

UNIVERSITY OF LEICESTER

THESIS SUBMITTED FOR THE DEGREE OF
DOCTOR OF PHILOSOPHY

Finitary Logics for Coalgebras with Branching

Author:
Christian KISSIG

Supervisor:
Dr. Alexander KURZ
Advisor:
Prof. Dr. Reiko HECKEL

February 2012

February 2012, London

© Christian Kissig, 2007-2012

The author reserves all rights to produce and distribute copies of this thesis, earlier drafts, and excerpts thereof.

Abstract

The purpose of this dissertation is to further previous work on coalgebras as infinite state-based transition systems and their logical characterisation with particular focus on infinite regular behaviour and branching.

Finite trace semantics is well understood [DR95] for nondeterministic labelled transition systems, and has recently [Jac04, HJS06] been generalised to a coalgebraic level where monads act as branching types for instance, of nondeterministic choice. Finite trace semantics then arises through an inductive construction in the Kleisli-category of the monad. We provide a more comprehensive definition of finite trace semantics, allowing for finitary branching types in Chapter 5. In Chapter 6 we carry over the ideas behind our definition of finite trace semantics to define infinite trace semantics.

Coalgebraic logics [Mos99] provide one approach to characterising states in coalgebras up to bisimilarity. Coalgebraic logics are Boolean logics with the modality ∇ . We define the Boolean dual of ∇ in the negation-free fragment of finitary coalgebraic logics in Chapter 7, showing that finitary coalgebraic logics are essentially negation free. Our proof is largely based on the previously established completeness of finitary coalgebraic logics [KKV08].

Finite trace semantics induces the notion of finite trace equivalence. In Chapter 8 we define coalgebraic logics for many relevant branching and transition types characterising states of coalgebras with branching up to finite trace equivalence. Under further assumptions we show that these logics are expressive.

Coalgebra automata allow us to state finitary properties over infinite structures essentially by a fix-point style construction. We use the dualisation of ∇ from Chapter 7 to

prove that coalgebra automata are closed under complementation in Chapter 10. This result completes a Rabin style [Rab69] correspondence between finitary coalgebraic logics and coalgebra automata for finitary transition types, begun in [Ven04, KV05].

The semantics of coalgebra automata is given in terms of parity graph games [GTW02]. In Chapter 9 we show how to structure parity graph games into rounds using the notion of players power [vB02] and how to normalise the interaction pattern between the players per round. From the latter we obtain the coinductive principle of game bisimulation.

Languages accepted by coalgebra automata are called regular. Regularity is commonly [Sip96, HMU03] disproved using the pumping lemma for regular languages. We define regular languages of coalgebras and prove a pumping lemma for these languages in Chapter 11.

Acknowledgements

First of all I thank Alexander Kurz. In his enthusiasm and dedication to the scholarly study of mathematics he has become a role model for me in more than the research presented in this dissertation. I also thank his research group at Leicester, which over the time consisted of Nick Bezhanishvili, Tadeusz Litak, Daniela Petrişan, and Suzuki Tomoyuki.

My scientific work would not have been possible without the influence of some great minds. Bart Jacobs sparked the definition of finite trace logics as it stands by a few notes he passed to me during a workgroup meeting in the Sierra Nevada, Spain in 2008, and after that he continued to support me with various comments and feedback. Yde Venema was my supervisor during my MSc studies in Amsterdam. He invited me later on for a research visit to the Institute of Logic, Language and Computation in 2008, at the end of which we wrote the joint paper on the complementation lemma for coalgebra automata. I am deeply grateful for all that I have learned from working with him.

I would also like to express my gratitude for the inspiring and helpful discussions with Jiří Adámek, Fillippo Bonchi, Vincenzo Ciancia, Roy Crole, Lucas Dixon, Fabio Gadducci, Neil Ghani, Ichiro Hasuo, Bartek Klin, Raul Leal, Alessandra Palmigiano, Dusko Pavlovic, Dirk Pattinson, Petter Remen, Alexandra Silva, Ana Sokolova, Mehrnoosh Sadrzadeh, Sam Staton, Emilio Tuosto, and Jacob Vosmaer.

During my time at Leicester, I helped in organising a series of seminars for PhD students jointly with Stephen Gorton, Daniela Petrişan, Hong-Qi Yu, Martin Birks, Frank Nebel, and Julien Lange. The generous financial support of the computer science department at Leicester enabled us to invite many speakers from various backgrounds and countries.

Following a seminar of Alexander Kurz, I co-initiated a study group in category theory. I would like to thank the other regular members Daniela Petrişan, Mauro Jaskelioff, Ondřej Rypáček, Carl Forsell, and Rob Myers, as much as the lecturers Alexander Kurz, Neil Ghani, and Dirk Pattinson.

In a parallel track I studied quantum logic from a categorical and topological viewpoint. I would like to point out the great efforts of Bob Coecke and the quantum group at Oxford. I gained much understanding of the matter from attending the Quantum, Physics and Logic workshops 2008-2009 and the spring school in the Foundational Structures in Quantum Computation and Information, 2010. I would like to thank the University of Oxford and the University of Leicester for the financial support.

My interest in topological quantum field theories (TQFTs) began with the summer school in Almeria of the Spanish topology meeting in 2009. I would like to thank the organisers and the University of Leicester for the financial support to attend. In Almeria I met Chris Kapulkin, who invited me to the University of Warsaw to give a talk on 3-dimensional topological quantum field theories and knot theory in the QFT seminar, he organised together with Marek Czarnecki. I shared the seminar day with Paweł Traczyk, whose inspiring introduction to knot theory keeps echoing in my mind. Apart from the organisers, I would like to thank the students union of the University of Warsaw and the University of Leicester for the financial support to participate in the QFT seminar.

I finished my dissertation while working as a software developer at the Search Technology Center of Microsoft in London. I would like to thank my colleagues for their understanding and encouragement, in particular Antonio Gulli and Mark Atherton.

The greatest gift I take away from Leicester are the friendships I have gained. I thank Emilio, Eleonora, Julien, Frank, Tadeusz, Daniela, Uğur, Mark, João and Adwoa for the opposite of everything else.

Above all, this work and my life during the last four years would not have been possible without two important women. My mother supported me generously, foremostly with advice and attention. Laura struggled with me, if not more. She kept challenging me and my points of view and at the end helped me understand about my reasons and priorities.

Contents

Abstract	iii
Acknowledgements	v
1 Introduction	1
1.1 Coalgebras in Computer Science	1
1.2 Parity Graph Games	4
1.3 Coalgebra Automata	5
1.4 Branching Types and Monads	7
1.5 The Semantics of Coalgebras	8
1.5.1 Final T -Coalgebra Semantics and T -Bisimilarity	9
1.5.2 Finite Trace Semantics	10
1.5.3 Infinite Trace Semantics	11
1.6 Coalgebraic Logics	13
1.7 Our Contributions in This Thesis	16
1.8 Outline	17
2 Notation	19
2.1 Set Theory	19
2.2 Logic	20
2.3 Category Theory	21

I	Foundations in Category Theory	23
3	A Review of Monads and Algebras over <i>Set</i>	25
3.1	Definition of Monads	25
3.2	Categories of Algebras for Monads	26
3.3	Eilenberg-Moore Categories of Commutative Monads	27
3.4	Examples	30
3.5	Commutative Monads in <i>Set</i>	35
4	Distributive Laws and Functor Liftings	37
4.1	Distributive Laws	37
4.1.1	Examples of Distributive Laws	38
4.1.2	The Existence of Distributive Laws	38
4.1.3	Some Properties of Distributive Laws	39
4.2	Kleisli-Lifting of Functors on <i>Set</i>	40
4.3	Continuous Extensions of Kleisli-Lifted Functors	42
II	Semantics of Coalgebras with Branching	45
5	Finite Trace Semantics	47
5.1	A Review of Generic Trace Theory	48
5.2	Non-Coinductive Finite Trace Semantics in Kleisli Categories	53
5.3	Finite Trace Semantics in Eilenberg-Moore Categories	58
6	Infinite Trace Semantics	61
6.1	Generic Infinite Trace Semantics	62
6.1.1	Change of Perspective	64
6.1.2	Generic Infinite Trace Equivalence	64
6.2	Plausible Continuations in Infinite Traces	65
6.3	Infinite Trace Semantics of Coalgebra Automata	69
6.4	Jacob's Infinite Trace Semantics	72

III	Coalgebraic Logics	75
7	The Complementation Lemma for Finitary Coalgebraic Logic	77
7.1	Preliminaries	78
7.2	A Review of the Completeness of Finitary Coalgebraic Logic	79
7.3	One-Step Semantics of Coalgebraic Logic	80
7.4	Complementation Lemma	81
8	Finitary Coalgebraic Logics for Finite Traces	87
8.1	Dual Adjunctions from Ambimorphic Objects	89
8.2	The Logic Functor	91
8.3	Finite Trace Logics as the Initial Algebra of the Logic Functor	92
8.4	Examples of Finite Trace Logics	94
8.4.1	An Example of a Logic Functor	94
8.4.2	Deterministic Streams	95
8.4.3	Finitarily Nondeterministic Streams	96
8.4.4	Streams with Finitary Graded Branching	96
8.4.5	Finitarily Probabilistic Streams	98
8.4.6	Path-Minimising Streams	99
8.5	Invariance of Finitary Trace Logics under Finite Trace Equivalence	100
8.6	Expressivity of Finitary Coalgebraic Logics for Finite Traces	101
IV	Coalgebraic Automata Theory	109
9	Game Bisimulations in Parity Graph Games	111
9.1	Preliminary Definitions	111
9.2	Unravelling Parity Graph Games	114
9.3	Structuring Parity Graph Games	115
9.4	Normalised Parity Graph Games	119
9.5	Game Bisimulations	120

10	Complementation of Coalgebra Automata	125
10.1	A Review of Nondeterministic Coalgebra Automata	125
10.2	Alternation	127
10.2.1	Alternating Coalgebra Automata	128
10.2.2	Semi-Transalternating Coalgebra Automata	128
10.2.3	Transalternating Coalgebra Automata	129
10.2.4	Basic Positions in Acceptance Games	130
10.3	Equivalence of Coalgebra Automata of various Branching Types	130
10.3.1	From Transalternating to Semi-Transalternating Automata	131
10.3.2	From Semi-Transalternating to Alternating Automata	131
10.4	Closure under Complementation	134
10.4.1	Complementation of Transalternating Coalgebra Automata	134
11	A Pumping Lemma for Regular Languages of Coalgebras in <i>Set</i>	137
11.1	Coalgebras in <i>Set</i> as Graphs	139
11.1.1	Reachable States	139
11.1.2	Generated Subcoalgebras	141
11.1.3	Unravelling Coalgebras in <i>Set</i>	141
11.1.4	Pumping Length for Coalgebras in <i>Set</i>	143
11.2	Pumping Coalgebras in <i>Set</i>	145
11.3	The Pumping Lemma	149
V	Conclusions	155
12	Conclusions	157
12.1	Summary of Contributions	157
12.1.1	Finite Trace Semantics	157
12.1.2	Infinite Trace Semantics	157
12.1.3	Finitary Coalgebraic Logics	158
12.1.4	Finite Trace Logics	158

12.1.5	Game Bisimulations for Parity Graph Games	158
12.1.6	Complementation Lemma for Coalgebra Automata	159
12.1.7	A Pumping Lemma for Regular Languages of Coalgebras in <i>Set</i> .	159
12.2	Some Open Questions and Directions for Future Work	159
12.2.1	Monads and Categories of Algebras	159
12.2.2	Finitary Coalgebraic Logics for Finite Traces	160
A	Set Theory	161
A.1	Basic Set Theory	161
A.2	Order Theory	162
B	Category Theory	163
B.1	Basic Category Theory	163
B.2	Limits and Colimits	168
B.2.1	Filtered Colimits and Finitary Functors	170
B.2.2	Equalisers and Coequalisers	171
B.2.3	Pointwise Construction of Kan Extensions	172
B.3	Monoidal Categories	173
B.4	Category Theory of <i>Set</i>	173
B.4.1	Standard and Weak-Pullback Preserving Functors	174
B.4.2	Relations and Relation Liftings in <i>Set</i>	175
B.4.3	Bases and Redistributions	178
C	Coalgebras	181
	List of Tables	185
	List of Figures	187
	Bibliography	187
	Index	196

Chapter 1

Introduction

1.1 Coalgebras in Computer Science

Algebraic Datatypes In computer science finite datastructures are classically described inductively. Finite binary trees with labels from a set Act , for instance, are either leafs or nodes with two successor binary trees, as in the following OCaml [Rem00] code.

```
type 'a bintree = Leaf of 'a | Node of 'a * 'a bintree * 'a bintree
```

The above type definition is algebraic in the sense that `bintree` assigns an operational meaning to `Leaf`, a unary function on the extension of `bintree`, and to `Node`, a binary function on the extension of `bintree`, for each label in a .

Coalgebras Coalgebraically, a binary tree t is a function $\sigma : S \rightarrow Act + Act \times S \times S$ where S is the set of subsets of t and σ assigns to each subtree $s \in S$ either a label from Act , if s is a leaf of t , or a label and two successor subtrees of s in t . The coalgebraic definition is coinductive [Rut96].

Definition 1.1.1 (*T-Coalgebras*). *Let C be a category, let S be an object in C , and let $T : C \rightarrow C$ be a functor on C , a T -coalgebra is a structure $\mathbb{S} = \langle S, \sigma \rangle$ where $\sigma : S \rightarrow TS$ is a morphism in C . We call S the carrier and T the transition type of \mathbb{S} .*

State-Based Coalgebras Coalgebras have states if they live in concrete categories, such as binary trees in *Set*. In our example above, the subtrees of t form the states of the

Algebras	Coalgebras
Operations	Observations
Congruence	Behavioural Equivalence
Initial Algebra	Final Coalgebra
Induction	Coinduction

Table 1.1: Concepts from Algebra and Coalgebra

coalgebra $\langle S, \sigma \rangle$. There are coalgebras which have no states, for instance coalgebras over locales. In state-based coalgebras we can distinguish states, and then call these coalgebras pointed. Pointed coalgebras play a significant role as models of coalgebra automata as in Chapter 10.

Definition 1.1.2 (Pointed T -Coalgebras). *When we distinguish a state s in a T -coalgebra $\mathbb{S} = \langle S, \sigma \rangle$, we call \mathbb{S} pointed and write $\mathbb{S} = \langle S, \sigma, s \rangle$. Coalgebra morphisms $f : \mathbb{S} \rightarrow \mathbb{S}'$ between pointed coalgebras $\mathbb{S} = \langle S, \sigma, s \rangle$ and $\mathbb{S}' = \langle S', \sigma', s' \rangle$ preserve the distinguished state, that is $s' = f(s)$.*

Coalgebras as Infinite Data Structures Unlike to the inductive definition, the coinductive definition allows us to define infinite binary trees. Table 1.2 shows examples of coalgebras relevant in computer science, more of which can be found in [Rut96, RJ97, Gum99]. More complex applications include

- communication protocols [HK07, Has08],
- object-orient programs [RTJ01, Kis05], and
- operating systems [HTS02].

Table 1.1¹ shows a comparison of concepts known in algebra and in coalgebras.

Next we give a preview of the essential definitions of coalgebras in *Set*, which can be found in introductory texts, for instance [Rut96, Gum99, Jac05]. The following definition is a straightforward concretisation of the definition of coalgebras above.

Definition 1.1.3 (T -Coalgebras in *Set*). *A T -coalgebra in the category *Set* is a structure $\mathbb{S} = \langle S, \sigma \rangle$ consisting of a set S and a function $\sigma : S \rightarrow TS$.*

¹Table 1.1 is an adaptation of Table 1.1 in [Kup06].

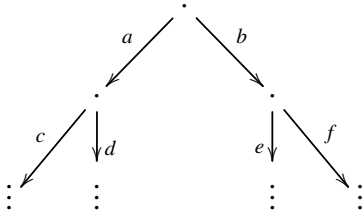
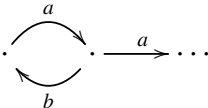
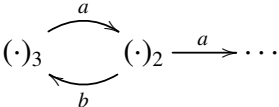
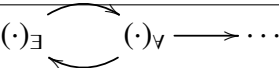
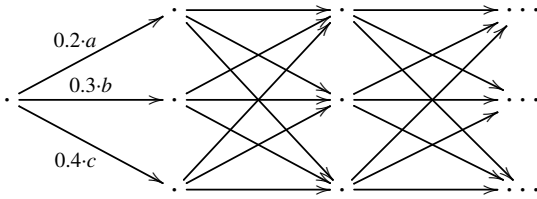
Structure	Example	Transition Type
Streams	$\cdot \xrightarrow{a} \cdot \xrightarrow{b} \cdot \xrightarrow{c} \dots$	$Act \times (-)$
Infinite Binary Trees		$Act \times (-) \times (-)$
Graphs		$\mathcal{P}(Act \times (-))$
Non-Deterministic Coalgebra Automata		$\mathbb{N} \times \mathcal{P}(Act \times (-))$
Parity Graph Games		$\{\exists, \forall\} \times \mathcal{P}(-)$
Hidden Markov Models		$\mathcal{D}_{\leq 1}(Act \times (-))$

Table 1.2: Examples of Coalgebras

We choose a categorical framework for coalgebras in *Set*. Aczel [Acz88], Barwise and Moss [BM04] established a theory of coalgebras over sets purely in set theory. Aczel replaced the wellfoundedness axiom of Zermelo-Fraenkel set theory [Jec06] with the non-wellfoundedness axiom and interpreted the resulting set theory in graph structures. The graphs of coalgebras in *Set*, we use in Chapter 11, generalise the graph models of Zermelo-Fraenkel theory with the non-wellfoundedness axiom from the categorical side.

Informally the transition type of a coalgebra \mathbb{S} can be thought of as providing a structure on the successor states. Coalgebra morphisms preserve the structure given by the transition type as follows.

Definition 1.1.4 (Coalgebra Morphisms). *A T -coalgebra morphism between T -coalgebras $\mathbb{S} = \langle S, \sigma \rangle$ and $\mathbb{S}' = \langle S', \sigma' \rangle$ in *Set* is a function $f : S \rightarrow S'$ making the following commute.*

$$\begin{array}{ccc} S & \xrightarrow{\sigma} & TS \\ f \downarrow & & \downarrow Tf \\ S' & \xrightarrow{\sigma'} & TS' \end{array} \quad (1.1)$$

1.2 Parity Graph Games

We briefly introduce parity graph games. For more details we refer the reader to Chapter 9. Parity graph games are two-player graph games. The vertices of the graph are the positions in the game, the edges the admissible moves. Positions, admissible moves and initial position form the arena of a graph game.

Definition 1.2.1 (Graph Games). *A graph game for two players, 0 and 1, is a structure $\mathcal{G} = \langle V_0, V_1, E, v_I, Acc \rangle$ consisting of*

- *disjoint sets V_0 and V_1 of positions assigned to 0 and 1 respectively,*
- *an edge relation $E \subseteq V_0 \cup V_1 \times V_0 \cup V_1$ of admissible moves,*
- *an initial position $v_I \in V_0 \cup V_1$, and*
- *an acceptance condition $Acc \in (V_0 \cup V_1)^\omega$.*

We have formulated the acceptance condition in the most general way. The acceptance condition is typically formulated in terms of the Büchi, Rabin, Muller, or parity acceptance condition. It has been shown, that the Büchi acceptance condition is strictly less expressive than the latter three acceptance conditions, and that the latter three are equally expressive. The main advantage of the last is that graph games with parity acceptance condition are historyfree determined [Mos91, EJ91, Zie98].

Definition 1.2.2 (Parity Acceptance Condition). *The parity acceptance condition Acc for a game $\mathcal{G} = \langle V_0, V_1, E, v_I, Acc \rangle$ is given in terms of a priority function $\Omega : V_0 \cup V_1 \rightarrow \mathbb{N}$, such that Acc contains precisely those infinite plays $p \in (V_0 \cup V_1)^\omega$ with the largest infinitely often occurring priority from $\{\Omega \in p(i) \mid i \in \omega\}$ being of parity 0.*

1.3 Coalgebra Automata

Coalgebra Automata and Fix-Point Logics Coalgebra automata were introduced by Venema in [Ven04] as an approach to introducing fix-point operators into Moss' coalgebraic logic. Essential to this idea is a correspondence between automaton states and formulas, which dates back to Rabin [Rab69] who used the correspondence to reduce the decidability problem of monadic second order logic of binary (finitely branching) trees, S2S, to the emptiness problem of binary tree automata.

Coalgebra Automata Generalising Classical Automata Venema observed that the classical types of automata, for instance word, tree, or graph automata, share a common structure $\langle Q, \theta, q_I, Acc \rangle$, where

- Q is a (finite) set of states,
- θ is a transition function,
- $q_I \in Q$ is a state distinguished as initial, and
- Acc is an acceptance condition

The transition function varies for each type of recognised input. The following are examples of transition functions for nondeterministic automata.

Type of Input	Transition Function
Words	$\theta : Q \rightarrow \mathcal{P}(Act \times Q)$
Binary Trees	$\theta : Q \rightarrow \mathcal{P}(Act \times Q \times Q)$
Graphs	$\theta : Q \rightarrow \mathcal{P}(Act \times \mathcal{P}Q)$

Table 1.3: Examples of Transition Functions for Nondeterministic Automata

Table 1.4 lists branching types commonly used for automata. We will formally introduce branching types in Chapter 3 as monads in *Set*.

Type of Choice	Transition Function
Determinism	$\theta : Q \rightarrow Act \times Q$
Nondeterminism	$\theta : Q \rightarrow \mathcal{P}(Act \times Q)$
Alternation	$\theta : Q \rightarrow \mathcal{P}_{\exists}\mathcal{P}_{\forall}(Act \times Q)$
Probabilism	$\theta : Q \rightarrow \mathcal{D}_{\leq 1}(Act \times Q)$

Table 1.4: Examples of Branching Types for Word Automata

The separation of the two powerset monads \mathcal{P}_{\exists} and \mathcal{P}_{\forall} is to emphasise, that the two are treated separately for acceptance. Probabilistic automata play no role in this dissertation, but have been added for completeness. The first three have been shown in [KV08, KV08] to be equivalent for all transition types. In Chapter 10 we will introduce two further branching types, which we will show equivalent to the first three.

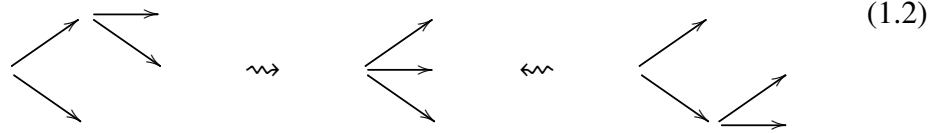
Acceptance Behaviour of Coalgebra Automata The acceptance behaviour of a T -coalgebra automaton in a pointed T -coalgebra is given in terms of an acceptance game, which is a parity graph game played by players \exists verifying acceptance and \forall disproving acceptance. The major advantage of this type of acceptance condition, is that it is equally expressive to Muller or Rabin acceptance condition, and that parity graph games are historyfree determined.

Coalgebra Automata as Coalgebras In particular in Chapters 6 and 11 we will take a coalgebraic point of view on coalgebra automata. Looking at the definition we see that T -coalgebra automata $\langle Q, \theta, q_I, \Omega \rangle$ contain a pointed \mathcal{PT} -coalgebra $\langle Q, \theta, q_I \rangle$. We can regard the acceptance condition induced by Ω as augmentation, which determines the semantics of $\langle Q, \theta, q_I \rangle$ as we show in Chapter 6.

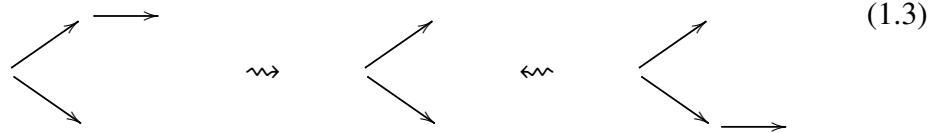
1.4 Branching Types and Monads

In the case of coalgebra automata we have seen several examples of branching types and anticipated that monads suitably formalise branching types. In the following we informally substantiate this claim and show how branching types can be added to coalgebras in general.

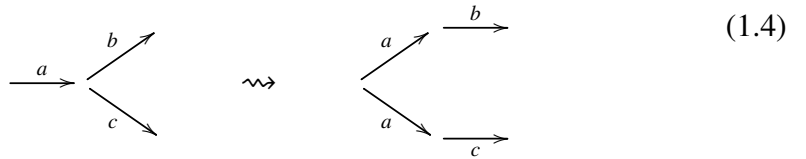
1. *Commutativity and Associativity* Branching types allow to make choices about branching at any stage. Consequently we can take branching upfront.


(1.2)

2. *Deterministic Subbehaviour* Branching types should allow trivial branching, that is where only one branching choice is possible.


(1.3)

3. *Distributivity* Branching distributes over transitions.


(1.4)

The above properties are faithfully represented by monads. A monad $\langle B, \mu, \eta \rangle$ is a functor B with an additional structure given by the natural transformations μ , the *multiplication* of the monad, and η , the *unit* of the monad. These natural transformations are subject to axiom 1. and 2. below.

Axiom 3. determines the interaction of the branching type with the transition type T , in terms of a natural transformation $\pi : TB \Rightarrow BT$, a *distributive law* of B and T .

The axioms correspond to the properties 1.-3. of branching types above.

1. *Multiplication*

$$\begin{array}{ccc}
 BBB & \xrightarrow{\mu_B} & BB \\
 \downarrow B\mu & \searrow & \downarrow \mu \\
 BB & \xrightarrow{\mu} & B
 \end{array} \quad (1.5)$$

2. *Unit*

$$\begin{array}{ccccc}
 B & \xrightarrow{\eta_B} & BB & \xleftarrow{B\eta} & B \\
 \searrow = & & \downarrow \mu & & \swarrow = \\
 & & B & &
 \end{array} \quad (1.6)$$

3. *Distributive Law*

$$\begin{array}{ccc}
 T & \xrightarrow{T\eta} & TB \\
 \downarrow \eta_T & \searrow \pi & \\
 BT & &
 \end{array} \quad
 \begin{array}{ccccc}
 TBB & \xrightarrow{\pi} & BTB & \xrightarrow{B\pi} & BBT \\
 \downarrow T\mu & & & & \downarrow \mu_T \\
 TB & \xrightarrow{\pi} & & & BT
 \end{array} \quad (1.7)$$

1.5 The Semantics of Coalgebras

Coalgebras do not capture coalgebraic behaviour succinctly, as they may contain distinct states with the same behaviour, and not comprehensively, as a single coalgebra may not capture all behaviours admissible for the transition type. Coalgebra semantics takes the role of a succinct and comprehensive structure capturing the behaviour of all coalgebras for a particular transition type.

In this thesis we will consider the following three kinds of semantics for coalgebras:

1. *final coalgebra semantics*, which distinguishes coalgebra states up to *T-bisimilarity*,
2. *finite trace semantics*, which distinguishes coalgebra states up to *finite trace equivalence*,
3. and *infinite trace semantics*, which distinguishes coalgebra states up to *infinite trace equivalence*.

In the following we briefly introduce each kind of semantics. For details on finite and infinite trace semantics we refer to Chapters 5 and 6 respectively.

1.5.1 Final T -Coalgebra Semantics and T -Bisimilarity

Recall that coalgebras and coalgebra morphisms for a transition type T over a base category C form a category, $\mathbf{Coalg}_T(C)$. If $\mathbf{Coalg}_T(C)$ has a final object $\mathbb{Z} = \langle Z, \xi \rangle$, we call \mathbb{Z} the *final T -coalgebra*. For each T -coalgebra $\mathbb{S} = \langle S, \sigma \rangle$ then exists a unique T -coalgebra morphism $f : \mathbb{S} \rightarrow \mathbb{Z}$.

$$\begin{array}{ccc} S & \xrightarrow{\sigma} & TS \\ f \downarrow & & \downarrow Tf \\ Z & \xrightarrow{\xi} & TZ \end{array} \quad (1.8)$$

For coalgebras in \mathbf{Set} , it turns out that f distinguishes states of \mathbb{S} up to T -bisimilarity, which is the largest T -bisimulation.

Being a final object, the final T -coalgebra is the limit of the empty diagram into $\mathbf{Coalg}_T(C)$. For $C = \mathbf{Set}$ and T finitary (accessible) we know [Wor05, AGT09] that the final T -coalgebra is the limit of the final T -sequence. The finitary prefix Seq^T of the final T -sequence in the presence of a final object 1 is defined by

$$\mathit{Seq}^T(n) := T^n 1 \text{ and } \mathit{Seq}^T(n \leq n+1) := T^n !_{T1} \text{ for all } n < \omega \quad (1.9)$$

as in the following diagram

$$1 \xleftarrow{!_{T1}} T1 \xleftarrow{\quad} \dots \xleftarrow{\quad} T^n 1 \xleftarrow{T^n !_{T1}} T^{n+1} 1 \xleftarrow{\quad} \dots \quad (1.10)$$

Definition 1.5.1 (*T -Bisimulations in \mathbf{Set}*). Let $\mathbb{S} = \langle S, \sigma \rangle$ be a T -coalgebra in \mathbf{Set} . We call a relation $R \subseteq S \times S$ a T -bisimulation on \mathbb{S} , if there is a function $r : R \rightarrow TR$ making the following commute.

$$\begin{array}{ccccc}
TS & \xleftarrow{T\pi_1} & TR & \xrightarrow{T\pi_2} & TS \\
\sigma \uparrow & & \uparrow r & & \uparrow \sigma \\
S & \xleftarrow{\pi_1} & R & \xrightarrow{\pi_2} & S
\end{array} \tag{1.11}$$

Definition 1.5.2 (*T-Bisimilarity in Set*). • *T-bisimilarity on a T-coalgebra \mathbb{S} is the largest T-bisimulation on \mathbb{S} .*

- *We say points s and s' in a coalgebra $\mathbb{S} = \langle S, \sigma \rangle$ are bisimilar, if there is a bisimulation $R \subseteq S \times S$ with $(s, s') \in R$.*
- *Pointed coalgebras $\mathbb{S} = \langle S, \sigma, s \rangle$ and $\mathbb{S}' = \langle S', \sigma', s' \rangle$ are bisimilar if (s, s') are bisimilar in $\langle S, \sigma \rangle + \langle S', \sigma' \rangle$.*

In this dissertation we will use that *T-bisimilarity* over a *T-coalgebra* $\mathbb{S} = \langle S, \sigma \rangle$ is contained in the kernel of a cone of \mathbb{S} over the final sequence of *T*, which is defined inductively as follows.

$$f_0 := !_S \text{ and } f_{n+1} := Tf_n \circ \sigma. \tag{1.12}$$

as in the following diagram.

$$\begin{array}{ccccccc}
& & S & \xrightarrow{\sigma} & TS & & \\
& f_0 \swarrow & & \searrow f_n & \searrow Tf_n & & \\
\{*\} & \longleftarrow \dots \longleftarrow & T^n\{*\} & \xleftarrow{T^n !_S} & T^{n+1}\{*\} & \longleftarrow \dots
\end{array} \tag{1.13}$$

In Appendix C we prove the following lemma.

Lemma 1.5.3. *For any T-coalgebra $\mathbb{S} = \langle S, \sigma \rangle$, any T-bisimulation R on \mathbb{S} is contained in the kernel of the cone $f : S \Rightarrow Seq^T$.*

1.5.2 Finite Trace Semantics

Classically [DR95], finite traces in a nondeterministic labelled transition system are given by words of consecutive labels. Recall that nondeterministic transition systems labelled

in a set Act are coalgebras of transition type $\mathcal{P}(\sqrt{} + Act \times (-))$, where $\sqrt{} + Act \times (-)$ is the transition type of words as coalgebras. Some results about the latter case can be found in [PT99].

When we generalise from words to coalgebras of other transition types T , T -coalgebras assume the role of the finite traces of \mathcal{PT} -coalgebras. This observation is due to Jacobs [Jac04]. In finite trace semantics, we distinguish \mathcal{P} and T as branching and transition type, respectively. As a branching type \mathcal{P} embodies non-determinism.

Final coalgebra semantics is not suitable for coalgebras with branching, such as coalgebra automata accepting finite input, as it distinguishes states, which are not bisimilar but have the same finite traces. The argument can be found in [BIM95]. For example the states x and x' in the coalgebras below are finite trace equivalent, but not bisimilar.

$$\begin{array}{ccc}
 x & \xrightarrow{a} & y \\
 & & \swarrow b \quad \searrow c \\
 & & z_0 \quad z_1
 \end{array}
 \qquad
 \begin{array}{ccc}
 x' & \xrightarrow{a} & y_0 \\
 & \searrow a & \\
 & & y_1 \\
 & & \xrightarrow{c} z_1
 \end{array}
 \qquad
 \begin{array}{ccc}
 & & y_0 \xrightarrow{b} z_0
 \end{array}
 \quad (1.14)$$

Jacobs [Jac04] defined finite trace semantics as a cone over the following ω -chain Seq_T , which is dually to Seq^T defined as follows by

$$Seq_T(n) := T^n 0 \text{ and } Seq_T(n \leq n+1) := T^n i_{T0} \text{ for all } n < \omega \quad (1.15)$$

as in the following diagram

$$0 \xrightarrow{i_{T1}} T0 \longrightarrow \dots \longrightarrow T^n 0 \xrightarrow{T^n i_{T0}} T^{n+1} 0 \longrightarrow \dots \quad (1.16)$$

1.5.3 Infinite Trace Semantics

The definition of infinite traces relates to the one of finite traces in the following sense. Recall that classically [DR95] finite traces in labelled transition systems are finite, that is ($\sqrt{-}$)-terminated, words of adjacent labels. Infinite traces may not be terminated. Infinite traces are generalised to coalgebras analogously to finite traces. In this dissertation we do

not consider the limit case. In the case of labelled transition systems, we consider thus only such words of length up to ω .

