\mathsf{gr}(f) = \{(x, f(x)) \mid x \in A\}.$$

Example 1.2.16 (Relation inclusion functor)**.** Every relation can be viewed as a $\mathscr{V}$-relation in the following sense: let $R : A \nrightarrow B$ be a relation in Rel. Then $\mathcal{I}(R) : A \nrightarrow B$ is a $\mathscr{V}$-relation in $\mathscr{V}$-mat, and $\mathcal{I}(R)(a, b) = \mathsf{e}$ if and only if $R(a, b)$. Otherwise $\mathcal{I}(R)(a, b) = \bot$. This is the morphism action of a functor $\mathcal{I} : \mathsf{Rel} \to \mathscr{V}$-$\mathsf{mat}$, which acts on objects as identity.

Example 1.2.17 ($\mathscr{V}$-graph functor)**.** We define the $\mathscr{V}$-graph functor $\mathsf{Gr} : \mathsf{Set} \to \mathscr{V}$-$\mathsf{mat}$ as the composition $\mathcal{I} \circ \mathsf{gr} : \mathsf{Set} \to \mathscr{V}$-$\mathsf{mat}$ and consider it a direct generalisation of the usual graph functor gr.

Chapter 2

Coalgebras and coalgebraic logics

In this chapter we will introduce the theory of coalgebras and their logics. Coalgebras have received much attention in the last decade, mainly because of their usefulness in computer science. As it is possible to use universal algebra to specify abstract datatypes [54], we can use coalgebras to model nonterminating systems, such as processes. Coalgebras can also be used for specification of object-oriented programming languages. The theory of universal coalgebra has been introduced in [49]. Introductory texts concerning coalgebras include [25], [32] and [3].

When talking about various systems and their behaviour, it is very often useful to abstract from discussing their inner states. In many cases we are much more interested in the observable behaviour of the systems. In program verification, we want to know whether the program enjoys certain observable properties: providing a defined output for some known input, never ending in a deadlock and so on. Given a deterministic automaton, we are interested in the language it accepts. It is therefore natural to ask whether there is a way to capture the notion of systems' behaviour. For many systems, the answer to these queries is positive: we can introduce logics with just enough expressive power to distinguish systems which have different behaviour. The first general treatment of such logics appeared in [45]. The reader may also wish to consult [38] or [20].

2.1 Examples of systems

Before we give the definition of a coalgebra, we are going to show few basic examples of different systems that can be modeled using coalgebras. To show the versatility of the notion of coalgebra, we first introduce these various systems without the use of coalgebraic formalism.

Example 2.1.1 (Streams)**.** Imagine a simple process-like program which regularly switches its internal state and can provide a information associated to its current state on demand. We can try to make a simple model of this program. Let S be the set of the internal states of the program and A the set of outputs

the program can provide. Then we can model the switches of internal states by a function $f : S \to S$. For each internal state we then associate the output. This can be done by a function $o : S \to A$.

Example 2.1.2 (Binary trees)**.** Imagine now we would like to describe a data structure which encodes a proper binary tree (where each node has either 0 or 2 children), which can possibly have infinite depth. Note that these kinds of structures are applicable in functional programming languages with lazy evaluation, for example in Haskell. This data structure can be modeled by a set N of nodes, a subset $B \subseteq N$ of nodes with two children, and two functions $l : B \to N$ and $r : B \to N$, which for each branching node associate its left and right child respectively. We can then draw a tree with a root $r \in N$ by unfolding, that is, if a node n that has already been drawn is contained in B, we draw its left and right children $l(n)$ and $r(n)$. This process of drawing may not terminate in finite time. However, it is possible to encode a tree with infinite depth by a model with a finite set N. Consider the set $N = \{n\}$ together with the functions $l : n \mapsto n$ and $r : n \mapsto n$. Then the drawing is an infinite complete binary tree.

Example 2.1.3 (Deterministic automata [30])**.** The standard definition of a deterministic automaton is as follows:
We have to be given a (finite) set of states S, a (finite) set of inputs I, a function $\delta : S \times I \to S$ which encodes how the automaton switches its internal states with respect to the input it gets, a start state s_0 and a set $F \subseteq S$ of final states.

The notion of a deterministic automaton is very important in automata theory and in computer science applications. It can be used as a programming scheme in text-searching programs, or to define any regular language. Equivalently, it can distinguish strings that follow the pattern of a regular expression from those that do not follow that pattern.

Example 2.1.4 (Kripke frames [12])**.** Kripke frames are of great importance in modal logic, they are the central notion in semantics of standard modal logic. A Kripke frame is a set S of states (sometimes denoted W as a set of "worlds") together with a relation $R : S \nrightarrow S$. This relation is called *accessibility relation* and $R(s, s')$ denotes that it is possible to get from a state s to the state s'.

Notice that the common characteristic for these very different examples is that they carry or can describe infinitary behaviour; streams never stop changing the state, trees can grow into infinite depths, deterministic automata can process any string, without any finite bound on the length on the string.

2.2 Coalgebras and systems

In this section we give the definition of a coalgebra and a coalgebraic momorphism and we show how this gives us a very general framework to talk about various systems. The examples of "coalgebraised" systems from the previous section are meant to convince the reader that this framework is indeed useful.

Definition 2.2.1 (Coalgebra)**.** A *coalgebra* for a endofunctor $T : \mathsf{Set} \to \mathsf{Set}$ (a T-coalgebra) is a pair (A, c), where A is a set and c is a function $c : A \to TA$.

We can think of A as of a set of inner states of the coalgebra c, and the shape of T determines the shape of the successor structure of states of A.

Coalgebras can be more generally defined for endofunctors of a base category different from Set. However, in this thesis we will restrict ourselves to endofunctors of Set.

Definition 2.2.2 (Coalgebraic momorphism)**.** Given two T-coalgebras $c : A \to TA$ and $d : B \to TB$, we say that $f : A \to B$ is a *coalgebraic momorphism* if the following diagram commutes:

$$\begin{array}{ccc} A & \xrightarrow{f} & B \\ {\scriptstyle c}\downarrow & & \downarrow{\scriptstyle d} \\ TA & \xrightarrow[Tf]{} & TB \end{array}$$

Coalgebras for a functor T and their coalgebraic momorphisms form a category Coalg. The composition of two coalgebraic momorphisms $f : A \to B$ and $g : A \to B$ is the function composition $g \circ f : A \to C$. The fact that these data indeed constitute a category is easy to check.

Now we are going to show that the examples of systems shown in Section 2.1 are in fact coalgebras.

Example 2.2.3 (Streams)**.** Streams with a fixed set of outputs A can be modeled as coalgebras for the functor $A \times \mathsf{Id}$. Indeed, if we have a stream with a set of states S, a switch function $f : S \to S$ and an output function $o : S \to A$, we can associate with this stream a coalgebra $c : S \to A \times S$ which acts on states as follows:

$$c(s) = (o(s), f(s)).$$

From this coalgebraic representation of our stream we can retrieve the previous one just by composing c with the respective projections of the product $A \times S$. Note how we have used the constant functor A to model output.

Example 2.2.4 (Binary trees)**.** In the case of binary trees (Example 2.1.2), it is apparent that we can form a function

$$\langle l, r \rangle : B \to N \times N,$$

which is, however, not in the form we need. There can be nodes in N which do not have any child. This is exactly an example of situation where different inner states of a system can lead to different successor structures. We can therefore use a sum of two different functors to model this situation. Let $1 = \{*\}$ be a one element set. Then we can use the functor $T = \mathsf{Id} \times \mathsf{Id} + 1$ to model coalgebras of binary trees. Define $c : N \to N \times N + 1$ as follows:

$$c(n) = \begin{cases} \langle l, r \rangle(n) & \text{if } n \in B \\ * & \text{otherwise.} \end{cases}$$

The fact that $c(n) = *$ just means that $n \notin B$.

Example 2.2.5 (Deterministic automata)**.** To capture the deterministic automata from the Example 2.1.3 coalgebraically, fix the finite input set I and the set $2 = \{0, 1\}$. We then claim that coalgebras for the functor $T = 2 \times \mathsf{Id}^I$ contain the information needed to reconstruct the standard definition of a deterministic automaton (excluding only the start state). Indeed, a coalgebra $c : S \to 2 \times S^I$ says for each state whether it is or is not a finish state, and gives a future state for each input from I.

Example 2.2.6 (Kripke frames)**.** In the case of Kripke frames, we have to transform the accessibility relation R to the form of a function. It is enough to observe that for each state $s \in S$ we can form a subset $S' = \{s' \in S \mid R(s, s')\}$. Kripke frames are therefore coalgebras for the powerset functor: $c : S \to \mathcal{P}S$.

These examples hopefully illustrate the versatility of coalgebras.

Generalisation of Kripke frames We are now going to introduce one additional functor, which will allow us to generalise Kripke frames to have a many-valued accessibility relation.

Example 2.2.7 ($\mathscr{V}$-powerset functor)**.** We are generalising the powerset functor. It is fruitful to view subsets of a set A as characteristic functions $\mu_Z : A \to 2$. More succintly, we see that on objects there is an isomorphism $\mathcal{P}(A) \cong 2^A$. For a given quantale $\mathscr{V}$, there is therefore a reasonable generalisation of the object part of the powerset functor. Let us define the object mapping $\mathcal{P}_\mathcal{V}$ as follows:

$$A \mapsto V^A$$

The action of $\mathcal{P}_\mathcal{V}$ on morphisms can be defined to extend the powerset functor morphism action in the following manner: given a function $f : A \to B$, the function $\mathcal{P}f$ is a function with domain 2^A and codomain 2^B. Therefore, it transforms functions of the form $\mu_Z : A \to 2$ to functions $\mu_{f[Z]} : B \to 2$. More in detail, a subset $Z \subseteq A$ (represented by a characteristic function μ_Z) is mapped to the function

$$b \mapsto \sup_{\substack{a \in A \\ f(a)=b}} \mu_Z(a).$$

It is then straightforward to generalise this such that for a function $f : A \to B$ we get a function $\mathcal{P}_\mathcal{V} f : V^A \to V^B$. Again, a characteristic function $\mu_Z : A \to V$ is then just mapped by $\mathscr{V} f$ to a function which acts as follows:

$$b \mapsto \bigvee_{\substack{a \in A \\ f(a)=b}} \mu_Z(a).$$

Note that we have used only the lattice structure underlying the quantale $\mathscr{V}$. It is indeed possible to define the $\mathscr{V}$-powerset functor for any complete lattice, which need not be a quantale.

We can use the newly introduced functor to form an interesting class of coalgebras.

Example 2.2.8 ($\mathscr{V}$-Kripke frames)**.** The coalgebras for the $\mathcal{P}_\mathcal{V}$ functor can be thought of as generalised Kripke frames with a many-valued accessibility relation. Given a coalgebra $c : S \to \mathcal{P}_\mathcal{V}S$, it is possible to view it as a set of states S together with a $\mathscr{V}$-relation $R : S \nrightarrow S$, where $R(s, s') = c(s)(s')$.

2.3 Logics for coalgebras

We have shown various examples of systems that can be modeled by coalgebras. For some of these systems we know of logics that capture their behaviour. First we show examples of such logics that are in some sense "ad hoc": they do not arise from a general theory which would provide us expressive logics parametric in the choice of the functor.

Example 2.3.1 (Streams)**.** Consider coalgebras for the functor $A \times \mathsf{Id}$, streams with outputs from A. Then there is a simple logic which exactly distinguishes behaviour of streams, not their inner structure. Construct a language $\mathcal{L}$ given be the following Backus-Naur form

$$\varphi ::= a \mid \langle \mathsf{next} \rangle \varphi,$$

where a is an element from A. The semantics of the logic is the expected one. For a coalgebra $c : S \to A \times S$, we define the forcing relation $\Vdash_c : A \nrightarrow \mathcal{L}$ inductively:

$$s \Vdash_c a \text{ iff } p_1(c(s)) = a$$
$$s \Vdash_c \langle \mathsf{next} \rangle \varphi \text{ iff } p_2(c(s)) \Vdash_c \varphi.$$

We will now show simple examples that show in what sense the logic we introduced captures the behaviour of streams: it is adequate, which means it is preserved by behavioural equivalence, and it is expressive, which means it can distinguish behaviorally nonequivalent states.

Consider $c : S \to A \times S$, where $A = \{a_1, a_2\}$ and $S = \{s_1, s_2\}$, the coalgebraic structure defined as $c(s_1) = (a_1, s_2)$, $c(s_2) = (a_2, s_1)$. Then take a second coalgebra, $d : W \to A \times W$, with $W = \{w\}$ and $d(w) = (a_1, w)$. Clearly the behaviour of c, starting from the state s_1, gives us a sequence of outputs $(a_1, a_2, a_1, \dots)$, whereas the behaviour of d starting from w is a sequence $(a_1, a_1, \dots)$. We can find a formula that holds in s_1, but does not hold in w: It is true that

$$s_1 \Vdash_c \langle \mathsf{next} \rangle a_2,$$

but it is definitely not the case that $w \Vdash_d \langle \mathsf{next} \rangle a_2$.

On the other hand, suppose we have a third coalgebra, $e : V \to A \times V$, where $V = \{v_1, v_2\}$ and the function e is defined as $e(v_1) = (a_1, v_2)$ and $e(v_2) = (a_1, v_1)$. Then again the behaviour of e starting from v_1 is a sequence of outputs $(a_1, a_1, \dots)$. And by a simple inductive argument over the length of formulas, we see that for any formula φ it holds that

$$w \Vdash_d \varphi \text{ iff } v_1 \Vdash_e \varphi.$$

These two examples should be sufficient to show in what sense the logic we introduced captures behaviour of streams: it is adequate, which means it is preserved by behavioural equivalence, and it is expressive, which means it can distinguish behaviourally nonequivalent states.

Example 2.3.2 (Kripke frames)**.** For coalgebras of the form $c : S \to \mathcal{P}_\omega S$, it is known that the standard modal logic with modalities $\Box$ and $\Diamond$ is expressive. We refer the reader to [12] for an extensive introduction into the field of modal logic.

From the preceding examples it is apparent that for some kinds of coalgebras it is possible to construct logics with interesting properties. In the following sections, we are going to introduce techniques used to construct logics for coalgebras for a large class of functors.

2.4 Relation lifting

Relation lifting is an operation on relations which allows us to take any relation between two given sets and returns a relation between datatypes constructed on top of the given sets. We are first going to show some examples of liftings for simple functors (datatypes). Then we will show a general definition for arbitrary functors, show some more examples using the general definition, and state an important theorem regarding the functoriality of the lifting operation.

The reason we are interested in this operation is that it will allow us to introduce semantics for Moss' coalgebraic modality and it will give us a notion of bisimulation. All these aspects will be addressed in more detail later in the text.

In [34] there is a detailed study of the properties of relation lifting together with detailed references.

2.4.1 First examples

Example 2.4.1 (Arrays of length 2)**.** Let us have a (two-valued) relation $R : A \nrightarrow B$ and a functor $\mathsf{Id} \times \mathsf{Id}$. This functor can be thought of as a datytype that creates, for a given type A, a datatype of arrays of length 2 for that type: $A \times A$. Is there a natural way to define a relation $A \times A \to B \times B$ given the relation R. We could then say that we can *lift* the relation R to live over arrays of length 2. It seems most natural to define the lifted relation denoted by $\overline{R}$ in the following way:

$$\overline{R}((a_1, a_2), (b_1, b_2)) \text{ if and only if } R(a_1, b_1) \text{ and } R(a_2, b_2).$$

This is what we would expect to be the result of relation lifting when applied to this specific example.

Moreover, this approach can be generalised to arrays of arbitrary fixed length, with the definition only slightly altered.

Example 2.4.2 (Lists)**.** Now suppose we do not want to work just with arrays of length 2, but with lists of different lengths. Let us first examine the simpler case of the functor Id^n for a given n. Taking again the relation $R : A \nrightarrow B$, let us define $\overline{R}$ as follows:

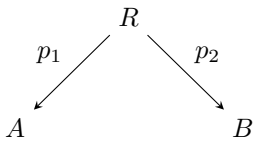
$$\overline{R}((a_i)_{i<n}, (b_i)_{i<n}) \text{ if and only if for all } i < n : R(a_i, b_i).$$

We will try to generalise this further and try to lift the relation R with the functor List. Intuitively, two lists should be in relation $\overline{R}$ if they have the same length and their corresponding elements are linked by relation R. Concretely, let us have two lists l_1 and l_2 of lengths m and n, so $l_1 = (a_i)_{i<m}$ and $l_2 = (b_i)_{i<n}$. We now say that $\overline{R}(l_1, l_2)$ if and only if $m = n$ and for every $i < m$ it holds that $R(a_i, b_i)$. Again, this seems to be the right way relation lifting should work in this specific case.

2.4.2 Relation lifting in general

In the previous two examples, we were given a relation over two sets, we constructed some datatypes over these two sets, and then we wanted to find out a suitable relation, which would relate the instances of these datatypes. We know that datatypes can be modeled by functors. Is there now a general definition, which would tell us how to lift any relation to work over any functor (datatype)? We are going to show now a span representation of relations, which will immediately allow us to define relation lifting for any functor. Relation lifting, defined via spans, first appeared in [7].

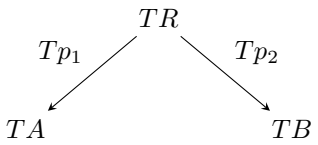
Definition 2.4.3 (Relations as spans)**.** If we have a relation $R : A \nrightarrow B$, we say that its *span representation* is a diagram



where R is the set of ordered pairs from $A \times B$ which are related, and the morphisms p_1 and p_2 are projections to A and B respectively.

See that p_1 and p_2 in the previous definition are just the projection morphisms of the product $A \times B$ precomposed with the inclusion morphism of R into $A \times B$.

Definition 2.4.4 (Relation lifting)**.** For a relation R given by the span representation as above and for a functor T, we can get the following diagram



by applying the functor T to the span representation. We say that two elements $\alpha \in TA$ and $\beta \in TB$ are in relation $\overline{T}R$ if there is a witness $\gamma \in TR$ such that it projects on α and β via Tp_1 and Tp_2, respectively. More explicitly, we say that the *relation lifting* $\overline{T}R$ of the relation R is defined as follows:

$$\overline{T}R = \{(\alpha, \beta) \mid (\exists \gamma \in TR) : Tp_1(\gamma) = \alpha \text{ and } Tp_2(\gamma) = \beta\}.$$

This definition generalises the preceding examples and allows us to compute some more involved examples as well.

Example 2.4.5 (Stream lifting)**.** Given a relation $R : A \nrightarrow B$ and a functor $C \times \mathsf{Id}$, the lifted relation $\overline{C \times \mathsf{Id}}(R) : C \times A \nrightarrow C \times B$ is defined this way:

$$\overline{C \times \mathsf{Id}}(R)((c_1, a)(c_2, b)) \quad \text{iff} \quad c_1 = c_2 \text{ and } R(a, b).$$

Example 2.4.6 (Lifting for polynomial functors)**.** Relation lifting for polynomial functors can be defined inductively by the following rules:

- For the constant functor C: $\overline{C}(R) = \mathsf{id}_C$.
- For the identity functor Id: $\mathsf{Id}(R) = R$.
- For the sum of functors $T + T'$:

$$\overline{T + T'}(R) = \overline{T}(R) \uplus \overline{T'}(R).$$

- For the product of functors $T \times T'$:

$$\overline{T \times T'}(R) = \{((\alpha, \alpha'), (\beta, \beta')) \mid \overline{T}(R)(\alpha, \beta) \text{ and } \overline{T}(R')(\alpha', \beta')\}.$$

Example 2.4.7 (Powerset lifting)**.** Suppose we work with the powerset functor $\mathcal{P}$ and want to find out how does the relation lifting $\overline{\mathcal{P}}R$ of the relation $R : A \nrightarrow B$ look like. Unraveling the definition, we see that two subsets $\alpha \subseteq A$ and $\beta \subseteq B$ are in the relation $\overline{\mathcal{P}}(R)(\alpha, \beta)$ exactly when they satisfy the Egli-Milner covering property, which means that the following two conditions must hold:

$$\forall a \in \alpha \; \exists b \in \beta : R(a, b)$$
$$\forall b \in \beta \; \exists a \in \alpha : R(a, b).$$

Example 2.4.8 (Finitary powerset lifting)**.** For the finitary powerset functor $\mathcal{P}_\omega$ the relation lifting is exactly of the same form as for the powerset functor $\mathcal{P}$. This is another general property of the lifting, see [34].

Example 2.4.9 ($\mathscr{V}$-powerset lifting)**.** We would now like to compute relation lifting for the $\mathcal{P}_\mathcal{V}$ functor.

Remembering the definition we know that for $\alpha \in \mathcal{P}_\mathcal{V}(A)$, $\beta \in \mathcal{P}_\mathcal{V}(B)$ we have

$$\overline{\mathcal{P}_\mathcal{V}}(R)(\alpha, \beta) \quad \text{iff} \quad (\exists w \in \mathcal{P}_\mathcal{V}(R)) : ((\mathcal{P}_\mathcal{V}(p_1)(w) = \alpha) \text{ and } (\mathcal{P}_\mathcal{V}(p_2)(w) = \beta)).$$

In the case of our functor $\mathcal{P}_\mathcal{V}$, we get that $\overline{\mathcal{P}_\mathcal{V}}(R)(\alpha, \beta)$ iff the following two conditions hold:

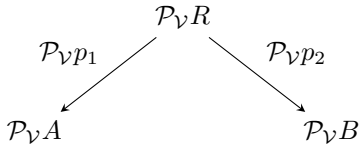
$$\forall a \in A : \bigvee_{R(a,b)} \beta(b) = \alpha(a),$$
$$\forall b \in B : \bigvee_{R(a,b)} \alpha(a) = \beta(b).$$

This follows directly from how the $\mathcal{P}_\mathcal{V}$ functor is defined to act on morphisms.