Böhm Trees In our definition of Chapter 6, infinite trace semantics assigns to the states of a (B, T) -coalgebra the infinite traces. We begin with an inductive definition of infinite trace semantics in the style of Böhm trees, which are used in lambda calculus [Bar81] to give a semantics to terms which admit an infinite reduction. Consider the following modification of the combinator Θ .

$$t := (\lambda x. \lambda f. (xx))(\lambda x. \lambda f. (xx)) \quad (1.17)$$

The term t admits an infinite sequence of β -reductions as follows.

$$\begin{array}{ll} (\lambda x. \lambda f. (xx))(\lambda x. \lambda f. (xx)) & \rightarrow_{\beta} \\ \lambda f. (\lambda x. \lambda f. (xx))(\lambda x. \lambda f. (xx)) & \rightarrow_{\beta} \\ \lambda f. \lambda f. (\lambda x. \lambda f. (xx))(\lambda x. \lambda f. (xx)) & \rightarrow_{\beta} \\ \dots & \end{array}$$

From these calculations it is obvious, that we can not give a finite semantics for t , that is a normalised term. In Böhm trees one defines the semantics of lambda terms without normal form to be \perp . Applying this idea to the above β -reduction sequence yields the following sequence.

$$\begin{array}{ll} \perp & \rightarrow_{\beta} \\ \lambda f. \perp & \rightarrow_{\beta} \\ \lambda f. \lambda f. \perp & \rightarrow_{\beta} \\ \dots & \end{array}$$

Coalgebra Automata and Infinite Trace Semantics Another example of infinite trace semantics can be found in automata operating on infinite input. Recall that T -coalgebra

automata are structures $\mathbb{A} = \langle Q, \theta, q_I, \Omega \rangle$. Coalgebraically these automata can be conceived as pointed coalgebras $\langle Q, \theta, q_I \rangle$ augmented with a priority function Ω . Without loss of generality, the latter can be assumed to determine the acceptance behaviour of \mathbb{A} on strictly infinite input² We argue that every such Ω determines an infinite trace semantics.

Coinductive Infinite Trace Semantics Although we set out with an inductive definition of infinite trace semantics, we argue that infinite trace semantics are not necessarily inductively definable. Furthermore according to the example of coalgebra automata, infinite trace semantics is not uniquely defined. We thus introduce a coinductive characterisation of infinite trace semantics in terms of invariants on sequences such as the one of inductive infinite trace semantics. Thereby we deviate from the previous approach of Jacobs [Jac04] and the parallel work of Cirstea [Cî10]. In Chapter 6 we will reconcile Jacobs' approach with ours in greater detail.

1.6 Coalgebraic Logics

Coalgebraic Logics Moss [Mos99] defined the formulas a of coalgebraic logic to be of the following form.

$$a ::= \top \mid \perp \mid \neg a \mid \bigwedge a \mid \bigvee a \mid \nabla a \quad (1.18)$$

The semantics of ∇ in a coalgebra $\mathbb{S} = \langle S, \sigma \rangle$ is defined by

$$\mathbb{S}, s \models \nabla \alpha \text{ if and only if } (\sigma(s), \alpha) \in \text{Rel}_T(\models) \quad (1.19)$$

Coalgebraic Logics as Initial Algebras Kurz [Kur01b, Kur01a] made precise that coalgebraic logics and coalgebras, which are the models of coalgebraic logics, correspond under Stone duality [Joh86]. Coalgebras live in the category *Set* of sets and functions, whereas coalgebraic logics live in the category *CABA* of complete atomic Boolean algebras. Coalgebraic logics is the initial algebra for a functor L , dual to the functor T , that is

²The meaning of infinite for *Set*-coalgebras will be introduced in Chapter 11.

$2^{L(-)} = T2^{(-)}$. For finitary T , L can concretely be defined by *Ind-Pro*-completion from the finitary case as in the following diagram.

$$\begin{array}{ccc}
 T \circlearrowleft \text{Set}^{op} & \xrightleftharpoons{2^{(-)}} & \text{CABA} \circlearrowright L \\
 \uparrow \text{Pro} & \text{Uf} & \uparrow \text{Ind} \\
 T_\omega \circlearrowleft \text{FinSet}^{op} & \xrightleftharpoons{2^{(-)}} & \text{FinBA} \circlearrowright 2^{T_\omega \text{Uf}}
 \end{array} \tag{1.20}$$

We are particularly interested in the following properties of coalgebraic logics.

- **Invariance under T -Bisimilarity** We have seen that T -bisimilarity is a semantical equivalence between T -coalgebra states. A coalgebraic logic is invariant under T -bisimilarity, if T -bisimilar states satisfy the same formulas. Invariance under T -bisimilarity has been established in [Mos99] for infinitary coalgebraic logics and restricts to the finitary case.
- **Expressivity** Conversely to the invariance under T -bisimilarity, a coalgebraic logic is expressive, if logically equivalent states of T -coalgebras are T -bisimilar. Expressivity has been established in the infinitary case in [Mos99]. The finitary case is more intricate, confere [Kup06]. Kupke, Kurz, and Venema proved [KKV04] that Kripke polynomial functors T and T -coalgebras lift to the Vietoris functor in the category of Stone spaces [Joh86] and Stone coalgebras. The latter form classes of models which have expressive coalgebraic logics.
- **Completeness** Completeness is to be understood relative to an axiom system, and means that formulas are valid in coalgebraic logics if they can be derived. The latter makes sense only if the logics are finitary. Kupke, Kurz, and Venema defined the set M of axioms for finitary coalgebraic logics and proved it complete in [KKV08].

Modal Logics Modal logics [BdRV01] allow us to describe properties of state based transition systems in a finitary manner. The set of formulas of modal logics is defined as follows.

$$\phi ::= \top \mid \perp \mid \phi \vee \phi \mid \phi \wedge \phi \mid \Box \phi \mid \Diamond \phi \tag{1.21}$$

where the semantics of \Box and \Diamond are given as functions

$$\llbracket \Box(-) \rrbracket_s, \llbracket \Diamond(-) \rrbracket_s : 2^S \Rightarrow 2^{\mathcal{P}S} \quad (1.22)$$

where $2 = \{\text{true}, \text{false}\}$ is the set of truth values. For a detailed account of the variants of modal logics, we refer to the established literature.

Modal Correspondence Theory In modal correspondence theory, van Benthem [vB77] proved that modal logics is the bisimulation invariant fragment of first order logics augmented with a binary relation symbol denoting transitions.

Coalgebraic Modal Logics Another approach to logics for coalgebras consists in coalgebraic modal logics [Pat01a, Pat01b, Pat03b, Pat03a, Pat04]. The idea underlying coalgebraic modal logics is the following. The semantics $\llbracket a \rrbracket$ of a formula a of modal logics is a subset $\llbracket a \rrbracket \subseteq Q$. Since implication on the logical side is contravariant to inclusion on the semantical side, $\llbracket a \rrbracket \in 2^Q$. Semantically, \Box and \Diamond are then natural transformations $2^Q \Rightarrow 2^{\mathcal{P}Q}$. The previous observation makes the generalisation to a coalgebraic level of generality possible in coalgebraic modal logics.

- Invariance under T -Bisimilarity
- Expressivity [Sch05]
- Completeness

Coalgebraic Logics and Coalgebraic Modal Logics \Box and \Diamond can [BPV08] be translated into coalgebraic logic such that

$$\Box a \rightsquigarrow \nabla \emptyset \vee \nabla \{a\} \text{ and } \Diamond a \rightsquigarrow \nabla \{a, \top\} \quad (1.23)$$

and backwards such that

$$\nabla a \rightsquigarrow \Box(\bigvee a) \wedge \bigwedge \Diamond a \quad (1.24)$$

Leal et alii [Lea07, Lea08, KL09] generalised the above translations to a coalgebraic level of generality, and thereby showed that coalgebraic logic and coalgebraic modal logics are

interdefineable for accessible transition types.

1.7 Our Contributions in This Thesis

Finite Trace Semantics We adapted the definition of finite trace semantics from Generic Trace Theory introduced by Jacobs by embedding into the Eilenberg-Moore category of the branching type. Our finite trace semantics is applicable to finitary branching types. Our definition of finite trace semantics uses continuous extensions of Kleisli-liftings of functors suggested by Alexander Kurz. The work of Chapter 5 is joint with Alexander Kurz.

Finitary Coalgebraic Logics for Finite Traces We define finitary coalgebraic logics for finite traces using dual adjunctions in Eilenberg-Moore categories induced by ambimorphic objects. The latter idea was suggested by Bart Jacobs. We prove finitary coalgebraic logics for finite traces invariant under finite trace equivalence and expressive under additional conditions. The work of Chapter 7 is joint with Alexander Kurz.

Infinite Trace Semantics We modify finite trace semantics from Generic Trace Theory and obtain an inductively defined generic infinite trace semantics in the style of Böhm trees. We then abstract properties of generic infinite trace semantics, and obtain a characterisation of infinite trace semantics which applies to generic infinite trace semantics, Jacob's infinite trace semantics, and the acceptance behaviour of coalgebra automata.

Complementation Lemma for Moss' Coalgebraic Logics In joint work with Yde Venema, we define the Boolean dual of ∇ in the negation-free fragment of finitary Moss' coalgebraic logics, and thereby show that finitary Moss' coalgebraic logics is essentially negation-free. The result makes use of the completeness of Moss' coalgebraic logics proven by Kurz, Kupke, and Venema. The definition of the Boolean dual of ∇ originates with Yde Venema, and has been refined by Alessandra Palmigiano and an anonymous referee to our paper [KV09].

Game Bisimulations We abstract the notion of rounds known for acceptance games of automata and find a structuring of parity graph games. Using the structure we define a

congruence relation, preserving the semantics of parity graph games. The definition and the results are joint with Yde Venema.

Complementation Lemma for Coalgebra Automata Using the complementation lemma and game bisimulations we show that languages of coalgebras accepted by coalgebra automata are closed under taking complements. Together with earlier results of Venema and Kupke, the latter result establishes a Rabin-style correspondence between coalgebra automata and Moss' coalgebraic logics augmented with fix-point operators. The results of Chapter 10 are joint with Yde Venema.

Pumping Lemma for Coalgebra Automata We generalise pumping from words to arbitrary coalgebras in *Set*, and prove that languages of such coalgebras accepted by coalgebra automata have the pumping property.

1.8 Outline

This thesis consists of five parts.

Foundations in Category Theory In Chapter 3 we review monads in *Set*, Eilenberg-Moore algebras of monads, and distributive laws of monads with functors. In Chapter 4 we review categories of algebras, the Kleisli- and Eilenberg-Moore-categories of monads, and the Kleisli-liftings of *Set*-functors and their continuous extension.

Semantics of Coalgebras with Branching In Chapter 5 we review Generic Trace Theory, and propose an alternative definition of finite trace semantics in Eilenberg-Moore Categories, which will play a central role in Chapter 8. In Chapter 6 we adapt the basic ideas of Generic Trace Theory to obtain a characterisation of infinite trace semantics.

Coalgebraic Logics In Chapter 7 we define the Boolean dual of Moss' modality ∇ and thereby show that Moss' coalgebraic logic is essentially negation-free. In Chapter 8 we propose generic coalgebraic logics characterising coalgebras with branching up to finite trace equivalence. We prove the logics invariant under finite trace equivalence and expressive under additional assumptions.

Coalgebraic Automata Theory In Chapter 9 we review parity graph games and introduce game bisimulations. In Chapter 10 we show that language of coalgebras accepted by coalgebra automata are closed under taking complements. Finally we show that languages of coalgebras accepted by state-finite coalgebra automata have the pumping property.

Chapter 2

Notation

2.1 Set Theory

- We write $Y \subseteq_{\omega} X$ to mean, that Y is a finite subset of X .
- Let $f : X \rightarrow Y$ be a function, we denote by $Gr(f) \subseteq X \times Y$ its graph.
- Let $f : X \rightarrow X$ be a function, we write f^n to denote the n -fold application of f , such that $f^1 = f$ and $f^{n+1}(x) = f(f^n(x))$ for all $x \in X$.
- Let X be a set, X^* denotes the set of (finite) words over X .
- ϵ denotes the empty sequence.
- Let X be a set, $X^+ := X^* \setminus \{\epsilon\}$ denotes the set of finite non-empty words over X .
- Let X be a set, $x \in X$, and $w \in X^*$, then $x.w$ denotes w with x prepended, and $w.x$ denotes w with x appended.
- Let $f : X \rightarrow Y$ be a function, and let $x \in X$ and $y \in Y$, we denote by $f\{x \mapsto y\}$ the function taking $x \mapsto y$ and $x' \mapsto f(x')$ for all $x' \in X \setminus \{x\}$.
- Let Y be a set with a distinguished element 0 , and let $x \in X$ and $y \in Y$, $\{x \mapsto y\}$ denotes the function $X \rightarrow Y$ taking $x \mapsto y$ and $x' \mapsto 0$ for all $x' \in X \setminus \{x\}$.

- We denote by $X \times Y := \{(x, y) \mid x \in X, y \in Y\}$ the cartesian product of X and Y . and by $\pi_X : X \times Y \rightarrow X$ the projection morphism for X , and similarly for Y . If $X = Y$, it may not be the case that $\pi_X = \pi_Y$.
- We denote by $X + Y := \{(0, x), (1, y) \mid x \in X, y \in Y\}$ the coproduct, that is disjoint sum, of X and Y , and by $\kappa_X : X \rightarrow X + Y$ the injection morphism for X , and similarly for Y . If $X = Y$, it may not be the case that $\kappa_X = \kappa_Y$.
- Let X be a set, $\Delta_X := \{(x, x) \mid x \in X\}$ denotes the diagonal relation.

For all sets X , Y , and X' with $X' \subseteq X$, and for all relations $R \subseteq X \times Y$, we denote

- $R[x] := \{y \in Y \mid (x, y) \in R\}$ for all $x \in X$,
- $R[X'] := \{y \in Y \mid \exists x \in X'. (x, y) \in R\}$ for all $X' \subseteq X$, and
- $R|_{X'} := \{(x, y) \mid x \in X', (x, y) \in R\}$ for all $X' \subseteq X$.

For our convenience we adopt the notation from lambda calculus [Bar81].

- $\lambda x \in X. t$ denotes the function $\{x \mapsto t \mid x \in X\}$.
- When X is clear from context we omit it and write $\lambda x. t$ for $\lambda x \in X. t$.

2.2 Logic

In the following let ϕ and ψ be formulas of a modal logic.

- $M \models \phi$ means M is a model of ϕ .
- $M, x \models \phi$ means that ϕ holds in M in state x .
- $\phi \vdash \psi$ means that ϕ entails ψ .
- $\vdash \phi$ means that ϕ holds.
- $\llbracket \phi \rrbracket_M := \{x \mid M, x \models \phi\}$ is the semantics of ϕ in the model M .

2.3 Category Theory

Below let \mathcal{C} and \mathcal{D} be categories and X an object of \mathcal{C} .

- id_X is the identity morphism.
- $Id_{\mathcal{C}}$ is the identity functor on \mathcal{C} .
- $(\times) \Pi$ is the (binary) categorical product.
- $(+) \Sigma$ is the (binary) categorical coproduct.
- We write X to denote the functor $\mathcal{D} \rightarrow \mathcal{C}$ taking all objects of \mathcal{D} to X and all morphisms to id_X .
- Let $T : \mathcal{C} \rightarrow \mathcal{C}$ be a function, we write T^n to denote the n -fold application, such that $T^0 = Id_{\mathcal{C}}$ and $T^{n+1} = T(T^n)$.

Suppose initial and final object exist in \mathcal{C} , respectively.

- 0 denotes the initial object of \mathcal{C}
- 1 denotes the final object of \mathcal{C} .
- $!_X$ is the initial object morphism $0 \rightarrow X$.
- $!_X$ is the final object morphism $X \rightarrow 1$.

Let $\langle \mathcal{C}, \otimes, I \rangle$ be a monoidal closed category.

- We write $[-, +]$ for $\mathcal{C}(-, +)$.

Composition of Arrows

We write the composition of arrows $f : X \rightarrow Y$ and $g : Y \rightarrow X$ as $g \circ f$ (read "g after f") and $f; g$ (read f composed with g). We use the latter notation in Rel , the category of sets and relation to comply with the standard notation in algebra, and the first notation otherwise.

Diagrams

Let $D : \mathcal{I} \rightarrow \mathcal{C}$ be a diagram over a category \mathcal{C} indexed in a category \mathcal{I} , then

- $\lim D$ is the limit of the diagram D .
- $\operatorname{colim} D$ is the colimit of the diagram D .
- $|D|$ is the discrete version of D , that is the restriction of D to $|\mathcal{I}|$, the discrete version of \mathcal{I} .

Part I

Foundations in Category Theory

Chapter 3

A Review of Monads and Algebras over *Set*

In the introduction we argued that functors are suitable transition types for coalgebras, and monads are suitable branching types. In this section we introduce monads and their algebras independently from coalgebras.

3.1 Definition of Monads

Monads are functors with the following additional structure.

Definition 3.1.1 (Monads). A monad $\langle B, \mu, \eta \rangle$ on a category C consists of

- a functor $B : C \rightarrow C$, and natural transformations
- $\eta : Id \Rightarrow B$ (unit), and
- $\mu : BB \Rightarrow B$ (multiplication)

subject to the following conditions.

$$\begin{array}{ccc}
 B & \xrightarrow{\eta_B} & BB & \xleftarrow{B\eta} & B \\
 \searrow & & \downarrow \mu & & \swarrow \\
 & & B & &
 \end{array}
 \qquad
 \begin{array}{ccc}
 BBB & \xrightarrow{\mu_B} & BB \\
 B\mu \downarrow & & \downarrow \mu \\
 BB & \xrightarrow{\mu} & B
 \end{array}
 \tag{3.1}$$

When the monad structure is clear from the context we omit mentioning the monad structure and denote the monad $\langle B, \mu, \eta \rangle$ as B .

3.2 Categories of Algebras for Monads

Definition 3.2.1 (Eilenberg-Moore Algebras for a Functor). *Let C be a category and $T : C \rightarrow C$ a functor on C . An Eilenberg-Moore algebra for T is a morphism $\alpha : TX \rightarrow X$ for an object X in C . We call X the carrier of α . A T -algebra morphism $\alpha \rightarrow \beta$ between T -algebras $\alpha : TX \rightarrow X$ and $\beta : TY \rightarrow Y$ is a morphism $f : X \rightarrow Y$ such that $\beta \circ Tf = f \circ \alpha$. The Eilenberg-Moore algebras for a functor T and their morphisms form a category, the Eilenberg-Moore category for the functor T .*

The definitions of Eilenberg-Moore algebras for a functor and their category extend to monads as follows.

Definition 3.2.2 (Eilenberg-Moore Algebras for a Monad). *An Eilenberg-Moore algebra for a monad B is an Eilenberg-Moore algebra $\alpha : BX \rightarrow X$ for the functor B satisfying the following additional conditions.*

1. $\alpha \circ \eta_X = id_X$
2. $\alpha \circ B\alpha = \alpha \circ \mu_X$

Morphisms between Eilenberg-Moore algebras of a monad are defined as the morphisms between the underlying Eilenberg-Moore algebras. The Eilenberg-Moore algebras for a monad B and their morphisms form a category, the Eilenberg-Moore category of B .

Remark 3.2.3. *Note that, the Eilenberg-Moore-category of the monad B embeds properly into the Eilenberg-Moore-category of the functor B .*

The category of Eilenberg-Moore algebras for a monad B over the category Set has a forgetful functor $U : B-Alg \rightarrow Set$ with a left adjoint $F : Set \rightarrow B-Alg$. The adjunction

$$\begin{array}{ccc} & F & \\ \text{Set} & \xrightarrow{\quad} & B-Alg \\ & U & \end{array} \quad (3.2)$$

is defined explicitly for all Eilenberg-Moore algebras $\alpha : BX \rightarrow X$ and $\beta : BY \rightarrow Y$, and B -algebra morphisms $f : \alpha \rightarrow \beta$ by

- $U(\alpha) := X$ and $U(f) := f$, and
- $F(X) := \mu_X$ and $F(f) := \mu_Y \circ Bf$.

Definition 3.2.4 (Kleisli-Categories for Monads B). The Kleisli-category $Kl(B)$ of a monad B is the category of free Eilenberg-Moore algebras for the monad B .

The following can be found in greater generality in Chapter VI.3 of MacLane [Mac98].

Lemma 3.2.5. *There is a unique embedding functor $K : Kl(B) \rightarrow B\text{-Alg}$, such that $F = KF'$ and $U' = UK$.*

3.3 Eilenberg-Moore Categories of Commutative Monads

Note that $B\text{-Alg}$ may not be symmetric monoidal closed, but for commutative monads it is. This is made precise in the following result, which is due to Kock [Koc70].

Theorem 3.3.1. *The following are equivalent*

1. B is a commutative monad.
2. $B\text{-Alg}$ is symmetric monoidal closed with \otimes being the Kleisli-lifting of \times .

We prove the two directions of this theorem separately below. In order to show that, 1 \implies 2, we use that commutative monads B admit a Kleisli-lifting of \times as in Corollary 4.2.3.

Proof. We prove the two directions separately. Let B be commutative, we show that the Kleisli-lifting $Kl(\times)$ of \times defines a symmetric monoidal structure on $B\text{-Alg}$.

By definition F commutes with the Kleisli-lifting, such that $FX(Kl(\times))FY = F(X \times Y)$. For non-free algebras A and A' we define the Kleisli-lifting as the continuous extension, that is the coequaliser of the following diagram.

$$FUFUA(Kl(\times))FUFUA' \xrightleftharpoons[\epsilon_{FUA,FUA'}]{FU\epsilon_{A,A'}} FUA(Kl(\times))FUA' \xrightarrow{\epsilon_{A,A'}} A \otimes A' \quad (3.3)$$

We need to verify the conditions of the symmetric monoidal closure.

1. There is an object I in $B\text{-Alg}$ with natural isomorphisms $\lambda_A : I \otimes A \rightarrow A$ and $\rho_A : A \otimes I \rightarrow A$.
2. There is a natural isomorphism $\sigma_{A,B} : A \otimes A' \rightarrow A' \otimes A$.

1. We choose $I := F\{*\}$ and observe the following.

$$\begin{array}{ccccc} FUFUA & \xrightleftharpoons[\epsilon_{FUA}]{\epsilon_{FUA}} & FUA & \xrightarrow{\epsilon_A} & A \\ \parallel & & \parallel & & \uparrow \lambda_A \\ F(UFUA \times \{*\}) & \xrightleftharpoons{\quad} & F(UA \times \{*\}) & & \\ \parallel & & \parallel & & \\ FUFUA \otimes I & \xrightleftharpoons[\epsilon_{FUA} \otimes id_I]{\epsilon_{FUA} \otimes id_I} & FUA \otimes I & \longrightarrow & A \otimes I \end{array} \quad (3.4)$$

and symmetrically,

$$\begin{array}{ccccc} FUFUA & \xrightleftharpoons[\epsilon_{FUA}]{\epsilon_{FUA}} & FUA & \xrightarrow{\epsilon_A} & A \\ \parallel & & \parallel & & \uparrow \rho_A \\ F(\{*\} \times UFUA) & \xrightleftharpoons{\quad} & F(\{*\} \times UA) & & \\ \parallel & & \parallel & & \\ I \otimes FUFUA & \xrightleftharpoons[id_I \otimes \epsilon_{FUA}]{id_I \otimes \epsilon_{FUA}} & I \otimes FUA & \xrightarrow{id_I \otimes \epsilon_A} & I \otimes A \end{array} \quad (3.5)$$

where λ and ρ exist because of the universal property of coequalisers.

2. Follows through similar considerations as follows.

$$\begin{array}{ccc}
 FUFUA \otimes FUFUA' & \rightrightarrows & FUA \otimes FUA' \longrightarrow A \otimes A' \\
 \parallel & & \parallel \\
 F(UFUA \times UFUA') & \rightrightarrows & F(UA \times UA') \\
 \parallel & & \parallel \\
 F(UFUA' \times UFUA) & \rightrightarrows & F(UA' \times UA) \\
 \parallel & & \parallel \\
 FUFUA' \otimes FUFUA & \rightrightarrows & FUA' \otimes FUA \longrightarrow A' \otimes A
 \end{array}
 \quad (3.6)$$

$\sigma_{A,A'}$

The converse direction follows in particular from the bilinearity of \otimes . We suppose $B\text{-Alg}$ is symmetric monoidal closed for $\otimes = Kl(\times)$, and show that B is commutative.

Bilinearity means in particular, that

$$(id_{FX} \otimes \epsilon_{FY}) \circ (\epsilon_{FX} \otimes id_{FUFY}) = (\epsilon_{FX} \otimes id_{FY}) \circ (id_{FUFX} \otimes \epsilon_{FY}) \quad (3.7)$$

commute natural in X and Y . Taking the adjoint transpose along $F \dashv U$ and using that $F(- \times +) = F(-) \otimes F(+)$, we obtain

$$\begin{array}{ccccc}
 FUFX \otimes FUFY & \xrightarrow{\epsilon_{FX} \otimes id_{FUFY}} & FX \otimes FUFY & \xrightarrow{id_{FX} \otimes \epsilon_{FY}} & FX \otimes FY \\
 \parallel & & \parallel & & \\
 F(UFX \times UFY) & \xrightarrow{f} & F(X \times UFY) & \xrightarrow{g} & F(X \times Y)
 \end{array}
 \quad (3.8)$$

$$BX \times BY \xrightarrow{f^\dagger} B(X \times BY) \xrightarrow{Bg^\dagger} BB(X \times Y) \xrightarrow{\mu_{X \times Y}} B(X \times Y)$$

and

$$\begin{array}{ccccc}
 FUFX \otimes FUFY & \xrightarrow{id_{FUFX} \otimes \epsilon_{FY}} & FUFX \otimes FY & \xrightarrow{\epsilon_{FX} \otimes id_{FY}} & FX \otimes FY \\
 \parallel & & \parallel & & \parallel \\
 F(UFX \times UFY) & \xrightarrow{f} & F(UFX \times Y) & \xrightarrow{g} & F(X \times Y)
 \end{array}
 \quad (3.9)$$

$$BX \times BY \xrightarrow{f^\dagger} B(BX \times Y) \xrightarrow{Bg^\dagger} BB(X \times Y) \xrightarrow{\mu_{X \times Y}} B(X \times Y)$$

□

3.4 Examples

In this section we introduce monads on *Set* which are of interest as branching types, and describe their Eilenberg-Moore algebras by means of axioms, which contribute to the examples of trace logics in Chapter 8.

The simplest kinds of monads we consider are the identity monad and lift monad. The commonly used finitary powerset and finitary multiset monad are subsumed by the semiring monad for the boolean semiring and the natural numbers with the usual arithmetic operations, respectively. The finitary multiset monad is also known as the Bag monad. From the class of semiring monads we furthermore consider min semiring monads, which played a role for instance in [Rut02].

In the following we define these examples of monads, and afterwards show syntax and axiomatisation of their algebras.

Definition 3.4.1 (Monads in *Set*). *Table 3.1 defines examples of monads, where*

- $\{x\}$ denotes a multiset singleton containing x once and \bigcup denotes multiset union,
- \mathcal{S} is a semiring $\mathcal{S} = \langle S, +, *, 0, 1 \rangle$,
- \mathcal{S}_{min} is a min semiring $\mathcal{S}_{min} = \{S, \min, +, \infty, 0\}$ as in Example 3.4.4,

Name	Functor B	Unit $\eta : Id \Rightarrow B$	Multiplication $\mu : B^2 \Rightarrow B$
Identity monad	Id	$\eta = id$	$\mu = id$
Lift monad	$\{\perp\} + Id$	$\eta_X(x) = \kappa_1 x$	$\mu_X = \begin{cases} \kappa_1 \perp \mapsto \kappa_1 \perp \\ \kappa_2 \kappa_1 \perp \mapsto \kappa_1 \perp \\ \kappa_2 \kappa_2 x \mapsto \kappa_2 x \end{cases}$
(Finitary) powerset monad	$\mathcal{P}(\mathcal{P}_\omega)$	$\eta_X(x) = \{x\}$	$\mu_X(\mathbb{X}) = \bigcup \mathbb{X}$
Bag monad	$(\mathbb{N}^{(-)})_\omega$	$\eta_X(x) = \{x\}$	$\mu_X(M) = \bigcup M$
Semiring monad	$(\mathcal{S}^{(-)})_\omega$	$\eta_X(x) = \{x \mapsto 1\}$	$\mu_X(M) = \lambda x. \sum_m M(m) * m(x)$
Min semiring monad	$(\mathcal{S}_{min}^{(-)})_\omega$	$\eta_X(x) = \{x \mapsto 1\}$	$\mu_X(M) = \lambda x. \text{Min}_m M(m) * m(x)$
(Sub-)distribution monad	$\mathcal{D}_{=1}(\mathcal{D}_{\leq 1})$	$\eta_X(x) = \{x \mapsto 1\}$	$\mu_X(D) = \lambda x. \sum_d D(d) * d(x)$

Table 3.1: Examples of Monads

Notation 3.4.2. We denote the function $X \rightarrow \mathbb{N}$ taking an element $x \in X$ to 1 and every other element of X to 0 by $\{x \mapsto 1\}$.

The following axioms of semimodules for semirings can be found in the standard literature, such as Chapter 14 of [Gol10].

Definition 3.4.3 (Semirings). *A semiring is an algebraic structure $\mathcal{S} = \langle S, +, *, 0, 1 \rangle$ satisfying the following axioms.*

$\langle S, +, 0 \rangle$ is a commutative monoid, so that for all $s, r, t \in S$,

1. *Unit-Element:* $s = s + 0$
2. *Commutativity:* $s + t = t + s$
3. *Associativity:* $s + (r + t) = (s + r) + t$

$\langle S, *, 1 \rangle$ is a monoid, so that for all $s, r, t \in S$

4. *Unit-Element:* $s = s * 1$
5. *Associativity:* $s * (r * t) = (s * r) * t$

The additive and multiplicative monoids interact such that for all $s, r, t \in S$,

6. *Annihilation:* $0 * s = 0 = s * 0$
7. *Distributivity:* $s * (r + t) = (s * r) + (s * t)$

An example of semirings are min semirings.

Example 3.4.4 (Min Semirings). *A min-semiring is an algebra $\mathcal{S}_{\min} = \langle S, \min, +, \infty, 0 \rangle$ satisfying the following axioms.*

The monoid $\langle S, \min, \infty \rangle$ is idempotent

1. *Idempotency:* $\min(s, s) = s$

The monoid $\langle S, +, 0 \rangle$ is commutative

2. *Commutativity:* $s + r = r + s$

0 is absorptive with respect to min,

3. *Absorption*: $\min(0, s) = 0$

Next we show syntax and axiomatisation of the Eilenberg-Moore algebras for the example monads above.

Proposition 3.4.5 (Algebras for the Monads in Definition 3.4.1). *1. The Eilenberg-Moore algebras for Id are sets X .*

2. The Eilenberg-Moore algebras for $Lift$ are pointed sets (X, \perp) , where $\perp \in X$.

3. The Eilenberg-Moore algebras $\alpha : \mathcal{P}X \rightarrow X$ for the powerset monad are complete (disjunctive) semilattices, satisfying the following axiom for all $(Y_{ij} \subseteq X)_{i \in I, j \in J}$.

(a) Bottom element: $\perp := \bigvee \emptyset$.

(b) Associativity: $\bigvee_{i \in I} \bigvee_{j \in J} Y_{ij} = \bigvee_{(i,j) \in I \times J} Y_{ij}$

4. The Eilenberg-Moore algebras $\alpha : \mathcal{P}X \rightarrow X$ for the finitary powerset monad are (disjunctive) semilattices, satisfying the following axioms for all $x, y, z \in X$.

(a) Bottom-Element: $\perp: x \vee \perp = x$

(b) Idempotency: $x \vee x = x$

(c) Commutativity: $x \vee y = y \vee x$

(d) Associativity: $x \vee (y \vee z) = (x \vee y) \vee z$

5. The Eilenberg-Moore algebras $\alpha : (\mathbb{N}^X)_\omega \rightarrow X$ for the finite multiset monad are semimodules satisfying the following axioms for all

(a) Unit-Element: $0 \cdot x = 0$ and $0 \oplus n \cdot x = n \cdot x$

(b) Commutativity: $n \cdot x \oplus m \cdot y = m \cdot y \oplus n \cdot x$

(c) Associativity: $n \cdot x \oplus (m \cdot y \oplus o \cdot z) = (n \cdot x \oplus m \cdot y) \oplus o \cdot z$

(d) Absorption: $n \cdot x \oplus m \cdot x = (n + m) \cdot x$

*6. Algebras for a semiring monad $(S^{(-)})_\omega$ for a semiring $S = \langle S, +, *, 0, 1 \rangle$ are algebras $\langle M, 0_M, \oplus, (s \cdot (-))_{s \in S} \rangle$ satisfying the following axioms.*

$\langle M, 0_M, \oplus \rangle$ is a commutative monoid, so that

(a) *Unit-element:* $0_M \oplus m = m$

(b) *Commutativity:* $m \oplus n = n \oplus m$

(c) *Associativity:* $m \oplus (n \oplus o) = (m \oplus n) \oplus o$

Such that the additive monoid and the semiring \mathcal{S} are compatible, such that

(a) $s \cdot m \oplus r \cdot m = (s + r) \cdot m$

(b) $s \cdot (r \cdot m) = (s * r) \cdot m$

7. *The algebras for the min-semiring monad $(\mathcal{S}^{(-)})_\omega$ where $\mathcal{S} = \langle S, \min, +, \infty, 0 \rangle$ are structures $\langle M, 0_M, \oplus, (s \cdot (-))_{s \in S} \rangle$ satisfying the following axioms.*

$\langle M, 0_M, \oplus \rangle$ is a commutative monoid, so that

(a) *Unit-element:* $0_M \oplus m = m$

(b) *Commutativity:* $m \oplus n = n \oplus m$

(c) *Associativity:* $m \oplus (n \oplus o) = (m \oplus n) \oplus o$

Such that the additive monoid and the semiring \mathcal{S} are compatible, such that

(a) *Aggregation:* $s \cdot m \oplus r \cdot m = \min(s, r) \cdot m$

(b) *Composition:* $s \cdot (r \cdot m) = (s + r) \cdot m$

8. *The Eilenberg-Moore algebras for the sub-distribution monad are convex sets satisfying the following axioms for all $x \in X$.*

(a) *Unit-Element:* $0 \cdot x = 0$ and $0 \oplus r \cdot x = r \cdot x$

(b) *Absorption:* $r \cdot x \oplus s \cdot x = (r + s) \cdot x$

(c) *Commutativity:* $r \cdot x \oplus s \cdot y = s \cdot y \oplus r \cdot x$

(d) *Associativity:* $r \cdot x \oplus (s \cdot y \oplus t \cdot z) = (r \cdot x \oplus s \cdot y) \oplus t \cdot z$

The proof is a basic exercise in algebra. Nevertheless the validity of the proposition above has been questioned by a reviewer of our submitted paper [KK10], so that we spell out

the proof for semiring monads. Recall that the latter subsumes finitary powerset, finitary multiset, semiring and min-semiring monads.

Proof. Soundness Let X be a set, $\mathcal{S} = \langle S, +, *, 0, 1 \rangle$ a semiring, and $\mathcal{M} = \langle (S^X)_\omega, \mu_X \rangle$ the free semiring monad for the semiring \mathcal{S} . We show that \mathcal{M} is closed under the operations of semimodules above, and that with these operations \mathcal{M} satisfies the axioms of semimodules. We define for all $m, n \in M$ and $s \in S$,

1. $0_M := \lambda x.0$,
2. $m \oplus n := \lambda x.m(x) + n(x)$, and
3. $s \cdot m := \lambda x.s * m(x)$.

Then $\langle M, \oplus, 0_M \rangle$ is an additive monoid, so that for all $m, n, o \in M$,

1. $m \oplus 0_M = \lambda x.m(x) + 0_M(x) = \lambda x.m(x) = m$,
2. $m \oplus n = \lambda x.m(x) + n(x) = \lambda x.n(x) + m(x) = n \oplus m$, and
3. $m \oplus (n \oplus o) = \lambda x.m(x) + (n \oplus o)(x) = \lambda x.m(x) + (n(x) + o(x)) = \lambda x.(m(x) + n(x)) + o(x) = \lambda x(m \oplus n)(x) + o(x) = (m \oplus n) \oplus o$,

and the monoid $\langle M, \oplus, 0_M \rangle$ is compatible with the semiring \mathcal{S} , so that for all $m, n \in M$ and $s \in S$,

1. $s \cdot m \oplus r \cdot m = (\lambda x.s * m(x)) \oplus (\lambda x.r * m(x)) = \lambda x.(s + r) * m(x) = (s + r) \cdot m$
2. $s \cdot (r \cdot m) = s \cdot (\lambda x.r * m(x)) = \lambda x.s * (r * m(x)) = \lambda x.(s * r) * m(x) = (s * r) \cdot m$

It suffices to show soundness for free \mathcal{S} -semimodules.

Completeness We show that any structure $\mathcal{M} = \langle M, \oplus, 0_M, (s \cdot (-))_{s \in S} \rangle$ for a semiring $\mathcal{S} = \langle S, +, *, 0, 1 \rangle$ satisfying the axioms of \mathcal{S} -semimodules as above defines an algebra for the monad $(S^{(-)})_\omega$ with the algebra map $\alpha : (S^M)_\omega \rightarrow M$, such that

$$\alpha(f) := \bigoplus \{f(m) \cdot m \mid m \in M, f(m) \neq 0\} \text{ for all } f \in (S^M)_\omega \quad (3.10)$$

where $\bigoplus\{m_0, \dots, m_n\}$ makes the ellipsis notation $m_0 \oplus \dots \oplus m_n$ formal. The above definition is well-stated, since f has finite support by definition. It remains to show that α is compatible with the monads laws.

- $\alpha \circ \eta_M = id_M$ follows from the definition of η_M since $\alpha(\eta_M(m)) = \alpha(\{m \mapsto 1\}) = 1 \cdot m = m$.
- $\alpha \circ \mu_M = \alpha \circ (\mathcal{S}^\alpha)_\omega$ from the associativity and commutativity of \oplus as well as the compatibility of \mathcal{M} with \mathcal{S} .

□

3.5 Commutative Monads in Set

Definition 3.5.1 (Strength Law). A strength law for a monad $\langle B, \mu, \eta \rangle$ is a morphism $st_{X,Y} : X \times BY \rightarrow B(X \times Y)$ natural in X and Y such that the following commute.

$$\begin{array}{ccc} X \times BY & \xrightarrow{id_X \times \eta_Y} & X \times BY \\ & \searrow \eta_{X \times Y} & \downarrow st_{X,Y} \\ & & B(X \times Y) \end{array} \quad (3.11)$$

$$\begin{array}{ccccc} X \times B^2 Y & \xrightarrow{id_X \times \mu_Y} & X \times BY & & \\ st_{X,BY} \downarrow & & \downarrow st_{X,Y} & & \\ B(X \times BY) & \xrightarrow{Bst_{X,Y}} & B^2(X \times Y) & \xrightarrow{\mu_{X \times Y}} & B(X \times Y) \end{array} \quad (3.12)$$

Strong monads are monads with a strength law as above. Commutative monads are monads with a double-strength law.

Definition 3.5.2 (Double-Strength Law). A $\langle B, \mu, \eta \rangle$ be a monad with strength law dst has a double strength law, if the following diagram commutes.

$$\begin{array}{ccccc} BX \times BY & \xrightarrow{st_{BX,Y}} & B(BX \times Y) & \xrightarrow{st_{Y,X}} & B^2(X \times Y) \\ st_{BY,X} \downarrow & & \searrow dst_{X,Y} & & \downarrow \mu_{X \times Y} \\ B(X \times BY) & \xrightarrow{Bst_{X,Y}} & B^2(X \times Y) & \xrightarrow{\mu_{X \times Y}} & B(X \times Y) \end{array} \quad (3.13)$$

In the above diagram we omitted the bijection $X \times Y \cong Y \times X$. The double strength law is the diagonal morphism $\text{dst}_{X,Y} : BX \times BY \rightarrow B(X \times Y)$ natural in X and Y . We call a monad $\langle B, \mu, \eta \rangle$ with a double strength law commutative.

Commutative monads admit unique extensions of multihomomorphisms, as in the following definition.

Definition 3.5.3 (Multihomomorphic Extensions). *Let $\langle B, \mu, \eta \rangle$ be a monad, a multihomomorphic extension of a morphism $f : X_0 \times \dots \times X_{n-1} \rightarrow Y$ is a morphism $\widehat{f} : BX_0 \times \dots \times BX_{n-1} \rightarrow BY$ making the following commute*

$$\begin{array}{ccc} X_0 \times \dots \times X_{n-1} & \xrightarrow{f} & Y \\ \eta_{X_0} \times \dots \times \eta_{X_{n-1}} \downarrow & & \downarrow \eta_Y \\ BX_0 \times \dots \times BX_{n-1} & \xrightarrow{\widehat{f}} & BY \end{array} \quad (3.14)$$

The equivalence stated in the following proposition above is proved by Manes and Mulry in [MM07]. For the proof we refer to the original paper.

Proposition 3.5.4. *Let B be a commutative monad, then each function $f : X_0 \times X_1 \rightarrow Y$ has precisely one multihomomorphic extension $\widehat{f} : BX_0 \times BX_1 \rightarrow BY$.*

Using Proposition 3.5.4 we can prove the following result, which will contribute to our construction of trace logics later on.

Lemma 3.5.5. *Let B be a commutative monad, then $B\emptyset \cong \emptyset$ or $B\emptyset \cong 1$.*

Proof. Let $\pi_0 : B\emptyset \times B\emptyset \rightarrow B\emptyset$ and $\pi_1 : B\emptyset \times B\emptyset \rightarrow B\emptyset$ be the projection morphisms. Both, π_0 and π_1 are multihomomorphic extensions of the isomorphism $f : \emptyset \times \emptyset \cong \emptyset$ as they both make $\pi \circ (\eta_\emptyset \times \eta_\emptyset) \cong \eta_\emptyset \circ f$ commute ($\pi \in \{\pi_0, \pi_1\}$). By Proposition 3.5.4, thus $\pi_0 \cong \pi_1$, from which it follows that $B\emptyset$ has at most one element. \square

Chapter 4

Distributive Laws and Functor Liftings

In this chapter we review the basic definition of the Kleisli-lifting of *Set*-functors. The new result of this chapter is a definition of the continuous extension of a Kleisli-lifting of a *Set*-functor, which we will use in Chapter 5. The details originate with Alexander Kurz.

4.1 Distributive Laws

Distributive laws as studied by Beck [Bec69], and Manes and Mulry [MM07], were connected to the question, whether the composition of two monads $\langle B_1, \mu_1, \eta_1 \rangle$ and $\langle B_2, \mu_2, \eta_2 \rangle$ yields a monad. $B_1 B_2$ is a monad if B_1 and B_2 commute, that is have a natural transformation $\pi : B_2 B_1 \Rightarrow B_1 B_2$. π yields the multiplication of $B_1 B_2$ as follows.

$$B_1 B_2 B_1 B_2 \xrightarrow{B_1 \pi_{B_2}} B_1 B_1 B_2 B_2 \xrightarrow{B_1 B_1 \mu_2} B_1 B_1 B_2 \xrightarrow{(\mu_1)_{B_2}} B_1 B_2 \quad (4.1)$$

In this dissertation we are interested in distributive laws between monads and functors.

Definition 4.1.1 (Distributive Laws). *A distributive law for a functor T and a monad $\langle B, \mu, \eta \rangle$ is a natural transformation $\pi : TB \Rightarrow BT$ commuting with the monad structure of B such that*

$$\begin{array}{ccc} T & \xrightarrow{T\eta} & TB \\ \eta_T \downarrow & \swarrow \pi & \\ BT & & \end{array} \quad \begin{array}{ccccc} TBB & \xrightarrow{\pi} & BTB & \xrightarrow{B\pi} & BBT \\ T\mu \downarrow & & & & \downarrow \mu_T \\ TB & \xrightarrow{\pi} & & & BT \end{array} \quad (4.2)$$

4.1.1 Examples of Distributive Laws

Distributive laws are not guaranteed to exist for all pairs of a monad and a functor. For our purposes the following cases are of sufficient interest.

Example 4.1.2. *A distributive law $\pi : TB \Rightarrow BT$ exists trivially, when*

1. $T = Id: \pi := id_B$, or

2. $B = Id: \pi := id_T$

The properties of distributive laws from Definition 4.1.1 can easily be reduced to the monad laws in Definition 3.1.1

Example 4.1.3. *Let $T(-) := \{\surd\} + Act \times (-)$ be a Set-functor for a fixed set Act . T is the functor which labels in Act or marks successful termination (\surd). With each of the monads in Section 3.4 T has a distributive law.*

1. $\pi : T\mathcal{P} \Rightarrow \mathcal{P}T: \pi_X(\surd) := \{\surd\}, \pi_X(a, Y \subseteq X) := \{(a, x) \mid x \in Y\}.$

2. $\pi : T(\mathbb{N}^-)_\omega \Rightarrow (\mathbb{N}^-)_\omega T: \pi_X(\surd) := \eta_{\{\ast\}+Act \times X}(\surd)$, and $\pi_X(a, m)(a, x) := \{(a, x) \mapsto m(x), (b, x) \mapsto 0, \ast \mapsto 0 \mid a \in Act, b \in Act, b \neq a, x \in X\}$

3. $\pi : T\mathcal{D} \Rightarrow \mathcal{D}T: \pi_X(\surd) := \eta_{\{\surd\}+Act \times X}(\surd)$, and $\pi_X(a, d) := \{(a, x) \mapsto d(x), (b, x) \mapsto 0, \surd \mapsto 0 \mid a \in Act, b \in Act, b \neq a, x \in X\}$ where $\mathcal{D} \in \{\mathcal{D}_{\leq 1}, \mathcal{D}_{=1}\}$

The distributive laws of Example 4.1.3 are easy, because T is defined from coproducts. In the next section we review the result that distributive laws exist also for functors made up from binary products given the monad is commutative.

4.1.2 The Existence of Distributive Laws

When B is the powerset monad, the existence of a distributive law with a functor T is closely related to the existence of a relation lifting $Rel_T(-)$ as the following proposition shows. Jacobs [Jac04] has accredited this result to Power. For the proof we refer to Jacobs' paper.

Proposition 4.1.4. *Let $T : \text{Set} \rightarrow \text{Set}$ be a functor preserving weak pullbacks, then there is a distributive law $\pi : T\mathcal{P} \Rightarrow \mathcal{P}T$ defined such that*

$$\pi_X(a) = \{b \in TX \mid (a, b) \in \text{Rel}_T(\epsilon)\} \quad (4.3)$$

Hasuo, Jacobs and Sokolova [HJS06] have also shown that the previous result generalises to commutative monads and shapely functors on *Set*. The latter uniformly cover many types used in algebraic specification, such as words, streams, trees, graphs, and lists. For details see also [Jay95].

Definition 4.1.5 (Shapely Functors). *Shapely functors on Set are defined inductively as follows*

$$T ::= Id \mid \text{Act} \mid T \times T \mid \coprod_I T \quad (4.4)$$

where *Act* is the constant functor with value *Act* and *I* is a set.

The following is Lemma 2.7 in [HJS06]. The proof is by induction on the structure of shapely functors. For the details we refer to the original paper.

Proposition 4.1.6. *Shapely functors T and commutative monads B have distributive laws.*

4.1.3 Some Properties of Distributive Laws

Distributive laws are in general not epic, and thus also not iso. The following lemma provides an explanation.

Lemma 4.1.7. *Distributive laws $\pi : TB \Rightarrow BT$ for monads B and functors T are epic, only if $\eta_T : T \Rightarrow BT$ is epic.*

Proof. This follows immediately from Axiom 4.2 of the definition of distributive laws. □

Thus the following is the only monad among the examples we consider, which has an epic distributive law.

Example 4.1.8. Where $B = Id$, the distributive law $\pi : TB \Rightarrow BT$ for any functor T is id_T and thus iso. Similarly, if $T = Id$, the distributive law $\pi : TB \Rightarrow BT$ with any monad B is id_B and thus iso.

Nevertheless, we can find many examples of distributive laws, which are monic.

Example 4.1.9. The following distributive laws are monic.

1. $\pi_{\mathcal{P}}^{1+Act \times (-)} : 1 + Act \times \mathcal{P}(-) \Rightarrow \mathcal{P}(1 + Act \times (-))$.
2. $\pi_{(\mathbb{N}^{(-)})_{\omega}}^{1+Act \times (-)} : 1 + Act \times (\mathbb{N}^{(-)})_{\omega} \Rightarrow (\mathbb{N}^{(1+Act \times (-))})_{\omega}$.
3. $\pi_{\mathcal{D}}^{1+Act \times (-)} : 1 + Act \times \mathcal{D}(-) \Rightarrow \mathcal{D}(1 + Act \times (-))$

4.2 Kleisli-Lifting of Functors on Set

A Kleisli-lifting of a functor commutes with the free B -algebra functor as in the following definition.

Definition 4.2.1 (Kleisli-Liftings of a Functor). *Let B be a monad and T a functor on Set , and let $F' : Set \rightarrow Kl(B)$ be the free functor into $Kl(B)$. A Kleisli-lifting \bar{T} of T in $Kl(B)$ is a functor \bar{T} making $\bar{T}F' \cong F'T$ commute.*

Kleisli-liftings are commonly defined from distributive laws. Both concepts are mutually interdefinable as in the following proposition.

Proposition 4.2.2. *The existence of a distributive law $\pi : TB \Rightarrow BT$ between a monad B and a functor T is equivalent to the existence of a functor lifting of T into the Kleisli-category $Kl(B)$ of B .*

Proof. Let π be a distributive law as above, we define the functor lifting $\bar{T} : Kl(B) \rightarrow Kl(B)$

- on objects $F'X$ of $Kl(B)$ as $\bar{T}F'X := F'TX$
- and on morphisms $f : F'X \rightarrow F'Y$ as $\bar{T}f := (\pi_X \circ f^{\dagger})^{\dagger 1}$.

¹ $(-)^{\dagger}$ yields the transpose of morphisms under the adjunction $F' \dashv U' : Set \rightarrow Kl(B)$

Given a Kleisli-lifting \bar{T} with $\bar{T}F' = FT$, we define $\pi := (\bar{T}\epsilon_F)^\dagger = U\tilde{T}\epsilon_F \circ \eta_{TB}$. It remains to verify the axioms of distributive laws from Definition 4.1.1. These axioms live in *Set*, but we prove their adjoint transposes in *B-Alg* under the adjunction $F \dashv U$, using that the homsets $B\text{-Alg}(F(-), +) \cong \text{Set}(-, U(+))$ are in bijection.

1. We compute the following.

$(\pi \circ T\eta)^\dagger =$	Definition of $(-)^\dagger$
$\epsilon_{FT} \circ F\pi \circ FT\eta =$	Definition of π
$\epsilon_{FT} \circ FU\tilde{T}\epsilon_F \circ F\eta_{TUF} \circ FT\eta =$	Naturality of η
$\epsilon_{FT} \circ FU\tilde{T}\epsilon_F \circ FUF\eta \circ F\eta_T =$	$FT = \tilde{T}F$
$\epsilon_{FT} \circ FU\tilde{T}\epsilon_F \circ FU\tilde{T}F\eta \circ F\eta_T =$	$FT = \tilde{T}F$
$\epsilon_{FT} \circ F\eta_T = (\eta_T)^\dagger$	Definition of $(-)^\dagger$

2. We compute the following.

$$\begin{aligned}
(\mu_T \circ B\pi \circ \pi_B)^\dagger &= && \text{Definition of } (-)^\dagger \\
\epsilon_{FT} \circ F(\mu_T \circ B\pi \circ \pi_B) &= && \text{Functoriality of } F \\
\epsilon_{FT} \circ F\mu_T \circ FUF\pi \circ F\pi_{UF} &= && \text{Definition of } \pi \\
\epsilon_{FT} \circ F\mu_T \circ FUFU\tilde{T}\epsilon_F \circ FU(F\eta_{TUF} \circ \tilde{T}\epsilon_{FUF}) \circ F\eta_{TUFUF} &= \\
\epsilon_{FT} \circ FU\tilde{T}\epsilon_F \circ FU\tilde{T}\epsilon_{FUF} \circ F\eta_{TUFUF} &= && \epsilon_F \circ \epsilon_{FUF} = F\mu \\
\epsilon_{FT} \circ FU\tilde{T}\epsilon_F \circ FU\tilde{T}F\mu \circ F\eta_{TUFUF} &= && \tilde{T}F = FT \\
\epsilon_{FT} \circ FU\tilde{T}\epsilon_F \circ FUFT\mu \circ F\eta_{TUFUF} &= && \text{Naturality of } \mu \\
\epsilon_{FT} \circ FU\tilde{T}\epsilon_F \circ F\eta_{TUF} \circ FT\mu &= && \text{Definition of } \pi \\
\epsilon_{FT} \circ F\pi \circ FT\mu &= (\pi \circ T\mu)^\dagger && \text{Definition of } (-)^\dagger
\end{aligned}$$

□

In conjunction with Proposition 4.1.6, the previous proposition induces the following corollary. This result is part of the work [HJS06] of Hasuo, Jacobs, and Sokolova.

Corollary 4.2.3. *Commutative monads admit the Kleisli-lifting of a shapely functor.*

4.3 Continuous Extensions of Kleisli-Lifted Functors

As in Definition 4.2.1 the Kleisli-lifting of a *Set*-functor T into the Eilenberg-Moore category $B\text{-Alg}$ of the monad B is a functor \tilde{T} commuting with the free functor $F : \text{Set} \rightarrow B\text{-Alg}$ such that

$$\tilde{T}F \cong FT \tag{4.5}$$

As $F = KF'^2$ and $\tilde{T}F' = F'T$, the above can easily be strengthened as

$$\tilde{T}K = K\tilde{T} \tag{4.6}$$

² K is the comparison functor $Kl(B) \rightarrow B\text{-Alg}$. See also Section VI.3 of [Mac98].

The latter characterises \tilde{T} as the left Kan extension of $K\bar{T}$ along K , where the natural transformation $\alpha : K\bar{T} \Rightarrow \tilde{T}K$ is a natural isomorphism. In the following we define the aforementioned left Kan extension pointwise.

Every B -algebra A arises from a coequaliser

$$FUFUA \xrightleftharpoons[\epsilon_{FUA}]{FU\epsilon_A} FUA \xrightarrow{\epsilon_A} A \quad (4.7)$$

in particular free B -algebras, $A = FX$.

$$FUFUF X \xrightleftharpoons[\epsilon_{FUF X}]{FU\epsilon_{FX}} FUF X \xrightarrow{\epsilon_{FX}} FX \quad (4.8)$$

Using³ $UK = U'$ and $F = KF'$, the above appears equivalent to

$$KF'U'F'U'F'X \xrightleftharpoons[K\epsilon_{F'U'F'X}]{KF'U'\epsilon_{F'X}} KF'U'F'X \xrightarrow{K\epsilon_{F'X}} KF'X \quad (4.9)$$

We do not make explicit that by ϵ in Diagram 4.9 and below we mean the preimage of ϵ from Diagram 4.8 under the full and faithful embedding $K : Kl(B) \rightarrow B\text{-Alg}$. Taking the image of Diagram 4.9 under \tilde{T} yields

$$\tilde{T}KF'U'F'U'F'X \xrightleftharpoons[\tilde{T}K\epsilon_{F'U'F'X}]{\tilde{T}KF'U'\epsilon_{F'X}} \tilde{T}KF'U'F'X \xrightarrow{\tilde{T}K\epsilon_{F'X}} \tilde{T}KF'X \quad (4.10)$$

We look for a natural transformation $\alpha : K\bar{T} \Rightarrow \tilde{T}K$. In fact α arises, when we define \tilde{T} as the coequaliser of $K\bar{T}F'U'\epsilon'_{F'X}$ and $K\bar{T}\epsilon'_{F'U'F'X}$.

$$K\bar{T}F'U'F'U'F'X \xrightleftharpoons[K\bar{T}\epsilon'_{F'U'F'X}]{K\bar{T}F'U'\epsilon'_{F'X}} K\bar{T}F'U'F'X \longrightarrow \tilde{T}KF'X \quad (4.11)$$

The following shows that the above defines \tilde{T} as the pointwise left Kan extension of $K\bar{T}$ along K .

Lemma 4.3.1. *Defined as above, $\tilde{T}A$ is $\text{colim}_{f:KX \rightarrow A} K\bar{T}X$.*

³We distinguish $F : \text{Set} \rightarrow B\text{-Alg}$ and $U : B\text{-Alg} \rightarrow \text{Set}$ from $F' : \text{Set} \rightarrow Kl(B)$ and $U' : Kl(B) \rightarrow \text{Set}$.

Proof. $\widetilde{T}A$ has a cocone in the comma category⁴ $(K \downarrow A)$, in particular any object $f : KF'X \rightarrow A$ of $(K \downarrow A)$ is transposed under $F \dashv U$ to $f^\dagger : X \rightarrow UA$. $K\overline{T}F'$ takes such a f^\dagger to $K\overline{T}F'f^\dagger : K\overline{T}F'X \rightarrow K\overline{T}F'UA$, which composes with the coequalising B -algebra morphism into $\widetilde{T}A$. This yields a component $K\overline{T}F'X \rightarrow \widetilde{T}A$ of the cocone of A .

For every cocone with object B there is a unique morphism $h : \widetilde{T}A \rightarrow B$, because the component $g : K\overline{T}F'U'F'X$ of cocone B coequalises $K\overline{T}F'U'\epsilon'_{F'X}$ and $K\overline{T}\epsilon'_{F'U'F'X}$. So that the existence and uniqueness of h follows from the definition of $\widetilde{T}A$ as the coequaliser object in (4.11). \square

Lemma 4.3.2. $\widetilde{T}K \cong K\overline{T}$.

Proof. The proof is an immediate consequence of the definition of \widetilde{T} . \square

Example 4.3.3. The continuous extension of the Kleisli-lifting of $T(-) = \{\sqrt{}\} + \text{Act} \times (-)$ for any monad is $\widetilde{T}(-) = F\{\sqrt{}\} + \text{Act} \cdot (-)$.

⁴See Definition B.1.18.

Part II

Semantics of Coalgebras with Branching

Chapter 5

Finite Trace Semantics

Finite trace semantics has been extensively studied for nondeterministic [vG90] and probabilistic [Seg95] labelled transition systems, and finds application for instance in the semantics of automata. In Generic Trace Theory [Jac04, HJS07] Jacobs, Hasuo and Sokolova have generalised finite trace semantics not only to a coalgebraic [PT99] level of generality in the transition type, but also to a wide range of branching types subsuming the two previously mentioned. Their approach had several shortcomings, which we will outline in Section 5.1. We suggest a slight modification in Sections 5.2 and 5.3, which allows us to eliminate several assumptions made in [HJS07] and thereby not only further increase the generality of the definition, but also clarify the construction.

Assumption 5.0.4. *In this chapter we assume that*

1. *B is a monad on \mathbf{Set} ,*
2. *T is a functor on \mathbf{Set} , and*
3. *there is a distributive law $\pi : TB \Rightarrow BT$.*

Henceforth we fix a distributive law π .

5.1 A Review of Generic Trace Theory

In Generic Trace Theory [HJS06], finite trace semantics of (B, T) -coalgebras is defined generically as cocones over the discrete ω -chain sequence $|Seq_{\bar{T}}|^1$ for the Kleisli-lifting \bar{T} of the functor T in $Kl(B)$.

Hasuo, Jacobs, and Sokolova have shown that the colimit of $Seq_{\bar{T}}$ coincides with the limit of the sequence formed by the stepwise projections along the initial sequence for \bar{T} , if the following assumptions are met. The resulting finite trace semantics and finite trace equivalence are coinductive.

- Assumption 5.1.1.** 1. *The Kleisli-category $Kl(B)$ can be enriched in the category $DCPO_{\perp}$ of directed complete partial orders \leq with bottom element \perp and continuous morphisms, such that the Kleisli-lifting \bar{T} of T is a functor enriched in $DCPO_{\perp}$.*
2. *Morphism composition is left-strict with respect to \perp , that is $\perp \circ f = \perp$ for all morphisms f .*

Objects of $DCPO_{\perp}$ are directed complete partial orders with bottom element.

Definition 5.1.2 (Directed Complete Partial Orders). *A partial order with bottom element is a set X augmented with a binary relation $\leq \subseteq X \times X$*

1. *reflexive, such that $x \leq x$ for all $x \in X$,*
2. *antisymmetric, such that $x = y$ whenever $x \leq y$ and $y \leq x$,*
3. *transitive, such that $x \leq z$ whenever $x \leq y$ and $y \leq z$ for any $x, y, z \in X$, and*
4. *there is a bottom element $\perp \in X$ with $\perp \leq x$ for all $x \in X$.*

A directed set in a partial order $\langle X, \leq \rangle$ is a set $Y \subset X$, such that

1. *Y is non-empty, and*
2. *for all $x, y \in Y$ there is a $z \in Y$ with $x \leq z$ and $y \leq z$.*

¹See Definition B.1.13 for $Seq_{\bar{T}}$ and Definitions B.1.3 and B.1.11 for discrete diagrams.

A partial order $\langle X, \leq \rangle$ is directed complete, if all directed sets $Y \subseteq X$ have a least upper bound in X with respect to \leq . We denote the least upper bound of Y as $\bigvee Y$.

Directed complete partial orders with bottom element and their morphisms form a category.

Definition 5.1.3 (The Category $DCPO_{\perp}$). *The objects of $DCPO_{\perp}$ are directed complete partial orders with bottom element, and morphisms between directed complete partial orders $\langle X, \leq_X, \perp_X \rangle$ and $\langle Y, \leq_Y, \perp_Y \rangle$ with bottom elements are functions $f : X \rightarrow Y$*

1. *commuting with the order such that $f(x) \leq_Y f(x')$ whenever $x \leq_X x'$,*
2. *preserving the bottom element $f(\perp_X) = \perp_Y$, and*
3. *continuous such that $f(\bigvee_X X') = \bigvee_Y f[X']$ for all directed sets X' in $\langle X, \leq_X \rangle$.*

Remark 5.1.4. *In Remark 3.6 of [HJS06], Jacobs et alii point out that, for their result to hold, \overline{T} need not necessarily be continuous, in the sense that $\overline{T}(\bigvee Y)$ need not be $\bigvee \overline{T}[Y]$ for any directed set $Y \subseteq Kl(B)(X, Y)$ for any pair of objects X, Y in $Kl(B)$.*

We give an explicit description of $DCPO_{\perp}$ -enrichments.

Definition 5.1.5 ($DCPO_{\perp}$ -Enrichment of $Kl(B)$). *An enrichment of the Kleisli-category $Kl(B)$ of a monad B in $DCPO_{\perp}$ is characterised by the following axioms.*

1. *For every pair of objects X and Y , the homset $Kl(B)(X, Y)$ is ordered by a directed complete partial order $\leq_{X,Y}$. We will write \leq , when X and Y are clear from context.*
2. *For every pair of objects X and Y , there is a bottom element \perp in $C(X, Y)$ with respect to \leq , such that $\perp \leq f$ for any morphism $f \in C(X, Y)$.*
3. *Composition is left-strict with respect to \leq , that is $\perp \circ f = \perp$ for all morphisms f .*
4. *Composition is monotone with respect to \leq , that is $g_1 \circ h \leq g_2 \circ h$ and $f \circ g_1 \leq f \circ g_2$ whenever $g_1 \leq g_2$ for all compatible morphisms f, g_1, g_2 , and h .*

\overline{T} is $DCPO_{\perp}$ -enriched, if

5. \bar{T} is monotone with respect to \leq , that is $\bar{T}(g_1) \leq \bar{T}(g_2)$ whenever $g_1 \leq g_2$

Under these assumptions the following classical result of Smyth and Plotkin [SP82] holds. For the proof we refer to the original literature.

Theorem 5.1.6. *In a category \mathcal{C} enriched in \mathbf{DCPO}_\perp , the ω -colimit of the initial sequence of an endofunctor T on \mathcal{C} coincides with the ω -limit of the sequence formed by the step-wise projections along the initial sequence of T , given either side exists.*

Definition 5.1.7 (Finite Trace Semantics in Generic Trace Theory). *In Generic Trace Theory, finite trace semantics tr^ω of a (B, T) -coalgebra $\mathbb{S} = \langle S, \sigma, s_I \rangle$ is defined by inductive construction of a cone $(tr_n^\omega)_{n < \omega}$*

$$tr_0^\omega = \perp_{Kl(B)(S, \emptyset)} \text{ and } tr_{n+1}^\omega = \bar{T} tr_n^\omega \circ \sigma \quad (5.1)$$

as in the following diagram

$$\begin{array}{ccccccc} & & S & \xrightarrow{\sigma} & \bar{T}S & & \\ & \swarrow^{tr_0^\omega} & & \searrow^{tr_n^\omega} & \searrow^{tr_{n+1}^\omega} & \searrow^{\bar{T}tr_n^\omega} & \\ \emptyset & & \dots & & \bar{T}^n \emptyset & \rightarrow & \bar{T}^{n+1} \emptyset \quad \dots \end{array} \quad (5.2)$$

Let $(d_n)_{n < \omega} : Seq_{\bar{T}} \Rightarrow colim(Seq_{\bar{T}})$ be the cocone of $colim(Seq_{\bar{T}})$ over $|Seq_{\bar{T}}|$.

$$\begin{array}{ccccccc} & & colim(Seq_{\bar{T}}) & & & & \\ & \swarrow^{d_0} & & \swarrow^{d_n} & \swarrow^{d_{n+1}} & & \\ \emptyset & & \dots & & \bar{T}^n \emptyset & \rightarrow & \bar{T}^{n+1} \emptyset \quad \dots \end{array} \quad (5.3)$$

Composing $(tr_n^\omega)_{n < \omega}$ with $(d_n)_{n < \omega}$ componentwise, yields a family of morphisms $(d_n \circ tr_n^\omega)_{n < \omega} : S \rightarrow colim(Seq_{\bar{T}})$. By enrichment in \mathbf{DCPO}_\perp , this family is a directed set in $Kl(B)(S, colim(Seq_{\bar{T}}))$ and has thus an upper bound. In Generic Trace Theory, finite trace semantics tr^ω is defined to be this upper bound.

Remark 5.1.8. Because $\perp_{Kl(B)(S, \emptyset)}$ is uniquely defined, $\bar{T}(tr_n^\omega)_{n < \omega}$ extends uniquely to a cone, which by definition of $(tr_n^\omega)_{n < \omega}$ coincides with $(tr_n^\omega)_{n < \omega}$.

Theorem 5.1.6 yields that the isomorphism $\text{colim}(\text{Seq}_{\bar{T}}) \cong \bar{T}\text{colim}(\text{Seq}_{\bar{T}})$ is the final \bar{T} -coalgebra taken in the enriched setting and that tr^ω is a \bar{T} -coalgebra morphism, so that finite trace semantics is coinductive.

Remark 5.1.9. *The definition of finite trace semantics in Generic Trace Theory has several drawbacks.*

1. *The enrichment in DCPO_\perp is not unique, in particular one may define $\text{Kl}(B)$ to be enriched over flat orders, and obtain finite trace semantics.*
2. *With the exception of trivial orderings, such as the flat ordering, the Kleisli-category of finitary monads can not be enriched in partial orders directed complete. A common example is the finitary powerset monad, which as a branching type otherwise has a well-understood finite trace semantics for many transition types.*

The following example emphasises the dependency of finite trace semantics on the choice of an order enrichment for $\text{Kl}(B)$.

Example 5.1.10 (Subset-Ordering on $\text{Kl}(\mathcal{P})$). *A natural choice for an ordering on morphisms in $\text{Kl}(\mathcal{P})$ is induced by the inclusion-ordering on subsets. Let X and Y be objects in $\text{Kl}(\mathcal{P})$, morphisms $f, g : X \rightarrow Y$ can be ordered such that $f \leq g$ if and only if $f(x) \subseteq g(x)$ for all $x \in X$. We need to verify that the above soundly defines an order enrichment on $\text{Kl}(\mathcal{P})$ as in Definition 5.1.5.*

1. *In the free algebras of \mathcal{P} , \subseteq is a complete partial order.*
2. *For each pair of objects X and Y , $\perp_{X,Y}(x) := \emptyset$ defines a morphism as $\perp_{X,Y}(x \cup y) = \perp_X \cup \perp_Y = \emptyset \cup \emptyset = \emptyset$.*
3. *For all $f : Y \rightarrow Z$, $f \circ \perp_{X,Y} = \perp_{X,Z}$.*
4. *For all $f : X \rightarrow Y$ and $g, g' : Y \rightarrow Z$ with $g \leq g'$, thus $g(y) \subseteq g'(y)$ for all $y \in Y$, and in particular for those y in the image of f . Thus for all $x \in X$, $g(f(x)) \subseteq g'(f(x))$ by functionality of f . For all $g : Y \rightarrow Z$ and $f, f' : X \rightarrow Y$ with $f \leq f'$, and for all $x \in X$, $f(x) \subseteq f'(x)$ and thus $g(f(x)) \subseteq g(f'(x))$ by monotonicity of g .*

The details of the following example appeared in [Has08].