In the case that the underlying lattice of $\mathscr{V}$ is a chain and α, β are functions with finite support (meaning that $\alpha(a) > \bot$ holds only for a finite number of elements $a \in A$), these two conditions change into much simpler requirements. It holds that $\overline{\mathcal{P}_\mathcal{V}}(R)(\alpha, \beta)$ if and only if both of the following conditions ar esatisfied:

$$\forall a \in A, \alpha(a) > \bot : \; \exists b \in B : (R(a, b) \wedge (\alpha(a) \leq \beta(b))),$$
$$\forall b \in B, \beta(b) > \bot : \; \exists a \in A : (R(a, b) \wedge (\beta(b) \leq \alpha(a))).$$

Why should this be true? We can help ourselves by looking again at the diagram used for defining relation liftings:



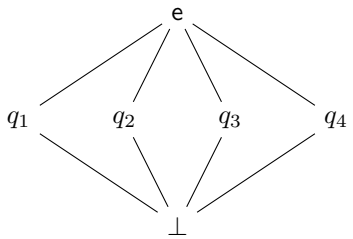
If for some $a \in A$ we know that $\alpha(a) = x > \bot$, it must be the case that there is a witness $w \in \mathcal{P}_\mathcal{V}(R)$ such that $w(a, b) = x$ for some $b \in B$ which is moreover related to a (it holds that $R(a, b)$). By definition of $\mathcal{P}_\mathcal{V} p_2$, it is now apparent that $\beta(b) \geq x$ and therefore $\alpha(a) \leq \beta(b)$. The second requirement can be obtained similarly. We have therefore proven that if the two elements α and β are in the lifted relation $\overline{\mathcal{P}_\mathcal{V}}(R)$, the lifting has the two properties listed above. In the other direction, we have to prove that if α and β satisfy these two properties, then they are in the lifted relation $\overline{\mathcal{P}_\mathcal{V}}(R)$. We just find an appropriate witness $w \in \mathcal{P}_\mathcal{V}(R)$. Set $w(a, b) = \alpha(a) \wedge \beta(b)$ for each pair $a \in A$, $b \in B$. The two properties assure us that $\mathcal{P}_\mathcal{V}(p_1)(w) = \alpha$ and $\mathcal{P}_\mathcal{V}(p_2)(w) = \beta$.

The finite support condition reminds us of the finite powerset functor: the elements α and β can be thought of as generalised subsets, "fuzzy" subsets of A or B. If α has finite support, it just means that α is a "finite fuzzy subset" of A, generalising the notion of finite subsets when working with the $\mathcal{P}_\omega$ functor.

Note 2.4.10. All Set endofunctors we have introduced so far preserve weak pullbacks, with the exception of the $\mathcal{P}_\mathcal{V}$ functor. The weak pullback preservation property depends on the choice of $\mathcal{V}$. In the case that the underlying lattice of the quantale $\mathcal{V}$ is itself a complete Heyting algebra, $\mathcal{P}_\mathcal{V}$ as a functor preserves weak pullbacks. We are going to prove this fact below. The property of weak pullback preservation is important with regard to relation lifting, as we will see in the next subsection.

Lemma 2.4.11 (Weak pullbacks and $\mathcal{P}_\mathcal{V}$)**.** *There exists a quantale $\mathcal{V}$ for which the functor $\mathcal{P}_\mathcal{V}$ does not preserve weak pullbacks.*

Proof. Take the 6-element quantale with a 4-element antichain (a subset of elements of V such that each pair from this subset is incomparable) and name the incomparable elements q_1, q_2, q_3, q_4.



Then let $A = \{a_1, a_2\}$, $B = \{b_1, b_2\}$, $C = \{c\}$ and consider the unique two functions $f : A \to C$ and $g : B \to C$. The pullback of f and g coincides with the product $A \times B$. Now for two elements $\alpha \in \mathcal{P}_\mathcal{V}(A)$, $\beta \in \mathcal{P}_\mathcal{V}(B)$ such that $\alpha(a_1) = q_1$, $\alpha(a_2) = q_2$ and $\beta(b_1) = q_3$, $\beta(b_2) = q_4$ we see that they get mapped to the same element by the maps $\mathcal{P}_\mathcal{V}(f)$ and $\mathcal{P}_\mathcal{V}(g)$. If $\mathcal{P}_\mathcal{V}$ preserved weak pullbacks, we would be able to find a witness $w \in \mathcal{P}_\mathcal{V}(A \times B)$ which would

project to α and β, or, in other words $\mathcal{P}_\mathcal{V}(p_1)(w) = \alpha$ and $\mathcal{P}_\mathcal{V}(p_2)(w) = \beta$. However, this is not possible, as there is no such function $w : A \times B \to V$, as can be easily checked (for example by trying all possible functions from $\mathcal{P}_\mathcal{V}(A \times B)$). □

Note that the underlying lattice was chosen not to be distributive, otherwise we could not construct such a counterexample. In fact, we shall prove that distributive lattices (or complete Heyting algebras for lattices with infinitely many elements) allow us to construct the witness function w.

Lemma 2.4.12 ($\mathcal{P}_\mathcal{V}$ preserves weak pullbacks for distributive $\mathcal{V}$)**.** *For each quantale $\mathcal{V}$ whose underlying lattice is a complete Heyting algebra, the associated functor $\mathcal{P}_\mathcal{V}$ does preserve weak pullbacks.*

Proof. Let us form a pullback (P, p_1, p_2) of two functions $f : A \to C$ and $g : B \to C$. The functor $\mathcal{P}_\mathcal{V}$ preserves weak pullbacks if and only if for each $\alpha \in \mathcal{P}_\mathcal{V}(A)$ and $\beta \in \mathcal{P}_\mathcal{V}(B)$ such that $\mathcal{P}_\mathcal{V}(f)(\alpha) = \mathcal{P}_\mathcal{V}(g)(\beta)$ there is a witness $w \in \mathcal{P}_\mathcal{V}P$ such that $\mathcal{P}_\mathcal{V}(p_1)(w) = \alpha$ and $\mathcal{P}_\mathcal{V}(p_2)(w) = \beta$.

The fact that $\mathcal{P}_\mathcal{V}(f)(\alpha) = \mathcal{P}_\mathcal{V}(g)(\beta)$ implies that for every $c \in C$ it holds that

$$\bigvee_{f(a)=c} \alpha(a) = \bigvee_{g(b)=c} \beta(b).$$

The witnessing function $w \in \mathcal{P}_\mathcal{V}(P)$ has to be of the form $w : P \to V$, therefore it assigns a value $v \in V$ to pairs (a, b) such that $f(a) = g(b)$. To say that $\mathcal{P}_\mathcal{V}(p_1)(w) = \alpha$ and $\mathcal{P}_\mathcal{V}(p_2)(w) = \beta$ is to say that for every $a \in A$

$$\bigvee_{\{b|(a,b)\in P\}} w(a, b) = \alpha(a)$$

holds, and for every $b \in B$

$$\bigvee_{\{a|(a,b)\in P\}} w(a, b) = \beta(b).$$

Let us define w as follows:

$$(a, b) \in P \mapsto \alpha(a) \wedge \beta(b).$$

Then we see that for each $a \in A$ it holds that

$$\bigvee_{\{b|(a,b)\in P\}} w(a, b) = \bigvee_{\{b|(a,b)\in P\}} (\alpha(a) \wedge \beta(b)) = \alpha(a) \wedge \bigvee_{\{b|(a,b)\in P\}} \beta(b),$$

and moreover we can infer that

$$\bigvee_{\{b|(a,b)\in P\}} \beta(b) = \bigvee_{g(b)=f(a)} \beta(b) = \bigvee_{f(a')=f(a)} \alpha(a') \leq \alpha(a),$$

which proves that indeed for every $a \in A$

$$\bigvee_{\{b|(a,b)\in P\}} w(a, b) = \alpha(a).$$

The second equality, namely that

$$\bigvee_{\{a|(a,b)\in P\}} w(a,b) = \beta(b)$$

holds for every $b \in B$ is proved in the same way. Therefore the functor $\mathcal{P}_{\mathcal{V}}$ preserves weak pullbacks. □

2.4.3 Properties of relation lifting

Relation lifting has many interesting and useful properties, some of which we are going to discuss here (without proofs). We mostly refer to [34] for the interested reader.

If we deal with a weak pullback preserving functor T (see Definition 1.1.32), it can be shown that the relation lifting can be used to define a functor $\overline{T}$ on the category Rel of sets and relations, and much more.

Theorem 2.4.13 (Functoriality of the lifting — [52])**.** *A set functor T preserves weak pullbacks if and only if there exists a 2-functor $\overline{T} : \mathsf{Rel} \to \mathsf{Rel}$ such that the following square commutes:*

$$\begin{array}{ccc} \mathsf{Rel} & \xrightarrow{\overline{T}} & \mathsf{Rel} \\ {\scriptstyle \mathsf{gr}}\uparrow & & \uparrow{\scriptstyle \mathsf{gr}} \\ \mathsf{Set} & \xrightarrow{T} & \mathsf{Set} \end{array}$$

Weak pullback preservation is necessary to keep $\overline{T}$ preserve the composition of relations.

Theorem 2.4.13 informs us about the following four properties of relation lifting for weak pullback preserving functor T:

- $\overline{T}$ preserves the identity: $\overline{T}(\mathsf{id}_A) = \mathsf{id}_{TA}$.
- $\overline{T}$ preserves composition: $\overline{T}(S \circ R) = \overline{T}(S) \circ \overline{T}(R)$.
- $\overline{T}$ is locally monotone: $R \leq S$ implies that $\overline{T}(R) \leq \overline{T}(S)$.
- $\overline{T}$ is an extension of T: $\overline{T}f = Tf$.

Relation lifting moreover has the following properties:

- $\overline{T}$ commutes with opposites: $\overline{T}(R^{op}) = (\overline{T}R)^{op}$.
- $\overline{T}$ commutes with restrictions: $\overline{T}(R \restriction_{A' \nrightarrow B'}) = \overline{T}(R) \restriction_{TA' \nrightarrow TB'}$

The properties we have introduced are going to be useful later in the text. We do not prove them here, we however prove analogous properties of the many-valued relation lifting later in Chapter 3. We refer the reader to [34] for a broader overview of relation lifting.

2.5 Finitary functors

Now we are going to introduce an important property of certain functors, which we will further require of every functor in later text for technical reasons.

Definition 2.5.1. We say that a set functor T is finitary if and only if for every set X it holds that

$$TX = \bigcup\{TY \mid Y \subseteq X, Y \text{ finite}\}.$$

This definition can help us when dealing with infinite sets. Suppose we have an infinite set X and take an element $\alpha \in TX$. Now we know that there is a finite set Y such that $\alpha \in TY$. This way we can easily forget that we were dealing with a large set and we can turn our attention to finite sets, which helps immensely in many cases.

Example 2.5.2 (Lists)**.** The List functor is a nice example of a finitary functor. Why is this functor finitary? We need to ask whether for any $\alpha \in \mathsf{List}(A)$ there is a finite subset $Z \subseteq A$ for which α is in $\mathsf{List}(Z)$. We know that α is a finite list of elements from A, say $(a_1, \ldots, a_n)$, with possible repetitions. Since the list is finite, there are only finitely many different elements of A, and therefore we can take the finite set $\{a_1, \ldots, a_n\}$ of these elements and denote the set by Z. Clearly, the list α is contained in the set $\mathsf{List}(Z)$ of all lists generated by Z, because every element of α is contained in the set Z.

The main message of this example should be that even when we are dealing with infinite alphabets, the actual words created from this alphabet have just a finite number of symbols, and therefore every single word can be written using a finite alphabet.

Example 2.5.3 (Finitary powerset)**.** The finitary powerset functor, denoted by $\mathcal{P}_\omega$, is a finitary functor. This functor assigns to a set A the set $\mathcal{P}_\omega A$ of all finite subsets of A. Similarly to the previous example, we will look at an arbitrary element $Z \in \mathcal{P}_\omega A$, so Z is a finite subset of A. Our task is to find some finite set B such that Z belongs to $\mathcal{P}_\omega B$, or, in other words, $Z \subseteq B$. See that we can always do this: just take the set B to be Z itself. We know that Z is finite and that $Z \in \mathcal{P}_\omega Z$, which is exactly what we needed.

The message is again that a finite subset Z of a possibly infinite set A can always be found by looking at subsets of an appropriate finite subset of A.

Example 2.5.4 (Counterexample — unbounded powerset)**.** Now we are going to look at an example of a functor that is *not* finitary. The unbounded powerset functor $\mathcal{P}$ is not finitary: suppose we take the set $\mathbb{N}$ of natural numbers and construct the set $\mathcal{P}\mathbb{N}$. The set $\mathbb{N}$ itself is an infinite set that is an element of $\mathcal{P}\mathbb{N}$, and there is no finite subset $M \subseteq \mathbb{N}$ such that $\mathbb{N}$ would be in $\mathcal{P}M$. This follows from the fact that any subset of a finite set is finite.

We have seen that not all functors are finitary. But there is a way to make a finitary modification of any functor.

Definition 2.5.5. For any functor T we define its finitary modification T_ω in the following way:

$$T_\omega A = \bigcup\{TZ \mid Z \subseteq A, Z \text{ finite}\},$$
$$T_\omega f = Tf.$$

See that T_ω indeed is finitary by definition. This allows us to rephrase the definition of a finitary functor: a functor T is finitary if and only if $T = T_\omega$.

Warning It is important to notice that finitarity of a functor T *does not* mean that the sets TX are finite for any X. It is even false to say that given a finite set X, the set TX would be finite. An easy counterexample is the set LX of words over the alphabet $X = \{a, b\}$: the alphabet is finite and still there is an infinite number of words over this alphabet.

Example 2.5.6 (Finitary $\mathcal{V}$-powerset functor)**.** The finitary modification of the $\mathcal{V}$-powerset functor behaves the same as the functor $\mathcal{P}_\mathcal{V}$ with a slight difference in the object part of the functor. For any set A, the object $\mathcal{P}_{\mathcal{V}_\omega}(A)$ consists of all functions $\alpha : A \to V$ which have finite support. That means that $\alpha \in \mathcal{P}_{\mathcal{V}_\omega}(A)$ exactly if cardinality of the set $\{a \mid \alpha(a) > \bot\}$ is finite.

Note 2.5.7. In the rest of the text, whenever we are going to talk about the $\mathcal{P}_\mathcal{V}$ functor, we will have its finitary modification in mind.

The discussion in this section leads us to the notion of *base*: we want to formalise and generalise the observation that each list $l \in \mathsf{List}(A)$, generated possibly from an infinite alphabet A, can be generated from a certain minimal alphabet A', which we will denote $\mathsf{Base}(l)$.

Definition 2.5.8 (Base)**.** For a finitary weak pullback preserving functor T we define a natural transformation

$$\mathsf{Base} : T \to \mathcal{P}_\omega$$

as follows: for a set A and an element $\alpha \in TA$,

$$\mathsf{Base}_A(\alpha) = \bigcap \{A' \subseteq A \mid \alpha \in TA'\}.$$

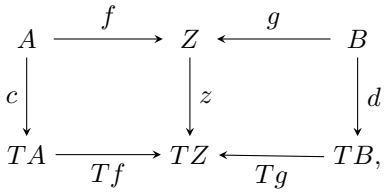
Proposition 2.5.9. *The transformation from Definition 2.5.8 is indeed well-defined and natural. For an element $\alpha \in TA$, the set $Z = \mathsf{Base}_A(\alpha)$ is the minimal finite set such that $\alpha \in TZ$.*

For more information about bases, see [34].

2.6 Bisimulation

Having introduced the notion of relation lifting, we can use it for defining one more core concept which is very useful, namely bisimulation. Bisimulation is closely related to the notion of behavioural equivalence, which we have informally introduced when talking about streams in Example 2.3.1. The notion of bisimulation is thoroughly dealt with in [50], we are only going to introduce the basic definitions.

Definition 2.6.1 (Behavioural equivalence ([37]))**.** Given two coalgebras $c : A \to TA$ and $d : B \to TB$, we say that two states $a \in A$ and $b \in B$ are *behaviourally equivalent* if there is a coalgebra $z : Z \to TZ$ and two coalgebraic morphisms $f : A \to Z$ and $g : B \to Z$ (as can be seen on the following commutative diagram)

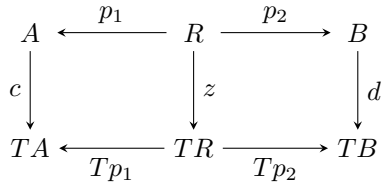


where moreover $f(a) = g(b)$ holds.

Definition 2.6.2 (Bisimulation ([49]))**.** We say that a relation $R : A \nrightarrow B$ is a *bisimulation* between coalgebras $c : A \to TA$ and $d : B \to TB$ if the following implication holds for every a, b:

$$R(a, b) \Rightarrow \overline{T}(R)(c(a), d(b)).$$

Equivalently, we say that R is a bisimulation if we can find a coalgebra $z : R \to TR$ such that the projections become coalgebraic morphisms, that is, the following diagram.



commutes. Two states a, b are said to be *bisimilar* if they are related by some bisimulation R.

2.7 Moss' logic

Now we have all the needed definitions to introduce logics for coalgebras for a wide range of functors, parametric in the choice of the functor. If someone gives us a weak pullback preserving functor T, we can form an expressive logic for T-coalgebras. We refer here to the paper [45] by Larry Moss which introduced the ideas presented here.

The logic is modal in nature: we shall introduce a modality ∇, which will be used to describe how the successor structure looks like in some state.

In the definitions that follow, let us fix a finitary weak pullback preserving functor T.

Definition 2.7.1 (Syntax)**.** We construct the language $\mathcal{L}$ of the coalgebraic Moss' logic inductively:

$$\begin{array}{lcl} \mathcal{L}_0 & \ni & \varphi ::= \bot \mid \neg\varphi \mid \varphi \wedge \varphi \mid \varphi \vee \varphi \\ \mathcal{L}_{T_i} & = & T(\mathcal{L}_i) \\ \mathcal{L}_{i+1} : & \ni & \varphi ::= \bot \mid \neg\varphi \mid \varphi \wedge \varphi \mid \varphi \vee \varphi \mid \nabla\alpha, \quad \text{where } \alpha \in \mathcal{L}_{T_i} \\ \mathcal{L} & = & \bigcup_i \mathcal{L}_i \end{array}$$

Observe that the construction of $\mathcal{L}$ proceeds by layers: we have the standard boolean part of the language with the $\bot$ element, negation, disjunction and

conjunction, then we add the modal part of the language and repeat. To see this construction in more detail, we point to the paper [34], Section 5.1.

Now in the semantic part we are going to use relation lifting to give meaning to modal formulas.

Definition 2.7.2 (Semantics)**.** For a coalgebra $c : A \to TA$, define the forcing relation $\Vdash_c$: $A \nrightarrow \mathcal{L}$ inductively as follows:

$$\begin{array}{lll} a \nVdash_c \bot, & & \\ a \Vdash_c \neg\varphi & \text{iff} & a \nVdash_c \varphi, \\ a \Vdash_c \varphi \wedge \psi & \text{iff} & a \Vdash_c \varphi \text{ and } a \Vdash_c \psi, \\ a \Vdash_c \varphi \vee \psi & \text{iff} & a \Vdash_c \varphi \text{ or } a \Vdash_c \psi, \\ a \Vdash_c \nabla\alpha & \text{iff} & \overline{T}(\Vdash_c)(c(a), \alpha). \end{array}$$

Well-definedness of this definition of semantics for our logic is discussed in Section 5.2 of [34] and it follows from the fact that relation lifting commutes with restrictions (Definition 1.2.4).

Theorem 2.7.3 (Expressivity)**.** *The introduced language is expressive with regard to behavioural equivalence. That is, for two coalgebras $c : A \to TA$ and $d : B \to TB$ two states a, b are behaviourally equivalent if and only if they satisfy the same formulas: for every $\varphi \in \mathcal{L}$ it holds that $a \Vdash_c \varphi$ if and only if $b \Vdash_d \varphi$.*

This theorem is a finitary variation of results in [45]: the original result allowed for using a non-finitary functor T, but to save the expressivity of the language, it is necessary to introduce infinitary connectives.

Now we are going to show examples of coalgebraic logics for streams, Kripke frames and $\mathscr{V}$-Kripke frames, especially the semantics of the ∇ modality. We are interested in coalgebras for the finitary modifications of the respective functors to get an expressive logic.

Example 2.7.4 (Streams)**.** In the case of streams for the functor $Z \times \mathsf{Id}$, we see that the syntax of the logic is

$$\varphi ::= \bot \mid \neg\varphi \mid \varphi \wedge \varphi \mid \varphi \vee \varphi \mid \nabla(z, \varphi),$$

where $z \in C$. For a coalgebra $c : A \to Z \times A$ and a state $a \in A$, we shall look at the meaning of the modal formula $\nabla(z, \varphi)$. Let $c(a) = (z', a')$. Then

$$a \Vdash_c \nabla(z, \varphi) \text{ iff } z = z' \text{ and } a' \Vdash_c \varphi.$$

Even though the logic introduced here and the logic from Example 2.3.1 are not the same, the logic introduced here has enough expressive power to distinguish two states that are not behaviourally equivalent.

Example 2.7.5 (Kripke frames)**.** In the case of finitely branching Kripke frames ($\mathcal{P}_\omega$-coalgebras), we see that the syntax of the logic is the same as in the case of streams, only the modality is different. We will therefore look at the syntax and semantics of this modality. To form a modal formula $\nabla\alpha$, we have to take a finite subset $\alpha \in \mathcal{P}_\omega(\mathcal{L})$ of formulas. Therefore $\alpha = \{\varphi_1, \ldots, \varphi_n\}$ for some $n \in \mathbb{N}$. Having a coalgebra $c : A \to \mathcal{P}_\omega(A)$, we get that $a \Vdash_c \nabla\alpha$ if and only if the following two properties hold:

$$\begin{array}{llll} \forall \varphi \in \alpha & \exists a' \in c(a) : & & a' \Vdash_c \varphi \\ \forall a' \in c(a) & \exists \varphi \in \alpha : & & a' \Vdash_c \varphi. \end{array}$$

This semantics copies the structure of the relation lifting for the $\mathcal{P}_\omega$ functor. But it is important to notice that this interesting modality can be defined by the standard modalities, $\Box$ and $\Diamond$ (where $\Diamond\alpha = \{\Diamond\varphi \mid \varphi \in \alpha\}$):

$$\nabla(\alpha) \equiv \Box \bigvee \alpha \wedge \bigwedge \Diamond\alpha.$$

The modality ∇ is actually interdefinable with $\Box$ and $\Diamond$, see [34].