Example 5.1.11 (Finite Trace Semantics for the Subset Ordering in $Kl(\mathcal{P})$). *Let the branching type be $B = \mathcal{P}$ and the transition type $T(-) = \{\sqrt{}\} + Act \times (-)$ with $Kl(\mathcal{P})$ enriched in the subset ordering. Let $\mathbb{S} = \langle S, \sigma \rangle$. For every $n \leq \omega$, tr_n^ω takes a state $s \in S$ into the set of finite traces, that is $\sqrt{}$ -terminated Act-words, of depth at most n in σ beginning from s . That $tr_n^\omega \leq tr_m^\omega$ means that $tr_n^\omega(s) \subseteq tr_m^\omega(s)$ for all $n \leq m < \omega$. The colimit in \leq then means the union, so that $tr^\omega(s) = \bigcup_{n < \omega} tr_n^\omega(s)$.*

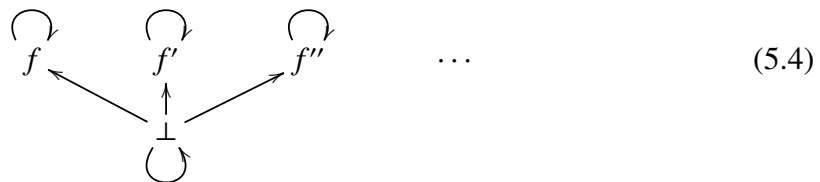
Definitions analogous to the subset ordering yield natural order enrichments for Kleisli-categories of the following monads.

Example 5.1.12. *Common to the following examples is that the order on morphisms is inherited from their codomain. Thus it suffices to describe the ordering on objects of the Kleisli-category, which are the free algebras of the branching type.*

1. *For the Bag-monad $(\mathbb{N}^{(-)})_\omega$ and a set X , let $m, n \in (\mathbb{N}^X)_\omega$. Then $m \leq n$ if and only if $m(x) \leq n(x)$ for all $x \in X$.*
2. *Let $D, E \in FX$ be sub-distributions over X , then $D \leq E$ if and only if $D(x) \leq E(x)$ for all $x \in X$.*

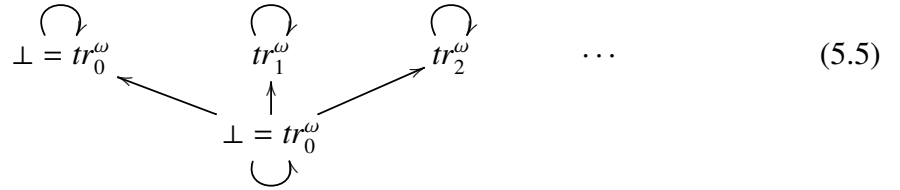
Remark 5.1.13. *For many relevant monads B , we can not find such an ordering on objects of $Kl(B)$. Among these monads are the strict distribution monad $\mathcal{D}_{=1}$ and the identity monad Id .*

Example 5.1.14 (Finite Trace Semantics for Flat Orders on $Kl(\mathcal{P})$). *An alternative to the natural ordering above is an enrichment in flat orders, such that precisely $\perp_{X,Y} \leq f$ and $f \leq f$ for all $f : X \rightarrow Y$, where \perp is defined as for the natural order: $\perp_{X,Y}(x) := \emptyset$. The following Hasse diagram depicts an example of a flat order.*



1. Flat orders are directed complete partial orders.
2. Define \perp as above.
3. For all $f : Y \rightarrow Z$, $\perp_{YZ} \circ f = \perp_{XZ}$, from which the following follows.
4. For all $f : Y \rightarrow Z$ and $g, g' : Y \rightarrow Z$ with $g \leq g'$, then either $g = g'$ from which $g \circ f = g' \circ f$ follows trivially, or $g = \perp$, so that $g \circ f = \perp$ and thus $g \circ f \leq g' \circ f$.
For all $g : Y \rightarrow Z$ and $f, f' : X \rightarrow Y$ with $f \leq f'$, either $f = f'$, so that $f \circ g = f' \circ g$ and thus $f \circ g \leq f' \circ g$, or $f = \perp_{YZ}$, so that $\perp_{YZ} \circ g = \perp_{XZ}$ and thus $f \circ g \leq f' \circ g$.

In our example let $\perp_{X,Y}$ be the adjoint transpose of the function taking every element of X into $\emptyset \in \mathcal{P}Y$ under $F \dashv U$. The stepwise embedding of the cone tr^ω yield a flat order as in the following Hasse diagram.



The colimit of tr^ω is easily seen to be tr_m^ω for some $m < \omega$, such that $tr_n^\omega = \perp$ for all $n < m$, and $tr_n^\omega = tr_m^\omega$ for all $n \geq m$. Because $tr_{n+1}^\omega := \overline{T}tr_n^\omega \circ \sigma^\dagger$, $tr_n^\omega = \perp$ for all $n < \omega$.

5.2 Non-Coinductive Finite Trace Semantics in Kleisli Categories

In this chapter we remove several assumptions from the definition of finite trace semantics in Generic Trace Theory, and thereby loose the coinductive nature of finite trace semantics and finite trace equivalence.

Remark 5.2.1. *Our definition of finite trace semantics has several advantages over the one in Generic Trace Theory, as it is*

1. well-defined for finitary branching types, such as finitary non-determinism, and

2. uniquely defined, that is independent from a choice of ordering.

Assumption 5.2.2. We assume that

1. $B\emptyset$ is not empty, and
2. there is for each set X a map $\iota_X : X \rightarrow B\emptyset$ factoring through the final object map such that $\iota_X = \iota_{\{*\}} \circ !_X$.

$$\begin{array}{ccc}
 X & \xrightarrow{\iota_X} & B\emptyset \\
 & \searrow !_X & \nearrow \iota_{\{*\}} \\
 & \{*\} &
 \end{array} \tag{5.6}$$

3. There is a distributive law $\pi : TB \Rightarrow BT$.

By the axiom of choice we may choose a candidate for $\iota_{\{*\}}$, so that 2 implies 3. Henceforth we fix a distributive law π .

Remark 5.2.3. The details of 2. of Assumption 5.2.2 replace the assumption in Generic Trace Theory, that $\iota_X^\dagger : FX \rightarrow F\emptyset$ is the bottom element in the $Kl(B)(FX, F\emptyset)$.

Many relevant branching and transition types satisfy the above assumptions, as the following examples show.

Example 5.2.4. The following monads satisfy 1 and 2 of Assumption 5.2.2.

1. \mathcal{P} satisfies the above conditions, as $\mathcal{P}\emptyset = \{\emptyset\}$ and $\iota_X : x \mapsto \emptyset$ for all $x \in X$, and for a similar argument \mathcal{P}_ω , too.
2. $\left(\mathbb{N}^{(-)}\right)_\omega$ satisfies the above conditions, as $\left(\mathbb{N}^\emptyset\right)_\omega = \{\dot{\emptyset}\}$ and $\iota_X : x \mapsto \dot{\emptyset}$ for all $x \in X$, where $\dot{\emptyset}$ denotes the empty multiset.
3. $\mathcal{D}_{\leq 1}$ satisfies the above conditions, as $\mathcal{D}_{\leq 1}\emptyset = \{\emptyset\}$ and $\iota_X : x \mapsto \{\emptyset\}$ for all $x \in X$, where \emptyset is the empty distribution.

Proof. It remains to verify 2 of Assumption 5.2.2 for the examples above.

1. If $X = \emptyset$, ι_X and $!_X$ are empty functions, so that Diagram 5.6 commutes trivially. Otherwise, $\iota_X(x) = \emptyset = \iota_{\{*\}}(*) = \iota_{\{*\}}!_X(x)$ for all $x \in X$. The same argument is sound for \mathcal{P}_ω .

2. The same argument applies to $(\mathbb{N}^{(-)})_\omega$. If $X = \emptyset$, ι_X and $!_X$ are empty functions, and Diagram 5.6 commutes. Otherwise, $\iota_X(x) = \emptyset = \iota_{\{*\}}(*) = \iota_{\{*\}}!_X(x)$ for all $x \in X$.
3. The same argument applies to $\mathcal{D}_{=1}$ as well. If $X = \emptyset$, ι_X and $!_X$ are empty functions, and Diagram 5.6 commutes. Otherwise, $\iota_X(x) = \emptyset = \iota_{\{*\}}(*) = \iota_{\{*\}}!_X(x)$ for all $x \in X$.

□

Example 5.2.5. Monads B with $B\emptyset = \emptyset$ do not satisfy 1 of Assumption 5.2.2. These include Id and $\mathcal{D}_{=1}$.

Finite trace semantics is defined as a cocone over $|Seq_{\bar{T}}|$ inductively from the morphism $\iota_S^\dagger : FS \rightarrow F\emptyset$, the transpose of ι_S under $F \dashv U$. Henceforth we denote $0 := F\emptyset$.

Definition 5.2.6 (Finite Trace Semantics). *The finite trace semantics of a (B, T) -coalgebra $\mathbb{S} = \langle S, \sigma \rangle$ is a cocone $(tr_n^\omega : S \rightarrow \bar{T}^n 0)_{n < \omega}$ inductively defined as*

$$tr_0^\omega = \iota_S \text{ and } tr_{n+1}^\omega := \bar{T}tr_n^\omega \circ \sigma^\dagger \quad (5.7)$$

for all $n < \omega$ as in the following diagram

$$\begin{array}{ccccccc}
 & FS & \xrightarrow{\sigma^\dagger} & \bar{T}FS & & & \\
 tr_0^\omega \swarrow & & & & \searrow \bar{T}tr_n^\omega & & \\
 0 & \dots & tr_n^\omega \searrow & \bar{T}^n 0 & \xrightarrow{tr_{n+1}^\omega} & \bar{T}^{n+1} 0 & \dots
 \end{array} \quad (5.8)$$

Example 5.2.7. Consider the following example of a (B, T) -coalgebra $\mathbb{S} = \langle S, \sigma \rangle$ with the branching type $B = (\mathbb{N}^{(-)})_\omega$ and the transition type $T(-) = \{\sqrt{}\} + Act \cdot (-)$.

$$\begin{array}{c}
 \begin{array}{ccc}
 & 1 \cdot a & \\
 & \curvearrowright & \\
 x & \xrightarrow{2 \cdot a} & y \\
 & \xleftarrow{3 \cdot b} & \\
 & 1 \cdot \sqrt{} & \\
 & \longrightarrow &
 \end{array}
 \end{array} \quad (5.9)$$

We compute the finite traces of x and y for a successively increasing depth as follows.

1. $tr_0^\omega(x) = 0$ and $tr_0^\omega(y) = 0$
2. $tr_1^\omega(x) = 1 \cdot a(tr_0^\omega(x)) + 2 \cdot a(tr_0^\omega(y)) = 1 \cdot a(0) + 2 \cdot a(0) = 0 + 0 = 0$ and $tr_1^\omega(y) = 1 \cdot \sqrt{} + 3 \cdot b(tr_0^\omega(x)) = 1 \cdot \sqrt{} + 3 \cdot b(0) = 1 \cdot \sqrt{}$

$$3. \ tr_2^\infty(x) = 1 \cdot a(tr_1^\infty(x)) + 2 \cdot a(tr_1^\infty(y)) = 1 \cdot a(0) + 2 \cdot a(1 \cdot \surd) = (2 * 1) \cdot a\surd = 2 \cdot a\surd$$

$tr_2^\infty(y)$ and tr_n^∞ for $n > 2$ can be computed similarly.

Finite trace semantics does in general not commute with the morphisms $(\overline{T}^n i_{\overline{T}0})_\omega$ of $Seq_{\overline{T}}$, because $\overline{T}^n i_{\overline{T}0}$ embeds traces of depth up to n into the set of potential traces of depth up to $n + 1$.

Taking ι_{T0} along BT^n yields a sequence of morphisms $(BT^n \iota_{T0} : BT^{n+1}0 \rightarrow BT^n0)_{n < \omega}$. These morphisms are the stepwise projections of the morphisms $(BT^n i_{BT0})_{n < \omega}$ along $USeq_{\overline{T}}$. Then we see that $tr^{\omega\dagger}$ commutes with the stepwise projections of the morphisms along $USeq_{\overline{T}}$. The following proposition is based on 2 of Assumption 5.2.2.

Proposition 5.2.8. $BT^n \iota_{T0} \circ tr_{n+1}^{\omega\dagger} = tr_n^{\omega\dagger}$ commutes for all $n < \omega$.

Proof. We prove the proposition by induction on n .

Because $\{*\}$ is final, $!_{BT0} \circ tr_1^{\omega\dagger} = !_S$ commutes. By 2. of Assumption 5.2.2, $\iota_{\{*\}} \circ !_S = tr_0^{\omega\dagger}$ and $\iota_{\{*\}} \circ !_S = \iota_{BT0}$ commute, so that $\iota_{BT0} \circ tr_1^{\omega\dagger} = tr_0^{\omega\dagger}$ and $\iota_{BT0} \circ tr_1^{\omega\dagger} = tr_0^{\omega\dagger}$ commute.

$$\begin{array}{ccc}
 S & \xrightarrow{\sigma} & BTS \\
 \downarrow tr_0^{\omega\dagger} & \searrow tr_1^{\omega\dagger} & \downarrow \mu_{T0} \circ B\pi_0 \circ BT tr_0^\omega \\
 B0 & \xleftarrow{\iota_{BT0}} & BT0 \\
 \swarrow \iota_{\{*\}} & \nearrow !_S & \swarrow !_S \\
 \{*\} & & \{*\}
 \end{array} \tag{5.10}$$

The induction step consists in the following chain of implications

$$tr_n^{\omega\dagger} = BT^n \iota_{T0} \circ tr_{n+1}^{\omega\dagger} \implies \tag{1.}$$

$$\mu_{T^{n+1}0} \circ B\pi_{T^n0} \circ BT tr_n^{\omega\dagger} = \mu_{T^{n+1}0} \circ B\pi_{T^n0} \circ BT BT^n \iota_{T0} \circ BT tr_{n+1}^{\omega\dagger} \implies \tag{2.}$$

$$\mu_{T^{n+1}0} \circ B\pi_{T^n0} \circ BT tr_n^{\omega\dagger} = BT^{n+1} \iota_{T0} \circ \mu_{T^{n+2}0} \circ B\pi_{T^{n+1}0} \circ BT tr_{n+1}^{\omega\dagger} \implies \tag{3.}$$

$$tr_{n+1}^{\omega\dagger} = BT^{n+1} \iota_{T0} \circ tr_{n+2}^{\omega\dagger}$$

by

1. functoriality of BT ,
2. naturality of μ and π , and
3. definition of tr^ω ,

respectively. □

Definition 5.2.9 (Finite Trace Equivalence). *Let $\mathbb{S} = \langle S, \sigma \rangle$ be a (B, T) -coalgebra. Finite trace semantics tr^ω for \mathbb{S} induces an equivalence relation $\sim_{tr^\omega} \subseteq S \times S$ with*

$$s \sim_{tr^\omega} s' \text{ if and only if } \forall n < \omega. tr_n^\omega(s) =_n tr_n^\omega(s') \quad (5.11)$$

for all $s, s' \in S$, where the equality $=_n$ is taken in the free B -algebra $FT^n\emptyset$. We call \sim_{tr^ω} the finite trace equivalence over \mathbb{S} induced by tr^ω .

Remark 5.2.10. *The statement of Definition 5.2.9 is equivalent to saying that finite trace equivalence is the kernel $\ker tr^\omega$ of finite trace semantics tr^ω .*

In order to be able to use Definition C.0.23 of T -bisimilarity as the largest T -bisimulation, we need to add the following assumption.

Assumption 5.2.11. *We assume that T preserves weak pullbacks.*

Proposition 5.2.12. *In any (B, T) -coalgebra, BT -bisimilar states are finite trace equivalent.*

Proof. Recall from Lemma C.0.26 that BT -bisimilarity is contained in the kernel of the cone $f : S \Rightarrow Seq^{BT}$. We show that the adjoint transpose of $tr^\omega : S \Rightarrow Seq_{\bar{T}}$ under $F \dashv U$ factors through f via a natural transformation $g : Seq^{BT} \Rightarrow BSeq_T$ defined inductively as

$$g_0 := \iota_{\{*\}} \text{ and } g_{n+1} := \mu_{T^{n+1}0} \circ B\pi_{T^n0} \circ BT g_n \quad (5.12)$$

for all $n < \omega$. It remains to show that $(tr_n^\omega)^\dagger = g_n \circ f_n$ commutes for all n . In the base case, $n = 0$, $\iota_{\{*\}} \circ !_S = \iota_S$ commutes by Assumption 5.2.2. The induction step consists in

the following chain of implications.

$$\begin{aligned}
(tr_n^\omega)^\dagger &= g_n \circ f_n \implies && \text{by functoriality of } BT \\
\mu \circ \pi \circ BT(tr_n^\omega)^\dagger \circ \sigma &= \mu \circ \pi \circ BT g_n \circ BT f_n \circ \sigma \implies && \text{by definition of } tr^\omega \\
(tr_{n+1}^\omega)^\dagger &= g_{n+1} \circ f_{n+1}
\end{aligned}$$

□

5.3 Finite Trace Semantics in Eilenberg-Moore Categories

In this section we mainly show that finite trace semantics for a branching type B defined in the Kleisli-category $Kl(B)$ embeds into the Eilenberg-Moore category $B\text{-Alg}$. In Chapter 8 we consider finite trace semantics in Eilenberg-Moore categories, only. In Proposition 5.3.3 we recover finite trace semantics as a final coalgebra semantics in the case where the final \widetilde{T} -sequence has a limit after ω steps. Then finite trace semantics and the induced finite trace equivalence are coinductive as under the limit-colimit coincidence assumed in Generic Trace Theory [HJS07]. In this section we make the same assumptions as in the previous one.

Assumption 5.3.1. *We assume that*

1. $B\emptyset$ is not empty, and
2. there is for each set X a map $\iota_X : X \rightarrow B\emptyset$ factoring through the final object map such that $\iota_Y = \iota_{\{*\}} \circ !_Y$.

$$\begin{array}{ccc}
X & \xrightarrow{\iota_X} & B\emptyset \\
& \searrow \scriptstyle !_X & \nearrow \scriptstyle \iota_{\{*\}} \\
& \{*\} &
\end{array} \tag{5.13}$$

3. There is a distributive law $\pi : TB \Rightarrow BT$.

Henceforth we fix a distributive law π .

Remark 5.3.2. *The definition of finite trace semantics is embedded along $K : Kl(B) \rightarrow B\text{-Alg}$ in $B\text{-Alg}$, since*

1. K preserves the initial object, and
2. K preserves the initial sequence because $K\bar{T} = \bar{T}K$ commutes.

As $B\text{-Alg}$ is complete, the final object 1 exists. In fact 1 is defined to be the B -algebra $\langle \{*\}, !_{B\{*\}} \rangle$. From 1 we obtain the ω^{op} -chain $Seq^{\bar{T}}$ inductively as in Equation (1.9). We show that finite trace semantics tr^ω commutes with $Seq^{\bar{T}}$.

Proposition 5.3.3. *For all $n < \omega$, $\bar{T}^n !_0 \circ Ktr_n^\omega = \bar{T}^n !_{\bar{T}1} \circ \bar{T}^{n+1} !_0 \circ Ktr_{n+1}^\omega$.*

Proof. We prove the proposition by induction over n . In the base case $n = 0$, $!_{\bar{T}1} \circ \bar{T} !_0 \circ tr_1 = !_0 \circ tr_0^\omega$ by finality of 1. The induction step follows from the following chain of implications.

$$\begin{aligned}
 \bar{T}^n !_{\bar{T}1} \circ \bar{T}^{n+1} !_0 \circ tr_{n+1} &= \bar{T}^n !_0 \circ tr_n \implies && \text{by functoriality of } \bar{T} \\
 \bar{T}^{n+1} !_{\bar{T}1} \circ \bar{T}^{n+2} !_0 \circ \bar{T} tr_{n+1} \circ \sigma &= \bar{T}^{n+1} !_0 \circ \bar{T} tr_n \circ \sigma \implies && \text{by definition of } tr^\omega \\
 \bar{T}^{n+1} !_{\bar{T}1} \circ \bar{T}^{n+2} !_0 \circ tr_{n+2} &= \bar{T}^{n+1} !_0 \circ \bar{T} tr_{n+1}^\omega
 \end{aligned}$$

□

Remark 5.3.4. *If the limit $\lim(Seq^{\bar{T}})$ exists, $\lim(Seq^{\bar{T}}) = \bar{T}\lim(Seq^{\bar{T}})$ are isomorphic, and the isomorphism forms the final \bar{T} -coalgebra. Then there is a unique \bar{T} -coalgebra morphism $h : FS \rightarrow \lim(Seq^{\bar{T}})$. If $Seq^{\bar{T}}$ terminates after ω steps, $tr^\omega = h$ coincide, and finite trace semantics and finite trace equivalence are coinductive.*

Chapter 6

Infinite Trace Semantics

In Chapter 5 we have defined finite trace semantics for (B, T) -coalgebras, which, informally speaking, assigns to each state of such a coalgebra a B -algebra term of its finite traces. In this chapter we define infinite trace semantics for (B, T) -coalgebras which assigns to each state a B -algebra term of its infinite traces.

We will see that the acceptance behaviour of nondeterministic coalgebra automata $\mathbb{A} = \langle Q, \theta, q_I, \Omega \rangle$ forms an infinite trace semantics of the underlying transition structure $\langle Q, \theta, q_I \rangle$ conceived as a (\mathcal{P}, T) -coalgebra parameterised in the priority function Ω . Consequently, infinite trace semantics is not uniquely defined.

Our approach to infinite trace semantics differs from the “possibly infinite traces” of Cirstea [Cî10]. “Possibly infinite traces” are defined based on Assumptions 5.1.1 of Generic Trace Theory, and the following.

- B is an affine monad [Jac94], so that that $B\{*\} = \{*\}$ are isomorphic.
- T preserves the limit of Seq^T , so that the final T -coalgebra exists.

Under these assumptions the assignment of “possibly infinite traces” to states in (B, T) -coalgebras is coinductive and unique, unlike our definition of infinite trace semantics.

In Section 6.1 we begin with an inductively defined infinite trace semantics in the style of Böhm trees [Bar81, Abr90], which is not necessarily coinductive. Then we characterise the codomain of generic infinite trace semantics and obtain a coinductively defined infinite trace semantics, which subsumes not only the generic infinite trace semantics, but also

the infinite trace semantics of Jacobs [Jac04] and the acceptance behaviour of coalgebra automata.

Assumption 6.0.5. *In this chapter we assume that*

1. B is a monad on Set , and
2. T is a functor on Set , such that
3. there is a distributive law $\pi : TB \Rightarrow BT$.

Henceforth we fix a distributive law π .

For later use in this Chapter, we define the following.

Definition 6.0.6. *Let $\mathbb{S} = \langle S, \sigma \rangle$ be a (B, T) -coalgebra, and let $\mu : BB \Rightarrow B$ be the multiplication of B and $\pi : TB \Rightarrow BT$ a distributive law for T and B , then we define for all $n < \omega$*

- $\sigma^n : S \rightarrow (BT)^n S$ such that $\sigma^0 := \text{id}$ and $\sigma^{n+1} := (BT)^n \sigma \circ \sigma^n$,
- $\pi^n : T^n B \Rightarrow BT^n$ such that $\pi^0 : \text{id}_B$ and $\pi^{n+1} := \pi_{T^n} \circ T\pi^n$,
- $\tau^n : (BT)^n \Rightarrow BT^n$ such that $\tau^0 := \eta$ and $\tau^{n+1} := \mu_{T^{n+1}} \circ B\pi_{T^n} \circ (BT)\tau^n$, and
- $\mu^n : B^n \Rightarrow B$ such that $\mu^0 := \eta$ and $\mu^{n+1} = \mu \circ B\mu^n$.

6.1 Generic Infinite Trace Semantics

We define generic infinite trace semantics tr^∞ inductively over the depth of horizon, such that tr_n^∞ describes the prefixes of infinite traces up to depth n . Informally, prefixes are traces terminated by $*$, which stands for unknown behaviour. At depth 0, tr_0^∞ assigns to each state the unknown behaviour $*$.

Definition 6.1.1 (Generic Infinite Trace Semantics). *Let $\mathbb{S} = \langle S, \sigma \rangle$ be a (B, T) -coalgebra. We define the generic infinite trace semantics of \mathbb{S} to be the cone $tr^\infty : FS \Rightarrow |FSeq^T|$, where $F : \text{Set} \rightarrow B\text{-Alg}$ is the free functor into $B\text{-Alg}$, Seq^T is the final T -sequence*

as in Definition B.1.14, and $|Seq^T|$ is the discrete version of Seq^T . Then tr^∞ is defined inductively such that

$$tr_0^\infty := F!_S \text{ and } tr_{n+1}^\infty := \bar{T}tr_n^\infty \circ \sigma^\dagger \quad (6.1)$$

as in the following diagram.

$$\begin{array}{ccccccc}
 & & FS & \xrightarrow{\sigma^\dagger} & \bar{T}FS & & \\
 & \swarrow tr_0^\infty & & \searrow tr_n^\infty & \searrow tr_{n+1}^\infty & \searrow \bar{T}tr_n^\infty & \\
 F\{*\} & \dots & & \bar{T}^n F\{*\} & & \bar{T}^{n+1} F\{*\} & \dots
 \end{array} \quad (6.2)$$

Remark 6.1.2. In general tr^∞ does not commute with the morphisms in $FSeq^T$, intuitively, because σ^\dagger appends to the front of traces, whereas $FT^n!_{T\{*\}}$ forgets at the end of trace prefixes. However, tr^∞ commutes laxly with the morphisms in $FSeq^T$ up to a relation, which expresses that taking a step along σ^\dagger we can not forget within the current horizon. We will introduce the relation as a span in the next section.

Before we abstract from tr^∞ , we consider an example of the generic infinite trace semantics of a (B, T) -coalgebra, which we have computed the finite trace semantics for in Example 5.2.7.

Example 6.1.3. Consider the following example of a (B, T) -coalgebra $\mathbb{S} = \langle S, \sigma \rangle$ with the branching type $B = (\mathbb{N}^{(-)})_\omega$ and the transition type $T(-) = \{\sqrt{}\} + Act \cdot (-)$.

$$\begin{array}{c}
 \begin{array}{ccc}
 & 1 \cdot a & \\
 & \curvearrowright & \\
 x & \xrightarrow{2 \cdot a} & y \\
 & \xleftarrow{3 \cdot b} & \\
 & 1 \cdot \sqrt{} & \\
 & \rightarrow &
 \end{array}
 \end{array} \quad (6.3)$$

We compute the infinite traces of x and y for a successively increasing depth as follows.

1. $tr_0^\infty(x) = *$ and $tr_0^\infty(y) = *$
2. $tr_1^\infty(x) = 1 \cdot a(tr_0^\infty(x)) + 2 \cdot a(tr_0^\infty(y)) = 1 \cdot a(*) + 2 \cdot a(*) = 3 \cdot a(*)$ and $tr_1^\infty(y) = 1 \cdot \sqrt{} + 3 \cdot b(tr_0^\infty(x)) = 1 \cdot \sqrt{} + 3 \cdot b(*)$.
3. $tr_2^\infty(x) = 1 \cdot a(tr_1^\infty(x)) + 2 \cdot a(tr_1^\infty(y)) = 1 \cdot a(3 \cdot a(*)) + 2 \cdot a(1 \cdot \sqrt{} + 3 \cdot b(*)) = (1 * 3) \cdot aa(*) + (2 * 1)a\sqrt{} + (2 * 3) \cdot ab(*) = 3 \cdot aa(*) + 2 \cdot a\sqrt{} + 6 \cdot ab(*)$.

$tr_2^\infty(y)$ and tr_n^∞ for $n > 2$ can be computed analogously.

6.1.1 Change of Perspective

As infinite trace semantics captures forgetting at the end, we will have to shift perspective in our definition of generic infinite trace semantics to identify it as a special case of infinite trace semantics as in Theorem 6.2.5. This shift in perspective consists in unravelling Definition 6.1.1 as follows.

$$\begin{array}{ccccc}
 S & \xrightarrow{\sigma} & \dots & \longrightarrow & (BT)^n S & \xrightarrow{(BT)^n \sigma} & (BT)^n BT S & \longrightarrow & \dots \\
 \downarrow (tr_0^\infty)^\dagger & & & & \downarrow (BT)^n (tr_0^\infty)^\dagger & & \downarrow (BT)^n BT (tr_0^\infty)^\dagger & & \\
 B\{*\} & & & & (BT)^n B\{*\} & & (BT)^n BT B\{*\} & & \\
 \downarrow \mu_{\{*\}} \circ B\pi_{\{*\}}^0 \circ \tau_{B\{*\}}^0 & & & & \downarrow \mu_{T^n\{*\}} \circ B\pi_{\{*\}}^n \circ \tau_{B\{*\}}^n & & \downarrow \mu_{T^{n+1}\{*\}} \circ B\pi_{\{*\}}^{n+1} \circ \tau_{B\{*\}}^{n+1} & & \\
 B\{*\} & & & & BT^n\{*\} & & BT^{n+1}\{*\} & &
 \end{array} \tag{6.4}$$

Lemma 6.1.4. *For all $n < \omega$, $(tr_n^\infty)^\dagger = \mu_{T^n\{*\}} \circ B\pi_{\{*\}}^n \circ \tau_{B\{*\}}^n \circ (BT)^n (tr_0^\infty)^\dagger \circ \sigma^n$.*

Proof. By definition of tr^∞ , $(\pi^n)_{n < \omega}$, $(\tau^n)_{n < \omega}$, and $(\sigma^n)_{n < \omega}$. □

6.1.2 Generic Infinite Trace Equivalence

Generic infinite trace semantics induces the following equivalence, identifying states with the same infinite traces.

Definition 6.1.5 (Generic Infinite Trace Equivalence). *Let $\mathbb{S} = \langle S, \sigma \rangle$ be a (B, T) -coalgebra and $tr^\infty : FS \Rightarrow |FSeq^T|$ the infinite trace semantics of \mathbb{S} . We then define the generic infinite trace equivalence \sim_{tr^∞} in \mathbb{S} as $\sim_{tr^\infty} := \ker tr^\infty$.*

In order to show that generic infinite trace equivalence contains T -bisimilarity as the largest T -bisimulation as in Definition C.0.23, we need to add the following assumption.

Assumption 6.1.6. *We assume that T preserves weak pullbacks.*

Proposition 6.1.7. *BT -bisimilar states are generic infinite trace equivalent.*

Proof. Let $\mathbb{S} = \langle S, \sigma \rangle$ be a (B, T) -coalgebra and let $tr^\infty : FS \Rightarrow |FSeq^T|$ be the generic infinite trace semantics of \mathbb{S} . We show by induction of the trace depth that the cone $f : S \Rightarrow Seq^{BT}$ of Definition 1.5.1 factors tr^∞ , such that $tr_n^{\infty \dagger} = \tau_{\{*\}}^n \circ f_n$ for all $n < \omega$ as follows.

$$\begin{array}{c}
 \begin{array}{ccccc}
 & & S & \xrightarrow{\sigma} & BTS \\
 & \swarrow^{(tr_0^\infty)^\dagger} & & \searrow^{(tr_n^\infty)^\dagger} & \\
 & B\{*\} & & BT^n\{*\} & \\
 \tau_{\{*\}}^0 \uparrow & \swarrow f_0 & & \searrow f_n & \\
 \{*\} & & & & BT^n\{*\}
 \end{array} \\
 \begin{array}{c}
 \{*\} \xleftarrow{\dots} (BT)^n\{*\} \xleftarrow{BT^n!_{T\{*\}}} (BT)^{n+1}\{*\} \\
 \tau_{\{*\}}^{n+1} \uparrow \\
 BT^{n+1}\{*\}
 \end{array}
 \end{array}
 \quad (6.5)$$

In the induction base, $n = 0$, we obtain $(tr_0^\infty)^\dagger = \tau_{\{*\}}^0 \circ f_0$ by definition of tr^∞ . The induction step consists in the following chain of implications.

$$\begin{aligned}
 (tr_n^\infty)^\dagger &= \tau_{\{*\}}^n \circ f_n && \implies \\
 \mu_{T^{n+1}\{*\}} \circ B\pi_{T^n\{*\}} \circ BT(tr_n^\infty)^\dagger \circ \gamma &= \mu_{T^{n+1}\{*\}} \circ B\pi_{T^n\{*\}} \circ BT(\tau_{\{*\}}^n \circ f_n) \circ \gamma && \iff \\
 (tr_{n+1}^\infty)^\dagger &= \tau_{\{*\}}^{n+1} \circ f_{n+1}
 \end{aligned}$$

where the second step follows from the definition of tr^∞ , τ , and f . □

6.2 Plausible Continuations in Infinite Traces

In Remark 6.1.2 we argued that infinite trace semantics does not commute with $FSeq^T$, that is $tr_n^\infty = \overline{TF}!_{T\{*\}} \circ tr_{n+1}^\infty$ does not hold in general for all $n < \omega$. In this section we define for each n a relation \leq_n on $Nat(S, FSeq^T)$ such that $tr_n^\infty \leq_n \overline{TF}!_{T\{*\}} \circ tr_{n+1}^\infty$ induced by a span on $BT^n\{*\}$ defined as follows.

Definition 6.2.1. For each $n < \omega$, we define the following span.

$$\begin{array}{ccccc}
 BT^n BT\{*\} & \xrightarrow{B\pi_{T\{*\}}^n} & BBT^n T\{*\} & \xrightarrow{\mu_{T^n T\{*\}}} & BT^n T\{*\} \\
 \downarrow BT^n !_{BT\{*\}} & & & & \downarrow BT^n !_{T\{*\}} \\
 BT^n \{*\} & & \leq_n & & BT^n \{*\}
 \end{array} \quad (6.6)$$

Example 6.2.2. In $tr_2^\infty(x)$ of Example 6.1.3 we find the following example of a span for $n = 1$.

$$\begin{array}{ccccc}
 2 \cdot a(3 \cdot b(*)) & \xrightarrow{B\pi_{T\{*\}}^1} & (2 * 3) \cdot ab(*) & \xrightarrow{\mu_{T^2\{*\}}} & 6ab* \\
 \downarrow BT^1 !_{BT\{*\}} & & & & \downarrow BT^1 !_{T\{*\}} \\
 2a* & & \leq_1 & & 6a*
 \end{array} \quad (6.7)$$

We show that $(\leq_n)_{n < \omega}$ defines a categorical enrichment of $BSeq^T[\omega]$, that is the image¹ of $BSeq^T$, in preorders². The enrichment of $BSeq^T$ induces an enrichment of $Nat(S, FSeq^T)$ by composition with the morphisms in $BSeq^T$.

Proposition 6.2.3. For each $n < \omega$, \leq_n is a preorder and thus a category.

1. Reflexivity If B and T preserve epis, for all $\phi \in BT^n \{*\}$, $\phi \leq_n \phi$.
2. Transitivity For each $n < \omega$ and $\phi, \psi, \xi \in BT^n \{*\}$, whenever $\phi \leq_n \psi$ and $\psi \leq_n \xi$, then $\phi \leq_n \xi$.

Moreover, morphism composition commutes with $(\leq_n)_{n < \omega}$, proving $(\leq_n)_{n < \omega}$ a categorical enrichment of $BSeq^T$.

3. Monotonicity $(\leq_n)_{n < \omega}$ is left and right monotone with respect to morphism composition

Proof. **1.** For each $\phi \in BT^n \{*\}$, we need a $\psi \in BT^n BT\{*\}$ such that $\phi = BT^n !_{T\{*\}} \circ \mu_{T^n T\{*\}} \circ B\pi_{T\{*\}}^n(\psi)$ and $\phi = BT^n !_{BT\{*\}}(\psi)$. Since $!_{T\{*\}}$ is surjective and, by the assumption, also its lifting $BT^1 !_{T\{*\}}$, there is a $\phi' \in BT^n T\{*\}$ such that $\phi = BT^n !_{T\{*\}}(\phi')$. Put $\psi := BT^n \eta_{T\{*\}}(\phi')$. Since $!_{T\{*\}} = !_{BT\{*\}} \circ \eta_{T\{*\}}$ commutes due to the finality of $\{*\}$ in Set , $BT^n !_{BT\{*\}}(\psi) = \phi$. Moreover $\phi' = \mu_{T^n T\{*\}} \circ B\pi_{T\{*\}}^n$. Hence ψ witnesses $\phi \leq_n \phi$.

¹See Definition B.1.12.

²Preorders are categories as in Example B.1.2.

2. Transitivity of \leq_n can be shown by multiplying the second B of $BT^n BT\{*\}$ in $\phi \leq_n \psi$ and the second B of $BT^n BT\{*\}$ in $\psi \leq_n \xi$. The following diagram clarifies what we mean.

$$\begin{array}{ccccc}
 BT^n BBT\{*\} & \xrightarrow{B\pi_{BT\{*\}}^n} & BBT^n BT\{*\} & & \\
 \downarrow BT^n \mu_{T\{*\}} & & \searrow \mu_{T^n BT\{*\}} & & \\
 BT^n BT\{*\} & \xrightarrow{\mu_{T^n T\{*\}} \circ B\pi_{T\{*\}}^n} & BT^n T\{*\} & & BT^n BT\{*\} \xrightarrow{\mu_{T^n T\{*\}} \circ B\pi_{T\{*\}}^n} BT^n T\{*\} \\
 \downarrow BT^n !_{BT\{*\}} & & \downarrow BT^n !_{T\{*\}} & & \downarrow BT^n !_{BT\{*\}} & & \downarrow BT^n !_{T\{*\}} \\
 \varphi \in BT^n\{*\} & & \psi \in BT^n\{*\} & & \psi \in BT^n\{*\} & & \xi \in BT^n\{*\}
 \end{array} \tag{6.8}$$

The upper row composes to $(\mu_{T^n T\{*\}} \circ B\pi_{T\{*\}}^n) \circ BT^n \mu_{T\{*\}}$, which is equivalent to the commutativity of the following diagram.

$$\begin{array}{ccccccc}
 BT^n BBT\{*\} & \xrightarrow{B\pi_{BT\{*\}}^n} & BBT^n BT\{*\} & \xrightarrow{\mu_{T^n BT\{*\}}} & BT^n BT\{*\} & \xrightarrow{B\pi_{T\{*\}}^n} & BBT^n T\{*\} \\
 \downarrow BT^n \mu_{T\{*\}} & & & & & & \downarrow \mu_{T^n T\{*\}} \\
 BT^n BT\{*\} & \xrightarrow{B\pi_{T\{*\}}^n} & BBT^n T\{*\} & \xrightarrow{\mu_{T^n T\{*\}}} & BT^n T\{*\} & &
 \end{array} \tag{6.9}$$

Proving the commutativity of the diagram boils down to a combinatorial argument on the definition of π^n , using the naturality of π and μ .

3. Composing with $BT^n !_{T\{*\}}$ preserves \leq , that is for all $n < \omega$ and $\phi, \psi \in BT^{n+1}\{*\}$, if $\phi \leq_{n+1} \psi$, then $BT^n !_{T\{*\}}(\phi) \leq_n BT^n !_{T\{*\}}(\psi)$. The following diagram depicts the monotonicity property.

$$\begin{array}{ccc}
 BT^n TBT\{*\} & \xrightarrow{B\pi_{T\{*\}}^{n+1}} & BBT^n T\{*\} \xrightarrow{\mu_{T^n T\{*\}}} BT^n TT\{*\} \\
 \downarrow BT^n !_{BT\{*\}} & & \downarrow BT^n !_{T\{*\}} \\
 BT^n T\{*\} & \leq_{n+1} & BT^n T\{*\} \\
 \downarrow BT^n !_{T\{*\}} & & \downarrow BT^n !_{T\{*\}} \\
 BT^n\{*\} & \leq_n & BT^n\{*\}
 \end{array} \tag{6.10}$$

Intuitively, we obtain the witness for \leq_n by taking the branching of the second B in $BT^n BT\{*\}$ on step to the front. We then obtain the span as depicted in the following diagram.

$$\begin{array}{ccccc}
& & B\pi_{T\{*\}}^{n+1} & & \\
& \nearrow & & \searrow & \\
BT^n T BT\{*\} & \xrightarrow{BT^n \pi_{T\{*\}}} & BT^n BT T\{*\} & \xrightarrow{\pi^n} & BBT^n T T\{*\} & \xrightarrow{\mu_{T^n T\{*\}}} & BT^n T T\{*\} \\
\downarrow BT^n T!_{BT\{*\}} & & \downarrow BT^n BT!_{T\{*\}} & & \downarrow BBT^n T!_{T\{*\}} & & \downarrow BT^n T!_{T\{*\}} \\
BT^n T\{*\} & & BT^n BT\{*\} & \xrightarrow{B\pi_{T\{*\}}^n} & BBT^n T\{*\} & \xrightarrow{\mu_{T^n T\{*\}}} & BT^n T\{*\} \\
& \searrow BT^n!_{T\{*\}} & \downarrow BT^n!_{BT\{*\}} & & & & \downarrow BT^n!_{T\{*\}} \\
& & BT^n\{*\} & \leq_n & & & BT^n\{*\}
\end{array} \tag{6.11}$$

The pentagram on the left commutes because $\{*\}$ is the final object in Set , the triangle on top by definition of π^{n+1} , and both squares by naturality of π^n and μ . \square

Definition 6.2.4. A cone $tr : FS \Rightarrow FSeq^T$ is an infinite trace semantics if

1. tr is stable under \bar{T} , that is $tr_{n+1} = \bar{T}tr_n \circ \sigma^\dagger$, and
2. tr is increasing, that is $tr_n \leq_n F!_{T\{*\}}tr_{n+1}$ for all $\phi \in FS$.

Theorem 6.2.5. Generic infinite trace semantics is an infinite trace semantics.

Proof. We verify the properties of infinite trace semantics for tr^∞ .

1. That tr^∞ is stable under \bar{T} is part of the definition of tr^∞ .
2. tr^∞ is increasing with respect to \leq .

$$\begin{array}{ccc}
(BT)^n S & \xrightarrow{(BT)^n \sigma} & (BT)^n B T S \\
\tau_S^n \downarrow & & \downarrow \tau_{B T S}^n \\
BT^n S & \xrightarrow{BT^n \sigma} & BT^n B T S \\
BT^n (tr_0^\infty)^\dagger \downarrow & & \downarrow BT^n B T (tr_0^\infty)^\dagger \\
BT^n B\{*\} & \xleftarrow{BT^n B!_{BT\{*\}}} & BT^n B B T\{*\} \\
\mu_{T^n\{*\}} \circ B\pi_{\{*\}}^n \downarrow & & \downarrow \mu_{T^n B T\{*\}} \circ B\pi_{B T\{*\}}^n \\
BT^n\{*\} & \xleftarrow{BT^n!_{BT\{*\}}} & BT^n B T\{*\} \\
& \wedge \downarrow & \downarrow B\pi_{T\{*\}}^n \\
& & B B T^n T\{*\} \\
& & \downarrow \mu_{T^n T\{*\}} \\
BT^n\{*\} & \xleftarrow{BT^n!_{T\{*\}}} & BT^n T\{*\}
\end{array} \tag{6.12}$$

All squares from the top commute in order

- (a) by naturality of τ ,
- (b) by finality of $\{*\}$, and
- (c) by naturality of μ and π^n .

According to Lemma 6.1.4 the left side of Diagram 6.12 defines tr_n^∞ . It remains to verify that the right hand side composes to tr_{n+1}^∞ .

$$\begin{aligned}
& \mu_{T^{n+1}\{*\}} \circ B\pi_{T\{*\}}^n \circ \mu_{T^n BT\{*\}} \circ B\pi_{BT\{*\}}^n \circ BT^n BT(tr_0^\infty)^\dagger \circ \tau_{BTS}^n = \text{naturality of } \pi \text{ and } \mu \\
& \mu_{T^{n+1}\{*\}} \circ B\pi_{\{*\}}^{n+1} \circ BT^{n+1}(tr_0^\infty)^\dagger \circ \mu_{T^{n+1}S} \circ B\pi_{TS}^n \circ \tau_{BTS}^n = \text{definition of } (\tau^n)_{n<\omega} \\
& \mu_{T^{n+1}\{*\}} \circ B\pi_{\{*\}}^{n+1} \circ BT^{n+1}(tr_0^\infty)^\dagger \circ \tau_{BTS}^{n+1}
\end{aligned}$$

□

Remark 6.2.6. *In Proposition 6.1.7 we have shown that bisimilar states are generically infinite trace equivalent. This result does not generalise to infinite trace semantics. As a technical reason we see that at depth 0 we may assign to bisimilar states different elements of $B\{*\}$.*

6.3 Infinite Trace Semantics of Coalgebra Automata

In this section we revisit coalgebra automata [Ven04] from a purely coalgebraic perspective. We argue that nondeterministic T -coalgebra automata $\mathbb{A} = \langle Q, \theta, q_I, \Omega \rangle$ are given by their transition function θ , which is a pointed (\mathcal{P}, T) -coalgebra $\langle Q, \theta, q_I \rangle$, and by their acceptance condition, which determines an infinite trace semantics of $\langle Q, \theta \rangle$. In Chapter 10 we give a more detailed account of coalgebra automata.

Recall from the introduction, Section 1.3, that coalgebra automata are structures $\mathbb{A} = \langle Q, \theta, q_I, \Omega \rangle$, where $\theta : Q \rightarrow \mathcal{P}TQ$ is the transition function of \mathbb{A} . Ω assigns to each state $q \in Q$ a rank $\Omega(q) \in \mathbb{N}$. The semantics of such a coalgebra automaton \mathbb{A} in a pointed T -coalgebra $\mathbb{S} = \langle S, \sigma, s_I \rangle$ is given in terms of a two-player parity graph game $\mathcal{G}(\mathbb{A}, \mathbb{S})$, the

acceptance game of \mathbb{A} and \mathbb{S} . The priority function Ω determines the outcome of infinite plays.

In the original definition of coalgebra automata, acceptance is defined from the initial states q_I and s_I of \mathbb{A} and \mathbb{S} as above. We may abstract from the initial states and say that \mathbb{A} accepts the state $s \in S$ of \mathbb{S} from a state $q \in Q$ if $\langle Q, \theta, q, \Omega \rangle$ accepts $\langle S, \sigma, s \rangle$. The set of such pairs (q, s) equals the winning region Win_\exists of \exists in the acceptance game $\mathcal{G}(\mathbb{A}, \mathbb{S})$. The latter gives rise to our coalgebraic definition of acceptance behaviour.

Definition 6.3.1 (Acceptance Behaviour). *Let $\mathbb{A} = \langle Q, \theta, q_I, \Omega \rangle$ be a nondeterministic T -coalgebra automaton and let $\mathbb{S} = \langle S, \sigma, s_I \rangle$ be a pointed T -coalgebra, we define $Acc_\mathbb{S}^\Omega := Q \times S \cap Win_\exists(\mathcal{G}(\mathbb{A}, \mathbb{S}))$.*

In this section we show that every T -coalgebra and every priority function Ω determines an infinite trace semantics of the (\mathcal{P}, T) -coalgebra $\langle Q, \theta, q_I \rangle$ underlying \mathbb{A} .

Lemma 6.3.2. *For all T -coalgebra automata $\mathbb{A} = \langle Q, \theta, q_I, \Omega \rangle$ and T -coalgebras $\mathbb{S} = \langle S, \sigma, s_I \rangle$, $Acc_\mathbb{S}^\Omega$ is a coalgebra morphism from $\langle Q, \theta \rangle \rightarrow \langle S, \sigma \rangle$, that is the following commutes.*

$$\begin{array}{ccc}
 Q & \xrightarrow{\theta} & PTQ \\
 \downarrow Acc_\mathbb{S}^\Omega & & \downarrow PT Acc_\mathbb{S}^\Omega \\
 \mathcal{P}S & \xrightarrow{\mathcal{P}\sigma} \mathcal{P}TS \xleftarrow{\mu_{TS}} \mathcal{P}\mathcal{P}TS \xleftarrow{\mathcal{P}\pi_S} \mathcal{P}T\mathcal{P}S &
 \end{array} \quad (6.13)$$

Proof. A coalgebra $\mathbb{S} = \langle S, \sigma, s \rangle$ is accepted by $\mathbb{A} = \langle Q, \theta, q, \Omega \rangle$, that is $s \in Acc_\mathbb{S}^\Omega(q)$, if and only if \exists has a winning in $\mathcal{G}(\mathbb{S}, \mathbb{A})$ from (q, s) . She can choose an element $a \in \theta(q)$ and a set $Z \subseteq Q \times S$ with $(a, \sigma(s)) \in Rel_T(Z)$, such that every element $(q', s') \in Z$ is a winning position for \exists , $s' \in Acc_\mathbb{S}^\Omega(q')$. \square

Theorem 6.3.3. *The acceptance behaviour of a T -coalgebra automaton defines an infinite trace semantics.*

We prove the following more general Theorem subsuming Theorems 6.4.3 and 6.3.3.

Theorem 6.3.4. *Let $\mathbb{S} = \langle S, \sigma \rangle$ be a (B, T) -coalgebra and let $\mathbb{S}' = \langle S', \sigma' \rangle$ be a T -coalgebra, such that $h : \sigma^\dagger \rightarrow (\eta_{TS'} \circ \sigma')^\dagger$ is a \bar{T} -coalgebra morphism in $Kl(T)$, then $h^\dagger : S \rightarrow BS'$ defines an infinite trace semantics tr .*

Proof. We argue that tr defined by its adjoint transpose $(tr)^\dagger$ by

$$(tr_0)^\dagger := B!_{S'} \circ h^\dagger \text{ and } (tr_{n+1})^\dagger := BT^{n+1}!_{S'} \circ \mu_{T^{n+1}S'} \circ B\pi_{T^n} \circ BT tr_n \circ \sigma \quad (6.14)$$

is an infinite trace semantics for θ . The above definition is inductive in the sense, that it appends to the front. Since infinite trace semantics is defined by forgetting at the end, we need to change perspective, and show that the above satisfies

$$(tr_n)^\dagger = BT^n!_{S'} \circ \mu_{T^n S'} \circ B\pi_{S'}^n \circ \tau_{BS'}^n \circ (BT)^n h^\dagger \circ \sigma^n \quad (6.15)$$

for all $n < \omega$.

1. That tr is stable under \overline{T} , so that $(tr_{n+1})^\dagger = \mu_{T^{n+1}\emptyset} \circ B\pi_{T^n\emptyset} \circ BT(tr_n)^\dagger \circ \sigma$ commutes, follows from the naturality of μ and π
2. The increasing sequence property follows from the commutativity of the four rectangles on the right in the diagram below for all $n < \omega$.

$$\begin{array}{ccccccc}
 S & \xrightarrow{\sigma} & \dots & \longrightarrow & (BT)^n S & \xrightarrow{(BT)^n \sigma} & (BT)^{n+1} S \\
 \downarrow h^\dagger & & & & \downarrow (BT)^n h^\dagger & & \downarrow (BT)^{n+1} h^\dagger \\
 & & & & (BT)^n BS' & & (BT)^{n+1} BS' \xlongequal{\quad} BT^n T BS' \\
 & & \mu_{T^n S'} \circ B\pi_{S'}^n \circ \tau_{BS'}^n \downarrow & & \downarrow \mu_{T^{n+1} S'} \circ B\pi_{S'}^{n+1} \circ \tau_{BS'}^{n+1} & & \downarrow BT^n \pi_{S'} \\
 BS' & \xrightarrow{B\sigma'} & \dots & \longrightarrow & BT^n S' & \xrightarrow{BT^n \sigma'} & BT^{n+1} S' \xleftarrow{\mu_{T^{n+1} S'} \circ B\pi_{TS'}^n} BT^n BT S' \\
 \downarrow B!_{S'} & & & & \downarrow BT^n !_{S'} & & \downarrow BT^n !_{S'} \\
 B\{*\} & \xleftarrow{!_{T\{*\}}} & \dots & \xleftarrow{\quad} & BT^n \{*\} & \xleftarrow{BT^n !_{T\{*\}}} & BT^{n+1} \{*\} \xleftarrow{\mu_{T^{n+1} 1} \circ B\pi_{T\{*\}}^n} BT^n BT \{*\}
 \end{array} \quad (6.16)$$

where

- the right top square commutes by definition of σ^n ,
- the right bottom square commutes by naturality of μ and π
- the left bottom square commutes by Lemma C.0.26, and

- the left top square commutes by Lemma 6.3.2 as in Diagram 6.17 below.

The following diagram commutes.

$$\begin{array}{ccccc}
 (BT)^n S & \xrightarrow{(BT)^n \sigma} & (BT)^n B T S & & (6.17) \\
 \downarrow (BT)^n h^\dagger & & \downarrow (PT)^n B T h^\dagger & & \\
 (BT)^n B S' & \xrightarrow{(BT)^n B \sigma'} (BT)^n B T S' & \xleftarrow{(BT)^n \mu_{TS'}} (BT)^n B B T S' & \xleftarrow{(BT)^n B \pi_{S'}} (BT)^n B T B S' & \\
 \downarrow \tau^n & \downarrow \tau^n & & & \downarrow \tau_{BS'}^{n+1} \\
 B T^n B S' & \xrightarrow{B T^n B \sigma'} B T^n B T S' & & & \\
 \downarrow B \pi_{S'}^n & \downarrow B \pi_{TS'}^n & & & \\
 B B T^n S' & \xrightarrow{B B T^n \sigma'} B B T^n T S' & \xleftarrow{B \pi_{S'}^{n+1}} B T^{n+1} B S' & & \\
 \downarrow \mu_{TS'}^n & \downarrow \mu_{T^{n+1} S'} & & & \\
 B T^n S' & \xrightarrow{B T^n \sigma'} B T^n T S' & & &
 \end{array}$$

- The top rectangle commutes by Lemma 6.3.2;
- the three left squares commute by naturality of σ , μ , and π , respectively; and
- the bottom right rectangle commutes by definition of τ .

□

6.4 Jacob's Infinite Trace Semantics

Jacobs [Jac04] defined infinite trace semantics for (\mathcal{P}, T) -coalgebras $\mathbb{S} = \langle S, \sigma \rangle$, where T admits a final coalgebra $\mathbb{Z} = \langle Z, \xi \rangle$. The final T -coalgebra contains all T -behaviours. Jacob's infinite trace semantics Tr of \mathbb{S} is a relation $Tr \subseteq S \times Z$, such that $Tr[\sigma(s)] \subseteq \xi[Tr[s]]$ commutes. Such a relation is not unique. In Jacob's infinite trace semantics we choose the largest such relation Tr , which is bound to exist by the Knaster-Tarski theorem.

Definition 6.4.1 (Jacob's Infinite Trace Semantics). *Let $\mathbb{S} = \langle S, \sigma \rangle$ be a (\mathcal{P}, T) -coalgebra for a transition type T admitting a final T -coalgebra $\mathbb{Z} = \langle Z, \xi \rangle$. Jacob's infinite trace semantics is a relation $Tr \subseteq S \times Z$ with the transpose $Tr^\dagger : S \rightarrow \mathcal{P}Z$, making the following diagram commute.*

$$\begin{array}{ccc}
 S & \xrightarrow{\sigma} & \mathcal{P}TS \\
 \text{\scriptsize Tr^\dagger} \downarrow & & \downarrow \text{\scriptsize $\mathcal{P}TTr^\dagger$} \\
 \mathcal{P}Z & \xrightarrow{\mathcal{P}\xi} \mathcal{P}TZ \xleftarrow{\mu_{TZ}} \mathcal{P}\mathcal{P}TZ \xleftarrow{\mathcal{P}\pi_Z} \mathcal{P}T\mathcal{P}Z &
 \end{array} \tag{6.18}$$

Remark 6.4.2. *That Diagram 6.18 commutes, means that Tr is a \bar{T} -coalgebra morphism $\sigma^\dagger \rightarrow (\eta_{TZ} \circ \xi)^\dagger$.*

The following theorem is an instance of Theorem 6.3.4.

Theorem 6.4.3. *Jacob's infinite trace semantics is an infinite trace semantics in the sense of Definition 6.2.4.*

Part III

Coalgebraic Logics

Chapter 7

The Complementation Lemma for Finitary Coalgebraic Logic

In this chapter we define the Boolean dual of ∇ , the modality in Moss' coalgebraic logic, and thereby show that Moss' coalgebraic logics is essentially negation free. ∇ furthermore plays a role in the definition of coalgebra automata, so that the complementation lemma for finitary coalgebraic logics in this chapter contributes to the complementation lemma for coalgebra automata in Chapter 10.

This chapter is joint work of the author with Venema published in [KV09].

Notation 7.0.4. *In this chapter we adopt the notational convention of [KKV08]. \mathcal{L} denotes the set of formulas in the finitary coalgebraic logic under discourse.*

$a, b, \dots \in \mathcal{L}$	$\alpha, \beta, \dots \in T_\omega \mathcal{L}$
$\phi, \psi, \dots \in \mathcal{P}_\omega \mathcal{L}$	$\Phi, \Psi, \dots \in T_\omega \mathcal{P}_\omega \mathcal{L}$
$A, B, \dots \in \mathcal{P}_\omega T_\omega$	

Table 7.1: Notational Convention for Coalgebraic Logic

Assumption 7.0.5. *Towards finitariness and completeness of coalgebraic logics, and towards stability of its semantics we make the following assumptions.*

1. $T : \text{Set} \rightarrow \text{Set}$ restricts to finite sets,
2. is standard¹, and

¹See Section B.4.1 for detailed treatment of standardness for weak-pullback preserving functors in *Set*.

3. *preserves weak pullbacks.*

7.1 Preliminaries

Firstly we recall Moss' coalgebraic logics from [Mos99]. Coalgebraic logics are parameterised in a functor T .

Definition 7.1.1 (Syntax of Coalgebraic Logic). *Formulas a of a coalgebraic logic \mathcal{L} are defined by the following grammar.*

$$a ::= \top \mid \perp \mid \neg a \mid a \wedge a \mid a \vee a \mid \nabla a \quad (7.1)$$

Definition 7.1.2 (Semantics of Coalgebraic Logic). *The semantics of a formula a of coalgebraic logic in a T -coalgebra $\mathbb{S} = \langle S, \sigma \rangle$ is given in terms of \models , for which we read $\mathbb{S}, s \models a$ as point s in \mathbb{S} satisfies a . \models is inductively defined as follows.*

1. $\mathbb{S}, s \models \top$
2. $\mathbb{S}, s \not\models \perp$
3. $\mathbb{S}, s \models a \wedge b$ if and only if $\mathbb{S}, s \models a$ and $\mathbb{S}, s \models b$
4. $\mathbb{S}, s \models a \vee b$ if and only if $\mathbb{S}, s \models a$ or $\mathbb{S}, s \models b$
5. $\mathbb{S}, s \models \nabla \alpha$ if and only if $(\sigma(s), \alpha) \in \text{Rel}_T(\models)$

When the modelling coalgebra is clear from the context, we omit of \mathbb{S} .

Notation 7.1.3. *We furthermore use the following abbreviations for T -coalgebras $\mathbb{S} = \langle S, \sigma \rangle$ and formulas a and b of \mathcal{L} .*

1. $\llbracket a \rrbracket_{\mathbb{S}} := \{s \in S \mid \mathbb{S}, s \models a\}$
2. $a \leq b$ if and only if $\llbracket a \rrbracket_{\mathbb{S}} \subseteq \llbracket b \rrbracket_{\mathbb{S}}$

$$\begin{array}{c}
\frac{\{b_1 \leq b_2 \mid (b_1, b_2) \in Z\}}{\{\nabla\alpha \leq \nabla\beta \mid (\alpha, \beta) \in Rel_T(Z)\}} \nabla 1 \\
\\
\frac{\{\nabla(T \wedge)\Phi \leq a \mid \Phi \in SRD(A)\}}{\wedge\{\nabla\alpha \mid \alpha \in A\} \leq a} \nabla 2 \\
\\
\frac{\nabla\alpha \leq a \mid (\alpha, \Phi) \in Rel_T(\epsilon)}{\nabla(T \vee)\Phi \leq a} \nabla 3 \\
\\
\frac{\{a \wedge \nabla\alpha' \leq \perp \mid \alpha' \in T_\omega(\phi), \alpha' \neq \alpha\} \top \leq \vee \phi}{a \leq \nabla\alpha} \nabla 4
\end{array}$$

Figure 7.1: Axiom System M for Finitary Coalgebraic Logics

7.2 A Review of the Completeness of Finitary Coalgebraic Logic

When the models of a logic \mathcal{L} admit complementation, negation is definable in \mathcal{L} if the axiomatisation of \mathcal{L} is complete. In [KKV08] Kupke, Kurz, and Venema have shown an axiomatisation of finitary coalgebraic logic, and have proved it complete. We review parts of this result in this section, as it is relevant for the dualisation of ∇ and the complementation of coalgebra automata in Chapter 10.

Theorem 7.2.1. *The axioms $\nabla(1-4)$ ² as in Figure 7.2 soundly and completely axiomatise finitary coalgebraic logics, that is*

$$\vdash_M a \leq b \iff a \models b \quad (7.2)$$

for all formulas $a, b \in \mathcal{L}$.

A stronger property than $\nabla 3$ tells us that disjunctions distribute over ∇ as in the following lemma. This result will play a role in Section 10.3.2.

Lemma 7.2.2.

$$\nabla(T \vee)\Phi \equiv \bigvee \{\nabla\alpha \mid (\alpha, \Phi) \in Rel_T(\epsilon)\}. \quad (7.3)$$

for all $\Phi \in T\mathcal{P}_\omega Q$ and $\alpha \in TQ$.

²See Definition B.4.17 for SRD .

Proof. We prove the lemma in two steps.

1. \leq is an instance of $\nabla 3$ for $a = \bigvee \{ \nabla \alpha \mid (\alpha, \Phi) \in \text{Rel}_T(\epsilon) \}$. $\nabla \alpha \leq \bigvee \{ \nabla \alpha \mid (\alpha, \Phi) \in \text{Rel}_T(\epsilon) \}$ for any $(\alpha, \Phi) \in \text{Rel}_T(\epsilon)$ is a fact of propositional logic.
2. It is a fact of propositional logic that $\epsilon; \text{Gr}(\bigvee) \subseteq \leq$. Because we assume T to be standard, $\text{Rel}_T(\epsilon; \text{Gr}(\bigvee)) \subseteq \text{Rel}_T(\leq)$ by 3. of Lemma B.4.12. Since $(\alpha, \Phi) \in \text{Rel}_T(\epsilon)$, $(\alpha, (T\bigvee)\Phi) \in \text{Rel}_T(\epsilon; \text{Gr}(\bigvee)) \subseteq \text{Rel}_T(\leq)$.

□

7.3 One-Step Semantics of Coalgebraic Logic

Many constructions in coalgebraic logic can be carried out in a one-step manner. We introduce one-step syntax and semantics for coalgebraic logic. Therefore we need to parameterise coalgebraic logic in a finite set Q of propositional variables.

Definition 7.3.1. *Given a Set-functor T , we define the functor T_∇ .*

$$T_\nabla Q := \{ \nabla \alpha \mid \alpha \in TQ \} \quad (7.4)$$

Definition 7.3.2 (Depth One Formulas). *Given a finite set Q of propositional variables we define the set $\text{Lat}_1 Q$ of depth one formulas over Q as $\text{Lat}_1 Q := \text{Lat} T_\nabla \text{Lat} Q$.*

We can define the one-step semantics of coalgebraic logics relative to a valuation of Q in the modelling coalgebra.

Definition 7.3.3. *Let $V : Q \rightarrow \mathcal{P}(X)$ be a valuation of Q in a T -coalgebra $\langle S, \sigma \rangle$, we define $\Vdash_0^V \subseteq S \times \text{Lat} Q$ and $\Vdash_1^V \subseteq TS \times \text{Lat}_1 Q$, as follows. For \Vdash_0^V , we define $s \Vdash_0^V q$ if $s \in V(q)$, $s \Vdash_0^V \bigwedge \tau$ ($\bigvee \tau$, respectively) if $s \Vdash_0^V a$ for all $a \in \tau$ (some $a \in \tau$, respectively); and we define a relation \Vdash_1^V such that $\sigma \Vdash_1^V \nabla \alpha$ if $(\sigma, \alpha) \in \text{Rel}_T(\Vdash_0^V)$, while for \bigwedge and \bigvee the same clauses apply as for \Vdash_0^V .*

Notation 7.3.4. We abbreviate

$$\llbracket a \rrbracket^V := \{s \in S \mid s \Vdash_1^V a\} \quad (7.5)$$

and for clarity of notation we write $V, s \Vdash_0 a$ instead of $s \Vdash_0^V a$, and $V, s \Vdash_1 a$ instead of $s \Vdash_1^V a$.

7.4 Complementation Lemma

Finally, we can define the Boolean dual Δ of ∇ . By Boolean dual we mean the following.

Definition 7.4.1 (Boolean Duals of Depth One Formulas). *Let a and b be depth one formulas over a finite set Q , we say a is the Boolean dual of b if for any valuation $V : Q \rightarrow \mathcal{P}S$ and its complement V^c , $\llbracket a \rrbracket^V$ is the set-complement of $\llbracket b \rrbracket^{V^c}$ in S .*

The idea underlying the following definition is of Venema. Alessandra Palmigiano and an anonymous referee of [KV09] suggested a simplification, which lead to the following definition.

Definition 7.4.2 (Boolean Dual of ∇). *For each finite set Q of propositional variables, and each $\alpha \in T_\omega Q$, we define the set $D(\alpha) \subseteq T_\omega \mathcal{P}Q$ as follows.*

$$D(\alpha) := \left\{ \Phi \in T_\omega \mathcal{P}_\omega \text{Base}(\alpha) \mid (\alpha, \Phi) \notin (Rel_T(\emptyset)) \right\} \quad (7.6)$$

Define $\Delta\alpha$ to abbreviate the following formula in Lat_1 .

$$\Delta\alpha := \bigvee \left\{ \nabla (T \bigwedge) (\Phi) \mid \Phi \in D(\alpha) \right\}. \quad (7.7)$$

Here we see the connective \bigwedge as a map $\bigwedge : \mathcal{P}_\omega Q \rightarrow Q$, so that $T \bigwedge : T_\omega \mathcal{P}_\omega Q \rightarrow T_\omega Q$.

Recall the definition of the set $Lat_1 Q$ of depth one formulas over a set Q , to be the set $Lat_{T\nabla} Lat Q$. If we want to admit depth one formulas with Δ , the Boolean dual of ∇ , we need to consider the following.

Theorem 7.4.3. *For all α , $\Delta\alpha$ and $\nabla\alpha$ are Boolean duals.*

Proof. Fix an arbitrary set S , an arbitrary Q -valuation V in S , and an arbitrary element σ of TS .

First assume that $V, \sigma \Vdash_1 \Delta\alpha$, that is, $V, \sigma \Vdash_1 \nabla(T \wedge) \Phi$ for some $\Phi \in D(\alpha)$. Then there is some relation $Y \subseteq \mathcal{P}(Q \times S)$ such that $Y \subseteq (\Vdash_0^V)^\circ; Gr(\wedge)$ and $(\Phi, \sigma) \in Rel_T(Y)$. In order to show that $V^c, \sigma \not\Vdash_1 \nabla\alpha$, suppose for contradiction that there is some relation Z such that $(\sigma, \alpha) \in Rel_T(Z)$ and $V^c, t \Vdash_0 q$ for all pairs $(t, q) \in Z$. It follows that $(\sigma, \alpha) \in Rel_T(Z)$ and $Z \cap \Vdash_0^V = \emptyset$.

Now consider the relation $R := Z; Y \subseteq \mathcal{P}Q \times Q$, then clearly $(\Phi, \alpha) \in Rel_T(R) = Rel_T(Z); Rel_T(Y)$. On the other hand, it follows from the definition of R that $R \subseteq \emptyset$, because for any $(\phi, q) \in R$ there is an $s \in S$ such that (i) $(\phi, s) \in Y$ implying $V, s \Vdash_0 p$ for all $p \in \phi$, and (ii) $(s, q) \in Z$ meaning that $V, s \not\Vdash_0 q$. But this gives the desired contradiction since $\Phi \in D(\alpha)$.

Conversely, assume that $V^c, \sigma \not\Vdash_1 \nabla\alpha$. In order to show that $V, \sigma \Vdash_1 \Delta\alpha$ we need to find some $\Phi \in D(\alpha)$ such that $V, \sigma \Vdash_1 \nabla(T \wedge)(\Phi)$. For this purpose, define a map $\phi : S \rightarrow \mathcal{P}Base(\alpha)$ by putting, for any $s \in S$, $\phi_s := \{q \in Base(\alpha) \mid V, s \Vdash_0 q\}$.