Example 2.7.6 ($\mathscr{V}$-Kripke frames)**.** Let us fix the quantale $\mathscr{V}$ to have as its underlying lattice the real interval $[0;1]$ with $e = \top = 1$ and $\vee = \sup$. Then the syntax of the ∇ modality looks like as follows: forming a formula $\nabla\alpha$, we take α from the set $\mathcal{P}_\mathcal{V}(\mathcal{L})$, which ammounts to choosing a finite number of formulas $\varphi_i \in \mathcal{L}$, and for each of these formulas associating a "weight", a number $\alpha(\varphi_i)$ from the interval $[0;1]$. More succintly, α may be thought of as a "fuzzy set" with finite support, taking as elements formulas from $\mathcal{L}$.

The semantics of the ∇ modality again follows easily from the relation lifting for the functor $\mathcal{P}_\mathcal{V}$. Fix a coalgebra $c : A \to \mathcal{P}_\mathcal{V}(A)$. We ask ourselves when $a \Vdash_c \nabla\alpha$ holds. This holds precisely when the following two properties hold:

$$\forall a' \in c(a), c(a)(a') > 0 : \ \exists \varphi \in \alpha : (a' \Vdash_c \varphi \wedge (c(a)(a') \leq \alpha(\varphi))),$$
$$\forall \varphi \in \alpha, \alpha(\varphi) > 0 : \ \exists a' \in c(a)(a') : (a' \Vdash_c \varphi \wedge (\alpha(\varphi) \leq c(a)(a'))).$$

These two requirements may seem complicated at first sight. We can understand them as straightforward generalisations of the standard lifting for the powerset functor. For a state a, think of $c(a)$ as of a fuzzy set of neighbors, and for $a' \in A$ of $c(a)(a')$ as of how close a is to a'. Then to say that $a \Vdash_c \nabla\alpha$ is to say that for every neighbor a' whose closeness $c(a)(a')$ to a is nonzero we have to have a formula φ for which $a' \Vdash_c \varphi$ and moreover the weight of φ in α is larger than closeness of a', and vice versa: for any formula φ with nonzero weight in α we have to find a neighbor a' of a which satisfies φ and its closeness is higher than the weight of φ.

Note 2.7.7 (Atomic propositions)**.** The logic we have introduced does not allow using atomic propositions, as we are used to them for example from standard propositional logic. This is the case because atomic propositions and their evaluation in each state of some T-coalgebra can be encoded functorially. What do we mean by that?

Let us have a coalgebra $c : A \to TA$, a set of atomic propositions At and for each state $a \in A$ an evaluation function $\mathsf{ev}_a : \mathsf{At} \to 2$. This corresponds to having a function $\mathsf{ev} : A \to 2^{\mathsf{At}}$. We can therefore say that we have a function $\langle c, \mathsf{ev}\rangle : A \to TA \times 2^{\mathsf{At}}$, which is in fact a coalgebra for the functor $T \times 2^{\mathsf{At}}$.

Chapter 3

Many-valued relation lifting

So far we have talked about coalgebras, which are used to model various systems, and about relation lifting, which we then used to define bisimulation and semantics of coalgebraic Moss' logic. We are now going to continue by introducing many-valued relation lifting, which we will then use in a similar manner to define many-valued variant of Moss' coalgebraic logic and to study the notion of many-valued bisimulation which stems from this approach.

To find out how to lift many-valued relations, we shall first look at several examples. In contrast with the definition of standard relation lifting, we will have to follow a different approach than that of using spans to define many-valued relation lifting, since no span representation is available for many-valued relations. We will first introduce many-valued relation lifting for polynomial functors via operations on many-valued relations. In order to extend this lifting on a wider class of functors, we will introduce functor presentations and show how can standard relation lifting be represented in this setting. When defining many-valued relation lifting in its full generality, we will restrict ourselves to complete Heyting algebras for technical reasons.

One possible approach to many-valued liftings is discussed in the papers [11] and [9]. These liftings stem from a different base category than Set and use enriched category very heavily. We are going to take a different approach and restrict ourselves to the category of sets. In [19] the authors study liftings of monads and lax monads to the category of many-valued relations. To our best knowledge, the approach we have chosen in this chapter has not appeared anywhere in the literature.

3.1 Intuitive examples

In this section we will reconsider the examples from subsection 2.4.1 and try to find an intuitive notion of many-valued relation lifting for them. Let us fix the quantale $\mathscr{V}$ for the rest of this section.

Example 3.1.1 (Arrays of length 2)**.** We now take a V-valued $\mathscr{V}$-relation $R : A \nrightarrow B$ and the functor $\mathsf{Id} \times \mathsf{Id}$ making the datatype of arrays of length 2. We will try to define a $\mathscr{V}$-relation $A \times A \nrightarrow B \times B$, given the relation R. Taking inspiration from Example 2.4.1, we define the lifted relation denoted by $\widehat{R}$ in

the following way:

$$\widehat{R}((a_1, a_2), (b_1, b_2)) = R(a_1, b_1) \otimes R(a_2, b_2).$$

We will again generalise the previous example to work with arrays of arbitrary length, or with the List functor.

Example 3.1.2 (Lists)**.** We now take the functor Id^n for a given n. This corresponds to a datatype of arrays of size n. If we want to lift the $\mathscr{V}$-relation $R : A \nrightarrow B$ to a $\mathscr{V}$-relation $\widehat{R} : A^n \nrightarrow B^n$, we proceed as follows:

$$\widehat{R}((a_i)_{i<n}, (b_i)_{i<n}) = \bigotimes_{i<n} R(a_i, b_i).$$

We again generalise this further and lift the $\mathscr{V}$-relation R with the functor List. As in the standard two-valued relation lifting, we want to relate two lists only if they have the same length, and then the extent to which the lists are related is computed by taking the relatedness of corresponding elements and using the tensor. Concretely, having two lists $l_1 = (a_i)_{i<m}$ and $l_2 = (b_i)_{i<n}$ of lengths m and n, we say that $\widehat{R}(l_1, l_2) = \bot$ if $m \neq n$ and otherwise

$$\widehat{R}(l_1, l_2) = \bigotimes_{i<n} R(a_i, b_i).$$

This seems to be a very natural generalisation of the standard relation lifting for lists.

3.2 Towards a general $\mathscr{V}$-lifting

In the previous section we have seen some examples which could serve as a "sanity check" for the general definition of many-valued relation lifting. Now we are going to focus our attention to finitary weak pullback preserving functors, which will be most important for the examples of coalgebras we are interested in. We will use the commutative diagram from Theorem 2.4.13 as a guidance of what the definition of many-valued relation lifting should satisfy. For an endofunctor T of Set, we shall try to find a suitable functor $\widehat{T}$ on the category of $\mathscr{V}$-matrices, such that it is moreover locally monotone (i.e. it is a 2-functor) and extends T. Moreover we would like it to be an extension of the standard relation lifting, so that it gives the usual results when applied to $\mathscr{V}$-matrices where $\mathscr{V}$ is the two-element boolean algebra (and the matrices coincide with relations).

Definition 3.2.1 (Many-valued relation lifting)**.** For a finitary Set endofunctor T preserving weak pullbacks, we say that a 2-functor $\widehat{T}$ on the category of many-valued matrices $\mathscr{V}$-mat is a $\mathscr{V}$-lifting of T, if and only if the following diagram commutes:

$$\begin{array}{ccc} \mathscr{V}\text{-mat} & \xrightarrow{\widehat{T}} & \mathscr{V}\text{-mat} \\ {\scriptstyle \mathsf{Gr}}\uparrow & & \uparrow{\scriptstyle \mathsf{Gr}} \\ \mathsf{Set} & \xrightarrow{T} & \mathsf{Set} \end{array}$$

The fact that we have a 2-functor again means that the functor $\widehat{T}$ is locally monotone, so that for two $\mathscr{V}$-relations R, S it holds: if $R \leq S$, then

$$\widehat{T}(R) \leq \widehat{T}(S).$$

Notice that we now cannot use the span representation of relations to define $\mathscr{V}$-lifting, since $\mathscr{V}$-relations generally do not have such representation. We will therefore first try to find a lifting for the polynomial functors, which will be the building blocks used in describing a much larger selection of functors later, namely finitary functors preserving weak pullbacks.

3.3 Many-valued relation lifting for polynomial functors

Polynomial functors are given by a simple inductive definition. We are going to use this fact for our good and define the relation lifting inductively as well. For each building block the definition of the $\mathscr{V}$-relation lifting is a straightforward generalisation of the standard relation lifting, and we will introduce it by defining various operations on $\mathscr{V}$-matrices.

3.3.1 Operations on $\mathscr{V}$-relations

In this subsection let us fix two $\mathscr{V}$-relations $R : A \nrightarrow B$ and $R' : A' \nrightarrow B'$. We shall view them now as matrices $R : A \times B \to V$ and $R' : A' \times B' \to V$ and introduce matrix operations which will be then used to define $\mathscr{V}$-lifting.

Constant operation We will introduce this operation for every set C. When applied to the matrix R, $C(R)$ just replaces the matrix R with the identity matrix id_C over the set C. Observe that this is just the action of the constant functor $C : \mathscr{V}\text{-}\mathsf{mat} \to \mathscr{V}\text{-}\mathsf{mat}$.

Identity operation The identity operation on matrices is simply the identity functor on the category $\mathscr{V}$-mat: for R, we have $\mathsf{Id}(R) = R$.

Sum operation We will denote the sum operation by the symbol $\boxplus$ and it will serve as "matrix gluing": given two matrices R, S, the matrix $R \boxplus S$ has R and S on the diagonal and all the other elements are set to $\bot$. More precisely, we get a relation

$$R \boxplus S : A + A' \nrightarrow B + B'.$$

On elements, $R \boxplus S(\alpha, \beta) = v$, where $v = R(a, b)$ if $\alpha = (a, 1)$ and $\beta = (b, 1)$. If $\alpha = (a', 2)$ and $\beta = (b', 2)$, then $v = R'(a', b')$. In all other cases, $v = R \boxplus S(\alpha, \beta) = \bot$.

Tensor operation We denote this operation by $\boxtimes$. The tensor of matrices R and R' is

$$R \boxtimes R' : A \times A' \nrightarrow B \times B',$$

and is defined elementwise in the spirit of Example 3.1.1:

$$R \boxtimes R'((a, a'), (b, b')) = R(a, b) \otimes R'(a', b').$$

Example 3.3.1. Let us fix a set $C = \{c_1, c_2\}$ and the unit interval quantale $\mathscr{V} = ([0; 1], \cdot, 1)$ with multiplication. Then let the two $\mathscr{V}$-relations $R : A \nrightarrow B$ and $S : X \to Y$ be defined by the following matrices:

R	b_1	b_2
a_1	1/2	1/3
a_2	1/5	0

S	y_1	y_2
x_1	0	0
x_2	0	1/7

Then the result of the tensor and sum operation applied to R and S is given by the following two matrices:

$\widehat{R \times S}$	(b_1, y_1)	(b_1, y_2)	(b_2, y_1)	(b_2, y_2)
(a_1, x_1)	0	0	0	0
(a_1, x_2)	0	$1/(2 \cdot 7)$	0	$1/(3 \cdot 7)$
(a_2, x_1)	0	0	0	0
(a_2, x_2)	0	$1/(5 \cdot 7)$	0	0

$\widehat{R + S}$	b_1	b_2	y_1	y_2
a_1	1/2	1/3	0	0
a_2	1/5	0	0	0
x_1	0	0	0	0
x_2	0	0	0	1/7

3.3.2 Lifting via operations

We shall now use the operations introduced in the previous subsection to define $\mathscr{V}$-relation lifting for polynomial functors.

Relation lifting for the constant functor Let $C : \mathsf{Set} \to \mathsf{Set}$ be the constant functor we want to lift. We denote its lifting by $\widehat{C}$ and it is simply the constant functor $C : \mathscr{V}\text{-}\mathsf{mat} \to \mathscr{V}\text{-}\mathsf{mat}$. Conoretely, this functor is defined by the following two actions:
Action on objects:

$$A \mapsto C$$

for any set A.
Action on morphisms:

$$(R : X \nrightarrow Y) \mapsto (\widehat{C}(R) = \mathsf{const}_C(R) = \mathsf{id}_C : C \nrightarrow C)$$

for any $\mathscr{V}$-relation R.

Relation lifting for the Id functor Given the identity functor $\mathsf{Id} : \mathsf{Set} \to \mathsf{Set}$, we define its lifted $\mathscr{V}$-mat counterpart $\widehat{\mathsf{Id}} : \mathscr{V}\text{-}\mathsf{mat} \to \mathscr{V}\text{-}\mathsf{mat}$ as the identity functor on $\mathscr{V}$-mat:
Action on objects:

$$X \mapsto X$$

for any set X.
Action on morphisms:

$$(R : X \nrightarrow Y) \mapsto (\mathsf{Id}(R) = R : X \nrightarrow Y)$$

for any matrix R.

Relation lifting for products of functors Lifting of a product of functors is the tensor. If we start with a functor $T = T_1 \times T_2$, knowing already that T_1 and T_2 have a many-valued relation lifting, we define the lifted functor $\widehat{T}$ according to the following two actions:
Action on objects:

$$X \mapsto T_1X \times T_2X$$

for any set X.
Action on morphisms:

$$(R : X \nrightarrow Y) \mapsto (\widehat{T}R : T_1X \times T_2X \nrightarrow T_1Y \times T_2Y),$$

where $\widehat{T}R = \widehat{T_1}(R) \boxtimes \widehat{T_2}(R)$, so on elements $\widehat{T}R$ acts as follows:

$$\widehat{T}R((\alpha_1, \alpha_2), (\beta_1, \beta_2)) = \widehat{T_1}R(\alpha_1, \beta_1) \otimes \widehat{T_2}R(\alpha_2, \beta_2)$$

for $\alpha_i \in T_iX$ and $\beta_i \in T_iY$.

Relation lifting for sums of functors Starting with a functor $T = T_1 + T_2$, where again T_1 and T_2 have a many-valued relation lifting, we define the lifted functor $\widehat{T}$ according to the following two actions:
Action on objects:

$$X \mapsto T_1X + T_2X$$

for any set X.
Action on morphisms:

$$(R : X \nrightarrow Y) \mapsto (\widehat{T}R : T_1X + T_2X \nrightarrow T_1Y + T_2Y),$$

where $\widehat{T}R = \widehat{T_1}(R) \boxplus \widehat{T_2}(R)$, so on elements

$$\widehat{T}R(\alpha, \beta) = \begin{cases} \widehat{T_i}R(\alpha, \beta) & \text{if } \alpha \in T_iX \text{ and } \beta \in T_iY \\ \bot & \text{otherwise.} \end{cases}.$$

Theorem 3.3.2. *All of the preceding constructions are indeed many-valued relation liftings, that is, they are 2-functorial and make the diagram in the Definition 3.2.1 commute.*

We shall prove this theorem part by part. Firstly, we should show that the constructions are in fact functorial.

Lemma 3.3.3. *The preceding constructions are functorial.*

Proof. The cases where we lift constant functors and the identity functor are immediate. When $T = T_1 \times T_2$, we check that $\widehat{T}(\mathsf{id}_A) = \mathsf{id}_{\widehat{T}A}$:

$$\widehat{T}(\mathsf{id}_A) = \widehat{T_1}(\mathsf{id}_A) \boxtimes \widehat{T_2}(\mathsf{id}_A) = \mathsf{id}_{T_1 A \times T_2 A} = \mathsf{id}_{\widehat{T}(A)}.$$

To prove that $\widehat{T}(S \circ R) = \widehat{T}(S) \circ \widehat{T}(R)$, it would come handy to first prove that

$$(\widehat{T_1}(S) \circ \widehat{T_1}(R)) \boxtimes (\widehat{T_2}(S) \circ \widehat{T_2}(R)) = (\widehat{T_1}(S) \boxtimes \widehat{T_2}(S)) \circ (\widehat{T_1}(R) \boxtimes \widehat{T_2}(R)).$$

This is just a special instance of the general law

$$(M \circ N) \boxtimes (O \circ P) = (M \boxtimes O) \circ (N \boxtimes P),$$

which we will now take for granted and prove it immediately after completing this proof. Knowing that this law holds,

$$\widehat{T}(S \circ R) = \widehat{T_1}(S \circ R) \boxtimes \widehat{T_2}(S \circ R) = (\widehat{T_1}(S) \circ \widehat{T_1}(R)) \boxtimes (\widehat{T_2}(S) \circ \widehat{T_2}(R)).$$

Then by using the law, we continue with

$$(\widehat{T_1}(S) \boxtimes \widehat{T_2}(S)) \circ (\widehat{T_1}(R) \boxtimes \widehat{T_2}(R)) = \widehat{T}(S) \circ \widehat{T}(R)$$

and this ends the proof of the product part of $\mathscr{V}$-lifting.

When $T = T_1 + T_2$, checking that $\widehat{T}(\mathsf{id}_A) = \mathsf{id}_{\widehat{T}A}$ is again easy:

$$\widehat{T}(\mathsf{id}_A) = \widehat{T_1}(\mathsf{id}_A) \boxplus \widehat{T_2}(\mathsf{id}_A) = \mathsf{id}_{T_1 A + T_2 A} = \mathsf{id}_{\widehat{T}(A)}.$$

Now does the $\mathscr{V}$-lifting for T preserve compositions? To prove that $\widehat{T}(S \circ R) = \widehat{T}(S) \circ \widehat{T}(R)$, we could use the same argument as in the previous case if we knew that there is a law

$$(M \circ N) \boxplus (O \circ P) = (M \boxplus O) \circ (N \boxplus P).$$

We are going to prove this law in the following lemma, and that will end the proof of functoriality of liftings for sums of functors. □

Lemma 3.3.4. *For any four $\mathscr{V}$-relations $M : M_1 \nrightarrow M_2$, $N : N_1 \nrightarrow M_1$, $O : O_1 \nrightarrow O_2$, $P : P_1 \nrightarrow O_1$, the following two laws hold:*

$$(M \circ N) \boxtimes (O \circ P) = (M \boxtimes O) \circ (N \boxtimes P)$$

$$(M \circ N) \boxplus (O \circ P) = (M \boxplus O) \circ (N \boxplus P).$$

Proof. The relation $(M \circ N) \boxtimes (O \circ P)$ is of the type $N_1 \times P_1 \nrightarrow M_2 \times O_2$, as is the relation $(M \boxtimes O) \circ (N \boxtimes P)$. Similarly, the relation $(M \circ N) \boxplus (O \circ P)$ is of the type $N_1 + P_1 \nrightarrow M_2 + O_2$, the same type as $(M \boxtimes O) \circ (N \boxtimes P)$ has. Take $n_1 \in N_1$, $p_1 \in P_1$, $m_2 \in M_2$ and $o_2 \in O_2$. Then

$$\begin{aligned}
&(M \circ N) \boxtimes (O \circ P)((n_1, p_1), (m_2, o_2)) \\
=&(M \circ N)(n_1, p_1) \otimes (O \circ P)(m_2, o_2) \\
=&\big(\bigvee_m N(n_1, m) \otimes M(m, m_2)\big) \otimes \big(\bigvee_o P(p_1, o) \otimes O(o, o_2)\big) \\
=&\bigvee_m \bigvee_o N(n_1, m) \otimes M(m, m_2) \otimes P(p_1, o) \otimes O(o, o_2) \\
=&\bigvee_{m,o} N \boxtimes P((n_1, p_1), (m, o)) \otimes M \boxtimes O((m, o), (m_2, o_2)) \\
=&(M \boxtimes O) \circ (N \boxtimes P)((n_1, p_1), (m_2, o_2)).
\end{aligned}$$

To prove the second law, we are only interested in the elements $\alpha \in N_1 + P_1$ and $\beta \in M_2 + O_2$ when either $\alpha = (n_1, 1)$ and $\beta = (m_2, 1)$, or $\alpha = (p_1, 2)$ and $\beta = (o_2, 2)$. In the first case, we see that

$$\begin{aligned}
&(M \circ N) \boxplus (O \circ P)(\alpha, \beta) \\
=&(M \circ N)(n_1, m_2) \\
=&\bigvee_m N(n_1, m) \otimes M(m, m_2) \\
=&\bigvee_{\gamma \in M_1 + O_1} N \boxplus P(\alpha, \gamma) \otimes M \boxplus O(\gamma, \beta) \\
=&(M \boxplus O) \circ (N \boxplus P)(\alpha, \beta).
\end{aligned}$$

The proof for the second case proceeds in the same manner. All the other cases are easily seen to yield $\bot$ as a result. □

The definition of $\mathscr{V}$-liftings requires the functor to be locally monotone as well.

Lemma 3.3.5. *For any polynomial functor $T : \mathsf{Set} \to \mathsf{Set}$, the operation $\widehat{T}$ is a 2-functor.*

Proof. We need to prove that for a polynomial functor T and for two relations $R : A \nrightarrow B$, $S : A \nrightarrow B$ where $R \leq S$, $\widehat{T}$ preserves the order on the relations. So $\widehat{T}R \leq \widehat{T}S$. We proceed by induction.