We claim that $\Phi := T\phi(\sigma)$ has the required properties. First of all, it follows by construction that $Gr(\wedge \circ \phi) \subseteq \Vdash_0^V$, so that $Gr(T \wedge); Gr(T\phi) \subseteq Rel_T(\Vdash_0^V)$. From this it is immediate that $V, \sigma \Vdash_1 \nabla T \wedge(\Phi)$. It remains to show that $\Phi \in D(\alpha)$. For that purpose, consider the relation $Z := \emptyset; Gr(\phi) \subseteq S \times Q$. It is easily verified that $V^c, s \Vdash_0 q$ for all $(s, q) \in Z$. Hence, we may derive from the assumption $V^c, \sigma \not\Vdash_1 \nabla\alpha$ that $(\sigma, \alpha) \notin Rel_T(Z) = Rel_T(\emptyset); Gr(T\phi)$. But then it follows from $(\sigma, \Phi) \in Gr(T\phi)$ that $(\Phi, \alpha) \notin (Rel_T(\emptyset))$, as required. \square

Definition 7.4.4 (The Base Dualisation Map). *Given a set Q we define the base dualisation map $\delta_0 : LatQ \rightarrow LatQ$ and the one-step dualisation map $\delta_1 : Lat_1Q \rightarrow Lat_1Q$ as follows:*

$$\begin{aligned}
 \delta_0(q) &:= q & \delta_1(\nabla\alpha) &:= \Delta(T\delta_0)\alpha \\
 \delta_0(\wedge \phi) &:= \bigvee \delta_0[\phi] & \delta_1(\wedge \phi) &:= \bigvee \delta_1[\phi] \\
 \delta_0(\vee \phi) &:= \bigwedge \delta_0[\phi] & \delta_1(\vee \phi) &:= \bigwedge \delta_1[\phi]
 \end{aligned} \tag{7.8}$$

Example 7.4.5. With $T = \mathcal{P}$ and $\alpha = \{q_1 \vee q_2, q_3 \wedge q_4\}$, we may calculate that

$$\begin{aligned}\delta_1(\nabla\alpha) &= \Delta\{q_1 \wedge q_2, q_3 \vee q_4\} \\ &= \nabla\emptyset \vee \nabla\{q_1 \wedge q_2\} \vee \nabla\{q_3 \vee q_4\} \vee \nabla\{(q_1 \wedge q_2) \wedge (q_3 \vee q_4), \top\}.\end{aligned}$$

The following is a corollary of the one-step complementation lemma. For the sake of completeness of our argument we carry out the proof.

Corollary 7.4.6. For any set Q of propositional variables and any $a \in \text{Lat}_1 Q$, the formulas a and $\delta_1(a)$ are Boolean duals.

The corollary depends on the following easy lemma.

Lemma 7.4.7. For any lattice term $a \in \text{Lat}Q$, a and $\delta_0(a)$ are Boolean duals.

Proof. For any pointed T -coalgebra (S, σ, s) and valuation $V : Q \rightarrow \mathcal{P}S$ we show that (*) $V, \sigma \Vdash_0 a$ if and only if $V^c, \sigma \nVdash_0 \delta_0(a)$ by induction on a .

1. If a is an element of Q , it is $\delta_0(a) = a$. As the point s is either in $V(a)$ or $V^c(a)$ but not both, the property (*) holds.
2. Let $\phi \subseteq_\omega Q$ and suppose that (*) holds for every formula $b \in \phi$. If $a = \bigwedge \phi$, then $\delta_0(a) = \bigvee (\mathcal{P}_\omega \delta_0) \phi$. Then $V, \sigma \Vdash_0 a$ if and only if $V, \sigma \Vdash_0 b$ for all $b \in \phi$, if and only if $V^c, \sigma \nVdash_0 \delta_0(b)$ for all $b \in \phi$, if and only if $V^c, \sigma \nVdash_0 \bigvee (\mathcal{P}_\omega \delta_0) \phi$, if and only if $V^c, \sigma \nVdash_0 \delta_0(a)$.
3. Through a symmetric argument, we obtain for $a = \bigvee \phi$, that $V, \sigma \Vdash_0 a$ if and only if $V, \sigma \Vdash_0 b$ for some $b \in \phi$, if and only if $V^c, \sigma \nVdash_0 \delta_0(b)$ for some $b \in \phi$, if and only if $V^c, \sigma \nVdash_0 \bigwedge \delta_0[\phi]$, if and only if $V^c, \sigma \nVdash_0 \delta_0(a)$.

□

Proof of Corollary 7.4.6. For the following argument fix a depth one formula $a \in \text{Lat}_1 Q$. Then a and $\delta_1(a)$ are Boolean duals if for any T -coalgebra (S, σ) and valuation $V : Q \rightarrow \mathcal{P}S$, $V, \sigma \Vdash_1 a$ if and only if $V^c, \sigma \nVdash_1 \delta_1(a)$.

In Theorem 7.4.3 set $\text{Lat}Q$ for Q . Every valuation $V : Q \rightarrow \mathcal{P}S$ extends uniquely to a valuation $\bar{V} : \text{Lat}Q \rightarrow \mathcal{P}S$ such that

1. $\bar{V}(q) := V(q)$ for all $q \in Q$, and
2. $\bar{V}(a \vee b) := \bar{V}(a) \cup \bar{V}(b)$ and
3. $\bar{V}(a \wedge b) := \bar{V}(a) \cap \bar{V}(b)$ for all $a, b \in LatQ$

For \bar{V} , Lemma 7.4.7 means that the following are equivalent

$$\begin{array}{ll}
 s \in \bar{V}(a) & \Longleftrightarrow \\
 V, \sigma \Vdash_0 a & \Longleftrightarrow \\
 V^c, \sigma \nVdash_0 \delta_0(a) & \Longleftrightarrow \\
 s \notin \bar{V}^c \delta_0(a) & \Longleftrightarrow
 \end{array}$$

Then using Theorem 7.4.3, we obtain the following chain of implications.

$$\begin{array}{ll}
 \bar{V}, \sigma \Vdash_1 \nabla \alpha & \Longleftrightarrow \\
 (\bar{V})^c, \sigma \nVdash_1 \Delta \alpha & \implies \\
 V, \sigma \Vdash_1 \nabla \alpha & \Longleftrightarrow \\
 V^c, \sigma \nVdash_1 \Delta(\mathcal{T} \delta_0) \alpha &
 \end{array}$$

In the remainder of the argument, we extend the previous over the lattice operators.

$$\begin{array}{ll}
 V, \sigma \Vdash_1 \phi \vee \psi & \Longleftrightarrow \\
 V, \sigma \Vdash_1 \phi \text{ or } V, \sigma \Vdash_1 \psi & \Longleftrightarrow \\
 V^c, \sigma \nVdash_1 \delta_1 \phi \text{ or } V^c, \sigma \nVdash_1 \delta_1 \psi & \Longleftrightarrow \\
 V^c, \sigma \nVdash_1 \delta_1 \phi \wedge \delta_1 \psi &
 \end{array}$$

and

$$V, \sigma \Vdash_1 \phi \wedge \psi \quad \Longleftrightarrow$$

$$V, \sigma \Vdash_1 \phi \text{ and } V, \sigma \Vdash_1 \psi \quad \Longleftrightarrow$$

$$V^c, \sigma \not\Vdash_1 \delta_1 \phi \text{ and } V^c, \sigma \not\Vdash_1 \delta_1 \psi \quad \Longleftrightarrow$$

$$V^c, \sigma \not\Vdash_1 \delta_1 \phi \vee \delta_1 \psi$$

□

Chapter 8

Finitary Coalgebraic Logics for Finite Traces

In Chapter 5 we have given a definition of finite trace semantics, and have shown that finite trace semantics induces a notion of finite trace equivalence. In this chapter we define logics to describe states in coalgebras with branching up to finite trace equivalence in the style of Moss coalgebraic logics [Mos99].

Recall, that Moss' coalgebraic logics for T -coalgebras in Set consists of

1. a dual adjunction $2^{(-)} \dashv Uf$ between Set and $CABA$, the category of complete atomic Boolean algebras, consisting of the contravariant powerset functor $2^{(-)} : Set \rightarrow CABA$ taking a set X to the set 2^X underlying the free Boolean algebra generated from X , and of the functor $Uf : CABA \rightarrow Set$ taking Boolean algebras A to the set UfA of ultrafilters over A , and
2. a logic functor L on $CABA$ with a denotation $\delta : L2^{(-)} \Rightarrow 2^{T(-)}$ defining respectively the syntax and semantics of the operator ∇ .
3. Given the above ingredients, Moss coalgebraic logics is the initial L -algebra¹ in $CABA$ and its semantics in a T -coalgebra $\langle S, \sigma \rangle$ is the initial L -algebra map into the L -algebra $2^\sigma \circ \delta : L2^S \rightarrow 2^S$.

¹See also Proposition 5 in [KKV08].

	Moss' Coalgebraic Logic	Finite Trace Logics
Base Category, Semantics	Set	$B-Alg$
Base Category, Logics	$CABA$	$B-Alg$
Dual Adjunction	$2^{(-)} \dashv Uf$	$[-, \Omega] \dashv [-, \Omega]$
Functor, Semantics	T	\tilde{T}
Functor, Logics	L	L
Denotation	$L2^{(-)} \Rightarrow 2^{T(-)}$	$L[-, \Omega] \Rightarrow [\tilde{T}(-), \Omega]$

Table 8.1: Comparison of Moss' Coalgebraic Logics and Finite Trace Logics

We define finite trace logics analogously. However, the Eilenberg-Moore category $B-Alg$ takes the role of the base category on the semantical and the logical side, and the Kleisli-lifted functor \tilde{T} takes the role of T above.

1. The dual adjunction $[-, \Omega] \dashv [-, \Omega]$ on the Eilenberg-Moore category $B-Alg$ of the branching type B generated from an ambimorphic object Ω takes the role of the dual adjunction $2^{(-)} \dashv Uf$ in the definition of Moss' coalgebraic logics.
2. The logic functor L on $B-Alg$ comes with a denotation $\delta : L[-, \Omega] \Rightarrow [\tilde{T}(-), \Omega]$.
3. Finite trace logics is then the initial L -algebra and its semantics in a (B, T) -coalgebra $\langle S, \sigma \rangle$ is the initial L -algebra morphism into the L -algebra $[\sigma^\dagger, \Omega] \circ \delta : L[S, \Omega] \rightarrow [S, \Omega]$ where $\sigma^\dagger : FS \rightarrow \tilde{T}FS$ is the adjoint transpose of σ under $F \dashv U$.

Table 8.1 summarises the analogy between finite trace logics and Moss' coalgebraic logics.

The idea underlying the definition of finite trace logics was suggested by Bart Jacobs. The approach lead to the definition of finite trace logics for semiring monads using a Morita-type dual adjunction between categories of semimodules. Alexander Kurz suggested to abstract from categories of semimodules to symmetric monoidal closed categories, which underlie the definition of finite trace logics in this chapter.

Assumption 8.0.8. *In this chapter we assume that*

1. *the branching type B is a commutative finitary monad on Set , and*
2. *the transition type T is a finitary functor on Set , such that*

3. *finite trace semantics is definable for (B, T) -coalgebras as in Chapter 5.*

Finite trace logics are parameterised in a dual adjunction on the Eilenberg-Moore category of the branching type, a logic functor and its denotation. The logics then arise as the initial algebra for the logic functor, its semantics by the logic functors denotation. In this section we will construct these ingredients.

8.1 Dual Adjunctions from Ambimorphic Objects

Definition 8.1.1 (Ambimorphic Objects). *An object Ω of a category C is said to be ambimorphic, if for all objects X of C , $C(X, \Omega)$ is an object of C .*

In the situation of Definition 8.1.1 $X \mapsto C(X, \Omega)$ defines a contravariant functor on C , which we denote as $[-, \Omega]$. Moreover, $[-, \Omega]$ forms a dual adjunction on C .

Proposition 8.1.2. *Let C be a symmetric monoidal closed category and Ω an ambimorphic object in C , then $[-, \Omega] \dashv [-, \Omega]$.*

Proof. $C(X, [Y, \Omega]) = C(Y, [X, \Omega])$ are isomorphic natural in X and Y , since $C(X, [Y, \Omega]) = C(X \otimes Y, \Omega) = C(Y \otimes X, \Omega) = C(Y, [X, \Omega])$. \square

Remark 8.1.3. *As the adjunction $[-, \Omega] \dashv [-, \Omega]$ is dual, the unit $v : Id \Rightarrow [[-, \Omega], \Omega]$ plays the role of the counit as well.*

The above proposition is applicable to our argument if B is commutative. In this case the Eilenberg-Moore category for B over Set is symmetric monoidal closed, which is an instance of the following theorem proved by Kock [Koc70].

Theorem 8.1.4. *The category of Eilenberg-Moore algebras for a commutative monad over a symmetric monoidal closed category is symmetric monoidal closed.*

Towards expressivity of finitary trace logics, we assume Ω to be a coseparator. In Chapter VI of [Joh86] discusses the coseparator condition in greater generality and detail.

Definition 8.1.5 (Coseparators). *A coseparator in a category C is an object Ω of C , such that for any pair of morphisms $f, g : X \rightarrow Y$ in C with $f \neq g$, there is a morphism $h : Y \rightarrow \Omega$, such that $h \circ f \neq h \circ g$*

We use the following property of coseparators in the Eilenberg-Moore category $B\text{-Alg}$ of B over Set .

Lemma 8.1.6. *An ambimorphic object Ω in $B\text{-Alg}$ is a coseparator if and only if the unit $v : (-) \rightarrow [[-, \Omega], \Omega]$ of the dual adjunction $[-, \Omega] \dashv [-, \Omega]$ is monic.*

Proof. Let Ω be a coseparator. Two distinct points a, b in a B -algebra A can be separated by functions $f, g : \{*\} \rightarrow UA$. Their adjoint transposes are B -algebra homomorphisms $f^\dagger, g^\dagger : F\{*\} \rightarrow A$ with $f \neq g$ and in particular $f* \neq g*$. By the coseparator condition there is a morphism $m : A \rightarrow \Omega$ such that $m \circ f^\dagger \neq m \circ g^\dagger$ and thus $m(a) = m \circ f^\dagger(*) \neq m \circ g^\dagger(*) = m(b)$, so that v is monic.

Suppose v is monic. Let $f, g : A \rightarrow A'$ be B -algebra morphisms such that $f \neq g$. Then there is an element $a \in UA$ such that $f(a) \neq g(a)$. Because v is monic, there is a morphism $m : A' \rightarrow \Omega$ such that $m(f(a)) \neq m(g(a))$ and thus $m \circ f \neq m \circ g$. \square

The coseparator condition is not too restrictive for our purposes.

Example 8.1.7. *For the branching types $\mathcal{P}, (\mathbb{N}^{(-)})_\omega$ and $\mathcal{D}_{\leq 1}$, $F\{\top\}$ is a coseparator.*

Assumption 8.1.8. *Below we assume the following*

1. *B is a commutative monad on Set , so that $B\text{-Alg}$ is symmetric monoidal closed and all objects of $B\text{-Alg}$ are ambimorphic, and*
2. *there is a coseparator Ω in $B\text{-Alg}$.*

We leave the question open which monads B admit a coseparating ambimorphic object in $B\text{-Alg}$.

Remark 8.1.9. *In most cases $F\{\top\}$ is a suitable coseparating ambimorphic object, because*

1. *$F\{\top\}$ exists uniformly for all branching types,*
2. *is a coseparator in most cases we consider, and*
3. *can be understood in terms of the operations and equations presenting B .*

The dual adjunction defined above is a dual equivalence only if the ambimorphic object or the branching type are trivial. We leave the following remark without proof.

Remark 8.1.10. *The dual adjunction induced by an ambimorphic object as above is not a dual equivalence in many relevant cases, such as for the Eilenberg-Moore categories of the monads in Section 3.4.*

Notation 8.1.11. *Henceforth in this chapter we abbreviate $[-, \Omega]$ by Q .*

8.2 The Logic Functor

If the dual adjunction $Q \dashv Q$ is a dual equivalence, the logic functor is the dual of the functor \widetilde{T} with $LQ \cong Q\widetilde{T}$. However, since the dual adjunction in many cases is not a dual equivalence, the logic functor is weakly dual, that is up to a natural transformation $\delta : LQ \Rightarrow Q\widetilde{T}(-)$, which we call the denotation of L .

Definition 8.2.1 (Logic Functors). *Let B be a branching type, T a transition type and let $\widetilde{T} : B\text{-Alg} \rightarrow B\text{-Alg}$ be the continuous extension of T as in Section 4.3. We call a functor $L : B\text{-Alg} \rightarrow B\text{-Alg}$ a logic functor for B and T if L comes with a natural transformation $\delta : LQ \Rightarrow Q\widetilde{T}$, the denotation of L .*

For most B and T , L is not unique, and neither is the assignment of the denotation. However, we can find a canonically defined logic functor, the free one.

Example 8.2.2 (Free Logic Functors). *The functor $L := Q\widetilde{T}Q$ has a denotation $\delta : LQ \Rightarrow Q\widetilde{T}$ defined on objects X as $\delta_X := Q\widetilde{T}v_X : Q\widetilde{T}QQ \Rightarrow Q\widetilde{T}$ where $v : Id \Rightarrow QQ$ is the unit of the dual adjunction $Q \dashv Q$.*

Although, the free logic functor exists and is easily computed, it is unsuitable for our definition of trace logics, as it not finitary for almost all B and T and hence likely does not admit an initial algebra.

8.3 Finite Trace Logics as the Initial Algebra of the Logic Functor

We define finite trace logics as the initial algebra for the logic functor L . Its semantics is determined by the initial L -algebra morphism and the denotation δ which gives a semantics to the modalities.

Definition 8.3.1 (Finite Trace Logics). *Let B , T , and \tilde{T} as above, L a logic functor and $\delta : LQ \Rightarrow Q\tilde{T}$ a denotation of L . We define finite trace logics as the initial L -algebra in $B\text{-Alg}$*

$$\mathcal{L}_{B,T,L,\delta} : LI \rightarrow I \quad (8.1)$$

The semantics $\llbracket - \rrbracket_{\mathbb{S}}$ of finite trace logics is defined in the lifting $\sigma^{\dagger} : FS \rightarrow \tilde{T}FS$ of a (B, T) -coalgebra $\mathbb{S} = \langle S, \sigma \rangle$ by the initial L -algebra morphism as follows.

$$\begin{array}{ccccc} LQFS & \xrightarrow{\delta_{FS}} & Q\tilde{T}FS & \xrightarrow{Q\sigma^{\dagger}} & QFS \\ \uparrow L\llbracket - \rrbracket_{\mathbb{S}} & & & & \uparrow \llbracket - \rrbracket_{\mathbb{S}} \\ LI & \xrightarrow{\mathcal{L}_{B,T,L,\delta}} & I & & \end{array} \quad (8.2)$$

Notation 8.3.2. • For any $\phi \in I$ and $s \in S$, we denote $\phi \Vdash_{\mathbb{S}} s$ to mean $s \in \llbracket \phi \rrbracket_{\mathbb{S}}$.

- When B , T , L , and δ are clear from the context, we omit the subscripts and denote finite trace logic $\mathcal{L}_{B,T,L,\delta}$ by \mathcal{L} .

The initial L -algebra $\mathcal{L} : LI \rightarrow I$ does not exist in general, but whenever L is finitary, then I is the colimit of the ω -chain Seq_L^2 , and \mathcal{L} is the isomorphism $LI \rightarrow I$. Let $c : I \Rightarrow \text{Seq}_L$ denote the colimiting cocone of I over Seq_L . The following is the standard construction of the initial algebra of a functor and can be found in [AK79].

Lemma 8.3.3. *For finitary L , the colimit I of Seq_L exists and $\mathcal{L} : LI \rightarrow I$ is an isomor-*

²See Definition B.1.13.

phism as in the following diagram.

$$\begin{array}{c}
\begin{array}{ccc}
& I & \\
\swarrow c_0 & \xleftarrow{\mathcal{L}} LI & \searrow c_{n+1} \\
0 \longrightarrow \dots \longrightarrow L^n 0 & \xrightarrow{L^n j_{L^0}} & L^{n+1} 0 \longrightarrow \dots
\end{array}
\end{array} \quad (8.3)$$

initial object in $B\text{-Alg}$. From $d_n = Qtr_n^\omega \circ e_n$ we obtain the following chain of equations.

$$\begin{aligned}
Q\sigma^\dagger \circ \delta_{FS} \circ Ld_n &= \\
Q\sigma^\dagger \circ \delta_{FS} \circ LQtr_n^\omega \circ Le_n &= \\
Q\sigma^\dagger \circ Q\widetilde{T}tr_n^\omega \circ \delta_{\widetilde{T}n0} \circ Le_n &= \\
Q(tr^\omega)_{n+1} \circ e_{n+1} &
\end{aligned}$$

□

The semantics $\llbracket - \rrbracket_{\mathbb{S}}$ of \mathcal{L} in a modelling coalgebra $\mathbb{S} = \langle S, \sigma \rangle$ coincides with the universal morphism of the colimit $I = \text{colim}(Seq_L)$ as in the following diagram.

$$\begin{array}{ccc}
\text{colim}Seq_L & \xrightarrow{\llbracket - \rrbracket_{\mathbb{S}}} & QFS \\
\uparrow c & & \uparrow tr^\omega \\
Seq_L & \xrightarrow[e]{} & Q|Seq_{\widetilde{T}}|
\end{array} \tag{8.6}$$

8.4 Examples of Finite Trace Logics

In this section we instantiate finite trace logics for common examples of branching types, to show that the abstract categorical definitions yield concrete axiomatisations. The transition type is uniformly the labelling functor $T = \{\sqrt{\cdot}\} + Act \cdot (-)$ for a finite set Act of labels. T is special as it is made up from colimits. Such functors have a straightforward Kleisli-lifting and admit a straightforward definition of a logic functor, as Q preserves colimits dually as limits. We emphasise that the definition of finite trace logics is not limited to functors built from colimits. We restrict ourselves to T , as T is the most common example studied in classical trace theory [DR95] and yields an easy presentation.

8.4.1 An Example of a Logic Functor

For functors \widetilde{T} built from colimits we can find a logic functor easily, because Q preserves colimits as limits. In our examples, we put $\Omega := F\{\ast\}$ in $Q := [-, \Omega]$. Let T be defined as

follows.

$$T(-) := \{\sqrt{}\} + Act \cdot (-) \quad (8.7)$$

Recall that the Kleisli-lifting of T in $B\text{-Alg}$ is the functor.

$$\widetilde{T}(-) = F\{\sqrt{}\} + Act \cdot (-) \quad (8.8)$$

For \widetilde{T} we obtain the logic functor

$$L(-) := QF\{\sqrt{}\} \times (-)^{Act} \quad (8.9)$$

with the denotation $\delta : LQ \Rightarrow Q\widetilde{T}$ generated from

$$m \in QF\{\sqrt{}\} \mapsto m \text{ and } (a, \phi) \in Act \cdot A \mapsto \{a \mapsto \phi\} \in A^{Act} \quad (8.10)$$

where $\{a \mapsto \phi\}$ our short-hand notation for a function, which is zero everywhere but ϕ at a . Below we will see, what zero means in concrete instantiations of B .

Remark 8.4.1. *As defined in (8.9), L is finitary.*

8.4.2 Deterministic Streams

The branching type of deterministic streams is the identity monad Id . Recall that the Eilenberg-Moore algebras of Id are sets.

Definition 8.4.2 (Finite Trace Logics for Deterministic Streams). *Formulas of finite trace logics for deterministic streams are of the following form.*

$$\phi := \phi \wedge \phi \mid \phi \vee \phi \mid \top \mid \perp \mid \sqrt{} \mid \langle a \rangle \phi \quad (8.11)$$

Finite trace logics for deterministic streams satisfy the following axioms.

1. *Unit-Element:* $\top \wedge \phi = \phi$ and $\perp \vee \phi = \phi$
2. *Absorption:* $\top \vee \phi = \top$ and $\perp \wedge \phi = \perp$

3. *Idempotency*: $\phi \vee \phi = \phi$ and $\phi \wedge \phi = \phi$
4. *Associativity*: $\phi \vee (\phi' \vee \phi'') = (\phi \vee \phi') \vee \phi''$ and $\phi \wedge (\phi' \wedge \phi'') = (\phi \wedge \phi') \wedge \phi''$
5. *Commutativity*: $\phi \vee \phi' = \phi' \vee \phi$ and $\phi \wedge \phi' = \phi' \wedge \phi$
6. *Distributivity*: $\phi \vee (\phi' \wedge \phi'') = (\phi \vee \phi') \wedge (\phi \vee \phi'')$ and $\phi \wedge (\phi' \vee \phi'') = (\phi \wedge \phi') \vee (\phi \wedge \phi'')$

for any formulas ϕ, ϕ', ϕ'' .

8.4.3 Finitarily Nondeterministic Streams

The branching type for finitary nondeterminism is the finitary powerset monad \mathcal{P}_ω . Recall that the Eilenberg-Moore algebras of \mathcal{P}_ω are (join) semi-lattices.

Definition 8.4.3 (Trace Logics for Nondeterministic Streams). *Formulas ϕ of finite trace logics are of the form*

$$\phi ::= \top \mid \phi \vee \phi \mid \top \mid \langle a \rangle \phi \quad (8.12)$$

Trace logics for nondeterminism satisfies the following axioms of semi-lattices

1. *Bottom-Element* \perp : $x \vee \perp = \perp$
2. *Idempotency*: $x \vee x = x$
3. *Commutativity*: $x \vee y = y \vee x$
4. *Associativity*: $x \vee (y \vee z) = (x \vee y) \vee z$

and the following axiom of compatibility with the transition structure

5. $\langle a \rangle (\phi \vee \phi') = \langle a \rangle \phi \vee \langle a \rangle \phi'$

8.4.4 Streams with Finitary Graded Branching

The bag monad $(\mathbb{N}^-)_\omega$ is the type of finitary graded branching. Its Eilenberg-Moore algebras are semimodules in the semiring $\langle \mathbb{N}, +, *, 0, 1 \rangle$ of natural numbers.

Definition 8.4.4 (Finite Trace Logics for Streams with Graded Branching). *Formulas of finite trace logics for streams with graded branching are of the form*

$$\phi ::= 0 \mid \phi \oplus \phi \mid n \cdot \phi \mid \sqrt{} \mid \langle a \rangle \phi \quad (8.13)$$

where $n \in \mathbb{N}$ and $a \in \text{Act}$.

Finite trace logics for streams with graded branching satisfy the following axioms of the \mathbb{N} -semiring $\langle \mathbb{N}, +, *, 0, 1 \rangle$ for any formulas ϕ, ϕ', ϕ''

1. *Unit-Element*: $0 \cdot \phi = 0$ and $0 \oplus n \cdot \phi = n \cdot \phi$
2. *Commutativity*: $n \cdot \phi \oplus m \cdot \phi' = m \cdot \phi' \oplus n \cdot \phi$
3. *Associativity*: $n \cdot \phi \oplus (m \cdot \phi' \oplus o \cdot z) = (n \cdot \phi \oplus m \cdot \phi') \oplus o \cdot z$
4. *Absorption*: $n \cdot \phi \oplus m \cdot \phi = (n + m) \cdot \phi$

where $n, m \in \mathbb{N}$, and the distributivity axiom, which stems from $\text{Act} \cdot I$ (in $LI = F\{\sqrt{}\} + \text{Act} \cdot I$ being a semiring modules).

5. *Aggregation*: $\langle a \rangle n \cdot \phi = n \cdot \langle a \rangle \phi$
6. *Branching*: $\langle a \rangle (\phi \oplus \phi') = \langle a \rangle \phi \oplus \langle a \rangle \phi'$

We obtain the language of trace logics for $(\mathbb{N}^-)_\omega$ and T by iteration along the initial L -sequence.

- $UF\emptyset = \{0\}$ contains the empty linear combination 0
- $ULF\emptyset = F\{\sqrt{}\} = \{n \cdot \sqrt{} \mid n \in \mathbb{N}\}$ contains the formulas specifying with which grade a successful termination can be reached
- $UL^2K\emptyset = U(F\{\sqrt{}\} + \text{Act} \cdot (F\{\sqrt{}\})) = \{n \cdot \sqrt{}, \langle a \rangle \cdot (n \cdot \sqrt{}) \mid n \in \mathbb{N}, a \in \text{Act}\}$ contains the formulas specifying the grade of successful termination and for each label $a \in \text{Act}$, the grade of successful termination after passing through a
- and so forth

8.4.5 Finitarily Probabilistic Streams

The branching type for finitary probabilism is the finitary sub-distribution monad $(\mathcal{D}_{\leq 1})_{\omega}$.

Its algebras are finitary convex cones.

Definition 8.4.5 (Finite Trace Logics for Streams with Finitary Probabilistic Branching).

Formulas of trace logics for probabilistic labelled transition systems are then

$$\phi ::= \sqrt{} \mid \langle a \rangle \phi \mid \bigoplus p \cdot \phi \quad (8.14)$$

where $p \in [0, 1]$ and $a \in \text{Act}$.

Finite trace logics for streams with finitary probabilistic branching satisfy the following axioms of finitary convex cones

1. *Unit-Element:* $0 \cdot x = 0$ and $0 \oplus r \cdot x = r \cdot x$
2. *Absorption:* $r \cdot x \oplus s \cdot x = (r + s) \cdot x$
3. *Commutativity:* $r \cdot x \oplus s \cdot y = s \cdot y \oplus r \cdot x$
4. *Associativity:* $r \cdot x \oplus (s \cdot y \oplus t \cdot z) = (r \cdot x \oplus s \cdot y) \oplus t \cdot z$

and the following axiom of compatibility with the transition structure

5. *Aggregation:* $\langle a \rangle n \cdot \phi = n \cdot \langle a \rangle \phi$
6. *Branching:* $\langle a \rangle (\phi \oplus \phi') = \langle a \rangle \phi \oplus \langle a \rangle \phi'$

The above axioms allow formulas of I to be normalised into linear combinations of formulas specifying individual finite Act-traces.

Probabilistic branching cannot be modelled with a semiring monad, because instead of binary linear combination, probabilism requires convex combinations. The sub-distribution monad provides the type for probabilistic labelled transition systems.

We can find the following two definitions for L . First, $L(-) = F\{\sqrt{}\} \times (-)^{\text{Act}}$ with denotation $\delta : LQK \Rightarrow QK\bar{T}$ being the natural isomorphism $F\{\sqrt{}\} \times [K(-), F\{*\}]^{\text{Act}} \cong$

$[F\{*\} + Act \cdot (-), F\{*\}]$. Second, $L(-) = F\{\sqrt{\cdot}\} + Act \cdot (-)$ with the denotation δ defined such that

$$\delta_X(p \cdot \sqrt{\cdot} \in F\{*\}) := \left\{ \begin{array}{l} q \cdot \sqrt{\cdot} \mapsto (p * q) \cdot \sqrt{\cdot} \\ \langle a \rangle_X \mapsto 0 \end{array} \mid a \in Act, x \in X \right\} \quad (8.15)$$

$$\delta_X(\langle a \rangle(f : X \rightarrow F\{*\}) \in Act \cdot DAlg(X, F\{*\})) := \left\{ \begin{array}{l} q \cdot \sqrt{\cdot} \mapsto 0 \\ \langle a \rangle_X \mapsto f(x) \mid b \in Act, b \neq a, x \in X \\ \langle b \rangle_X \mapsto 0 \end{array} \right\} \quad (8.16)$$

Because products and coproducts do not coincide in $DAlg$, we obtain in fact two different logics. We choose the more natural functor $L(-) = F\{\sqrt{\cdot}\} + (-)^{Act}$. The initial sequence L resembles the one for graded labelled transition systems, because also for $\mathcal{D}_{\leq 1}$, $[F\{*\}, F\{*\}] \cong F\{*\}$. Note that the choice of L and δ has implications on the expressivity of the so obtained trace logics.

8.4.6 Path-Minimising Streams

The branching type of path-minimising streams is the semiring monad $(S^{(-)})_{\omega}$ for the min-semiring $S_{min} = (\mathbb{N} \cup \{\infty\}, \min, +, \infty, 0)$. Labelled transition systems of cost optimal paths are branching in the semiring monad for the min-semiring S_{min} . We read natural numbers and ∞ as costs, where ∞ means unattainable. Costs accumulate along paths (+) and at each node of branching, the cost optimal branch is chosen (min).

Definition 8.4.6 (Finite Trace Logics for Path-Minimising Streams). *Formulas $\phi, (\phi_i)_{i \in I}$ of finite trace logics for path-minimising streams are of the following shape.*

$$\phi ::= \bigoplus_i n_i \cdot \phi_i \mid \langle a \rangle \phi \quad (8.17)$$

where the indexing set I is a finite.

Finite trace logics for path-minimising streams satisfy the following axioms of min-semiring semimodules

1. *Unit-Element*: $0 \cdot \phi = 0$ and $0 \oplus n \cdot \phi = n \cdot \phi$
2. *Commutativity*: $n \cdot \phi \oplus m \cdot \phi' = m \cdot \phi' \oplus n \cdot \phi$
3. *Associativity*: $n \cdot \phi \oplus (m \cdot \phi' \oplus o \cdot z) = (n \cdot \phi \oplus m \cdot \phi') \oplus o \cdot z$
4. *Absorption*: $n \cdot \phi \oplus m \cdot \phi = n \cdot \phi$ if $n \leq m$

where $n, m \in \mathbb{N}$, and the following axioms of compatibility with the transition structure

1. *Aggregation*: $\langle a \rangle n \cdot \phi = n \cdot \langle a \rangle \phi$
2. *Branching*: $\langle a \rangle (\phi \oplus \phi') = \langle a \rangle \phi \oplus \langle a \rangle \phi'$

Formulas for generic trace logics for such transition systems inhabit (right) min-semiring semimodules. The semantic map $\llbracket - \rrbracket$ assigns to a formula and a point in the modelling coalgebra the least grade with which that point satisfies the formula. In free (unquotiented) trace logics, a formula can be thought of as a statement about the costs of the cost-optimal finite paths with given labels.

8.5 Invariance of Finitary Trace Logics under Finite Trace Equivalence

A logic is invariant under logical equivalence if semantically equivalent states are logically equivalent, that is satisfy the same formulas. The semantic equivalence in our case is trace equivalence. If trace semantics is final coalgebra semantics for \widetilde{T} in $B\text{-Alg}$ such as in Generic Trace Theory [HJS06], invariance under finite trace equivalence follows from $\llbracket - \rrbracket_{\mathbb{S}}$ factoring through the final coalgebra morphism tr_{σ}^{ω} .

Proposition 8.5.1. *If the final \widetilde{T} -coalgebra $\mathbb{Z} = \langle Z, \xi \rangle$ exists in $B\text{-Alg}$, finite trace logics for any logic functor L and denotation δ is invariant under finite trace equivalence.*

Proof. We show that $\llbracket - \rrbracket_S$ factors through Qtr^ω as in the following diagram.

$$\begin{array}{ccccc}
 & QFS & \xleftarrow{Q\sigma^\dagger} & Q\tilde{T}FS & \xleftarrow{\delta_{FS}} & LQFS \\
 & \uparrow Qtr^\omega & & \uparrow Q\tilde{T}tr^\omega & & \uparrow LQtr^\omega \\
 \llbracket - \rrbracket_S & QZ & \xleftarrow{Q\xi} & Q\tilde{T}Z & \xleftarrow{\delta_Z} & LQZ \\
 & \uparrow \llbracket - \rrbracket_Z & & \uparrow L\llbracket - \rrbracket_Z & & \\
 I & & \xleftarrow{\mathcal{L}_{B,T,L,\delta}} & LI & &
 \end{array} \tag{8.18}$$

The left square commutes, because it is the image of a commuting square; the right square commutes by naturality of δ . Thus both squares prove that Qtr^ω is an L -algebra morphisms from $Q\xi \circ \delta_Z$ to $Q\sigma^\dagger \circ \delta_X$. The initiality of $\mathcal{L}_{B,T,L,\delta}$ proves that the left and right triangle commute. \square

Where L and \tilde{T} are finitary, the invariance under finite trace equivalence does not depend on the existence of a final coalgebra, informally speaking, because finite trace equivalence is defined inductively along the discrete initial \tilde{T} -sequence.

Theorem 8.5.2. *Finite trace logics are invariant under finite trace equivalence for all logic functors L and denotations δ .*

Proof. Recall that $I = \text{colim}(Seq_L)$ has a cocone c over Seq_L . It follows from the universal property of colimits that c is jointly epic. From the latter and $Qtr^\omega \circ e = \llbracket - \rrbracket_S \circ c$ ³ follows the invariance under finite trace equivalence at once. \square

8.6 Expressivity of Finitary Coalgebraic Logics for Finite Traces

Recall the definition of $\llbracket - \rrbracket_S$ from Diagram (8.2) as the initial L -algebra morphism. Finite trace logics are expressive, if $\llbracket - \rrbracket_S : I \rightarrow QFS$ is epic. However, epicness is unlikely, as S is not restricted in its cardinality. It suffices to have \mathcal{L} epic on $Qtr^\omega[QFS]$. In our proof of expressivity we use an argument with a similar rationale to the one in Klin [Kli07].

³ $e : Seq_L \Rightarrow QSeq_{\tilde{T}}$ is defined in 8.3.4

Below we need the natural transformation $\delta^* : \widetilde{T}Q \Rightarrow QL$ defined from δ as follows.

$$\widetilde{T}Q \xrightarrow{v_{\widetilde{T}Q}} QQ\widetilde{T}Q \xrightarrow{Q\delta_Q} QLQQ \xrightarrow{QL\eta} QL \quad (8.19)$$

Theorem 8.6.1. $\mathcal{L}_{B,T,L,\delta}$ is expressive if

1. Ω is a coseparating ambimorphic object,
2. δ^* is monic,
3. \widetilde{T} preserves monomorphisms, and
4. \widetilde{T} and L are finitary.

Proof. Let $\mathbb{S} = \langle S, \sigma \rangle$ be a (B, T) -coalgebra, and let $a, b \in FS$ be points in the lifted \widetilde{T} -coalgebra σ^\dagger . Then expressivity of $\mathcal{L}_{B,T,L,\delta}$ means that a and b can be discerned by a formula ϕ , such that $a \Vdash_{\mathbb{S}} \phi$ and $b \not\Vdash_{\mathbb{S}} \phi$, if $tr^\omega(a) \neq tr^\omega(b)$. Abstractly, expressivity means that $\Vdash_{\mathbb{S}}$ factors through tr^ω followed by a monomorphism m as in the diagram below.

$$\begin{array}{ccccc}
 & & \widetilde{T}\Vdash_{\mathbb{S}} & & \\
 \widetilde{T}FS & \xrightarrow{\widetilde{T}tr^\omega} & \widetilde{T}colim(Seq_{\widetilde{T}}) & \xrightarrow{\widetilde{T}m} & \widetilde{T}QI \\
 \uparrow \sigma^\dagger & & \uparrow \alpha & & \downarrow \delta^* \\
 FS & \xrightarrow{tr^\omega} & colim(Seq_{\widetilde{T}}) & \xrightarrow{m} & QI \\
 & & \downarrow & & \uparrow QL \\
 & & \Vdash_{\mathbb{S}} & &
 \end{array} \quad (8.20)$$

We carry out the details of the proof in the following order below.

1. We define $m : colim(Seq_{\widetilde{T}}) \rightarrow Qcolim(Seq_L)$ and prove it monic in Lemma 8.6.4.
2. We prove $\Vdash_{\mathbb{S}} := m \circ tr^\omega$ the adjoint transpose of $\llbracket - \rrbracket_{\mathbb{S}}$ in Lemma 8.6.5.
3. The left square commutes, $\alpha \circ tr^\omega = \widetilde{T}tr^\omega \circ \sigma^\dagger$, by definition of tr^ω .
4. The right pentagon commutes, $\delta^* \circ \widetilde{T}m \circ \alpha = QL \circ m$, by definition of m .

□

Since our proof of expressivity is situated on the semantics' side, we begin with the definition of the adjoint transpose of $\delta : LQ \Rightarrow Q\tilde{T}$ under $Q \dashv Q$.

Definition 8.6.2 (Transpose of the Denotation). *Define the transpose $\delta^* : \tilde{T}Q \Rightarrow QL$ of the denotation $\delta : LQ \Rightarrow Q\tilde{T}$ as*

$$\delta^* := QL\nu \circ Q\delta_Q \circ \nu_{\tilde{T}Q} \quad (8.21)$$

Further below we will need the iterated version $(\delta^*)^n : \tilde{T}^n Q \Rightarrow QL^n$ of δ^* for all $n < \omega$.

$$(\delta^*)^0 = id_Q \text{ and } (\delta^*)^{n+1} := \delta_{L^n}^* \circ \tilde{T}(\delta^*)^n \quad (8.22)$$

δ^* gives us the explicit description of the adjoint transpose $e^\dagger : Seq_{\tilde{T}} \Rightarrow QSeq_L$ of e defined in 8.3.4, such that

$$e_0^\dagger = i_{Q0} \text{ and } e_{n+1}^\dagger = \delta_{L^n 0}^* \circ \tilde{T}e_n^\dagger. \quad (8.23)$$

Lemma 8.6.3. *e^\dagger is the adjoint transpose of e , such that $e_n^\dagger = Qe_n \circ \nu_{L^n 0}$ for all $n < \omega$.*

Proof. We prove the lemma by induction on n . In the base case $n = 0$, $e_0^\dagger = i_{Q0} = Q i_{Q0} \circ \nu_0 = Qe_0 \circ \nu_0$ commutes by initiality of 0. As an induction hypothesis we suppose

$e_n^\dagger = Qe_n \circ v_{\tilde{T}^n_0}$ for some $n < \omega$.

$$\begin{aligned}
e_{n+1}^\dagger &= && \text{definition of } e^\dagger \\
\delta_{L^n_0}^* \circ \tilde{T}e_n^\dagger &= && \text{definition of } \delta^* \\
QL\nu_{L^n_0} \circ Q\delta_{QL^n_0} \circ v_{\tilde{T}QL^n_0} \circ \tilde{T}e_n^\dagger &= && \text{naturality of } \nu \\
QL\nu_{L^n_0} \circ Q\delta_{QL^n_0} \circ QQ\tilde{T}e_n^\dagger \circ v_{\tilde{T}^{n+1}_0} &= && \text{induction hypothesis} \\
QL\nu_{L^n_0} \circ Q\delta_{QL^n_0} \circ QQ\tilde{T}Qe_n \circ QQ\tilde{T}v_{\tilde{T}^n_0} \circ v_{\tilde{T}^{n+1}_0} &= && \text{naturality of } \delta \\
QL\nu_{L^n_0} \circ QLQQe_n \circ QLQv_{\tilde{T}^n_0} \circ Q\delta_{\tilde{T}^n_0} \circ v_{\tilde{T}^{n+1}_0} &= && \text{naturality of } \nu \\
QLe_n \circ QL\nu_{Q\tilde{T}^n_0} \circ QLQv_{\tilde{T}^n_0} \circ Q\delta_{\tilde{T}^n_0} \circ v_{\tilde{T}^{n+1}_0} &= && Q\nu \circ v_Q = id_Q \\
QLe_n \circ Q\delta_{\tilde{T}^n_0} \circ v_{\tilde{T}^{n+1}_0} &= && \text{definition of } e \\
Qe_{n+1} \circ v_{\tilde{T}^{n+1}_0} &= &&
\end{aligned}$$

□

e^\dagger yields an explicit definition of the adjoint transpose $\Vdash_{\mathbb{S}} : FS \rightarrow QI$ of $\llbracket - \rrbracket_{\mathbb{S}} : I \rightarrow QFS$.

$$\begin{array}{ccc}
FS & \xrightarrow{\quad \Vdash_{\mathbb{S}} \quad} & Qcolim(Seq_L) \\
\searrow \scriptstyle \text{\textit{tr}}^\omega & \begin{array}{c} \xrightarrow{\quad colim(Seq_{\tilde{T}}) \quad} \xrightarrow{\quad m \quad} \xrightarrow{\quad} \\ \downarrow \scriptstyle d^\circ \\ \xrightarrow{\quad} \end{array} & \xrightarrow{\quad} \\
& |Seq_{\tilde{T}}| & \xrightarrow{\quad e^\dagger \quad} QSeq_L \\
& \swarrow \scriptstyle \text{\textit{tr}}^\omega & \nwarrow \scriptstyle Q^c
\end{array} \tag{8.24}$$

where d° is defined as in Equation 8.26.

The morphism $m : colim(Seq_{\tilde{T}}) \rightarrow QI = Qcolim(Seq_L) = lim(Seq^L)$ is defined as the universal morphism for the limit $lim(Seq^L)$ with respect to the cone of $colim(Seq_{\tilde{T}})$ over Seq^L in the following diagram.

$\lim(QSeq_L)$ are jointly monic, as for any object A of $B\text{-Alg}$ with morphisms $f, g : A \rightarrow \text{colim}(Seq_{\bar{T}})$, it is $m \circ f = m \circ g$ if and only if $e \circ d^\circ \circ f = e \circ d^\circ \circ g$ which implies $f = g$. \square

We define $\Vdash_{\mathbb{S}} := m \circ tr^\omega$, and prove $\Vdash_{\mathbb{S}}$ the adjoint transpose of $\llbracket - \rrbracket_{\mathbb{S}}$ under $Q \dashv Q$.

Lemma 8.6.5. $\Vdash_{\mathbb{S}} = Q\llbracket - \rrbracket_{\mathbb{S}} \circ \nu_{FS}$.

Proof. The proof is summarised in the following diagram.

$$\begin{array}{ccccc}
 FS & \xrightarrow{\quad \Vdash_{\mathbb{S}} \quad} & Qcolim(Seq_L) & & \\
 \downarrow tr^\omega & \xrightarrow{\nu_{FS}} & QQFS & \xrightarrow{Q\llbracket - \rrbracket_{\mathbb{S}}} & \\
 |Seq_{\bar{T}}| & \xrightarrow{\nu_{Seq_{\bar{T}}}} & QQ|Seq_{\bar{T}}| & \xrightarrow{Qe} & QSeq_L \\
 & \searrow e^\dagger & & &
 \end{array}
 \quad (8.27)$$

- The left square commutes because of the naturality of ν .
- The right square is the image of Diagram 8.6 under Q and thus commutes.
- We have shown the bottom triangle to commute in Lemma 8.6.3
- The outer quadrangle commutes by definition of $\Vdash_{\mathbb{S}}$.

Thus $\Vdash_{\mathbb{S}} = Q\llbracket - \rrbracket_{\mathbb{S}} \circ \nu_{FS}$ commutes. \square

Example 8.6.6. *In order to assert that finite trace logics for (B, T) -coalgebras as in Theorem 8.6.1 are expressive, we assume that*

1. Ω is a coseparating ambimorphic object,
2. δ^* is monic.
3. \bar{T} preserves monomorphisms, and
4. \bar{T} and L are finitary.

We claim that these assumptions are met in common cases by the examples of finite trace logics in Section 8.4. In all cases, $T = \{\sqrt{}\} + \text{Act} \cdot \text{Id}$ for a finite set Act of labels, so that uniformly $\bar{T} = F\{\sqrt{}\} + \text{Act} \cdot \text{Id}$ and $L = QF\{\sqrt{}\} \times (-)^{\text{Act}}$ and $\Omega = F\{*\}$.

1. *There are coseparating ambimorphic objects Ω as of Example 8.1.7.*
2. *δ^* is monic, since δ is iso. We prove this claim separately below.*
3. *\widetilde{T} preserves monomorphisms.*
4. *\widetilde{T} and L are finitary.*

We can show that δ^* is monic generically for the examples of finite trace logics above.

Therefor we separate the claim into the generators $\delta_X^* \upharpoonright_{\kappa_0 F\{\sqrt{\cdot}\}}$ and $\delta_X^* \upharpoonright_{\kappa_1 Act \cdot QX}$. Recall that $Q = [-, \Omega]$.

$$\begin{array}{ccc}
 F\{\sqrt{\cdot}\} + Act \cdot QX & \xrightarrow{v_{F\{\sqrt{\cdot}\} + Act \cdot QX}} & QQ(F\{\sqrt{\cdot}\} + Act \cdot QX) \\
 & & \downarrow Q\delta_{Q(F\{\sqrt{\cdot}\} + Act \cdot QX)} \\
 Q(QF\{\sqrt{\cdot}\} \times X^{Act}) & \xleftarrow{Q(id_{QF\{\sqrt{\cdot}\}} \times v_X^{Act})} & Q(QF\{\sqrt{\cdot}\} \times (QQX)^{Act})
 \end{array} \tag{8.28}$$

$$\begin{array}{ccc}
 \kappa_0 \phi \in F\{\sqrt{\cdot}\} \vdash & \longrightarrow & \lambda m \in QF\{\sqrt{\cdot}\}.m(\phi) \\
 \delta_X^* \upharpoonright_{\kappa_0 F\{\sqrt{\cdot}\}} \downarrow & & \downarrow \\
 \lambda m \in QF\{\sqrt{\cdot}\}.m(\phi) & \longleftarrow & \lambda m \in QF\{\sqrt{\cdot}\}.m(\phi)
 \end{array}$$

$$\begin{array}{ccc}
 \kappa_1 \kappa_a m' \in Act \cdot QX \vdash & \longrightarrow & \lambda M \in Q(F\{\sqrt{\cdot}\} + Act \cdot QX).M(\kappa_1 \kappa_a m') \\
 \delta_X^* \upharpoonright_{\kappa_1 Act \cdot QX} \downarrow & & \downarrow \\
 \lambda g \in X^{Act}.m(g(a)) & \longleftarrow & \lambda(h \in QF\{\sqrt{\cdot}\}, f \in (QQX)^{Act}).f(a)(m)
 \end{array}$$

Then

- $\delta_X^* \upharpoonright_{\kappa_0 F\{\cdot\}}$ is monic by Lemma 8.1.6 because Ω is a coseparator, and
- $\delta_X^* \upharpoonright_{\kappa_1 Act \cdot QX}$ is monic because g ranges over the whole of X^{Act} .

Part IV

Coalgebraic Automata Theory

Chapter 9

Game Bisimulations in Parity Graph Games

In this chapter we briefly review the basic definitions of parity graph games, and show that each parity graph game is equivalent to one, where we can distinguish some positions as basic. These basic positions mark rounds in the plays of these parity graph games. Within each round we can normalise the interaction pattern of the players using the notion of the players power introduced by van Benthem in [vB02]. The normalisation lays foundation to game bisimulations, which are congruence relations on basic positions. The work of this chapter is joint with Yde Venema.

9.1 Preliminary Definitions

We recall the formal definition of parity graph games [GTW02].

Definition 9.1.1 (Parity Graph Games). *A two-player parity graph game $\mathcal{G} = \langle V_0, V_1, E, v_I, \Omega \rangle$ is played by two players, 0 and 1, on a bipartite graph $\langle V_0, V_1, E, v_I \rangle$, the arena of \mathcal{G} , where*

- $V = V_0 \cup V_1$ with $V_0 \cap V_1 = \emptyset$ is the set of positions,
- $E \subseteq V \times V$ is the edge relation of admissible moves, and
- $v_I \in V$ is distinguished as the initial position.

The priority function $\Omega : v \rightarrow \mathbb{N}$ assigns to each position $v \in V$ a natural number, and is image-finite. A position v is called *terminal* if $E[v] = \emptyset$.

Notation 9.1.2. We abbreviate $V := V_0 \cup V_1$.

Definition 9.1.3 (Plays). A play of a parity graph game $\mathcal{G} = \langle V_0, V_1, E, v_I, \Omega \rangle$ is a finite or infinite sequence $p \in V^* \cup V^\omega$ of positions such that $p(0) = v_I$ and for all $i \leq |p|$, $(p(i), p(i+1)) \in E$. The total plays of \mathcal{G} are the infinite plays $p \in V^\omega$ and the finite plays $p \in V^*$ with $E[\text{last}(p)] = \emptyset$. The initial plays are the plays which are not total.

Definition 9.1.4 (Winning Condition). A total play p of a parity graph game $\mathcal{G} = \langle V_0, V_1, E, v_I, \Omega \rangle$ is winning for player $\Pi \in \{0, 1\}$ if and only if

1. p is finite and $\text{last}(p) \in V_{1-\Pi}$, or
2. p is infinite and the largest priority occurring infinitely often is of parity Π .

Notation 9.1.5. Since we identify players with their assigned parity, we can write $1 - \Pi$ for the opponent of Π .

We introduce strategies as partial functions on the set of positions, defined precisely on the positions of one player. This definition is suitable because parity graph games are history-free determined [Mos91, EJ91, Zie98]. The latter means, that a player has a winning strategy in a parity graph game if and only if she has one which takes into consideration the current position only. For the full details and the proof of the following theorem we refer to the original literature.

Theorem 9.1.6. Parity graph games are history-free determined.

Definition 9.1.7 (Strategies). A strategy of a player Π in a parity graph game $\mathcal{G} = \langle V_0, V_1, E, v_I, \Omega \rangle$ is a partial function $f: V \rightarrow V$, which is

1. defined precisely on positions v from V_Π with $E[v] \neq \emptyset$ and
2. consistent with E , such that $\text{Gr}(f) \subseteq E$.

Definition 9.1.8 (Plays Consistent with Strategies). *A play p of a parity graph game $\mathcal{G} = \langle V_0, V_1, E, v_I, \Omega \rangle$ is consistent with a strategy f of a player $\Pi \in \{0, 1\}$, if for all $i < |p| - 1$, $f(p(i))$ is defined and $f(p(i)) = p(i + 1)$ whenever $p(i) \in V_\Pi$.*

Definition 9.1.9 (Winning Strategies). *A strategy f of a player Π in a parity graph game $\mathcal{G} = \langle V_0, V_1, E, v_I, \Omega \rangle$ is winning for Π if all plays consistent with f are winning for Π .*

Definition 9.1.10 (Winning Positions). *A position v in a game $\mathcal{G} = \langle V_0, V_1, E, v_I, \Omega \rangle$ is winning for player Π if Π has a winning strategy in the parity graph game $\langle V_0, V_1, E, v, \Omega \rangle$. The set of winning positions for Π is called the winning region for Π in \mathcal{G} , and is denoted by $\text{Win}_\Pi^\mathcal{G}$.*

Definition 9.1.11 (Equivalence of Parity Graph Games). *Two parity graph games $\mathcal{G} = \langle V_0, V_1, E, v_I, \Omega \rangle$ and $\mathcal{G}' = \langle V'_0, V'_1, E', v'_I, \Omega' \rangle$ are called equivalent if each player Π has a winning strategy in \mathcal{G} if and only if she has one in \mathcal{G}' .*

Proposition 9.1.12. *Any parity graph game $\mathcal{G} = \langle V_0, V_1, E, v_I, \Omega \rangle$ is equivalent to one where each play is infinite.*

Proof. Any play of \mathcal{G} is finite if it ends in a position $v \in V_\Pi$ of a player Π where Π can not move, that is $E[v] = \emptyset$. \mathcal{G} can be transformed into an equivalent game $\mathcal{G}' = \langle V'_0, V'_1, E', v'_I, \Omega' \rangle$ by

- adding states v_0 and v_1 to V_0 and V_1 , respectively,
- with edges (v_Π, v_Π) for both $\Pi \in \{0, 1\}$, and
- priorities $\Omega'(v_\Pi) := \Pi$ for both $\Pi \in \{0, 1\}$.

Then for each state $v \in V_\Pi$ with $E[v] = \emptyset$ we add an edge $(v, v_{1-\Pi})$ to E' . v_Π is a winning position for player Π determined in an infinite play. E does not admit dead ends, so that all plays in \mathcal{G} are infinite. Moreover, instead of getting stuck, a player has to move into the winning region of the opponent, so that indeed both games are equivalent. \square

9.2 Unravelling Parity Graph Games

The unravelling of a graph is a tree, that is an acyclic root graph. In the following we show that the unravelling of the arena of a parity graph games defines a parity graph game. The unravelled parity graph game is equivalent to the original one.

Definition 9.2.1 (Unravelling Parity Graph Games). *Let $\mathcal{G} = \langle V_0, V_1, E, v_I, \Omega \rangle$ be a parity graph game, we define a parity graph game $T^{\mathcal{G}} = \langle V'_0, V'_1, E', v'_I, \Omega' \rangle$ such that*

- $V' := \{w \in V^* \mid w \neq \epsilon, w(0) = v_I, \forall n < |w| - 1. (w(n), w(n+1)) \in E\}$
- $V'_\Pi := \{w \in V' \mid \text{last}(w) \in V_\Pi\}$ for each $\Pi \in \{0, 1\}$
- $(w, w') \in E'$ if $|w'| = |w| + 1$, $w = w'|_{|w|}$, and $(\text{last}(w), \text{last}(w')) \in E$,
- $v'_I := (v_I)^1$, and
- $\Omega'(w) := \Omega(\text{last}(w))$.

Unravelling a parity graph game preserves its semantics.

Proposition 9.2.2. *Any $\mathcal{G} = \langle V_0, V_1, E, v_I, \Omega \rangle$ and its unravelling $T^{\mathcal{G}} = \langle V'_0, V'_1, E', v'_I, \Omega' \rangle$ are equivalent.*

This proposition can easily be proven by establishing a game bisimulation between \mathcal{G} and $T^{\mathcal{G}}$. The concept of game bisimulations will be introduced shortly, and specialises in the case of Proposition 9.2.2 as follows.

Proof. Positions w in $T^{\mathcal{G}}$ correspond to positions $\text{last}(w)$ in \mathcal{G} , in the sense (1) that each player has choices w' in $T^{\mathcal{G}}$ corresponding to choices $\text{last}(w')$ in \mathcal{G} . Thus every play $p' = w_0 w_1 \dots$ in $T^{\mathcal{G}}$ corresponds to a play $p = \text{last}(w_0) \text{last}(w_1) \dots$ of the same length l such that at each index $i < l$, (2) $\Omega'(w_i) = \Omega(\text{last}(w_i))$ and thus p is winning for player Π if and only if p is winning for Π . (1) and (2) are part of Definition 9.2.1. \square

¹ (v_I) denotes the one-character word v_I .

9.3 Structuring Parity Graph Games

We give a structure to parity graph games by distinguishing a set of positions, which together with the terminal positions delimit rounds in parity graph games.

Definition 9.3.1 (Basic Sets of Positions). *Given a parity graph game $\mathcal{G} = \langle V_0, V_1, E, v_I, \Omega \rangle$, call a set $B \subseteq V$ of positions basic if*

1. *the initial position of \mathcal{G} belongs to B ;*
2. *any full play starting at some $b \in B$ either ends in a terminal position or it passes through another position in B ; and*
3. *$\Omega(v) = 0$ if and only if $v \notin B$.*

We call positions in B basic.

Requiring a basic set of positions does not restrict the generality of parity graph games. In particular we may choose the whole set of positions to be basic. The obtained structure on the parity graph game is then trivial, however.

Proposition 9.3.2. *Every parity graph game is equivalent to one which we can identify a basic set of positions in.*

Proof. Let $\mathcal{G} = \langle V_0, V_1, E, v_I, \Omega \rangle$ be a parity graph game, we define $\mathcal{G}' := \langle V_0, V_1, E, v_I, \Omega' \rangle$ with $\Omega'(v) := \Omega(v) + 2$ for all $v \in V$. Then V itself forms a basic set B of positions, as we show with the following argument concurrent to Definition 9.3.1.

1. $v_I \in B$ as $\Omega'(v_I) \neq 0$.
2. Any position $v \in B$ is either terminal or $E[v] \neq \emptyset$, in which case $E[v] \subseteq B$. Thus any play passing through a basic position v either terminates immediately after v or passes through a basic position from $E[v]$ next.
3. By definition of Ω' , $\Omega'(v) > 0$ for all $v \in V$.

It remains to show \mathcal{G} and \mathcal{G}' equivalent. First, we see that \mathcal{G} and \mathcal{G}' have the same arena, thus in every position v of \mathcal{G} both players have the same choices as in v in \mathcal{G}' , so that strategies are transferable between the two games, and the set of plays consistent with a strategy in one game coincides with the set of plays consistent with the same strategy in the other game. Second, adding 2 uniformly to the priority of positions preserves their parity. Thus a strategy for one player in one game is winning if and only if it is winning for the same player in the other game. \square

Basic positions delimit rounds of plays.

Definition 9.3.3 (Local Game Trees). *Let B be a basic set of positions in a parity graph game $\mathcal{G} = \langle V_0, V_1, E, v_I, \Omega \rangle$. The local game tree associated with a basic position $b \in B$ is defined as the following bipartite tree $T^b = \langle V_0^b, V_1^b, E^b, v_I^b \rangle$. Let V^b be the set of those finite paths β starting at b , of which the only basic positions are $\text{first}(\beta) = b$ and possibly $\text{last}(\beta)$. The bipartition of V^b is given through the bipartition of V , that is $V_\Pi^b := \{\beta \in V^b \mid \text{last}(\beta) \in V_\Pi\}$ for both players $\Pi \in \{0, 1\}$. The root v_I^b is (b) . The edge relation is defined as $E^b := \{(\beta, \beta v) \mid v \in E[\text{last}(\beta)]\}$.*

A node $\beta \in V^b$ is a leaf of T^b if $|\beta| > 1$ and $\text{last}(\beta) \in B$; we let $\text{Leaves}(T^b)$ denote the set of leaves of T^b , and put $N(b) := \{\text{last}(\beta) \mid \beta \in \text{Leaves}(T^b)\}$.

For the semantics of a parity graph game, only the basic positions matter. Intuitively, the local games delimited by basic and terminal positions can be normalised to local games consisting of only one choice of each player. The normalisations makes use of the concept of the power of players [vB02].

Definition 9.3.4 (Powers). *Let B be a basic set related to some parity graph game $\mathcal{G} = \langle V_0, V_1, E, v_I, \Omega \rangle$, and let b be a basic position. By induction on the height of a node $\beta \in V^b$, we define, for each player Π , the power of Π at β as a collection $P_\Pi^\mathcal{G}(\beta)$ of subsets of $N(b)$:*

- *If $\beta \in \text{Leaves}(T^b)$, we put, for each player Π ,*

$$P_\Pi^\mathcal{G}(\beta) := \{\{\text{last}(\beta)\}\}. \quad (9.1)$$

- If $\beta \notin \text{Leaves}(T^b)$, we put

$$P_{\Pi}^{\mathcal{G}}(\beta) := \begin{cases} \bigcup \{P_{\Pi}^{\mathcal{G}}(\gamma) \mid \gamma \in E^b(\beta)\} & \text{if } \beta \in V_{\Pi}^b, \\ \left\{ \bigcup_{\gamma \in E^b(\beta)} Y_{\gamma} \mid Y_{\gamma} \in P_{\Pi}^{\mathcal{G}}(\gamma), \text{ all } \gamma \right\} & \text{if } \beta \in V_{1-\Pi}^b. \end{cases} \quad (9.2)$$

Finally, we define the power of Π at b as the set $P_{\Pi}^{\mathcal{G}}(b) := P_{\Pi}^{\mathcal{G}}((b))$.

Perhaps some special attention should be devoted to the paths β in T^b with $E^b(\beta) = \emptyset$. If such a β is a leaf of T^b , then the definition above gives $P_{\Pi}^{\mathcal{G}}(\beta) = P_{1-\Pi}^{\mathcal{G}}(\beta) = \{\text{last}(\beta)\}$. But if β is not a leaf of T^b , then we obtain, by the inductive clause of the definition:

$$P_{\Pi}^{\mathcal{G}}(\beta) := \begin{cases} \emptyset & \text{if } \beta \in V_{\Pi}^b, \\ \{\emptyset\} & \text{if } \beta \in V_{1-\Pi}^b. \end{cases} \quad (9.3)$$

Definition 9.3.5 (Local Games). Let $\mathcal{G} = \langle V_0, V_1, E, v_I, \Omega \rangle$ be a parity graph game with basic set $B \subseteq V$. We define a game \mathcal{G}^b local to a basic position $b \in B$ in \mathcal{G} as the (finite length) graph game played by 0 and 1 on the local game tree $T^b = \langle V^b, E^b \rangle$. Plays of this game are won by a player Π if their opponent $1 - \Pi$ gets stuck, and end in a tie if the last position of the play is a leaf of T^b .

Definition 9.3.6 (Local Strategies). A local strategy of a player Π in a local game $\mathcal{G}^b = \langle V_0^b, V_1^b, E^b, v_I^b \rangle$ is a partial function $f : V_{\Pi}^b \rightarrow V^b$ defined on such β if and only if $E^b(\beta)$ is non-empty, then $f(\beta) \in E^b(\beta)$. Such a local strategy for player Π is surviving if it guarantees that Π will not get stuck, and thus does not lose the local game; and a local strategy is winning for Π if it guarantees that her opponent $1 - \Pi$ gets stuck.

Consider the play of the local game \mathcal{G}^b in which 0 and 1 play local strategies f_0 and f_1 , respectively. If this play ends in a leaf β of the local game tree, we let $\text{Res}(f_0, f_1)$ denote the basic position $\text{last}(\beta)$; if one of the players gets stuck in this play, we leave $\text{Res}(f_0, f_1)$ undefined. Given a local strategy f_0 for player 0, we define

$$X_{f_0} := \{\text{Res}(f_0, f_1) \mid f_1 \text{ a local strategy for player 1}\}, \quad (9.4)$$

and similarly we define X_{f_1} for a strategy f_1 for player 1.

Proposition 9.3.7. *Let \mathcal{G} , B and b as in Definition 9.3.4, let $\Pi \in \{0, 1\}$ be a player, and let W be a subset of $N(b)$. Then the following are equivalent.*

1. $P_{\Pi}^{\mathcal{G}}(b)$ is non-empty.
2. Π has a surviving strategy f in \mathcal{G}^b such that $X_f \in P_{\Pi}^{\mathcal{G}}$.

Proof. We prove the two directions separately.

Given a set $W \in P_{\Pi}^{\mathcal{G}}(b)$, we define a surviving strategy f_W for Π in the local game \mathcal{G}^b such that each play consistent with f_W will either end in win of Π , that is a default of $1 - \Pi$, or in a position in W . We define f_W partially and recursively over plays of T^b , such that in each position β along a play consistent with f (*) there is a set $W' \in P_{\Pi}^{\mathcal{G}}(\beta)$ with $W' \subseteq W$. In the following let β be a position along a play consistent with f_W .

- If $\beta \in V_{\Pi}^b$, then there is a position $\beta' \in E^b(\beta)$ such that $W \in P_{\Pi}^{\mathcal{G}}(\beta)$ by definition of $P_{\Pi}^{\mathcal{G}}$. For each such β' , we can extend f_W by $f_W(\beta) := \beta'$.
- If $\beta \in V_{1-\Pi}^b$, then for all positions $\beta' \in E^b(\beta)$, there is a set $W'_{\beta'} \in P_{\Pi}^{\mathcal{G}}(\beta')$ with $W'_{\beta'} \subseteq W$ by definition of $P_{\Pi}^{\mathcal{G}}(\beta)$. If some $W'_{\beta'}$ is empty, Π defeats $1 - \Pi$ in β' . Moreover $W = \bigcup_{\beta' \in E^b(\beta)} (W'_{\beta'})$.

As all plays of \mathcal{G}^b are finite, each play consistent with f will end either in a win for Π or in $N(b)$. In the latter case, (*) will hold in a position in β with $\text{last}(\beta) \in N(b)$, which by definition of $P_{\Pi}^{\mathcal{G}}$ means that $\text{last}(\beta) \in W$. Finally, f_W can be arbitrarily extended to a fully defined local strategy.

Let Π have a surviving strategy f in \mathcal{G}^b . Then each play of \mathcal{G}^b consistent with f ends either in a position β where $P_{\Pi}^{\mathcal{G}} = \{\emptyset\}$, or in a position $\beta \in N(b)$, where $P_{\Pi}^{\mathcal{G}}(\beta) = \{\text{last}(\beta)\}$. Beginning with these positions, we prove by induction over the length of plays in \mathcal{G}^b consistent with f that $(X_f)_{\beta} \in P_{\Pi}^{\mathcal{G}}(\beta)$ for each position β in such a play.

- If $\beta \in V_{\Pi}^b$, $(X_f)_{\beta} = (X_f)_{\beta'} \in P_{\Pi}^{\mathcal{G}}(\beta)$ as $P_{\Pi}^{\mathcal{G}}(f(\beta)) \subseteq P_{\Pi}^{\mathcal{G}}(\beta)$.
- If $\beta \in V_{1-\Pi}^b$, there is a set $(X_f)_{\beta} \in P_{\Pi}^{\mathcal{G}}(\beta)$ define as $(X_f)_{\beta} = \bigcup_{\beta' \in E^b(\beta)} (X_f)_{\beta'}$ by definition of $P_{\Pi}^{\mathcal{G}}$.

In (b) put $X_f := (X_f)_{(b)}$. □

Proposition 9.3.8. *Let b be a basic position in a parity graph game \mathcal{G} and let $U \subseteq N(b)$ such that $U \notin [P_{\Pi}^{\mathcal{G}}(b)]$, then there is a set $V \in P_{1-\Pi}^{\mathcal{G}}(b)$ with $U \cap V = \emptyset$.*

Proof. We prove by induction on the length of plays in \mathcal{G}^b , that for each position β in \mathcal{G}^b , (*) if there is a set $U \subseteq N(b)$ with $U \notin [P_{\Pi}^{\mathcal{G}}(\beta)]$, then there is a set $V \in P_{1-\Pi}^{\mathcal{G}}(\beta)$ with $U \cap V = \emptyset$.