- Constant functor C: $\widehat{C}R = \mathsf{id}_C = \widehat{C}S$.
- Identity functor Id: $\widehat{\mathsf{Id}}R = R \leq S = \widehat{\mathsf{Id}}S$.
- Sum of functors $T + T'$: Take $\alpha = (a, i) \in (T + T')A$ and $\beta = (b, j) \in (T + T')B$. Either $i \neq j$ and then

  $$\widehat{T + T'}R(\alpha, \beta) = \bot = \widehat{T + T'}S(\alpha, \beta)$$

  or $i = j = 1$ and then

  $$\widehat{T + T'}R(\alpha, \beta) = \widehat{T}R(a, b) \leq \widehat{T}S(a, b) = \widehat{T + T'}S(\alpha, \beta)$$

or $i = j = 2$ and then

$$\widehat{T + T'}R(\alpha, \beta) = \widehat{T'}R(a, b) \leq \widehat{T'}S(a, b) = \widehat{T + T'}S(\alpha, \beta)$$

- Product of functors $T \times T'$: For any $(\alpha, \alpha') \in (T \times T')A$ and $(\beta, \beta') \in (T \times T')B$ we have that

$$\begin{aligned}\widehat{T \times T'}R((\alpha, \alpha'), (\beta, \beta')) &= \widehat{T}R(\alpha, \beta) \otimes \widehat{T}R(\alpha', \beta') \\ &\leq \widehat{T}S(\alpha, \beta) \otimes \widehat{T}S(\alpha', \beta') \\ &= \widehat{T \times T'}S((\alpha, \alpha'), (\beta, \beta')).\end{aligned}$$

We have shown that $\mathscr{V}$-liftings of polynomial functors are 2-functors. □

The $\mathscr{V}$-lifting does moreover have to commute with the graph functor.

Lemma 3.3.6. *$\mathscr{V}$-lifting of polynomial functors commutes with the graph functor:*

$$\widehat{T} \circ \mathsf{Gr} = \mathsf{Gr} \circ T.$$

Proof. We show the inductive argument for polynomial functors:

- Constant functor C:

$$(\widehat{\mathsf{gr}} \circ C)(f) = \mathsf{Gr}(\mathsf{id}_C) = \widehat{C}(\mathsf{Gr}(f)).$$

- Identity functor Id:

$$(\mathsf{Gr} \circ \mathsf{Id}) = \mathsf{Gr} = (\widehat{\mathsf{Id}} \circ \mathsf{Gr}).$$

- Sum of functors $T + T'$:

$$(\mathsf{Gr} \circ (T + T')) = (\mathsf{Gr} \circ T) \boxplus (\mathsf{Gr} \circ T') = (\widehat{T} \circ \mathsf{Gr}) \boxplus (\widehat{T'} \circ \mathsf{Gr}) = (\widehat{T} \boxplus \widehat{T'}) \circ \mathsf{Gr} = \widehat{T + T'} \circ \mathsf{Gr}.$$

- Product of functors $T \times T'$:

$$(\mathsf{Gr} \circ (T \times T')) = (\mathsf{Gr} \circ T) \boxtimes (\mathsf{Gr} \circ T') = (\widehat{T} \circ \mathsf{Gr}) \boxtimes (\widehat{T'} \circ \mathsf{Gr}) = (\widehat{T} \boxtimes \widehat{T'}) \circ \mathsf{Gr} = \widehat{T \times T'} \circ \mathsf{Gr}.$$

□

After carefully reading this proof, the reader has surely noticed that we have used a few unproved properties: namely that the graph functor distributes over sum and product operations on functors.

Lemma 3.3.7. *The graph functor* Gr *distributes over sums of endofunctors of* Set*:*

$$(\mathsf{Gr} \circ (T + T')) = (\mathsf{Gr} \circ T) \boxplus (\mathsf{Gr} \circ T').$$

It also distributes over sums of endofunctors of $\mathscr{V}$-mat*:*

$$(\widehat{T} \circ \mathsf{Gr}) \boxplus (\widehat{T'} \circ \mathsf{Gr}) = (\widehat{T} \boxplus \widehat{T'}) \circ \mathsf{Gr}.$$

In the same manner, Gr *distributes over products of endofunctors of* Set *and* $\mathscr{V}$-mat*.*

We are not going to prove these properties. Instead we prove Lemma 3.3.6 more conceptually (and this time not partially) in a different way. We are going to show that $\mathscr{V}$-lifting extends the standard notion of relation lifting, which will give us a nice pictorial proof of Lemma 3.3.6 and moreover it is an important observation on its own.

Lemma 3.3.8 ($\mathscr{V}$-lifting is an extension of standard lifting)**.** *$\mathscr{V}$-lifting for polynomial functors is an extension of standard relation lifting, that is, the following diagram*

$$\begin{array}{ccc} \mathscr{V}\text{-mat} & \xrightarrow{\widehat{T}} & \mathscr{V}\text{-mat} \\ {\scriptstyle\mathcal{I}}\uparrow & & \uparrow{\scriptstyle\mathcal{I}} \\ \mathsf{Rel} & \xrightarrow{\overline{T}} & \mathsf{Rel} \end{array}$$

commutes.

Proof. This lemma follows directly from the fact that standard relation lifting for polynomial functors can be defined as in Example 2.4.6, and from the definition of the $\mathcal{I}$ functor (Example 1.2.16).

□

We can now finally present the pictorial proof of Lemma 3.3.6.

Proof. We already know that standard relation lifting for polynomial functors makes the diagram

$$\begin{array}{ccc} \mathsf{Rel} & \xrightarrow{\overline{T}} & \mathsf{Rel} \\ {\scriptstyle\mathsf{gr}}\uparrow & & \uparrow{\scriptstyle\mathsf{gr}} \\ \mathsf{Set} & \xrightarrow{T} & \mathsf{Set} \end{array}$$

commute, and also that the diagram from Lemma **??** commutes. By composing the two diagrams, we see that

$$\begin{array}{ccc} \mathscr{V}\text{-mat} & \xrightarrow{\widehat{T}} & \mathscr{V}\text{-mat} \\ {\scriptstyle\mathcal{I}}\uparrow & & \uparrow{\scriptstyle\mathcal{I}} \\ \mathsf{Rel} & \xrightarrow{\overline{T}} & \mathsf{Rel} \\ {\scriptstyle\mathsf{gr}}\uparrow & & \uparrow{\scriptstyle\mathsf{gr}} \\ \mathsf{Set} & \xrightarrow{T} & \mathsf{Set} \end{array}$$

commutes as well. Because the $\mathscr{V}$-graph functor Gr is defined as the composition $\mathcal{I} \circ \mathsf{gr}$, the proof is complete. □

We have now shown a reasonable many-valued generalisation of relation liftings for polynomial functors. In order to extend this generalisation to a larger class of functors, we are going to introduce presentations of functors in the next section. These will allow us to use the $\mathscr{V}$-lifting of polynomial functors to define $\mathscr{V}$-lifting for arbitrary weak pullback preserving finitary functors.

3.4 Presentations of finitary functors

Presentations of functors allow us to "disassemble" finitary functors by means of certain special polynomial functors and natural transformations. They are discussed for example in [5] or in [40]. We have seen an example of a functor presentation before, even if we did not mention it explicitly, in Example 1.1.34.

Example 3.4.1. (List to powerset, revisited) There is a natural transformation $\varepsilon : \mathsf{List} \to \mathcal{P}$, which for any set A contains a function $\varepsilon_A : \mathsf{List}(A) \to \mathcal{P}A$:

$$(a_i)_{i<n} \mapsto \{a_i \mid i < n\}.$$

The function ε_A is a surjection for any set A, and so the natural transformation ε is a natural epimorphism. The requirements of naturality and surjectivity will be central to the notion of functor presentations.

The natural transformation ε from previous example can be used to define the relation lifting of the $\mathcal{P}$ functor using the relation lifting for the List functor. This observation is the main reason why we are interested in functor presentations in general.

Relation lifting for $\mathcal{P}$ using List. Let us recall from Example 2.4.8 that standard relation lifting for the finitary powerset $\mathcal{P}_\omega$ functor lifts the relation $R : A \nrightarrow B$ to the relation $\mathcal{P}_\omega(R) : \mathcal{P}_\omega(A) \nrightarrow \mathcal{P}_\omega(B)$, where $\mathcal{P}_\omega(R)(\alpha, \beta)$ if and only if

$$\forall a \in \alpha\ \exists b \in \beta :\ R(a,b) \quad \text{and} \quad \forall b \in \beta\ \exists a \in \alpha :\ R(a,b).$$

Observe that we can equivalently say that $\mathcal{P}_\omega(R)(\alpha, \beta)$ holds if and only if there are two lists $l_\alpha \in \mathsf{List}(A)$, $l_\beta \in \mathsf{List}(B)$ such that l_α "represents" α, meaning that $\varepsilon_A(l_\alpha) = \alpha$, l_β represents β, so that it holds $\varepsilon_B(l_\beta) = \beta$, and moreover $\overline{\mathsf{List}}(R)(l_\alpha, l_\beta)$, the two lists l_α and l_β are related by the relation R lifted to the functor List. We can see this easily by the means of the following commutative diagram:

$$\begin{array}{ccccc}
\mathsf{List}(A) & \xleftarrow{\mathsf{List}(p_1)} & \mathsf{List}(R) & \xrightarrow{\mathsf{List}(p_2)} & \mathsf{List}(B) \\
\downarrow{\scriptstyle \varepsilon_A} & & \downarrow{\scriptstyle \varepsilon_R} & & \downarrow{\scriptstyle \varepsilon_B} \\
\mathcal{P}_\omega(A) & \xleftarrow[\mathcal{P}_\omega(p_1)]{} & \mathcal{P}_\omega(R) & \xrightarrow[\mathcal{P}_\omega(p_2)]{} & \mathcal{P}_\omega(B)
\end{array}$$

The subsets $\alpha \in \mathcal{P}_\omega(A)$ and $\beta \in \mathcal{P}_\omega(B)$ are in the lifted relation $\overline{\mathcal{P}_\omega}(R)$ if and only if there is a witness $w \in \mathcal{P}_\omega(R)$ such that it projects onto α and β. Since ε is a natural epimorphism, it means that there is a list witness $l_w \in \mathsf{List}(R)$

which represents w, so $\varepsilon_R(l_w) = w$, and by the naturality of the squares in the diagram we get the lists l_α and l_β as the list projections of w. Conversely, if there are lists l_α and l_β representing α and β, and it holds that $\overline{\mathsf{List}(R)}(l_\alpha, l_\beta)$, then by the definition of relation lifting we get the list witness $l_w \in \mathsf{List}(R)$ and again by naturality of ε and surjectivity of ε_R we get a witness $w \in \mathcal{P}_\omega(R)$ which witnesses the fact that α and β are in the lifted relation $\overline{\mathcal{P}_\omega}(R)$. We have therefore described relation lifting for $\mathcal{P}_\omega$ in terms of the relation lifting for List, using a natural epimorphism $\varepsilon : \mathsf{List} \to \mathcal{P}_\omega$. This example and generalisations to arbitrary finitary functors is discussed for example in [40]; we are going to present some of the theory now and then generalise our present observation to define $\mathscr{V}$-liftings for finitary functors.

Definition 3.4.2 (Polynomial presentations of finitary functors)**.** We say that a finitary functor $T : \mathsf{Set} \to \mathsf{Set}$ is *polynomially presented* by a polynomial functor H and a natural transformation $\varepsilon : H \to T$ if ε is a natural epimorphism.

Using polynomial presentations, we can compute relation liftings for polynomially presentable finitary functors which do not need to be polynomial.

The following lemma is a straightforward generalisation of the principles shown in the example with List and finitary powerset functor.

Lemma 3.4.3 (Relation lifting via presentations)**.** *Having a finitary* Set *endofunctor T presented by a functor H and a natural epimorphism ε, and having a relation $R : A \nrightarrow B$, the following two properties are equivalent:*

- *$\overline{T}(R)(\alpha, \beta)$ holds for $\alpha \in T(A)$, $\beta \in T(B)$.*
- *There exist $t_\alpha \in HA$, $t_\beta \in HB$ such that $\varepsilon_A(t_\alpha) = \alpha$, $\varepsilon_B(t_\beta) = \beta$ and $\overline{H}(R)(t_\alpha, t_\beta)$.*

Proof. If $\overline{T}(R)(\alpha, \beta)$, find the witness $w \in T(R)$ and choose a representing element $t_w \in H(R)$ such that $\varepsilon_R(t_w) = w$ (which exists because ε_R is surjective). Then by naturality of ε there are elements $H(p_1)(w)$,$H(p_2)(w)$ representing α and β (see the diagram below)

$$\begin{array}{ccccc} H(A) & \xleftarrow{H(p_1)} & H(R) & \xrightarrow{H(p_2)} & H(B) \\ \downarrow{\scriptstyle \varepsilon_A} & & \downarrow{\scriptstyle \varepsilon_R} & & \downarrow{\scriptstyle \varepsilon_B} \\ T(A) & \xleftarrow[T(p_1)]{} & T(R) & \xrightarrow[T(p_2)]{} & T(B) \end{array}$$

and t_w moreover witnesses that $\overline{H}(R)(H(p_1)(w), H(p_2)(w))$. Conversely, having t_α and t_β, we get t_w which by naturality gives us an element $w = \varepsilon_R(t_w)$ witnessing that $\overline{T}(R)(\alpha, \beta)$ holds. □

Theorem 3.4.4. *Every finitary* Set *functor is polynomially presentable by a* canonical *presentation.*

Proof. Fix the finitary functor T. Then we construct the polynomial functor

$$H_\Sigma = \coprod_{n \in \mathbb{N}} \mathsf{Id}^n \times T(n)$$

and we have to show that there is a natural epimorphism $\varepsilon : H_\Sigma \to T$. For a set A, we have to define how $\varepsilon_A : \coprod_{n\in\mathbb{N}} A^n \times T(n) \to T(A)$ acts on elements. An element $t \in H_\Sigma(A)$ is a pair $(\mathbf{a}, \sigma)$, where $\mathbf{a}$ is a function of type $n \to A$ and $\sigma \in T(n)$. From these data we need to get an element from $T(A)$. By applying the functor T to $\mathbf{a}$, we get a function $T(\mathbf{a}) : T(n) \to T(A)$. Since σ is an element of $T(n)$, we can define $\varepsilon_A(t)$ to be the element $T(\mathbf{a})(\sigma)$. It remains to show that this is an epimorphism. This property follows from the fact that T is finitary. Having $\alpha \in T(A)$, this means that $\alpha \in T(A')$, where A' is a finite subset of A. Denote by n the number of elements of A'. There is a bijection $\mathbf{b} : n \to A'$, and since bijections are preserved by applying a functor, $T(\mathbf{b}) : T(n) \to T(A')$ is a bijection as well. Choose then the element $\sigma \in T(n)$ corresponding to α under the bijection $T(\mathbf{b})$, and now it is obvious that $t = (\mathbf{b}, \sigma)$ represents α. The naturality of ε can be proved easily. The following diagram

$$\begin{array}{ccc} H_\Sigma(A) & \xrightarrow{H_\Sigma(f)} & H_\Sigma(B) \\ {\scriptstyle \varepsilon_A}\downarrow & & \downarrow{\scriptstyle \varepsilon_B} \\ T(A) & \xrightarrow[T(f)]{} & T(B) \end{array}$$

commutes for any A,B and f. Take an arbitrary $t_\alpha = (\mathbf{a}, \sigma) \in H_\Sigma(A)$, which represents some $\alpha \in T(A)$. We see that $\alpha \mapsto \beta$ by $T(f)$ for some $\beta \in T(B)$. Observe that $H_\Sigma(f)$ maps $t_\alpha = (\mathbf{a}, \sigma)$ to a pair $(f \circ \mathbf{a}, \sigma)$. Then we need to know that this pair represents β via ε_B. This is inded true: we take

$$n \xrightarrow{\mathbf{a}} A \xrightarrow{f} B$$

and apply the functor T to it, so we get

$$T(n) \xrightarrow{T(\mathbf{a})} T(A) \xrightarrow{T(f)} T(B).$$

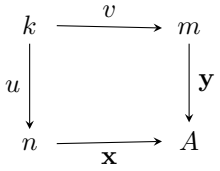
Applying this function to σ, it gets mapped to α by $T(\mathbf{a})$, and α gets mapped to β by $T(f)$ by assumption. The collection of morphisms ε is therefore a natural epimorphism. □

We say that the presentation we have described in the previous proof is the *canonical presentation* of T. Every finitary functor T therefore has a canonical presentation [40]. Because of this fact, Lemma 3.4.3 is a strong statement.

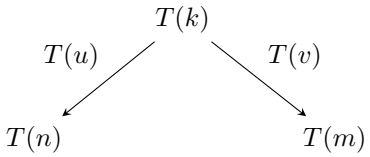
Notation Given a functor T with a canonical presentation consisting of H_Σ and ε, we talk of the presentation as of a pair (H_Σ, ε), and if a given element $\alpha \in T(A)$ is presented by $t_\alpha = (\mathbf{a}, \sigma)$, we write the pair $(\mathbf{a}, \sigma)$ in term notation $\sigma(\mathbf{a})$, which reminds us that σ can be thought of as an operation which accepts a vector $\mathbf{a} : n \to A$ of elements from A.

Canonical presentations of weak pullback preserving functors enjoy an important property, which we are going to use when defining $\mathscr{V}$-lifting for finitary functors.

Definition 3.4.5 (Dominated presentation [5])**.** A presentation (H_Σ, ε) of a functor T is called *dominated* if for any two terms $\sigma(\mathbf{x})$, $\tau(\mathbf{y})$ representing $\alpha \in T(A)$ (where $\mathbf{x} : n \to A$, $\mathbf{y} : m \to A$ and $\sigma \in T(n)$, $\tau \in T(m)$) there is an operation $\rho \in T(k)$ together with two maps $u : k \to n$ and $v : k \to m$ such that $T(u)(\rho) = \sigma$, $T(v)(\rho) = \tau$ and $\mathbf{x} \circ u = \mathbf{y} \circ v$. In diagrams,



commutes and the span



maps ρ to σ and τ respectively, ensuring that both of the terms $\rho(\mathbf{x} \circ u)$ and $\rho(\mathbf{y} \circ v)$ represent α.

We are going to introduce here one technical property of functors, which we will implicitly assume in the following text.

Definition 3.4.6 (Sound functor [5])**.** Let 1 be the constant Set functor sending all sets to 1 and 1_0 the same functor with the difference that $1_0(\emptyset) = \emptyset$. A Set functor T is *sound* if it preserves inclusions and every natural transformation $\varepsilon : 1_0 \to T$ has a unique extension to a natural transformation $\widehat{\varepsilon} : 1 \to T$.

Every functor that is not sound can be easily transformed into one that is sound just by changing its actions with respect to the empty set. We will therefore assume throughout the text that the functors we are working with do have the soundness property without mentioning this fact.

Theorem 3.4.7 ([5])**.** *The canonical presentation of a sound (Definition 3.4.6) finitary weak pullback preserving functor is dominated.*

The fact that canonical presentations of weak pullback preserving functors are dominated will help us with the proof that $\mathscr{V}$-lifting of finitary functors preserves compositions.

3.5 Many-valued lifting for finitary functors

Now we are going to use the technical results introduced in the previous section to lift finitary functors to the category $\mathscr{V}$-mat in general and show that for weak pullback preserving functors this gives a $\mathscr{V}$-lifting in the sense of the Definition 3.2.1.

First let us look at Lemma 3.4.3 and restate it in a way that will be useful when defining $\mathscr{V}$-lifting for a finitary functor T: Having T presented by (H, ε), then for a relation $R : A \nrightarrow B$, the following two properties are equivalent:

- $\overline{T}(R)(\alpha, \beta)$ holds for $\alpha \in T(A)$, $\beta \in T(B)$.
- There exist $t_\alpha \in HA$, $t_\beta \in HB$ such that $\varepsilon_A(t_\alpha) = \alpha$, $\varepsilon_B(t_\beta) = \beta$ and $\overline{H}(R)(t_\alpha, t_\beta)$.

We can rewrite the equivalence as an algebraic rule

$$\overline{T}(R)(\alpha, \beta) = \bigvee_{t_\alpha, t_\beta} \overline{H}(R)(t_\alpha, t_\beta).$$

The notation is simplified: we are computing the supremum over all pairs of terms (t_α, t_β) such that t_α represents α and t_β represents β. In the case of standard relation lifting, this just tells us that we look for *some* two terms representing α and β which are related. In its generalised version, this algebraic rule will tell us that we are looking for the "best" representatives of α and β and take the value of their lifting as defining the value of $\mathscr{V}$-lifting for α and β.

Definition 3.5.1 ($\mathscr{V}$-lifting of finitary functors)**.** Given a $\mathscr{V}$-relation $R : A \nrightarrow B$ and a non-polynomial finitary weak pullback preserving functor T, we define the lifted relation $\widehat{T}R : TA \nrightarrow TB$ by the following formula for $\alpha \in TA$ and $\beta \in TB$ using the canonical presentation (H_Σ, ε):

$$\widehat{T}(R)(\alpha, \beta) = \bigvee_{t_\alpha, t_\beta} \widehat{H_\Sigma}(R)(t_\alpha, t_\beta).$$

Does Definition 3.5.1 give us really a $\mathscr{V}$-lifting, is it functorial and does it make the diagram

$$\begin{array}{ccc} \mathscr{V}\text{-mat} & \xrightarrow{\widehat{T}} & \mathscr{V}\text{-mat} \\ \mathsf{Gr} \uparrow & & \uparrow \mathsf{Gr} \\ \mathsf{Set} & \xrightarrow{T} & \mathsf{Set} \end{array}$$

from the Definition 3.2.1 commute? The answer is positive if we restrict ourselves from a general quantale $\mathscr{V}$ to a complete Heyting algebra. In the case of a general quantale $\mathscr{V}$-lifting does not have to be functorial.

Example 3.5.2. Take $\mathscr{V} = ([0, 1], \cdot, 1)$ as a quantale and $A = \{a\}, B = \{b\}, C = \{c_1, c_2\}$. Set two relations $R : A \nrightarrow B$, $S : B \nrightarrow C$ as follows: $R(a, b) = \frac{1}{2}$, $S(b, c_1) = \frac{1}{3}$, $S(b, c_2) = \frac{1}{5}$. Then

$$\widehat{T}(S \circ R)(\{a\}, \{c_1, c_2\}) = \frac{1}{60},$$

but

$$\widehat{T}(S) \circ \widehat{T}(R)(\{a\}, \{c_1, c_2\}) = \frac{1}{30}.$$

Let us now assume that $\otimes = \wedge$ and $e = \top$. For better distinction, we are going to use the tensor notation in places where it should be if we were working with a general quantale. We are then going to switch to $\wedge$ notation when it is necessary to use some of the properties of $\wedge$.

Theorem 3.5.3. *Suppose $\mathscr{V}$ is a complete Heyting algebra. Then the definition of $\widehat{T}$ from 3.5.1 gives an endofunctor of $\mathscr{V}$-*mat*.*

Proof. Let us first show that $\widehat{T}(\mathsf{id}_X) = \mathsf{id}_{\widehat{T}X}$. By definition, $\widehat{T}X = TX$ and

$$id_{\widehat{T}X} : TX \nrightarrow TX$$

is defined as

$$(\mathsf{id}_{\widehat{T}X})(\alpha, \beta) = \begin{cases} e & \text{if } \alpha = \beta \\ \bot & \text{otherwise.} \end{cases}$$

It is up to us to show that $\widehat{T}(\mathsf{id}_X)$ defines the same relation. But see that

$$\widehat{T}(\mathsf{id}_X)(\alpha, \alpha) = \bigvee_{t_\alpha, t'_\alpha} \widehat{H_\Sigma}(\mathsf{id}_X)(t_\alpha, t'_\alpha) \geq \widehat{H_\Sigma}(\mathsf{id}_X)(t_\alpha, t_\alpha) = e$$

and that if we take $\alpha \neq \beta$, then

$$\widehat{T}(\mathsf{id}_X)(\alpha, \beta) = \bigvee_{t_\alpha, t_\beta} \widehat{H_\Sigma}(\mathsf{id}_X)(t_\alpha, t_\beta) = \bot$$

since $\widehat{H_\Sigma}(\mathsf{id}_X)(t_\alpha, t_\beta) = \bot$ for any $t_\alpha \neq t_\beta$.

The proof that

$$\widehat{T}(S \circ R) = \widehat{T}S \circ \widehat{T}R$$

is more involved. We need to prove it for any two arbitrary $\mathscr{V}$-relations $R : A \nrightarrow B$ and $S : B \nrightarrow C$. Thus, for any $\alpha \in TA$ and $\gamma \in TC$, we ask ourselves whether

$$\widehat{T}(S \circ R)(\alpha, \gamma) = \widehat{T}S \circ \widehat{T}R(\alpha, \gamma).$$

From one side, we derive that by definition of $\mathscr{V}$-lifting for finitary functors

$$\widehat{T}(S \circ R)(\alpha, \gamma) = \bigvee_{t_\alpha, t_\gamma} (\widehat{H_\Sigma}(S \circ R)(t_\alpha, t_\gamma))$$

and by the definition of $\mathscr{V}$-relation composition

$$\bigvee_{t_\alpha, t_\gamma} (\widehat{H_\Sigma}(S \circ R)(t_\alpha, t_\gamma)) = \bigvee_{t_\alpha, s, t_\gamma} (\widehat{H_\Sigma}R(t_\alpha, s) \otimes \widehat{H_\Sigma}S(s, t_\gamma)).$$

From the other side, we see that (by $\mathscr{V}$-relation composition)

$$\widehat{T}(S) \circ \widehat{T}(R)(\alpha, \gamma) = \bigvee_{\beta} (\widehat{T}R(\alpha, \beta) \otimes \widehat{T}S(\beta, \gamma))$$

and from the definition of $\mathscr{V}$-lifting

$$\bigvee_{\beta} (\widehat{T}R(\alpha, \beta) \otimes \widehat{T}S(\beta, \gamma)) = \bigvee_{\beta} \Big(\big(\bigvee_{t_\alpha, t_\beta} \widehat{H_\Sigma}R(t_\alpha, t_\beta) \big) \otimes \big(\bigvee_{t'_\beta, t_\gamma} \widehat{H_\Sigma}S(t'_\beta, t_\gamma) \big) \Big).$$

We now want to prove that the following equality

$$\bigvee_{t_\alpha, s, t_\gamma} (\widehat{H_\Sigma}R(t_\alpha, s) \otimes \widehat{H_\Sigma}S(s, t_\gamma)) = \bigvee_{\beta} \Big(\big(\bigvee_{t_\alpha, t_\beta} \widehat{H_\Sigma}R(t_\alpha, t_\beta) \big) \otimes \big(\bigvee_{t'_\beta, t_\gamma} \widehat{H_\Sigma}S(t'_\beta, t_\gamma) \big) \Big)$$

holds. To prove that

$$\widehat{T}(S \circ R)(\alpha, \gamma) \leq \widehat{T}(S) \circ \widehat{T}(R)(\alpha, \gamma) \tag{3.1}$$

is easy: it suffices to find for for any triple t_α, s, t_γ a quadruple $t_\alpha, t_\beta, t'_\beta, t_\gamma$ witnessing the inequality 3.1. Denote by β the element represented by s. Then we can set $t_\beta = t'_\beta = s$, and this is a witness for the inequality.

But does the other inequality hold as well? It would be enough to find for every quadruple $t_\alpha, t_\beta, t'_\beta, t_\gamma$ a triple t'_α, s, t'_γ such that

$$\widehat{H_\Sigma} R(t_\alpha, t_\beta) \otimes \widehat{H_\Sigma} S(t'_\beta, t_\gamma) \leq \widehat{H_\Sigma} R(t'_\alpha, s) \otimes \widehat{H_\Sigma} S(s, t'_\gamma),$$

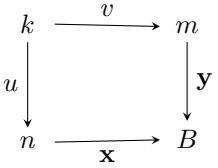
pointwise witnessing the second inequality,

$$\widehat{T}(S) \circ \widehat{T}(R)(\alpha, \gamma) \leq \widehat{T}(S \circ R)(\alpha, \gamma). \tag{3.2}$$

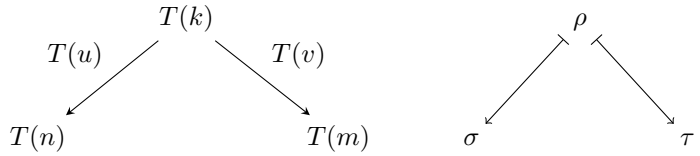
Here we are going to use the fact that we have a dominated presentation, which in turn comes from the fact that T preserves weak pullbacks.

Let $t_\alpha = \sigma(\mathbf{a})$, $t_\gamma = \tau(\mathbf{c})$ for some $\sigma \in T(n)$, $\tau \in T(m)$ (and therefore the vectors are of the type $\mathbf{a} : n \to A$ and $\mathbf{c} : m \to C$). We can assume for the other two terms t_β and t'_β from the quadruple that they are of the form $t_\beta = \sigma(\mathbf{x})$ and $t'_\beta = \tau(\mathbf{y})$ (with $\mathbf{x} : n \to B$ and $\mathbf{y} : m \to B$). This is so because if the terms t_α and t_β had a different operation as terms, their $\mathscr{V}$-lifting $\widehat{H_\Sigma} R(t_\alpha, t_\beta)$ would be equal to $\bot$, and the same reasoning applies for the terms t'_β and t_γ.

Since both t_β and t'_β represent β and since (H_Σ, ε) is a dominated presentation, there is an operation $\rho \in T(k)$ together with two maps $u : k \to n$ and $v : k \to m$ such that the diagram



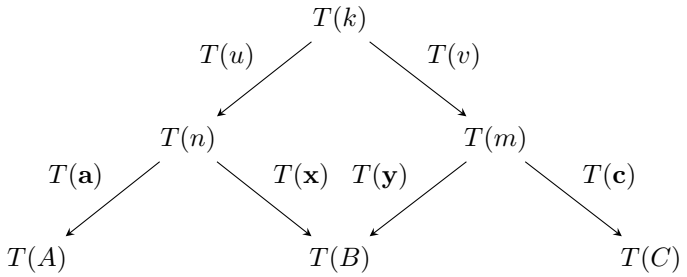
commutes and the operation ρ gets mapped to σ and τ by $T(u)$ and $T(v)$:



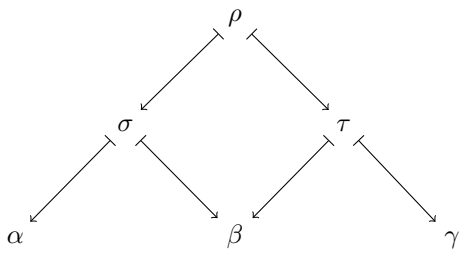
Therefore, $\rho(\mathbf{x} \circ u)$ represents β. Set $s = \rho(\mathbf{x} \circ u)$. We now just have to find suitable terms t'_α and t'_γ which would witness the inequality

$$\widehat{H_\Sigma} R(t_\alpha, t_\beta) \otimes \widehat{H_\Sigma} S(t'_\beta, t_\gamma) \leq \widehat{H_\Sigma} R(t'_\alpha, s) \otimes \widehat{H_\Sigma} S(s, t'_\gamma).$$

The terms t'_α and t'_γ are now forced to have ρ as the operation, since $s = \rho(\mathbf{x} \circ u)$. By looking at the following diagram,



we immediately see what to choose as t'_α and t'_γ:



Let $t'_\alpha = \rho(\mathbf{a} \circ u)$ and $t'_\gamma = \rho(\mathbf{c} \circ v)$. These terms represent α and γ respectively. To prove that

$$\widehat{H_\Sigma} R(t_\alpha, t_\beta) \otimes \widehat{H_\Sigma} S(t'_\beta, t_\gamma) \leq \widehat{H_\Sigma} R(t'_\alpha, s) \otimes \widehat{H_\Sigma} S(s, t'_\gamma),$$

we show that

$$\widehat{H_\Sigma} R(t_\alpha, t_\beta) \leq \widehat{H_\Sigma} R(t'_\alpha, s)$$

and that

$$\widehat{H_\Sigma} S(t'_\beta, t_\gamma) \leq \widehat{H_\Sigma} S(s, t'_\gamma).$$

By the definition of $\mathscr{V}$-lifting for polynomial functors,

$$\widehat{H_\Sigma} R(t_\alpha, t_\beta) = \bigotimes_{i<n} R(\mathbf{a}(i), \mathbf{x}(i)).$$

In the same spirit,

$$\widehat{H_\Sigma} R(t'_\alpha, s) = \bigotimes_{i<k} R(\mathbf{a}(u(i)), \mathbf{x}(u(i))).$$

Now we use the fact that $\otimes = \wedge$ and $\mathsf{e} = \top$ to prove that

$$\bigotimes_{i<n} R(\mathbf{a}(i), \mathbf{x}(i)) \leq \bigotimes_{i<k} R(\mathbf{a}(u(i)), \mathbf{x}(u(i))).$$

We can equivalently write that we want to prove

$$\bigwedge_{i<n} R(\mathbf{a}(i), \mathbf{x}(i)) \leq \bigwedge_{i<k} R(\mathbf{a}(u(i)), \mathbf{x}(u(i))),$$

and for proving this we use the fact that

$$\bigwedge_{i<n} R(\mathbf{a}(i), \mathbf{x}(i)) = \bigwedge \{R(\mathbf{a}(i), \mathbf{x}(i)) \mid i < n\}$$

and that

$$\bigwedge_{i<k} R(\mathbf{a}(u(i)), \mathbf{x}(u(i))) = \bigwedge \{R(\mathbf{a}(u(i)), \mathbf{x}(u(i))) \mid i < k\}.$$

This is useful because we moreover know that

$$\{R(\mathbf{a}(u(i)), \mathbf{x}(u(i))) \mid i < k\} \subseteq \{R(\mathbf{a}(i), \mathbf{x}(i)) \mid i < n\},$$

and since the infimum operation $\wedge$ is monotone, we can indeed infer that

$$\bigwedge_{i<n} R(\mathbf{a}(i), \mathbf{x}(i)) \leq \bigwedge_{i<k} R(\mathbf{a}(u(i)), \mathbf{x}(u(i))).$$

To prove the inequality

$$\widehat{H_\Sigma} S(t'_\beta, t_\gamma) \leq \widehat{H_\Sigma} S(s, t'_\gamma),$$

we would have to repeat the same procedure we have undertaken now, so we are going to omit it. This concludes the proof that $\widehat{T}$ is functorial. □

The proof that $\widehat{T}$ is 2-functorial is really easy in comparison with the proof of functoriality.

Lemma 3.5.4 ($\mathscr{V}$-lifting is a 2-functor)**.** *$\mathscr{V}$-lifting of a finitary functor $T : \mathsf{Set} \to \mathsf{Set}$ is a 2-functor $\widehat{T} : \mathscr{V}\text{-}\mathsf{mat} \to \mathscr{V}\text{-}\mathsf{mat}$.*

Proof. For a finitary functor T we use its presentation (H_Σ, ε), where H_Σ is a polynomial functor, so we are going to use this fact. Given two relations $R : A \nrightarrow B$ and $S : A \nrightarrow B$ with $R \leq S$, we see that

$$\widehat{T}R(\alpha, \beta) = \bigvee_{t_\alpha, t_\beta} \widehat{H_\Sigma} R(t_\alpha, t_\beta) \leq \bigvee_{t_\alpha, t_\beta} \widehat{H_\Sigma} S(t_\alpha, t_\beta) = \widehat{T}S(\alpha, \beta).$$

□

To prove that $\widehat{T}$ commutes with graphs when T is finitary, we use the same trick as for the polynomial functors: we prove that $\widehat{T}$ is an extension of the standard relation lifting.

Lemma 3.5.5. *$\mathscr{V}$-lifting for finitary functors is an extension of standard relation lifting, that is, the following diagram*

$$\begin{array}{ccc} \mathscr{V}\text{-}\mathsf{mat} & \xrightarrow{\widehat{T}} & \mathscr{V}\text{-}\mathsf{mat} \\ \uparrow{\scriptstyle \mathcal{I}} & & \uparrow{\scriptstyle \mathcal{I}} \\ \mathsf{Rel} & \xrightarrow{\overline{T}} & \mathsf{Rel} \end{array}$$

commutes.

Proof. Take a (two-valued) relation $R : A \nrightarrow B$. For $\alpha \in TA$ and $\beta \in TB$ The value $\widehat{T}(\mathcal{I}(R))(\alpha, \beta)$ is either e or $\bot$. Suppose it is e. Then

$$\bigvee_{t_\alpha, t_\beta} \widehat{H_\Sigma}(\mathcal{I}(R))(t_\alpha, t_\beta) = \mathsf{e}.$$

This happens exactly when there exist t_α,t_β such that

$$\widehat{H_\Sigma}(\mathcal{I}(R))(t_\alpha, t_\beta) = \mathsf{e},$$

which can happen if and only if

$$\overline{H_\Sigma}(R)(t_\alpha, t_\beta)$$

holds. Equivalently, we may say that

$$\bigvee_{t_\alpha, t_\beta} \overline{H_\Sigma}(R)(t_\alpha, t_\beta) = \mathcal{I}(\overline{T}(R))(\alpha, \beta) = \mathsf{e}.$$

We have therefore proven that

$$\widehat{T}(\mathcal{I}(R))(\alpha, \beta) = \mathsf{e} \text{ iff } \mathcal{I}(\overline{T}(R))(\alpha, \beta) = \mathsf{e}.$$

From this we can immediately deduce that

$$\widehat{T}(\mathcal{I}(R)) = \mathcal{I}(\overline{T}(R)),$$

which is what we had to prove. □

By the series of proofs in this section, we have proved that the definition of $\widehat{T}$ for a finitary weak pullback preserving functor T gives us a $\mathscr{V}$-lifting when $\mathscr{V}$ is a complete Heyting algebra.

3.6 Some properties of $\mathscr{V}$-lifting

In this section we are going to state and prove important properties of $\mathscr{V}$-lifting which we will use heavily in Chapter 4.

Lemma 3.6.1. *$\mathscr{V}$-lifting of finitary functors commutes with opposites (Definition 1.2.12):*

$$\widehat{T}(R^{op}) = \widehat{T}(R)^{op}.$$

Proof. This technical lemma follows quite quickly from the fact that $\mathscr{V}$-lifting is symmetric in nature. We fix a $\mathscr{V}$-relation $R : A \nrightarrow B$ and then proceed inductively. We first prove the assertion for polynomial endofunctors of Set and then we use presentations to prove the assertion for general finitary functors.

- Constant functor C: $\widehat{C}(R^{op}) = \mathsf{id}_C = \mathsf{id}_C^{op} = \widehat{C}(R)^{op}$.
- Identity functor Id: $\widehat{Id}(R^{op}) = R^{op} = \widehat{Id(R)}^{op}$.

- Sum of functors $T + T'$:

$$\begin{aligned}\widehat{T + T'}(R^{op}) &= (\widehat{T} \boxplus \widehat{T'})(R^{op}) = \widehat{T}(R^{op}) \boxplus \widehat{T'}(R^{op}) \\ &= \widehat{T}(R)^{op} \boxplus \widehat{T'}(R)^{op} = (\widehat{T} \boxplus \widehat{T'})(R)^{op}.\end{aligned}$$

- Product of functors $T \times T'$:

$$\begin{aligned}\widehat{T \times T'}(R^{op}) &= (\widehat{T} \boxtimes \widehat{T'})(R^{op}) = \widehat{T}(R^{op}) \boxtimes \widehat{T'}(R^{op}) \\ &= \widehat{T}(R)^{op} \boxtimes \widehat{T'}(R)^{op} = (\widehat{T} \boxtimes \widehat{T'})(R)^{op}.\end{aligned}$$

Consider a presentation $\varepsilon : H_\Sigma \to T$ (see 3.4.4). Then for any pair $\alpha \in TA$, $\beta \in TB$ the following equation

$$\begin{aligned}\widehat{T}(R^{op})(\beta, \alpha) &= \bigvee_{t_\beta, t_\alpha} \widehat{H_\Sigma}(R^{op})(t_\beta, t_\alpha) \\ &= \bigvee_{t_\beta, t_\alpha} \widehat{H_\Sigma}(R)^{op}(t_\beta, t_\alpha) \\ &= \widehat{T}(R)^{op}(\beta, \alpha)\end{aligned}$$

holds. □

Lemma 3.6.2. *Given a finitary functor T presented by (H_Σ, ε) and a $\mathscr{V}$-relation $R : A \nrightarrow B$, the $\mathscr{V}$-lifting $\widehat{T}(R)$ can be computed by restricting to bases:*

$$\widehat{T}(R)(\alpha, \beta) = \widehat{T}(R \restriction_{\mathsf{Base}(\alpha) \nrightarrow \mathsf{Base}(\beta)})(\alpha, \beta).$$

Proof. Observe that by definition of $\widehat{T}$

$$\widehat{T}(R \restriction_{\mathsf{Base}(\alpha) \nrightarrow \mathsf{Base}(\beta)})(\alpha, \beta) = \bigvee_{t_\alpha, t_\beta} \widehat{H_\Sigma}(R \restriction_{\mathsf{Base}(\alpha) \nrightarrow \mathsf{Base}(\beta)})(t_\alpha, t_\beta)$$

holds. Moreover, because every $t_\alpha \in H_\Sigma(\mathsf{Base}(\alpha))$ is an element of $H_\Sigma(A)$ and every $t_\beta \in H_\Sigma(\mathsf{Base}(\beta))$ is an element of $H_\Sigma(B)$, we get that

$$\bigvee_{t_\alpha, t_\beta} \widehat{H_\Sigma}(R \restriction_{\mathsf{Base}(\alpha) \nrightarrow \mathsf{Base}(\beta)})(t_\alpha, t_\beta) \leq \bigvee_{t_\alpha, t_\beta} \widehat{H_\Sigma}(R)(t_\alpha, t_\beta)$$

holds. From this, the inequality

$$\widehat{T}(R \restriction_{\mathsf{Base}(\alpha) \nrightarrow \mathsf{Base}(\beta)})(\alpha, \beta) \leq \widehat{T}(R)(\alpha, \beta)$$

follows immediately. To prove the inequality

$$\widehat{T}(R)(\alpha, \beta) \leq \widehat{T}(R \restriction_{\mathsf{Base}(\alpha) \nrightarrow \mathsf{Base}(\beta)})(\alpha, \beta),$$

we look again at the definition of $\mathscr{V}$-lifting for T:

$$\widehat{T}(R \restriction_{\mathsf{Base}(\alpha) \nrightarrow \mathsf{Base}(\beta)})(\alpha, \beta) = \bigvee_{t_\alpha, t_\beta} \widehat{H_\Sigma}(R \restriction_{\mathsf{Base}(\alpha) \nrightarrow \mathsf{Base}(\beta)})(t_\alpha, t_\beta).$$

We need to prove that

$$\bigvee_{t_\alpha, t_\beta} \widehat{H_\Sigma}(R)(t_\alpha, t_\beta) \leq \bigvee_{t_\alpha, t_\beta} \widehat{H_\Sigma}(R \restriction_{\mathsf{Base}(\alpha) \nrightarrow \mathsf{Base}(\beta)})(t_\alpha, t_\beta).$$

For each pair of terms $t_\alpha \in H_\Sigma A$ and $t_\beta \in H_\Sigma B$ we try to find a pair of terms $t'_\alpha \in H_\Sigma \mathsf{Base}(\alpha)$ and $t'_\beta \in H_\Sigma \mathsf{Base}(\beta)$ such that

$$\widehat{H_\Sigma}(R)(t_\alpha, t_\beta) \leq \widehat{H_\Sigma}(R \restriction_{\mathsf{Base}(\alpha) \nrightarrow \mathsf{Base}(\beta)})(t'_\alpha, t'_\beta). \tag{3.3}$$

We are interested only in pairs $t_\alpha = \sigma(\mathbf{x})$, $t_\beta = \sigma(\mathbf{y})$ for some $\sigma \in Tn$ and $\mathbf{x} : n \to A$, $\mathbf{y} : n \to B$. If the terms t_α and t_β did not have the same operation σ, their lifting would be $\widehat{H_\Sigma}(R)(t_\alpha, t_\beta) = \bot$.

We are going to restrict the vectors $\mathbf{x}$ and $\mathbf{y}$ to get new vectors $\mathbf{x}'$ and $\mathbf{y}'$ such that their codomains are $\mathsf{Base}(\alpha)$ and $\mathsf{Base}(\beta)$ respectively. Then we will find an operation ρ which would ensure that $\rho(\mathbf{x}')$ represents α and $\rho(\mathbf{y}')$ represents β. And finally we shall show that our choice $t'_\alpha = \rho(\mathbf{x}')$ and $t'_\beta = \rho(\mathbf{y}')$ satisfies inequality (3.3) that we need.

Taking the vector $\mathbf{x}$, form the pullback

$$\begin{array}{ccc} P & \xhookrightarrow{p_2} & n \\ {\scriptstyle p_1}\downarrow & & \downarrow{\scriptstyle \mathbf{x}} \\ \mathsf{Base}(\alpha) & \xhookrightarrow[i]{} & A \end{array}$$

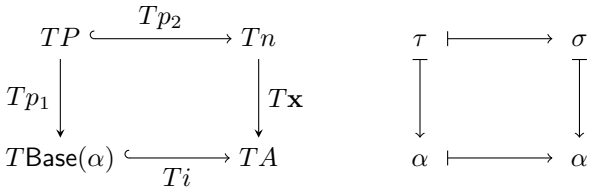
where i is an inclusion map $\mathsf{Base}(\alpha) \hookrightarrow A$ and therefore p_2 is an inclusion as well. For the vector $\mathbf{y}$, form the pullback

$$\begin{array}{ccc} P' & \xhookrightarrow{p'_2} & n \\ {\scriptstyle p'_1}\downarrow & & \downarrow{\scriptstyle \mathbf{y}} \\ \mathsf{Base}(\beta) & \xhookrightarrow[j]{} & B \end{array}$$

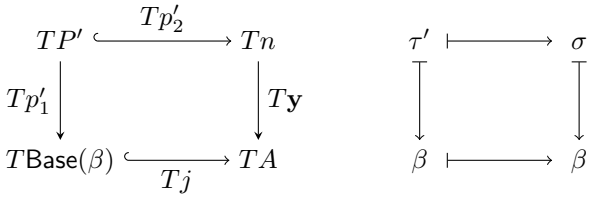
with j being the inclusion map $\mathsf{Base}(\beta) \hookrightarrow B$. By the same reasoning as with the first pullback, the map p'_2 is an inclusion as well. Lastly, form a third pullback

$$\begin{array}{ccc} Q & \xhookrightarrow{q_2} & P' \\ {\scriptstyle q_1}\downarrow & & \downarrow{\scriptstyle p'_2} \\ P & \xhookrightarrow[p_2]{} & n, \end{array}$$

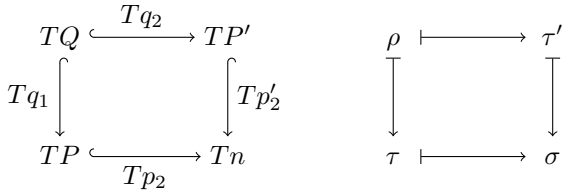
which gives us a categorical description of the intersection $Q = P \cap P'$. We shall now use the property that T preserves weak pullbacks and apply the functor to the three pullbacks we have just formed. The weak pullback



allows us to find an operation $\tau \in TP$ for $\alpha \in T\mathsf{Base}(\alpha)$ and $\sigma \in Tn$. Similarly, the weak pullback



finds an operation $\tau' \in TP'$ for $\beta \in T\mathsf{Base}(\beta)$ and $\sigma \in Tn$. The last weak pullback in the following diagram



tells us that there is an operation $\rho \in TQ$ such that it projects onto τ and τ'. We can say that ρ is an operation since $Q \cong k$ for some $k \leq n$. Observe now that we have two vectors $\mathbf{x}' = (p_1 \circ q_1) : Q \to \mathsf{Base}(\alpha)$ and $\mathbf{y}' = (p_1' \circ q_2) : Q \to \mathsf{Base}(\beta)$, for which it holds that $\rho(\mathbf{x}')$ represents α and $\rho(\mathbf{y}')$ represents β. The vectors $\mathbf{x}'$ and $\mathbf{y}'$ have as codomains the respective bases. We now have to prove that

$$\widehat{H_\Sigma}(R)(\sigma(\mathbf{x}), \sigma(\mathbf{y})) \leq \widehat{H_\Sigma}(R \restriction_{\mathsf{Base}(\alpha) \nrightarrow \mathsf{Base}(\beta)})(\rho(\mathbf{x}'), \rho(\mathbf{y}'))$$

holds. From the definition of $\mathscr{V}$-lifting of polynomial functors and from the fact that $\otimes = \wedge$, we get that

$$\widehat{H_\Sigma}(R)(\sigma(\mathbf{x}), \sigma(\mathbf{y})) = \bigwedge_{i<n} R(\mathbf{x}(i), \mathbf{y}(i))$$

and

$$\widehat{H_\Sigma}(R \restriction_{\mathsf{Base}(\alpha) \nrightarrow \mathsf{Base}(\beta)})(\rho(\mathbf{x}'), \rho(\mathbf{y}')) = \bigwedge_{i<k} R(\mathbf{x}'(i), \mathbf{y}'(i)).$$

From the commutativity of the weak pullback diagrams, we see that the direct image of k under $\mathbf{x}'$ is contained in the direct image of n under $\mathbf{x}$:

$$\mathbf{x}'[k] \subseteq \mathbf{x}[n].$$

It is also true that

$$\mathbf{y}'[k] \subseteq \mathbf{y}[n]$$

holds. Then we can finally deduce that

$$\bigwedge_{i<n} R(\mathbf{x}(i), \mathbf{y}(i)) \leq \bigwedge_{i<k} R(\mathbf{x}'(i), \mathbf{y}'(i)).$$

Therefore we know that the inequality

$$\bigvee_{t_\alpha, t_\beta} \widehat{H_\Sigma}(R)(t_\alpha, t_\beta) \leq \bigvee_{t_\alpha, t_\beta} \widehat{H_\Sigma}(R \restriction_{\mathsf{Base}(\alpha) \nrightarrow \mathsf{Base}(\beta)})(t_\alpha, t_\beta)$$

holds and this in turn implies that we have proven the inequality

$$\widehat{T}(R)(\alpha, \beta) \leq \widehat{T}(R \restriction_{\mathsf{Base}(\alpha) \nrightarrow \mathsf{Base}(\beta)})(\alpha, \beta),$$

and the proof is complete. □

Lemma 3.6.3. *$\mathscr{V}$-lifting of finitary functors commutes with restrictions (Definition 1.2.13), that is, for a relation $R : A \nrightarrow B$ and two subsets $A' \subseteq A$ and $B' \subseteq B$, the following equality*

$$\widehat{T}(R \restriction_{A' \nrightarrow B'}) = \widehat{T}(R) \restriction_{TA' \nrightarrow TB'}$$

holds.

Proof. We proceed by induction. First we prove the lemma for polynomial functors and then we use presentations to prove the lemma for general finitary functors.

- Constant functor C: $\widehat{C}(R \restriction_{A' \nrightarrow B'}) = \mathsf{id}_C = \mathsf{id}_C = \widehat{C}(R) \restriction_{C \nrightarrow C}$.
- Identity functor Id: $\widehat{Id}(R \restriction_{A' \nrightarrow B'}) = R \restriction_{A' \nrightarrow B'} = \widehat{Id(R)} \restriction_{A' \nrightarrow B'}$.
- Sum of functors $T + T'$:

$$\begin{aligned}
\widehat{T + T'}(R \restriction_{A' \nrightarrow B'}) &= (\widehat{T} \boxplus \widehat{T'})(R \restriction_{A' \nrightarrow B'}) \\
&= \widehat{T}(R \restriction_{A' \nrightarrow B'}) \boxplus \widehat{T'}(R \restriction_{A' \nrightarrow B'}) \\
&= \widehat{T}(R) \restriction_{TA' \nrightarrow TB'} \boxplus \widehat{T'}(R) \restriction_{T'A' \nrightarrow T'B'} \\
&= (\widehat{T} \boxplus \widehat{T'})(R) \restriction_{(T+T')A' \nrightarrow (T+T')B'} .
\end{aligned}$$

- Product of functors $T \times T'$:

$$\begin{aligned}
\widehat{T \times T'}(R \restriction_{A' \nrightarrow B'}) &= (\widehat{T} \boxtimes \widehat{T'})(R \restriction_{A' \nrightarrow B'}) \\
&= \widehat{T}(R \restriction_{A' \nrightarrow B'}) \boxtimes \widehat{T'}(R \restriction_{A' \nrightarrow B'}) \\
&= \widehat{T}(R) \restriction_{TA' \nrightarrow TB'} \boxtimes \widehat{T'}(R) \restriction_{T'A' \nrightarrow T'B'} \\
&= (\widehat{T} \boxtimes \widehat{T'})(R) \restriction_{(T \times T')A' \nrightarrow (T \times T')B'} .
\end{aligned}$$

Consider a presentation $\varepsilon : H_\Sigma \to T$: for any pair $\alpha \in TA'$, $\beta \in TB'$, we want to check whether

$$\widehat{T}(R \restriction_{A' \nrightarrow B'}) = \widehat{T}(R) \restriction_{TA' \nrightarrow TB'}$$

holds. This is immediate from the fact $\widehat{T}$ commutes with restriction to bases:

$$\begin{aligned}\widehat{T}(R \restriction_{A' \nrightarrow B'})(\alpha, \beta) &= \widehat{T}(R \restriction_{\mathsf{Base}(\alpha) \nrightarrow \mathsf{Base}(\beta)})(\alpha, \beta) \\ &= \widehat{T}(R)(\alpha, \beta) \\ &= \widehat{T}(R) \restriction_{TA' \nrightarrow TB'} (\alpha, \beta).\end{aligned}$$

□

3.7 Examples of $\mathscr{V}$-liftings

We are now going to show examples of $\mathscr{V}$-liftings for functors that will then be used in Chapter 4 to define many-valued logics for coalgebras of these functors.

Example 3.7.1 (Lifting of the stream functor)**.** Given a $\mathscr{V}$-relation $R : A \nrightarrow B$ and a functor $Z \times \mathsf{Id}$ (recall Example 2.1.1), the lifted $\mathscr{V}$-relation $\widehat{Z \times \mathsf{Id}}(R) : Z \times A \nrightarrow Z \times B$ is defined this way:

$$\widehat{Z \times \mathsf{Id}}(R)((z_1, a)(z_2, b)) = \begin{cases} R(a, b) & \text{if } z_1 = z_2 \\ \bot & \text{otherwise.} \end{cases}$$

Example 3.7.2 (Lifting of the finitary powerset)**.** Given the relation $R : A \nrightarrow B$ and two elements $\alpha \in \mathcal{P}_\omega A$, $\beta \in \mathcal{P}_\omega B$, we look at the Example 2.4.8 to see how to find out whether $\overline{\mathcal{P}_\omega}(R)(\alpha, \beta)$ holds. We know that α and β are in the lifted relation $\overline{\mathcal{P}_\omega}(R)$ when the following two properties

$$\begin{aligned}&\forall a \in \alpha \; \exists b \in \beta : R(a, b) \\ &\forall b \in \beta \; \exists a \in \alpha : R(a, b)\end{aligned}$$

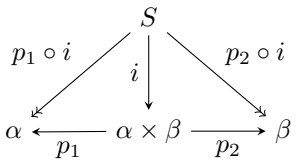
hold. The $\mathscr{V}$-lifting for $\mathcal{P}_\omega$ takes a $\mathscr{V}$-relation $R : A \nrightarrow B$ and constructs a $\mathscr{V}$-relation $\widehat{\mathcal{P}_\omega}(R) : \mathcal{P}_\omega A \nrightarrow \mathcal{P}_\omega B$. For two subsets $\alpha \subseteq A$ and $\beta \subseteq B$, we get that

$$\widehat{\mathcal{P}_\omega}(R)(\alpha, \beta) = \bigvee_{t_\alpha, t_\beta} \widehat{H_\Sigma}(R)(t_\alpha, t_\beta),$$

where (H_Σ, ε) is the canonical presentation of $\mathcal{P}_\omega$. However, if we computed the $\mathscr{V}$-lifting explicitly using this presentation, we would find out that the result is the same when we use the List presentation of $\mathcal{P}_\omega$. We are therefore going to compute the $\mathscr{V}$-lifting using the List presentation, which is slightly easier to compute with. See that

$$\widehat{\mathcal{P}_\omega}(R)(\alpha, \beta) = \bigvee_{l_\alpha, l_\beta} \widehat{\mathsf{List}}(R)(l_\alpha, l_\beta),$$

where $l_\alpha = \mathbf{x} \in \mathsf{List}(A)$ and $l_\beta = \mathbf{y} \in \mathsf{List}(B)$ are the lists representing α and β. When the Heyting algebra $\mathscr{V}$ is a Gödel chain (Definition 1.2.8), we can use its linearity and compute the value of $\widehat{\mathcal{P}_\omega}(R)(\alpha, \beta)$ by finding the best subset $S \subseteq \alpha \times \beta$ such that



the composition of the inclusion map with product projections p_1 and p_2 yields surjective maps and the value

$$\bigwedge_{(a,b)\in S} R(a,b)$$

is maximal. Observe that we have a canonical subset S with a special shape: for each $a \in \alpha$ we choose $b_a \in \beta$ such that $R(a, b_a)$ is maximal. Analogously, for each $b \in \beta$ we choose $a_b \in \alpha$ such that $R(a_b, b)$ is maximal. We then put $S = \{(a, b_a) \mid a \in \alpha\} \cup \{(a_b, b) \mid b \in \beta\}$.

See that this view is a generalisation of the situation with standard relation lifting, because $\overline{\mathcal{P}_\omega}(R)(\alpha, \beta)$ holds exactly if we have a subset $S \subseteq \alpha \times \beta$ where for each $a \in \alpha$ we have some b such that $(a, b) \in S$, and for each $b \in \beta$ we have some a such that $(a, b) \in S$; and moreover, for all $(a, b) \in S$ it holds that $R(a, b)$.

We can also write explicitly that

$$\widehat{\mathcal{P}_\omega}(R)(\alpha,\beta) = \left(\bigwedge_{a\in\alpha}\bigvee_{b\in\beta} R(a,b)\right) \wedge \left(\bigwedge_{b\in\beta}\bigvee_{a\in\alpha} R(a,b)\right),$$

which is a direct generalisation of the Egli-Milner lifting.

Example 3.7.3 (Lifting of $\mathscr{V}$-powerset)**.** Let $\mathscr{V}$ now be a complete Heyting algebra. The $\mathscr{V}$-lifting for the functor $\mathcal{P}_\mathcal{V}$ is slightly more complicated. Given again a $\mathscr{V}$-relation $R : A \nrightarrow B$, we can ask how much does

$$\widehat{\mathcal{P}_\mathcal{V}}(R)(\alpha, \beta)$$

hold for some $\alpha \in \mathscr{V}(A)$ and $\beta \in \mathscr{V}(B)$. Observe that $\alpha : A \to V$ can now be viewed as a "fuzzy subset" of A and $\beta : B \to V$ as a fuzzy subset of B. The value $\alpha(a)$ for some $a \in A$ just tells us the weight of a in α.

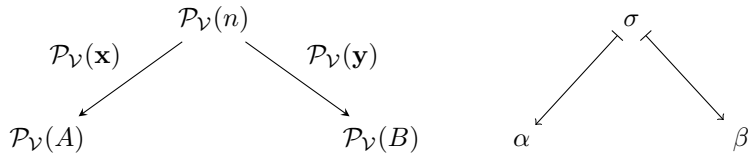
By looking at the definition of $\mathscr{V}$-lifting, we see that

$$\widehat{\mathcal{P}_\mathcal{V}}(R)(\alpha,\beta) = \bigvee_{t_\alpha, t_\beta} H_\Sigma(R)(t_\alpha, t_\beta),$$

holds. To be able to compute the $\mathscr{V}$-lifting more explicitly, let us assume that $\mathcal{P}_\mathcal{V}$ is a Gödel chain in the rest of the example. In that case, due to linearity of $\mathscr{V}$, there is just one ideal witness (t_α, t_β) which determines the value of the lifting. We shall determine which pair that is. Suppose that

$$\widehat{\mathcal{P}_\mathcal{V}}(R)(\alpha,\beta) \neq \bot$$

holds. Then it must be the case that t_α and t_β share the same operation, that is, they are of the form $t_\alpha = \sigma(\mathbf{x})$ and $t_\beta = \sigma(\mathbf{y})$ for some $\sigma \in \mathcal{P}_\mathcal{V}(n)$ and for some two vectors $\mathbf{x} : n \to A$ and $\mathbf{y} : n \to B$. This already tells us something about the shape of the elements α and β. Observe the following diagram



and see that we can conclude that for each $a \in A$ which is in the support of α (meaning that $\alpha(a) > \bot$), there has to exist an element $b \in B$ such that $\alpha(a) \leq \beta(b)$. It remains to show what choice of σ, $\mathbf{x}$ and $\mathbf{y}$ gives the largest value of $\mathscr{V}$-lifting for α and β. We know that

$$\widehat{\mathcal{P}_\mathcal{V}}(R)(\alpha, \beta) = \bigwedge_{i<n} R(\mathbf{x}(i), \mathbf{y}(i)),$$

so we will try to maximise the value of $R(\mathbf{x}(i), \mathbf{y}(i))$ for each i. We can construct σ, $\mathbf{x}$ and $\mathbf{y}$ step by step. Since $\mathcal{P}_\mathcal{V}$ is finitary, we know that α and β have finite support and the construction is going to terminate. Let s_α be the ordered list of elements from the support of α and p be the size of s_α. In the same manner, let s_β be the ordered list of elements from the support of β and q the size of s_β. Then we set $n = p + q$ and start defining σ, $\mathbf{x}$ and $\mathbf{y}$.

For each $i \in \mathbb{N}$ such that $i < p$, let us set

$$\mathbf{x}(i) = s_\alpha(i)$$

and

$$\sigma(i) = \alpha(s_\alpha(i)).$$

By this setting we will ensure that $\sigma(\mathbf{x})$ represents α. Now we shall fill the respective part of the vector $\mathbf{y}$ so that we maximise the value of the $\mathscr{V}$-lifting. Therefore, for each $i < p$ we set that

$$\mathbf{y}(i) = b$$

where b is an element from s_β such that $\sigma(i) \leq \beta(b)$ and which maximises $R(s_\alpha(i), b)$. We know that such an element exists, and we have to force $\sigma(i) \leq \beta(b)$, because otherwise $\sigma(\mathbf{y})$ could no longer represent β. Thus we have completed the first part of constructing $\mathbf{x}$, $\mathbf{y}$ and σ. Now for each $i \in \mathbb{N}$ such that $i < q$, let us set

$$\mathbf{y}(i + p) = s_\beta(i)$$

and

$$\sigma(i + p) = \beta(s_\beta(i)).$$

The vector $\mathbf{y}$ and the operation σ are now completely defined and together they represent β. We now only need to complete the definition of the vector $\mathbf{x}$. Similarly as in the first part of the construction, for each $i < q$ we set

$$\mathbf{x}(i + p) = a$$

where a is an element from s_α such that $\sigma(i + p) \leq \alpha(a)$ and which maximises $R(a, s_\beta(i + p))$. The construction of the terms t_α and t_β is now complete.

Since the construction of the best terms is quite complicated, we now rephrase the main result. Let us have a $\mathscr{V}$-relation $R : A \nrightarrow B$ and two elements

$\alpha \in \mathcal{P}_{\mathcal{V}}(A)$, $\beta \in \mathcal{P}_{\mathcal{V}}(B)$. We compute the $\mathscr{V}$-lifting $\widehat{\mathcal{P}_{\mathcal{V}}}(R)(\alpha, \beta)$: for each $a \in A$ which belongs to the support of α and find a $b \in B$ with $\alpha(a) \leq \beta(b)$ such that $v_a = R(a, b)$ is maximal. Then for each $b \in B$ which belongs to the support of β we find an element $a \in A$ with $\beta(b) \leq \alpha(a)$ such that $v_b = R(a, b)$ is maximal. We can then compute that

$$\widehat{\mathcal{P}_{\mathcal{V}}}(R)(\alpha, \beta) = \bigwedge \left(\{ v_a \mid a \in A \} \cup \{ v_b \mid b \in B \} \right).$$

To state an explicit formula, we can say that

$$\widehat{\mathcal{P}_{\mathcal{V}}}(R)(\alpha, \beta) = \left(\bigwedge_{a \in \mathsf{supp}(\alpha)} \bigvee_{\substack{b \in \mathsf{supp}(\beta) \\ \alpha(a) \leq \beta(b)}} R(a, b) \right) \wedge \left(\bigwedge_{b \in \mathsf{supp}(\beta)} \bigvee_{\substack{a \in \mathsf{supp}(\alpha) \\ \beta(b) \leq \alpha(a)}} R(a, b) \right)$$

holds, where $\mathsf{supp}(\alpha)$ marks the support of α.

Chapter 4

Many-valued coalgebraic logics

In the previous chapter we have introduced the notion of many-valued lifting, $\mathscr{V}$-lifting, which we are now going to use to define many-valued coalgebraic logic. Since the $\mathscr{V}$-lifting is a generalisation of the standard two-valued relation lifting, the syntax and semantics for the logic very much resembles the syntax and semantics introduced in Section 2.7. After introducing the logic in its generality, we will look at specific examples of logics for various systems. Then we shall study the notion of bisimulation which comes from the $\mathscr{V}$-lifting, and finally we shall prove that our logic is expressive in the case that $\mathscr{V}$ is a Gödel chain.

4.1 Syntax and semantics

Let us fix a complete Heyting algebra $\mathscr{V}$ with the operations $\wedge$, $\vee$ and $\to$ (for residuum). We are going to introduce the syntax which resembles the syntax of Moss' logic from Definition 2.7.1. We are going to alter the connectives to deal with the fact that we are working with a Heyting algebra and not the two-valued Boolean algebra. Moreover, we include the elements of the Heyting algebra into the syntax of our language. We shall use these elements in the proof of expressivity of the introduced logic, which we shall call Moss' $\mathscr{V}$-logic, or shortly $\mathscr{V}$-logic.

Definition 4.1.1 (Syntax of Moss' $\mathscr{V}$-logic)**.** With a fixed set of atomic propositions At, we construct the language $\mathcal{L}$ of the coalgebraic logic inductively (i going through $\mathbb{N}$):

$$\begin{array}{lcll} \mathcal{L}_0 & \ni & \varphi ::= \bot \mid v \mid s \mid \varphi \wedge \varphi \mid \varphi \vee \varphi \mid \varphi \to \varphi & \\ \mathcal{L}_{T_i} & = & T(\mathcal{L}_i) & \\ \mathcal{L}_{i+1}: & \ni & \varphi ::= \bot \mid v \mid s \mid \varphi \wedge \varphi \mid \varphi \vee \varphi \mid \varphi \to \varphi \mid \nabla\alpha, & \text{where } \alpha \in \mathcal{L}_{T_i} \\ \mathcal{L} & = & \bigcup_i \mathcal{L}_i, & \end{array}$$

where $v \in V$, $s \in \mathsf{At}$.

The definition of semantics uses the operations of the Heyting algebra $\mathscr{V}$.

Definition 4.1.2 (Semantics of Moss' $\mathscr{V}$-logic)**.** For a coalgebra $c : A \to TA$ together with the evaluation function $\mathsf{ev}_a : \mathsf{At} \to V$ for every state $a \in A$, define the forcing $\mathscr{V}$-relation $\Vdash_c \colon A \nrightarrow \mathcal{L}$ inductively as follows:

$$\begin{array}{lcl} a \Vdash_c \bot & = & \bot, \\ a \Vdash_c v & = & v, \\ a \Vdash_c s & = & \mathsf{ev}_a(s), \\ a \Vdash_c \varphi \wedge \psi & = & a \Vdash_c \varphi \wedge a \Vdash_c \psi, \\ a \Vdash_c \varphi \vee \psi & = & a \Vdash_c \varphi \vee a \Vdash_c \psi, \\ a \Vdash_c \varphi \to \psi & = & a \Vdash_c \varphi \to a \Vdash_c \psi, \\ a \Vdash_c \nabla\alpha & = & \widehat{T}(\Vdash_c)(c(a), \alpha), \end{array}$$

where we again denote by v the elements from V and by s the atomic propositions from At.

4.2 Examples of logics

In this section we are going to show examples of $\mathscr{V}$-logics for functors whose $\mathscr{V}$-liftings we have computed in Section 3.7.

Example 4.2.1 (Logic for streams)**.** In the case of streams for the functor $Z \times \mathsf{Id}$, we see that the syntax of the logic is

$$\varphi ::= \bot \mid v \mid s \mid \varphi \wedge \varphi \mid \varphi \vee \varphi \mid \varphi \to \varphi \mid \nabla(z, \varphi),$$

where $z \in Z$. For a coalgebra $c : A \to Z \times A$ and a state $a \in A$, we shall look at the meaning of the modal formula $\nabla(z, \varphi)$. Let $c(a) = (z', a')$. Then

$$a \Vdash_c \nabla(z, \varphi) = \begin{cases} a' \Vdash_c \varphi & \text{if } z = z' \\ \bot & \text{otherwise.} \end{cases}$$

Example 4.2.2 (Logic for the functor $\mathcal{P}_\omega$)**.** In the case of the functor $\mathcal{P}_\omega$, the syntax of the logic is

$$\varphi ::= \bot \mid v \mid s \mid \varphi \wedge \varphi \mid \varphi \vee \varphi \mid \varphi \to \varphi \mid \nabla\alpha,$$

where α is a finite set of formulas from $\mathcal{L}$. Given a coagebra $c : A \to \mathcal{P}_\omega A$ and considering that $\mathscr{V}$ is a Gödel chain, the semantics of the modality ∇ is defined as follows:

$$a \Vdash_c \nabla\alpha = \left(\bigwedge_{a' \in c(a)} \bigvee_{\varphi \in \alpha} a' \Vdash_c \varphi \right) \wedge \left(\bigwedge_{\varphi \in \alpha} \bigvee_{a' \in c(a)} a' \Vdash_c \varphi \right).$$

Observe that $\mathcal{P}_\omega$-coalgebras are the same structures as those considered by Bou et al. in [14] as crisp Kripke frames. Our logical language is different, but the box and diamond modalities they introduce are interdefinable with the ∇ modality presented here.

Example 4.2.3 (Logic for the functor $\mathcal{P}_\mathcal{V}$)**.** We are going to show the $\mathscr{V}$-logic semantics of the ∇ modality for the functor $\mathcal{P}_\mathcal{V}$. Let us again have a coalgebra

$c : A \to \mathcal{P}_{\mathcal{V}} A$ and consider a Gödel chain $\mathscr{V}$. Then, as we have computed in Section 3.7, the semantics of ∇ is given by the following formula:

$$a \Vdash_c \nabla\alpha = \left(\bigwedge_{a' \in \mathsf{supp}(c(a))} \bigvee_{\substack{\varphi \in \mathsf{supp}(\alpha) \\ c(a)(a') \leq \alpha(\varphi)}} a' \Vdash_c \varphi \right) \wedge \left(\bigwedge_{\varphi \in \mathsf{supp}(\alpha)} \bigvee_{\substack{a' \in \mathsf{supp}(c(a)) \\ \alpha(\varphi) \leq c(a)(a')}} a' \Vdash_c \varphi \right).$$

4.3 Bisimulations

The standard two-valued relation lifting allows us to define bisimulations in a succint way: we say that for two given coalgebras $c : A \to TA$, $d : B \to TB$, a relation $R : A \nrightarrow B$ is a bisimulation if for any pair $R(a, b)$ it follows that $\overline{T}(B)(c(a), d(b))$ (see Definition 2.6.2). We say that the states a,b related by R are bisimilar. In the case that T preserves weak pullbacks, bisimilarity precisely captures behavioural equivalence (Definition 2.6.1).

In this section we shall look at the notion of $\mathscr{V}$-bisimulation which we can define with use of $\mathscr{V}$-lifting as a direct generalisation of standard bisimulation. We will prove that this $\mathscr{V}$-bisimulation closely corresponds with the notion of the two-valued bisimulation.

Definition 4.3.1 ($\mathscr{V}$-bisimulation)**.** A $\mathscr{V}$-relation $R : A \nrightarrow B$ is a *$\mathscr{V}$-bisimulation* for coalgebras $c : A \to TA$ and $d : B \to TB$ if for every $a \in A$ and $b \in B$

$$R(a, b) \leq \widehat{R}(c(a), d(b))$$

holds.

Definition 4.3.2 ($\mathscr{V}$-bisimilarity)**.** For coalgebras $c : A \to TA$, $d : B \to TB$ and a pair of states $a \in A$ and $b \in B$, we call a and b *$\mathscr{V}$-bisimilar* if there is a $\mathscr{V}$-bisimulation R such that $R(a, b) > \bot$.

Remark 4.3.3. From the properties of the $\mathscr{V}$-relation lifting, namely that it commutes with opposites and is functorial, it follows that $\mathscr{V}$-bisimilarity is reflexive, transitive and symmetric, i.e. an equivalence relation.

Lemma 4.3.4 (Bisimilarity implies $\mathscr{V}$-bisimilarity)**.** *A standard bisimulation can be interpreted as a $\mathscr{V}$-bisimulation. Therefore, if two states are bisimilar, they are $\mathscr{V}$-bisimilar as well.*

Proof. We are going to use the fact that $\mathscr{V}$-lifting is an extension of the two-valued lifting, see Lemma 3.3.8 and Lemma 3.5.5. In short, the diagram

$$\begin{array}{ccc} \mathscr{V}\text{-mat} & \xrightarrow{\widehat{T}} & \mathscr{V}\text{-mat} \\ \mathcal{I} \uparrow & & \uparrow \mathcal{I} \\ \mathsf{Rel} & \xrightarrow{\overline{T}} & \mathsf{Rel} \end{array}$$

commutes. Therefore, given a bisimulation $R : A \nrightarrow B$, we can prove that $\mathcal{I}(R) : A \nrightarrow B$ is a $\mathscr{V}$-bisimulation. By the fact that R is a bisimulation, we know that $R(a, b)$ implies $\overline{T}(R)(c(a), d(b))$. Then we want to prove that

$$\mathcal{I}(R)(a, b) \leq \widehat{T}(\mathcal{I}(R))(c(a), d(b)).$$

We know that either $\mathcal{I}(R)(a, b) = \bot$ or $\mathcal{I}(R)(a, b) = \mathsf{e}$. In the first case the inequality holds trivially. When $\mathcal{I}(R)(a, b) = \mathsf{e}$, we know that $R(a, b)$ holds from the definition of $\mathcal{I}$. Since R is a bisimulation, we can deduce that $\overline{T}(R)(c(a), d(b))$ holds, and therefore $\mathcal{I}(\overline{T}(R))(c(a), d(b)) = \mathsf{e}$. Since

$$\mathcal{I} \circ \overline{T} = \widehat{T} \circ \mathcal{I},$$

we now know that

$$\widehat{T}(\mathcal{I}(R))(c(a), d(b)) = \mathsf{e}.$$

But this implies that

$$\mathcal{I}(R)(a, b) \leq \widehat{T}(\mathcal{I}(R))(c(a), d(b)),$$

and $\mathcal{I}(R)$ is truly a $\mathscr{V}$-bisimulation.

This in turn implies that two bisimilar states a and b are $\mathscr{V}$-bisimilar as well, since by being bisimilar we know there is a bisimulation R such that $R(a, b)$ holds, and therefore the relation $\mathcal{I}(R)$ is a $\mathscr{V}$-bisimulation for which $\mathcal{I}(R)(a, b) > \bot$. □

Definition 4.3.5 (Weak cut)**.** Given a $\mathscr{V}$-relation $R : A \nrightarrow B$, we say that its *weak cut* is a relation $\mathcal{W}(R) : A \nrightarrow B$, where $\mathcal{W}(R)(a, b)$ holds if and only if $R(a, b) > \bot$.

Lemma 4.3.6 ($\mathscr{V}$-bisimilarity implies bisimilarity)**.** *For any $\mathscr{V}$-bisimulation, its weak cut is a bisimulation. Therefore, if two states are $\mathscr{V}$-bisimilar, they are bisimilar as well.*

Proof. Having a $\mathscr{V}$-bisimulation $R : A \nrightarrow B$, is its weak cut $\mathcal{W}(R) : A \nrightarrow B$ a bisimulation?

We know that $R(a, b) \leq \widehat{T}(R)(c(a), d(b))$. If we want to be sure that $\mathcal{W}(R)$ is a bisimulation, we need to know whether for every a,b with $\mathcal{W}(R)(a, b)$ we have that $\overline{T}(\mathcal{W}(R))(c(a), d(b))$.

Let us assume that $\mathcal{W}(R)(a, b)$ holds. This means that $R(a, b) > \bot$. Therefore $\widehat{T}(R)(c(a), d(b)) > \bot$ follows from the fact that R is a $\mathscr{V}$-bisimulation. This means that there are two terms $t_{c(a)}, t_{d(b)}$ such that

$$\widehat{H_\Sigma}(R)(t_{c(a)}, t_{d(b)}) > \bot$$

holds. Analysing the terms more closely, this means that we have $t_{c(a)} = \sigma(\mathbf{x})$ and $t_{d(b)} = \sigma(\mathbf{y})$ for some $\sigma \in T(n)$ and $\mathbf{x} : n \to A$, $\mathbf{y} : n \to B$ such that

$$R(\mathbf{x}(i), \mathbf{y}(i) > \bot$$

for every $i < n$. Therefore for every $i < n$ we know that

$$\mathcal{W}(R)(\mathbf{x}(i), \mathbf{y}(i))$$

holds and because of that we can infer that

$$\overline{H_\Sigma}(R)(t_{c(a)}, t_{d(b)})$$

holds. By Lemma 3.4.3 this in turn implies $\overline{T}(\mathcal{W}(R))(c(a), d(b))$, which tells us that $\mathcal{W}(R)$ is indeed a bisimulation: this is what we wanted to prove.

Now we can easily infer that two states a,b which are $\mathscr{V}$-bisimilar, are bisimilar as well. There is a $\mathscr{V}$-bisimulation $R : A \nrightarrow B$ for which we have $R(a, b) > \bot$. We now know that $\mathcal{W}(R)$ is a bisimulation and that $\mathcal{W}(R)(a, b)$ holds. The states a and b are then indeed bisimilar. □

Remark 4.3.7. We could have defined the notion of $\mathscr{V}$-bisimilarity by saying that a and b are bisimilar if and only if there is a $\mathscr{V}$-bisimulation R such that

$$R(a, b) = \mathsf{e}.$$

Observe that this notion coincides with the notion of $\mathscr{V}$-bisimilarity given above, and therefore coincides with the notion of bisimilarity: in particular, in Lemma 4.3.4 we actually proved that when a and b are bisimilar, then the standard bisimulation R connecting them provides us with a $\mathscr{V}$-bisimulation with $\mathcal{I}(R)(a, b) = \mathsf{e}$.

Moreover observe that (assuming $\bot \neq \mathsf{e}$ in $\mathscr{V}$) if there is a $\mathscr{V}$-bisimulation R with $R(a, b) = \mathsf{e}$, it is at the same time true that $R(a, b) > \bot$ and by Lemma 4.3.6 a and b are bisimilar.

4.4 Expressivity

Now we are going to show that the $\mathscr{V}$-logic we have introduced is expressive. This will follow immediately from the fact that $\mathscr{V}$-lifting is an extension of the two-valued lifting. Moreover, if $\mathscr{V}$ is a Gödel chain, we shall prove a stronger version of expressiveness for our logic.

Let us review the important definitions we are going to use in this section. We say that given two coalgebras $c : A \to TA$ and $d : B \to TB$, their states $a \in A$ and $b \in B$ are *behaviourally equivalent* (Definition 2.6.1) if there is a coalgebra $z : Z \to TZ$ and two coalgebra morphisms $f : A \to Z$ and $g : B \to Z$ such that the following diagram

$$\begin{array}{ccccc} A & \xrightarrow{f} & Z & \xleftarrow{g} & B \\ {\scriptstyle c}\downarrow & & \downarrow{\scriptstyle z} & & \downarrow{\scriptstyle d} \\ TA & \xrightarrow[Tf]{} & TZ & \xleftarrow[Tg]{} & TB, \end{array}$$

commutes and it holds that $f(a) = g(b)$.

We know that Moss' logic, which was introduced in Section 2.7, is expressive: if two states $a \in A$ and $b \in B$ satisfy the same formulas, then the states are behaviourally equivalent. By contraposition, two states that are not behaviourally equivalent can be distinguished by some formula. We shall use this contrapositive statement in the following theorem.

We first prove that the fragment of the language $\mathcal{L}$ without atomic propositions is expressive — this merely follows from the fact that $\mathscr{V}$-lifting extends

the two-valued relation lifting, together with the fact that the Boolean algebra $\{0,1\}$ embeds into $\mathscr{V}$. That is on $\{\bot, \mathsf{e}\}$, the Heyting algebra $\mathscr{V}$ behaves classically.

Theorem 4.4.1 (Expressivity of the closed fragment)**.** *The closed fragment of the $\mathscr{V}$-logic for a finitary functor T is expressive. For two coalgebras $c : A \to TA$ and $d : B \to TB$, if the states $a \in A$ and $b \in B$ are not behaviourally equivalent, then there is a formula $\varphi \in \mathcal{L}$ such that*

$$a \Vdash_c \varphi \neq b \Vdash_d \varphi.$$

Proof. Let us write $\mathcal{L}^M$ for the language of Moss' logic from Section 2.7, and write $\Vdash^M$ for the forcing relation of Moss' logic. We know that this logic is expressive. This means that we can find a formula $\varphi \in \mathcal{L}$ of the Moss' logic such that $a \Vdash^M_c \varphi$ holds and also $b \nVdash^M_d \varphi$ holds. See that if we had a notion of negation in the language $\mathcal{L}$ of Moss' $\mathscr{V}$-logic, the language $\mathcal{L}^M$ would be a subset of $\mathcal{L}$. Let us for this moment define a syntactic shortcut for the pseudo-complement (negation)

$$\neg\psi \equiv \psi \to \bot$$

and add to the language $\mathcal{L}$ the negation connective by introducing this shortcut. Observe that for the elements $\bot \in V$ and $\mathsf{e} \in V$ this gives us semantically the notion of standard negation as we know it from the 2-element Boolean algebra: $\neg\bot$ has the same semantics as e and $\neg\mathsf{e}$ has the same semantics as $\bot$. We shall now prove that if we interpret the formulas of $\mathcal{L}^M$ in Moss' $\mathscr{V}$-logic, we get the same semantics as in the standard Moss' logic. More succintly, we prove that

$$\mathcal{I}(\Vdash^M_c) \; = \Vdash_c \restriction_{A \nrightarrow \mathcal{L}^M} .$$

This is now almost immediate by looking at the semantics of Moss' logic from Definition 2.7.2 and the semantics of $\mathscr{V}$-logic from Definition 4.1.1. The proof follows by induction on the complexity of φ. The only nontrivial case is to prove that for formulas of the form $\nabla\alpha$ we get that

$$a \; \mathcal{I}(\Vdash^M_c) \; \nabla\alpha = a \Vdash_c \nabla\alpha$$

holds. But now we can use Lemma 3.3.8 and Lemma 3.5.5 to conclude the proof. Since the following diagram

$$\begin{array}{ccc} \mathscr{V}\text{-mat} & \xrightarrow{\widehat{T}} & \mathscr{V}\text{-mat} \\ \uparrow{\scriptstyle \mathcal{I}} & & \uparrow{\scriptstyle \mathcal{I}} \\ \mathsf{Rel} & \xrightarrow{\overline{T}} & \mathsf{Rel} \end{array}$$

commutes, we can say that

$$a \Vdash_c \nabla\alpha = \widehat{T}(\Vdash_c)(c(a), \alpha)$$

by definition of semantics for ∇, then

$$\widehat{T}(\Vdash_c)(c(a), \alpha) = \widehat{T}(\mathcal{I}(\Vdash^M_c))(c(a), \alpha)$$

follows from the inductive hypothesis. By the commutativity of the diagram above we may infer that

$$\widehat{T}(\mathcal{I}(\Vdash_c^M))(c(a),\alpha) = \mathcal{I}(\overline{T}(\Vdash_c^M))(c(a),\alpha),$$

and from this last line we see that $a \Vdash_c \nabla\alpha = \mathsf{e}$ if and only if $a \Vdash_c^M \nabla\alpha$ holds, and similarly $a \Vdash_c \nabla\alpha = \bot$ if and only if we know that $a \nVdash_c^M \nabla\alpha$.

We have therefore shown that the formula φ which distinguishes the states a and b in the standard Moss' logic can be used to distinguish the states in the $\mathscr{V}$-logic as well. □

Observe that in the proof of expressivity of the closed fragment we did not need any special property of the Heyting algebra $\mathscr{V}$ (apart from knowing that it has two distinct elements $\bot$ and e). When $\mathscr{V}$ is moreover a Gödel chain, we can provide a finer analysis and a direct proof of adequacy and expressivity of $\mathscr{V}$-logic in Theorem 4.4.4.

First we introduce the technical notion of a *strict cut*.

Definition 4.4.2 (Strict cut)**.** Given a $\mathscr{V}$-relation $R : A \nrightarrow B$, we say that its *strict cut* is a relation $\mathcal{S}(R) : A \nrightarrow B$ such that $\mathcal{S}(R)(a,b)$ holds if and only if $R(a,b) = \mathsf{e}$.

Lemma 4.4.3. *Let $\mathscr{V}$ be a Gödel chain. For a finitary functor T and a relation $R : A \nrightarrow B$, the following inequality*

$$\mathcal{S}\widehat{T}(R) \leq \overline{T}\mathcal{S}(R)$$

holds.

Proof. We want to prove that

$$\mathcal{S}\widehat{T}(R)(\alpha,\beta) \leq \overline{T}\mathcal{S}(R)(\alpha,\beta)$$

holds for any $\alpha \in TA$ and $\beta \in TB$. Let $\mathcal{S}\widehat{T}(R)(\alpha,\beta)$ hold. This means that

$$\widehat{T}(R)(\alpha,\beta) = \mathsf{e}.$$

By the definition of $\mathscr{V}$-lifting,

$$\bigvee_{t_\alpha, t_\beta} \widehat{H_\Sigma}(R)(t_\alpha, t_\beta) = \mathsf{e}.$$

Since $\mathscr{V}$ is a chain, this implies that there are two terms t_α and t_β representing α and β, which moreover satisfy

$$\widehat{H_\Sigma}(R)(t_\alpha, t_\beta) = \mathsf{e}.$$

Therefore $t_\alpha = \sigma(\mathbf{x})$ and $t_\beta = \sigma(\mathbf{y})$ for some $\sigma \in Tn$ and $\mathbf{x} : n \to A$, $\mathbf{y} : n \to B$. Now these two vectors satisfy that

$$\bigwedge_{i<n} R(\mathbf{x}(i), \mathbf{y}(i)) = \mathsf{e},$$

which can be true only if for every $i < n$ it holds that $R(\mathbf{x}(i), \mathbf{y}(i)) = \mathsf{e}$. Therefore, we know that for every $i < n$ it holds that $\mathcal{S}(R)(\mathbf{x}(i), \mathbf{y}(i))$. By Lemma 3.4.3, we can now deduce that there is a witness $t_w \in H_\Sigma(\mathcal{S}(R))$ such that $H_\Sigma(p_1)(t_w) = t_\alpha$ and $H_\Sigma(p_2)(t_w) = t_\beta$ and that

$$\overline{T}\mathcal{S}(R)(\alpha, \beta)$$

holds. This concludes the proof. □

Theorem 4.4.4 (Expressivity of $\mathscr{V}$-logic)**.** *Let $\mathscr{V}$ be a Gödel chain and $c : A \to TA$ and $d : B \to TB$ coalgebras.*

1. *(Adequacy.) Let $R : A \nrightarrow B$ be a $\mathscr{V}$-bisimulation. Then for all φ from the closed fragment of $\mathcal{L}$*
$$(R(a, b) \wedge a \Vdash_c \varphi) \leq b \Vdash_d \varphi.$$
2. *(Expressivity.) The relation $R : A \nrightarrow B$ defined as*
$$R(a, b) \text{ iff for all closed } \varphi \in \mathcal{L} : a \Vdash_c \varphi = b \Vdash_d \varphi$$
is a bisimulation.

Remark 4.4.5. Observe that part 1 of Theorem 4.4.4 indeed entails adequacy: suppose a and b are $\mathscr{V}$-bisimilar. Therefore (recall from Remark 4.3.7) there is a $\mathscr{V}$-bisimulation R with $R(a, b) = e$. Then by part 1 of Theorem 4.4.4 it holds that

$$a \Vdash_c \varphi \leq b \Vdash_d \varphi$$

and by a similar argument using R^{op} (which is again a $\mathscr{V}$-bisimulation), the other inequality

$$b \Vdash_d \varphi \ \leq a \Vdash_c \varphi$$

follows. Thus it holds that $a \Vdash_c \varphi = b \Vdash_d \varphi$.

Part 2 of Theorem 4.4.4 entails expressivity: if a and b are not $\mathscr{V}$-bisimilar, then they are not bisimilar and from the definition of R in part 2 it follows that $\neg R(a, b)$ and therefore there exists a formula $\varphi \in \mathcal{L}$ such that

$$a \Vdash_c \varphi \neq b \Vdash_d \varphi.$$

Proof of Theorem 4.4.4. We first prove part 2.

The (two-valued) relation R relates states that satisfy the same formulas. Suppose $R(a, b)$ holds. We then have to prove that $\overline{T}(R)(c(a), d(b))$ holds to prove that R is a bisimulation.

For each $a_i \in \mathsf{Base}(c(a))$ and $b_j \in \mathsf{Base}(d(b))$ such that $R(a_i, b_j)$ does not hold, fix a formula φ_{ij} such that

$$a_i \Vdash_c \varphi_{ij} \neq b_j \Vdash_d \varphi_{ij}.$$

Then define a function $f : \mathsf{Base}(c(a)) \to \mathcal{L}$ as follows (where $\phi \leftrightarrow \psi$ is a syntactic shortcut for $\phi \to psi \wedge \psi \to \phi$):

$$a_i \mapsto \bigwedge_{\neg R(a_i, b_j)} (\varphi_{ij} \leftrightarrow a_i \Vdash_c \varphi_{ij}) .$$

From the fact that $v \leftrightarrow v' = \mathsf{e}$ iff $v = v'$, we can deduce that

$$a_i \Vdash_c f(a_i) = \mathsf{e} \tag{4.1}$$

and, if it does not hold that $R(a_i, b_j)$, then

$$b_j \Vdash_d f(a_i) < \mathsf{e}. \tag{4.2}$$

Now we are going to use the function

$$Tf : T\mathsf{Base}(c(a)) \to T\mathcal{L}$$

to define an element $\alpha \in T\mathcal{L}$ to be

$$\alpha = (Tf)(c(a)).$$

Observe that from equation (4.1) we see that the following diagram

$$\mathsf{Base}(c(a)) \overset{\mathsf{Gr}(f)}{\underset{\Vdash_c}{\Downarrow}} \mathcal{L}$$

commutes, and by the fact that $\mathscr{V}$-lifting for T is a 2-functor (Lemma 3.5.4), we see that the following diagram

$$T\mathsf{Base}(c(a)) \overset{\widehat{T}(\mathsf{Gr}(f))}{\underset{\widehat{T}(\Vdash_c)}{\Downarrow}} T\mathcal{L}$$

commutes as well. From the fact that $\mathscr{V}$-lifting commutes with graphs, that is,

$$\widehat{T}(\mathsf{Gr}(f)) = \mathsf{Gr}(Tf),$$

we infer that it also holds that

$$T\mathsf{Base}(c(a)) \overset{\mathsf{Gr}(Tf)}{\underset{\widehat{T}(\Vdash_c)}{\Downarrow}} T\mathcal{L}$$

commutes. And since $\alpha = (Tf)(c(a))$, we now know that $c(a)\widehat{T}(\Vdash_c)\alpha = \mathsf{e}$. By the definition of the semantics for ∇, this allows us to say that

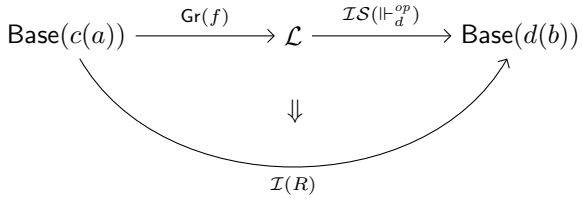
$$a \Vdash_c \nabla\alpha = \mathsf{e}$$

holds. By assumption, we know that $R(a, b)$ holds, and therefore $b \Vdash_d \nabla\alpha = \mathsf{e}$ as well, since a and b satisfy all the formulas to the same extent. From the definition of the semantics for ∇ we know that $d(b)\widehat{T}(\Vdash_d)\alpha = \mathsf{e}$. Using the strict cut, we can write that

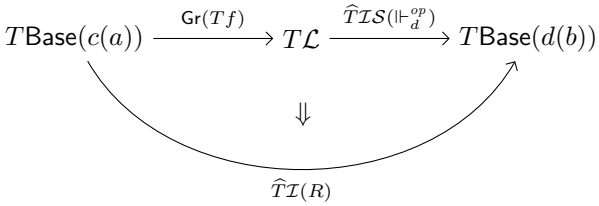
$$\mathcal{S}(\widehat{T}(\Vdash_d))(d(b), \alpha) \tag{4.3}$$

holds.

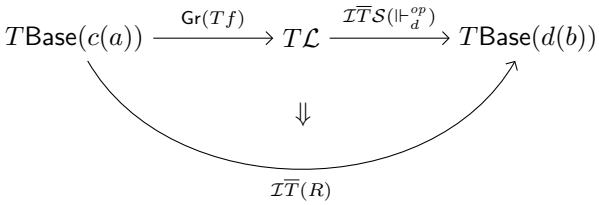
Now by equation 4.2 we know that when $R(a_i, b_j)$ does not hold, then $b_j \Vdash_d f(a_i) < \mathsf{e}$, and therefore it does not hold that $b_j \mathcal{S}(\Vdash_d) f(a_i)$. This can be succintly written by the following commutative diagram:



Applying $\widehat{T}$ to this diagram, we obtain that



commutes. Because $\mathscr{V}$-lifting is an extension of the two-valued lifting, we can write that



holds. Observe that since $\alpha = (Tf)(c(a))$, it would now suffice to show that

$$\overline{T}\mathcal{S}(\Vdash_d^{op})(\alpha, d(b))$$

is true in order to prove that $c(a)\overline{T}(R)d(b)$ holds. By Lemma 4.4.3 we know that

$$\mathcal{S}\widehat{T}(\Vdash_d^{op}) \leq \overline{T}\mathcal{S}(\Vdash_d^{op})$$

holds, and by equation 4.3 we know that $\mathcal{S}(\widehat{T}(\Vdash_d))(d(b), \alpha)$ holds, so it is also true that

$$\overline{T}\mathcal{S}(\Vdash_d)(d(b), \alpha).$$

Because relation lifting commutes with opposites,

$$\overline{T}\mathcal{S}(\Vdash_d^{op})(\alpha, d(b))$$

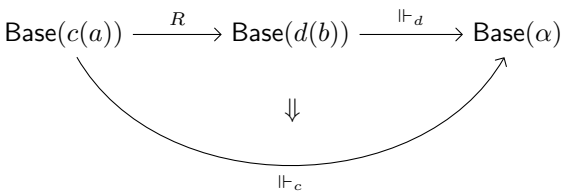
holds as well. We have therefore shown that

$$c(a)\overline{T}(R)d(b)$$

holds, and R is a bisimulation.

Now we prove part 1 of the theorem by an easy induction on the complexity of φ. The only interesting case is when $\varphi = \nabla\alpha$, where $\alpha \in T\mathcal{L}$.

Consider the following diagram:



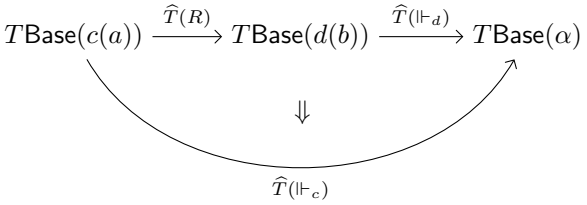
It commutes: Fix $a' \in \mathsf{Base}(c(a))$ and $\psi \in \mathsf{Base}(\alpha)$. We have to show that

$$\bigvee_{b'} R(a', b') \wedge b' \Vdash_d \psi \leq a \Vdash_c \psi$$

holds. If we fix b', then by induction hypothesis it holds that

$$R(a', b') \wedge b' \Vdash_d \psi \leq a \Vdash_c \psi.$$

Now by lifting the above diagram we get that



commutes. It shows that

$$\bigvee_{\sigma \in T\mathsf{Base}(d(b))} \left(\widehat{T}(R)(c(a), \sigma) \wedge \widehat{T}(\Vdash_d)(\sigma, \alpha)\right) \leq \widehat{T}(\Vdash_c)(c(a), \alpha).$$

Thus, since $d(b) \in T\mathsf{Base}(d(b))$, it holds that

$$\widehat{T}(R)(c(a), d(b)) \wedge \widehat{T}(\Vdash_d)(d(b), \alpha) \leq \widehat{T}(\Vdash_c)(c(a), \alpha).$$

Recall now that

$$R(a, b) \leq \widehat{T}(R)(c(a), d(b))$$

and by definition of semantics of ∇, the following two equalities

$$\widehat{T}(\Vdash_d)(d(b), \alpha) = b \Vdash_d \nabla\alpha$$

$$\widehat{T}(\Vdash_c)(c(a), \alpha) = a \Vdash_c \nabla\alpha$$

hold. Thus it holds that

$$R(a, b) \wedge b \Vdash_d \nabla\alpha \leq a \Vdash_c \nabla\alpha$$

and the proof is complete. □

Note 4.4.6. In the diagrams we have used implicitly the fact that $\mathscr{V}$-lifting commutes with restrictions.

Remark 4.4.7. Observe that the same proof would work for the full language $\mathcal{L}$ of our $\mathscr{V}$-logic and not just for the closed fragment. It would be enough just to incorporate the notion of *atomic harmony* into the notion of $\mathscr{V}$-bisimilarity: for two states a,b to be $\mathscr{V}$-bisimilar, we would demand that

$$a \Vdash_c s = b \Vdash_d s$$

holds for all atomic propositions s.

We have now concluded that the $\mathscr{V}$-logic we have introduced is expressive when $\mathscr{V}$ is a Gödel chain. In the next chapter we shall summarise the results of the thesis and talk about possible future work.

Summary

In this chapter we shall summarise the results of the thesis and propose possibilites for future work.

Results

We have proved that there is a way to define many-valued relation lifting for a large class of functors with the use of functor presentations. This result is an extension of the classical relation lifting result [52]. We have moreover proved some additional nice properties of the many-valued relation lifting that have proved useful. Our definition of many-valued relation lifting works when we consider complete Heyting algebras as the algebras describing many-valuedness.

We have used the introduced notion of many-valued relation lifting to define semantics for many-valued coalgebraic logic, which is a direct generalisation of Moss' logic. This logic has been shown to be expressive when we allow the truth-values of the semantics to be taken from a Gödel chain. The proof used properties of many-valued relation lifting which we have proved beforehand. Lastly we have shown some interesting examples of the introduced logic.

Future work

It would be natural to try and find an axiomatisation of our logic and present proofs of soundness and completeness. We should therefore study the notion of a distributive law arising from the relation lifting.

Since we proved the expressivity results for $\mathscr{V}$-logic in the case of $\mathscr{V}$ being a Gödel chain, it would be nice to find out whether there is a more direct way to define $\mathscr{V}$-lifting for Gödel chains.

In this thesis we have studied only Moss' logic. We could therefore try to study modalities arising from a many-valued logical connection (or many-valued predicate liftings) as well.

Bibliography

[1] P. Aczel. *Non-Well-Founded Sets*, volume 14 of *CSLI Lecture Notes*. CSLI Publications, 1988.

[2] P. Aczel and N. Mendler. A final coalgebra theorem. In D. H. Pitt, A. Poigne, and D.E. Rydeheard, editors, *Category Theory and Computer Science*, volume 389. Springer, 1989.

[3] J. Adámek. Introduction to coalgebra. *Theory and Applications of Categories*, 14:157–199, 2005.

[4] J. Adámek. A logic of coequations. In *Computer Science Logic, 19th International Workshop (CSL 2005)*, pages 70–86, 2005.

[5] J. Adámek, H. P. Gumm, and V. Trnková. Presentation of set functors: A coalgebraic perspective. *Journal of Logic and Computation*, 20(5):991–1015, 2010.

[6] S. Awodey. *Category Theory (Oxford Logic Guides)*. Oxford University Press, USA, 2 edition, August 2010.

[7] M. Barr. Relational algebras. In S. Mac Lane, H. Applegate, M. Barr, B. Day, E. Dubuc, Phreilambud, A. Pultr, R. Street, M. Tierney, and S. Swierczkowski, editors, *Reports of the Midwest Category Seminar IV*, volume 137 of *Lecture Notes in Mathematics*, pages 39–55. Springer Berlin / Heidelberg, 1970.

[8] M. Barr and Ch. Wells, editors. *Category theory for computing science, 2nd ed.* Prentice Hall International Ltd., Hertfordshire, UK, 1995.

[9] M. Bílková, A. Kurz, D. Petrisan, and J. Velebil. Relation lifting, with an application to the many-valued cover modality. *submitted (2012)*.

[10] M. Bílková, A. Palmigiano, and Y. Venema. Proof systems for the coalgebraic cover modality. In *Advances in Modal Logic*, pages 1–21, 2008.

[11] Marta Bílková, Alexander Kurz, Daniela Petrişan, and Jiří Velebil. Relation liftings on preorders and posets. In A. Corradini, B. Klin, and C. Cîrstea, editors, *Algebra and Coalgebra in Computer Science*, volume 6859 of *Lecture Notes in Computer Science*, pages 115–129. Springer Berlin Heidelberg, 2011.

[12] P. Blackburn, M. de Rijke, and Y. Venema. *Modal Logic*. Cambridge Tracts in Theoretical Computer Science. Cambridge University Press, 2002.

[13] M. Bonsangue and A. Kurz. Duality for logics of transition systems. In V. Sassone, editor, *Foundations of Software Science and Computational Structures, 8th International Conference (FoSSaCS'05)*, volume 3441, pages 455–469. Springer, 2005.

[14] F. Bou, F. Esteva, L. Godo, and R. Rodríguez. On the minimum many-valued modal logic over a finite residuated lattice. *Journal of Logic and Computation*, 21(5):739–790, October 2011.

[15] L. Běhounek, U. Bodenhofer, and P. Cintula. Relations in fuzzy class theory:: Initial steps. *Fuzzy Sets and Systems*, 159(14):1729 – 1772, 2008. Formal Methods for Fuzzy Mathematics, Approximation and Reasoning, Part I.

[16] L. Běhounek and P. Cintula. Fuzzy logics as the logics of chains. *Fuzzy Sets and Systems*, 157(5):604 – 610, 2006. What is Fuzzy Logic.

[17] X. Caicedo and R. Rodriguez. Standard gödel modal logics. *Studia Logica*, 94:189–214, 2010.

[18] P. Cintula, P. Hájek, and C. Noguera. *Handbook of Mathematical Fuzzy Logic - volume 1*. Number 37 in Studies in Logic, Mathematical Logic and Foundations. College Publications, London, 2011.

[19] Hofmann D. Clementino, M. M. On extensions of lax monads. *Theory and Applications of Categories [electronic only]*, 13:41–60, 2004.

[20] C. Cîrstea, A. Kurz, D. Pattinson, L. Schröder, and Y. Venema. Modal logics are coalgebraic. *The Computer Journal*, 54(1):31–41, 2011.

[21] M. Fitting. Many-valued modal logics. In *Fundamenta Informaticae*, pages 365–448. Kluwer Academic Publishers, 1992.

[22] M. Fitting. Many-valued modal logics ii. *Fundamenta Informaticae*, 17, 1992.

[23] N. Galatos, P. Jipsen, T. Kowalski, and H. Ono. *Residuated Lattices: an algebraic glimpse at substructural logics*, volume 151. Elsevier Science, 2007.

[24] J. Goguen. *L-fuzzy sets*. Defense Technical Information Center, 1966.

[25] H. P. Gumm. *Universelle Coalgebra*.

[26] P. Hájek. *Metamathematics of Fuzzy Logic*. Trends in Logic. Springer, 2001.

[27] P. Hájek. Making fuzzy description logic more general. *Fuzzy Sets and Systems*, 154(1):1 – 15, 2005.

[28] P. Hájek, D. Harmancová, F. Esteva, P. Garcia, and L. Godo. On modal logics for qualitative possibility in a fuzzy setting. In *Proceedings of the Tenth international conference on Uncertainty in artificial intelligence*, UAI'94, pages 278–285, San Francisco, CA, USA, 1994. Morgan Kaufmann Publishers Inc.

[29] G. Hansoul and B. Teheux. Completeness results for many-valued lukasiewicz modal systems and relational semantics. 2006.

[30] J. E. Hopcroft, R. Motwani, and J. D. Ullman. *Introduction to Automata Theory, Languages, and Computation: Introduction to Automata Theory, Languages, and Computation - Introduction to Automata Theory, Languages, and Computation: International Edition 3/E*. Addison Wesley, 2007.

[31] B. Jacobs. Many-sorted coalgebraic modal logic: a model-theoretic study. *Theoretical Informatics and Applications*, 35(1):31–59, 2001.

[32] B. Jacobs. *Introduction to Coalgebra: Towards Mathematics of States and Observations*. 2012.

[33] C. Kupke, A. Kurz, and Y. Venema. Completeness of the finitary moss logic. In *Advances in Modal Logic*, pages 193–217, 2008.

[34] C. Kupke, A. Kurz, and Y. Venema. Completeness for the coalgebraic cover modality. *Logical Methods in Computer Science*, 8(3), 2012.

[35] C. Kupke and Y. Venema. Closure properties of coalgebra automata. In *Proceedings of the Twentieth Annual IEEE Symposium on Logic in Computer Science (LICS 2005)*, pages 199–208. IEEE Computer Society Press, 2005.

[36] C. Kupke and Y. Venema. Coalgebraic automata theory: basic results. *Logical Methods in Computer Science*, 4(4), 2008.

[37] A. Kurz. Coalgebras and modal logic, essli notes. 2001.

[38] A. Kurz. Specifying coalgebras with modal logic. *Theoretical Computer Science*, 260(1,2):119 – 138, 2001. Coalgebraic Methods in Computer Science 1998.

[39] A. Kurz. Coalgebras and their logics, 2006. SIGACT News.

[40] A. Kurz and R. Leal. Modalities in the stone age: A comparison of coalgebraic logics. *Theoretical Computer Science*, 430(0):88 – 116, 2012. Mathematical Foundations of Programming Semantics (MFPS XXV).

[41] F. W. Lawvere and S. H. Schanuel. *Conceptual mathematics - a first introduction to categories*. Cambridge University Press, 1997.

[42] T. Lukasiewicz and U. Straccia. Managing uncertainty and vagueness in description logics for the semantic web. *Web Semantics: Science, Services and Agents on the World Wide Web*, 6(4):291 – 308, 2008. Semantic Web Challenge 2006/2007.

[43] S. Mac Lane. *Categories for the Working Mathematician (Graduate Texts in Mathematics)*. Springer, 2nd edition, September 1998.

[44] G. Metcalfe and N. Olivetti. Proof systems for a gödel modal logic. In *Proceedings of the 18th International Conference on Automated Reasoning with Analytic Tableaux and Related Methods*, TABLEAUX '09, pages 265–279, Berlin, Heidelberg, 2009. Springer-Verlag.

[45] L. Moss. Coalgebraic logic. *Annals of Pure and Applied Logic*, 96, 1999.

[46] S. Ovchinnikov. An introduction to fuzzy relations. In D. Dubois and H. Prade, editors, *Fundamentals of Fuzzy Sets*, volume 7 of *The Handbooks of Fuzzy Sets Series*, pages 233–259. Springer US, 2000.

[47] Rosický J. Paseka, J. Quantales. *Fundamental Theories of Physics*.

[48] D. Pattinson. Coalgebraic modal logic: soundness, completeness and decidability of local consequence. *Theoretical Computer Science*, 309:177–193, 2003.

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

[50] D. Sangiorgi and J. J. M. M. Rutten. *Advanced Topics in Bisimulation and Coinduction*. Cambridge Tracts in Theoretical Computer Science. Cambridge University Press, 2011.

[51] L. Schröder. Expressivity of coalgebraic modal logic: The limits and beyond. *Theoretical Computer Science*, 390:230–247, 2008.

[52] V. Trnková. Relational automata in a category and their languages. In Marek Karpinski, editor, *Fundamentals of Computation Theory*, volume 56 of *Lecture Notes in Computer Science*, pages 340–355. Springer Berlin / Heidelberg, 1977.

[53] Y. Venema. Automata and fixed point logics: a coalgebraic perspective. *Electronic Notes in Theoretical Computer Science*, 204:2004, 2004.

[54] W. Wechler. *Universal algebra for computer scientists*. EATCS monographs on theoretical computer science. Springer-Verlag, 1992.

[55] L. A. Zadeh. Fuzzy sets. *Information and Control*, 8(3):338 – 353, 1965.

[56] L. A. Zadeh. Similarity relations and fuzzy orderings. *Information Sciences*, 3(2):177 – 200, 1971.