- Suppose β in $N(b)$, if $\beta \in V_{\Pi}^b$ then $[P_{\Pi}^{\mathcal{G}}(\beta)] = N(b)$, and if $\beta \in V_{1-\Pi}^b$, then $P_{\Pi}^{\mathcal{G}}(\beta) = \emptyset$, so that β satisfies (*).
- If otherwise $\beta \notin N(b)$ and $\beta \in V_{\Pi}^b$, every $U \notin [P_{\Pi}^{\mathcal{G}}(\beta)]$ does not occur in $[P_{\Pi}^{\mathcal{G}}(\beta')]$ of any $\beta' \in E^b[\beta]$. Assuming (*) as the induction hypothesis, there is a corresponding $V_{\beta'}$ in $P_{1-\Pi}^{\mathcal{G}}(\beta')$ for any β' with $V_{\beta'} \cap U = \emptyset$. Their union $\bigcup_{\beta' \in E^b[\beta]} V_{\beta'}$ does not intersect U either, and occurs in $P_{1-\Pi}^{\mathcal{G}}(\beta)$ by definition of $P_{1-\Pi}^{\mathcal{G}}$.
- If $\beta \notin N(b)$ and $\beta \in V_{1-\Pi}^b$, $U \notin [P_{\Pi}^{\mathcal{G}}(\beta)]$ means that there is a successor $\beta' \in E^b[\beta]$ such that $U \notin [P_{\Pi}^{\mathcal{G}}(\beta)]$, so that there is a set $V \in P_{1-\Pi}^{\mathcal{G}}(\beta')$ with $U \cap V = \emptyset$ taking (*) as the induction hypothesis. Then $V \in P_{1-\Pi}^{\mathcal{G}}(\beta)$ as $P_{1-\Pi}^{\mathcal{G}}(\beta') \subseteq P_{1-\Pi}^{\mathcal{G}}(\beta)$ by definition of $P_{1-\Pi}^{\mathcal{G}}$.

□

9.4 Normalised Parity Graph Games

Previously we argued that the concept of a players power allows us to normalise the interaction pattern between the players of a parity graph game. Formally, we obtain parity graph games of the following shape.

Definition 9.4.1 (Normalised Parity Graph Games). *Let $\mathcal{G} = \langle V_0, V_1, E, v_I, \Omega \rangle$ be a parity graph game with a basic set B of positions, the normalised parity graph game of \mathcal{G} has an arena*

$$\langle B \cup \mathcal{P}PB, \mathcal{P}B, Gr(P)_0 \cup \in, v_I, \Omega \rangle \quad (9.5)$$

In the above definition the edge relation is determined by P_0 . The basic set of positions of \mathcal{G} is basic in the normalised game. Every round in the normalised game consists of a choice of 0 followed by one of 1.

Definition 9.4.2 (Normal Parity Graph Games). *A normal parity graph game is a structure $\langle B, P, b_I, \Omega \rangle$ with*

- *a single sort B of positions,*
- *a function $P_0 : B \rightarrow \mathcal{P}\mathcal{P}B$,*
- *a position $v_I \in B$ distinguished as initial,*
- *a priority function $\Omega : B \rightarrow \mathbb{N}$.*

Normalised parity graph games are normal, which we emphasise by using the same notation in both definitions. The latter definition will provide a frame to prove game bisimulations congruence relations in the next section.

9.5 Game Bisimulations

Definition 9.5.1 (Game Bisimulations). *Let $\mathcal{G} = \langle V_0, V_1, E, \Omega \rangle$ and $\mathcal{G}' = \langle V'_0, V'_1, E', \Omega' \rangle$ be parity graph games with basic sets B and B' , respectively, and let Π and Π' be (not necessarily distinct) players in \mathcal{G} and \mathcal{G}' , respectively.*

A Π, Π' -game bisimulation is a binary relation $Z \subseteq B \times B'$ satisfying for all $v \in V$ and $v' \in V'$ with vZv' the structural conditions

- $(\Pi, \text{forth}) \forall W \in P_\Pi^\mathcal{G}(v). \exists W' \in P_{\Pi'}^{\mathcal{G}'}(v'). \forall w' \in W'. \exists w \in W. (w, w') \in Z,$
- $(1 - \Pi, \text{forth}) \forall W \in P_{1-\Pi}^\mathcal{G}(v). \exists W' \in P_{1-\Pi'}^{\mathcal{G}'}(v'). \forall w' \in W'. \exists w \in W. (w, w') \in Z,$
- $(\Pi, \text{back}) \forall W' \in P_{\Pi'}^{\mathcal{G}'}(v'). \exists W \in P_\Pi^\mathcal{G}(v). \forall w \in W. \exists w' \in W'. (w, w') \in Z,$
- $(1 - \Pi, \text{back}) \forall W' \in P_{1-\Pi'}^{\mathcal{G}'}(v'). \exists W \in P_{1-\Pi}^\mathcal{G}(v). \forall w \in W. \exists w' \in W'. (w, w') \in Z,$

and the priority conditions

- (parity) $\Omega(v) \bmod 2 = \Pi$ if and only if $\Omega'(v') \bmod 2 = \Pi'$,
- (contraction) for all $v, w \in V$ and $v', w' \in V'$ with $(v, v') \in Z$ and $(w, w') \in Z$, $\Omega(v) \leq \Omega(w)$ if and only if $\Omega(v') \leq \Omega(w')$.

In Condition (parity) above we identify players with their characteristic parity. Note that in fact there are only two kinds of game bisimulations: the $(0, 0)$ -bisimulations coincide with the $(1, 1)$ -bisimulations, and the $(0, 1)$ -bisimulations coincide with the $(1, 0)$ -bisimulations.

Game bisimulations are congruence relations. This result follows immediately from the definition of game bisimulations.

Proposition 9.5.2. *Game bisimulations are congruence relations.*

Proof. We verify the properties of equivalence relations.

- Reflexivity: Let \mathcal{G} be a parity graph game with a basic set B of positions, then $\Delta_B = \{(b, b) \mid b \in B\}$ is a (Π, Π) -game bisimulation. Δ_B meets the structural conditions trivially as it is the diagonal relation on the basic positions of the same arena, and meets the priority conditions as the basic positions on both sides have the same priority.
- Symmetry: Let \mathcal{G} and \mathcal{G}' be parity graph games with basic sets B and B' of positions, and let $Z \subseteq B \times B'$ be a (Π, Π') -game bisimulation. Then Z^{-1} is a (Π', Π) -game bisimulation. That Z^{-1} meets the structural conditions follows from the symmetry in the definition of game bisimulations, and that it meets the priority conditions follows from Z meeting the priority conditions.
- Transitivity: Let \mathcal{G} , \mathcal{G}' , and \mathcal{G}'' be parity graph games with basic sets B , B' , and B'' of positions, and let $Z \subseteq B \times B'$ be a (Π, Π') -game bisimulation and let $Z' \subseteq B' \times B''$ be a (Π', Π'') -game bisimulation. Then $Z; Z'$ is a (Π, Π'') -game bisimulation, as $Z; Z'$ satisfies the structural compositions, as (Π, forth) , $(1 - \Pi, \text{forth})$, (Π, back) , and $(1 - \Pi, \text{back})$ for Z and Z' compose, and $Z; Z'$ satisfies the priority conditions as (parity) and (contraction) compose.

Additionally, game bisimulations are compatible with transitions in parity graph games as in Definition 9.4.2. Formally, let B be a basic set of positions in a parity graph game \mathcal{G} , then there is an operator taking sets $\mathbb{W} \in \mathcal{P}\mathcal{P}B$ to a normalised interaction pattern where first Π chooses a set $W \in \mathbb{W}$, and then $1 - \Pi$ chooses a basic position from W . Next we show that game bisimulation is compatible with this operator.

- **Congruence:** Let \mathcal{G} and \mathcal{G}' be parity graph games with basic sets B and B' of positions, and let $Z \subseteq B \times B'$ be a (Π, Π') -game bisimulation between \mathcal{G} and \mathcal{G}' . Fix an arbitrary pair $(b, b') \in Z$ of game bisimilar positions. For any $W \in P_{\Pi}^{\mathcal{G}}(b)$ there is a set $W' \in P_{\Pi'}^{\mathcal{G}'}(b')$, such that for any $p \in W$ there is a game bisimilar $p' \in W'$ and vice versa. Symmetrically, there is for any such W' such a W . Moreover, the previous two conditions hold also for $1 - \Pi$ in place of Π . These four conditions are the converses of (Π, forth) , $(1 - \Pi, \text{forth})$, (Π, back) , and $(1 - \Pi, \text{back})$.

□

In informal terms, the following theorem states that bisimilar games \mathcal{G} and \mathcal{G}' are equivalent in the sense, that any player Π has a winning strategy in \mathcal{G} if and only if she has one in \mathcal{G}' . Thus the theorem confirms soundness of our definition of game bisimulations.

Theorem 9.5.3. *Let \mathcal{G} and \mathcal{G}' be parity graph games and let Π and Π' be players in the respective game. If the initial positions of \mathcal{G} and \mathcal{G}' are in a (Π, Π') -game bisimulation, then Π has a winning strategy in \mathcal{G} if and only if Π' has one in \mathcal{G}' .*

Proof. Let $\mathcal{G} = \langle V_0, V_1, E, \Omega \rangle$ and $\mathcal{G}' = \langle V'_0, V'_1, E', \Omega' \rangle$ be parity graph games with initial positions v_0 and v'_0 and basic sets B and B' of positions, respectively. We show that for every strategy f of Π in \mathcal{G} from v_0 , there is a strategy f' of Π' in \mathcal{G}' from v'_0 winning if f is. The converse direction will follow through the symmetry in the definition of game bisimulation.

We define f' such that the basic positions in any play p of \mathcal{G} from v_0 consistent with f are Z -related to the basic positions in some play p' of \mathcal{G}' from v'_0 consistent with f' , and vice versa. The latter direction will follow through the symmetry in the definition of game bisimulation.

Given f , we define f' inductively and show that all basic positions b and b' in p and p' respectively meet the induction hypothesis that $(b, b') \in Z$. Implicitly we make use of Proposition 9.3.7.

Let the play of \mathcal{G} be in the basic position b , let the play of \mathcal{G}' be in the basic position b' , and let $(b, b') \in Z$. Since f is winning, there is a set $W \in P_{\Pi}^{\mathcal{G}}(b)$ by definition of game bisimulations $Z[W] \in P_{\Pi'}^{\mathcal{G}'}(b')$. By Proposition 9.3.7, Π' has a local strategy $f'^{b'}$ in $\mathcal{G}'^{b'}$ such that $X_{f'^{b'}} = Z[W]$. Put $f'(last(\beta)) := f'^{b'}(\beta)$ for all $\beta \in \mathcal{G}'^{b'}$. That f' is consistently defined on positions in the round after b' , follows by construction of \mathcal{G}' and in particular by the definition of basic positions.

If Π wins p immediate after b , $\emptyset \in P_{\Pi}^{\mathcal{G}}(b)$, so that $Z[\emptyset] = \emptyset \in P_{\Pi'}^{\mathcal{G}'}(b')$ and Π' wins in p' immediately after b' .

It remains to treat the situation, where p is infinite and thus where p' is infinite. As the basic positions in p and p' are stepwise related via Z , the largest priority occurring infinitely often in p are of parity Π if and only if the largest priority occurring infinitely often in p' are of parity Π' . Thus if Π wins p , then Π' wins p' . \square

Chapter 10

Complementation of Coalgebra

Automata

Coalgebra automata were introduced by Venema in [Ven04] as an approach to adding fix-point operators to Moss' coalgebraic logic. The idea is manifested in a correspondence between coalgebra automata and coalgebraic logic. The (initial) states of coalgebra automata \mathbb{A} correspond to formulas ϕ , such that \mathbb{A} accepts precisely those pointed coalgebras $\mathbb{S} = \langle S, \sigma, s_I \rangle$ satisfying ϕ at s .

In previous work, Kupke and Venema [KV05, KV08] have shown that coalgebra automata are closed under union, intersection, existential and universal projecting corresponding to disjunction, conjunction, existential and universal quantification, respectively. Then Venema and the author [KV09] have established that coalgebra automata are closed under complementation, corresponding to negation on the logical side. The latter constitutes the main contribution of this chapter.

Assumption 10.0.4. *In this chapter we assume $T : \mathbf{Set} \rightarrow \mathbf{Set}$ to be a standard, weak-pullback preserving functor which restricts to the category of finite sets.*

10.1 A Review of Nondeterministic Coalgebra Automata

A well-studied form of automata are nondeterministic automata. Nondeterministic and deterministic word automata are equally expressive. In fact the one can effectively be

translated into the other, as shown for instance in [Saf88]. Kupke and Venema [KV08] have proved a similar result for coalgebra automata.

Definition 10.1.1 (Nondeterministic Coalgebra Automata). *Let T be a Set-functor, a non-deterministic T -coalgebra automaton is a tuple $\langle Q, \theta, q_I, \Omega \rangle$ consisting of*

- *a finite set Q of automaton states,*
- *a transition function $\theta : Q \rightarrow \mathcal{P}TQ$,*
- *an automaton state $q_I \in Q$ distinguished as initial, and*
- *a priority function $\Omega : Q \rightarrow \mathbb{N}$.*

The transition function $\theta : Q \rightarrow \mathcal{P}TQ$ takes automaton states $q \in Q$ to sets $\theta(q) \subseteq TQ$ of T -structured automaton states. The automaton is to choose from each subset nondeterministically.

The subsets of TQ form the carrier of the free (disjunctive) semi-lattice over TQ . This observation gives rise to the equivalent definition of nondeterministic coalgebra automata with a transition function of type $Q \rightarrow \text{SLat}^\vee T_\nabla Q$. We call such automata coalgebra automata *in logical form*.

The acceptance behaviour of coalgebra automata is given in terms of parity graph games parameterised in a T -coalgebra automaton and a T -coalgebra to be recognised.

Definition 10.1.2 (Acceptance Games for Nondeterministic Coalgebra Automata). *Let $\mathbb{A} = \langle Q, \theta, q_I, \Omega \rangle$ be a T -coalgebra automaton and let $\mathbb{S} = \langle S, \sigma, s_I \rangle$ a T -coalgebra. The acceptance game $\mathcal{G}(\mathbb{A}, \mathbb{S})$ is the parity graph game $\langle V_\exists, V_\forall, E, v_I, \Omega_{\mathcal{G}} \rangle$ defined as in Table 10.1.*

Note that we may assign positions from $Q \times S$ to either player as there is only one choice. Only if we assign a player, the game graph is bipartite and the acceptance game is well-defined as a two-player graph game. However, in later arguments it will be handy to leave the assignment ambiguous.

Because there is only one choice in positions from $Q \times S$, we may easily merge the first two lines and make the admissible moves of $(\theta(q), s)$ the admissible moves of (q, s) .

Position	Player	Set of Admissible Moves	$\Omega_{\mathcal{G}}$
$(q, s) \in Q \times S$	-	$\{(\theta(q), s)\}$	$\Omega(q)$
$(\bigvee \tau, s) \in SLat^{\vee} T_{\nabla} Q \times S$	\exists	$\{(a, s) \mid a \in \tau\}$	0
$(\nabla \alpha, s) \in T_{\nabla} Q \times S$	\exists	$\{Z \subseteq Q \times S \mid (\alpha, \sigma(s)) \in Rel_T(Z)\}$	0
$Z \subseteq Q \times S$	\forall	Z	0

Table 10.1: Acceptance Games for Nondeterministic Coalgebra Automata

The presentation of the acceptance game as in Table 10.1 is to facilitate the presentation of the acceptance game in logical form.

Definition 10.1.3 (Languages and Equivalence of Coalgebra Automata). *We call the set of coalgebras accepted by a coalgebra automaton \mathbb{A} the language of \mathbb{A} . Two coalgebra automata with the same language are called equivalent.*

10.2 Alternation

Nondeterminism means that \exists makes a choice before the transition step. This type of choice can be logically completed giving a choice to \forall before the transition step. The so obtained type of automata are *alternating*. Similarly, we may want to add a choice for \exists and \forall after the transition step. We call such automata *transalternating*. For our argument, we need an intermediate form, called *semi-transalternating* automata. In addition to the alternating choice before the transition step, only \forall can choose after the transition step.

All of these kinds of coalgebra automata share a common structure, $\langle Q, \theta, q_I, \Omega \rangle$, where only the branching type of θ may vary. Table 10.2 summarises the branching types of transition functions, we consider.

Name	Type of Transition Function
Deterministic	$Q \rightarrow T_{\nabla} Q$
Nondeterministic	$Q \rightarrow SLat^{\vee} T_{\nabla} Q$
Alternating	$Q \rightarrow Lat T_{\nabla} Q$
Semi-Transalternating	$Q \rightarrow Lat T_{\nabla} SLat^{\wedge} Q$
Transalternating	$Q \rightarrow Lat T_{\nabla} Lat Q$

Table 10.2: Branching Types of Coalgebra Automata

where

Position	Player	Set of Admissible Moves	$\Omega_{\mathcal{G}}$
$(q, s) \in Q \times S$	-	$\{(\theta(q), s)\}$	$\Omega(q)$
$(\bigwedge \tau, s) \in LatT_{\forall}Q \times S$	\forall	$\{(a, s) \mid a \in \tau\}$	0
$(\bigvee \tau, s) \in LatT_{\exists}Q \times S$	\exists	$\{(a, s) \mid a \in \tau\}$	0
$(\nabla \alpha, s) \in T_{\forall}Q \times S$	\exists	$\{Z \subseteq Q \times S \mid (\alpha, \sigma(s)) \in Rel_T(Z)\}$	0
$Z \subseteq Q \times S$	\forall	Z	0

Table 10.3: Acceptance Games for Alternating Automata

- $SLat^{\vee}$ ($SLat^{\wedge}$) is the *Set*-functor assigning to a set X the carrier of the free disjunctive (conjunctive) semi-lattice generated over X , and
- Lat is the *Set*-functor assigning to a set X the carrier of the free lattice generated over X .

10.2.1 Alternating Coalgebra Automata

Definition 10.2.1 (Alternating Coalgebra Automata). *An alternating coalgebra automaton is a structure $\mathbb{A} = \langle Q, \theta, q_I, \Omega \rangle$ where*

- Q is the finite set of states,
- $\theta : Q \rightarrow LatTQ$ is the transition function,
- $q_I \in Q$ is the state distinguished as initial, and
- $\Omega : Q \rightarrow \mathbb{N}$ is the priority function.

Definition 10.2.2 (Acceptance Games of Alternating Coalgebra Automata). *The notion of an alternating T -automata accepting a pointed T -coalgebra $\mathbb{S} = \langle S, \sigma, s_I \rangle$ is defined in terms of the parity graph game $\mathcal{G}(\mathbb{A}, \mathbb{S})$ given by Table 10.5 \mathbb{A} accepts a pointed coalgebra \mathbb{S} if and only if \exists has a winning strategy in the acceptance game $\mathcal{G}(\mathbb{A}, \mathbb{S})$.*

10.2.2 Semi-Transalternating Coalgebra Automata

Definition 10.2.3 (Semi-Transalternating Automata). *A semi-transalternating T -automaton $\mathbb{A} = \langle Q, \theta, q_I, \Omega \rangle$ consists of*

Position	Player	Set of Admissible Moves	$\Omega_{\mathcal{G}}$
$(q, s) \in Q \times S$	-	$\{(\theta(q), s)\}$	$\Omega(q)$
$(\bigwedge_{i \in I} a_i, s) \in Lat_1 Q \times S$	\forall	$\{(a_i, s) \mid i \in I\}$	0
$(\bigvee_{i \in I} a_i, s) \in Lat_1 Q \times S$	\exists	$\{(a_i, s) \mid i \in I\}$	0
$(\nabla \alpha, s) \in T_{\nabla} Lat Q \times S$	\exists	$\{Z \subseteq Lat Q \times S \mid (\alpha, \sigma(s)) \in Rel_T(Z)\}$	0
$Z \subseteq Lat Q \times S$	\forall	Z	0
$(\bigwedge_{i \in I} a_i, s) \in SLat^{\wedge} Q \times S$	\forall	$\{(a_i, s) \mid i \in I\}$	0

Table 10.4: Acceptance Games for Semi-Transalternating Automata

- a finite set Q of states,
- a transition function $\theta : Q \rightarrow Lat T_{\nabla} SLat^{\wedge} Q$,
- a state $q_I \in Q$ distinguished as initial, and
- a priority function $\Omega : Q \rightarrow \mathbb{N}$.

Definition 10.2.4 (Acceptance Games for Semi-Transalternating Automata). Let $\mathbb{A} = \langle Q, q_I, \theta, \Omega \rangle$ be a semi-transalternating automaton and let $\mathbb{S} = \langle S, \sigma, s_I \rangle$ be a T -coalgebra. The acceptance game $\mathcal{G}(\mathbb{A}, \mathbb{S})$ is the parity graph game $\langle V_{\exists}, V_{\forall}, E, v_I, \Omega_{\mathcal{G}} \rangle$ with $v_I = (q_I, s_I)$ and V_{\exists} , V_{\forall} , E and $\Omega_{\mathcal{G}}$ given by Table 10.5.

10.2.3 Transalternating Coalgebra Automata

Definition 10.2.5 (Transalternating Automata). A transalternating T -automaton $\mathbb{A} = \langle Q, \theta, q_I, \Omega \rangle$ consists of

- a finite set Q of states,
- a transition function $\theta : Q \rightarrow Lat_1 Q$,
- a state $q_I \in Q$ distinguished as initial, and
- a priority function $\Omega : Q \rightarrow \mathbb{N}$.

Definition 10.2.6 (Acceptance Games for Transalternating Automata). Let $\mathbb{A} = \langle Q, q_I, \theta, \Omega \rangle$ be a transalternating automaton and let $\mathbb{S} = \langle S, \sigma, s_I \rangle$ be a pointed T -coalgebra. The

Position	Player	Set of Admissible Moves	$\Omega_{\mathcal{G}}$
$(q, s) \in Q \times S$	-	$\{(\theta(q), s)\}$	$\Omega(q)$
$(\bigwedge_{i \in I} a_i, s) \in Lat_1 Q \times S$	\forall	$\{(a_i, s) \mid i \in I\}$	0
$(\bigvee_{i \in I} a_i, s) \in Lat_1 Q \times S$	\exists	$\{(a_i, s) \mid i \in I\}$	0
$(\nabla \alpha, s) \in T_{\nabla} Lat Q \times S$	\exists	$\{Z \subseteq Lat Q \times S \mid (\alpha, \sigma(s)) \in Rel_T(Z)\}$	0
$Z \subseteq Lat Q \times S$	\forall	Z	0
$(\bigwedge_{i \in I} a_i, s) \in Lat Q \times S$	\forall	$\{(a_i, s) \mid i \in I\}$	0
$(\bigvee_{i \in I} a_i, s) \in Lat Q \times S$	\exists	$\{(a_i, s) \mid i \in I\}$	0

Table 10.5: Acceptance Games for Transalternating Automata

acceptance game $\mathcal{G}(\mathbb{A}, \mathbb{S})$ is the parity graph game $\mathcal{G}(\mathbb{A}, \mathbb{S}) = \langle V_{\exists}, V_{\forall}, E, v_I, \Omega_{\mathcal{G}} \rangle$ with $v_I = (q_I, s_I)$ and $V_{\exists}, V_{\forall}, E$ and $\Omega_{\mathcal{G}}$ given by Table 10.5.

10.2.4 Basic Positions in Acceptance Games

In the acceptance games for coalgebra automata of any of the above branching types we can distinguish certain positions as basic.

Definition 10.2.7 (Basic Positions in Acceptance Games). *Let $\mathbb{A} = \langle Q, \tau, q_I, \Omega \rangle$ be a T -coalgebra automaton and $\mathbb{S} = \langle S, \sigma, s_I \rangle$ be a pointed coalgebra recognised by \mathbb{A} . In the acceptance game $\mathcal{G}(\mathbb{A}, \mathbb{S})$, positions from $Q \times S$ are called basic.*

Basic positions allow us to dissect plays of acceptance games into rounds. The structure of these rounds is indicated in Tables 10.1 and 10.3. The dissection facilitates coinductive proofs on the semantics of coalgebra automata. We make use of the technique in particular to prove the soundness of the complementation algorithm in Section 10.4. In Chapter 9 we have introduced basic sets of positions in the more general context of parity graph games.

10.3 Equivalence of Coalgebra Automata of various Branching Types

The classes of coalgebra automata of the previously mentioned branching types are equivalent, and in fact effectively mutually undefinable.

Embedding automata into a class of automata with richer structure, such as

$$\begin{aligned} & \text{Nondeterministic} \rightsquigarrow \text{Alternating} \rightsquigarrow \\ & \text{Semi-Transalternating} \rightsquigarrow \text{Transalternating}, \end{aligned}$$

obviously preserves their semantics. The equivalence between nondeterministic and alternating automata, and in particular

$$\text{Alternating} \rightsquigarrow \text{Nondeterministic}$$

has been established by Kupke and Venema in [KV05]. In [KV09] Venema and the author have given effective translations

$$\text{Transalternating} \rightsquigarrow \text{Semi-Transalternating} \rightsquigarrow \text{Alternating}.$$

We repeat the argument in the remainder of this section.

10.3.1 From Transalternating to Semi-Transalternating Automata

Proposition 10.3.1. *There is an effective algorithm transforming a given transalternating automaton into an equivalent semi-transalternating one.*

Definition 10.3.2. *Let $\mathbb{A} = \langle Q, \theta : Q \rightarrow \text{Lat} T \text{Lat} Q, q_I, \Omega \rangle$ be a transalternating automaton, we define a semi-transalternating automaton $\mathbb{A}^\circ = \langle Q, \theta^\circ, q_I, \Omega \rangle$ such that $\theta^\circ(a) := f \circ \theta$ where $f : \text{Lat}_1 Q \rightarrow \text{Lat } T_\nabla \text{SLat}^\wedge Q$ is the map in Lemma 7.2.2.*

Then the following is a corollary of Definition 10.3.2 and Lemma 7.2.2.

Corollary 10.3.3. *\mathbb{A} and \mathbb{A}° are equivalent, for any transalternating automaton \mathbb{A} .*

10.3.2 From Semi-Transalternating to Alternating Automata

In this section we prove, that semi-transalternating and alternating automata are equally expressive. In particular we prove an effective transformation of the first into the second as in the following theorem.

Theorem 10.3.4. *There is an effective algorithm transforming a given semi-transalternating automaton into an equivalent alternating one.*

The transition function of the semi-transalternating automaton \mathbb{A} is a function $\theta : Q \rightarrow \text{Lat } T_{\nabla} \text{SLat}^{\wedge} Q$, whereas the transition function of the alternating automaton \mathbb{A}° is a function $\theta^{\circ} : Q^{\circ} \rightarrow \text{Lat } T_{\nabla} Q^{\circ}$. The idea underlying the transformation is to push the transalternation choice over to the next transition step.

Figure 10.3.2 illustrates the idea on a play of the acceptance games $\mathcal{G}(\mathbb{A}, \mathbb{S})$ and $\mathcal{G}(\mathbb{A}^{\circ}, \mathbb{S})$. The intervals on the left and right demark rounds of $\mathcal{G}(\mathbb{A}, \mathbb{S})$ and $\mathcal{G}(\mathbb{A}^{\circ}, \mathbb{S})$ respectively.

In a first attempt, one may make $\bigwedge_k q'_k$, $\bigwedge_k q''_k$, and so forth states of \mathbb{A}° , but then we can not consistently assign priorities to $\bigwedge_{k \in K} q'_k$ knowing $\{q'_k \mid k \in K\}$ alone. If we assign the priority of the state seen before, we preserve the semantics of \mathbb{A} . The following definition contains these two ideas.

Definition 10.3.5. Let $\mathbb{A} = \langle Q, \theta, q_I, \Omega \rangle$ be a semi-transalternating automaton. We define the alternating automaton $\mathbb{A}^{\circ} = \langle Q^{\circ}, \theta^{\circ}, q_I^{\circ}, \Omega^{\circ} \rangle$ by putting

- $Q^{\circ} := Q \times \text{SLat}^{\wedge} Q$,
- $q_I^{\circ} := (q_I, q_I)$,
- $\Omega^{\circ}(a, b) := \Omega(a)$, and
- $\theta^{\circ} : (Q \times \text{SLat}^{\wedge} Q) \rightarrow \text{Lat } T_{\nabla}(Q \times \text{SLat}^{\wedge} Q)$ is given by

$$\theta^{\circ}(q, a) := \bigwedge_{p \geq a} \text{Lat } T_{\nabla} \kappa_p(\theta(p)). \quad (10.1)$$

For any $q \in Q$, $\kappa_q : \text{SLat}^{\wedge} Q \rightarrow (Q \times \text{SLat}^{\wedge} Q)$ is the map defined by $\kappa_q(a) := (q, a)$ for all $a \in \text{SLat}^{\wedge} Q$.

Proposition 10.3.6. Let \mathbb{A} be a semi-transalternating automaton, then \mathbb{A} and \mathbb{A}° are equivalent.

Proof. Instead of proving $\mathcal{G}(\mathbb{A}, \mathbb{S})$ and $\mathcal{G}(\mathbb{A}^{\circ}, \mathbb{S})$ equivalent directly, we use Proposition 9.2.2 and prove $T^{\mathcal{G}(\mathbb{A}, \mathbb{S})}$ and $T^{\mathcal{G}(\mathbb{A}^{\circ}, \mathbb{S})}$ equivalent.

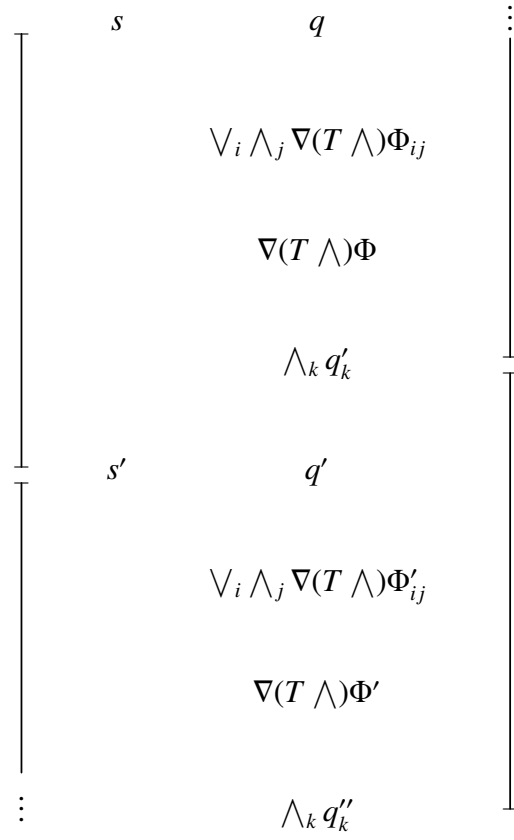


Figure 10.1: Transformation of Semi-Transalternating into Alternating Automata

Every strategy f of a player Π in $T^{\mathcal{G}(\mathbb{A}, \mathbb{S})}$ extends uniquely to a strategy f° in $T^{\mathcal{G}(\mathbb{A}^\circ, \mathbb{S})}$ by putting $f^\circ((v_I), \bigwedge \{(v_I)\}) := (v_I)$, and conversely every strategy f° of Π in $T^{\mathcal{G}(\mathbb{A}^\circ, \mathbb{S})}$ restricts to a strategy in $T^{\mathcal{G}(\mathbb{A}, \mathbb{S})}$ such that every play p consistent with f corresponds to a play p° consistent with f° such that $p^\circ = (\bigwedge \{v_I^T\}).p$. Then p° is finite if and only if p is finite. If p° is finite, the last positions of p° and p belong to the same player. If p° is infinite, the largest priorities occurring infinitely often in p° and p are the same and thus have the same parity. Thus player Π wins p° if and only if she wins p , and f° is winning for Π if and only if f is, and $T^{\mathcal{G}(\mathbb{A}, \mathbb{S})}$ and $T^{\mathcal{G}(\mathbb{A}^\circ, \mathbb{S})}$ are equivalent. \square

10.4 Closure under Complementation

Theorem 10.4.1 (Complementation Lemma for Coalgebra Automata). *The class of alternating coalgebra automata is closed under taking complements.*

We prove the above theorem by devising an algorithm complementing transalternating automata and using that alternating and transalternating automata are effectively mutually translatable as in Section 10.3.

10.4.1 Complementation of Transalternating Coalgebra Automata

We treat states of transalternating coalgebra automata as formulas, thus the transition $\theta(q)$ from an automaton state q is a depth one formula $\theta(q) \in \text{Lat}_1 Q$ over Q . Our definition of complementing automata uses the step-wise dualisation which we introduced in Definition 7.4.4.

Definition 10.4.2 (Complements of Transalternating Automata). *The complement of a transalternating T -coalgebra automaton $\mathbb{A} = \langle Q, \theta, q_I, \Omega \rangle$ is the transalternating automaton $\mathbb{A}^c = \langle Q, \theta^c, q_I, \Omega^c \rangle$ defined with $\theta^c(q) := \delta_1(\theta(q))$ and $\Omega^c(q) := \Omega(q) + 1$, for all $q \in Q$.*

Lemma 10.4.3. *Let \mathbb{A} be a transalternating T -automaton, and \mathbb{S} a T -coalgebra. Given a*

basic position (q, s) in $\mathcal{G}(\mathbb{A}, \mathbb{S})$, and a relation $Z \subseteq Q \times S$, we have:

$$Z \in [P_{\exists}(q, s)] \iff V_Z, \sigma(s) \Vdash_1 \theta(q)$$

$$Z \in [P_{\forall}(q, s)] \iff V_Z, \sigma(s) \Vdash_1 \delta_1(\theta(q))$$

The following proposition shows, that \mathbb{A}^c is indeed the complement of \mathbb{A} .

Proposition 10.4.4. *For every transalternating T -coalgebra automaton \mathbb{A} , the automaton \mathbb{A}^c accepts precisely those pointed T -coalgebras that are rejected by \mathbb{A} .*

Proof. Clearly it suffices to prove for a given T -coalgebra \mathbb{S} , state q of \mathbb{A} , and point s in \mathbb{S} , that

$$(q, s) \in \text{Win}\exists(\mathcal{G}(\mathbb{A}^c, \mathbb{S})) \text{ if and only if } (q, s) \in \text{Win}\forall(\mathcal{G}(\mathbb{A}, \mathbb{S})). \quad (10.2)$$

We prove the above by means of a game bisimulation.

First we note that $Q \times S$ is a basic set in both acceptance games, $\mathcal{G}(\mathbb{A}, \mathbb{S})$ and $\mathcal{G}(\mathbb{A}^c, \mathbb{S})$. The main observation is that the diagonal relation $Id_{Q \times S} := \{(q, s), (q, s) \mid q \in Q, s \in S\}$ is an \forall, \exists -game bisimulation between $\mathcal{G}(\mathbb{A}, \mathbb{S})$ and $\mathcal{G}(\mathbb{A}^c, \mathbb{S})$. Since it is immediate from the definitions that the diagonal relation satisfies the *priority conditions*, it is left to check the structural conditions.

Leaving the other three conditions for the reader, we establish the condition (\exists, forth) as an immediate consequence of the following claim:

$$\text{for all } Z \in P_{\exists}^{\mathcal{G}}(q, s) \text{ there is a } Z_{\forall} \subseteq Z \text{ such that } Z_{\forall} \in P_{\forall}^{\mathcal{G}^c}(q, s), \quad (10.3)$$

which can be proved via the following chain of implications:

$$\begin{aligned}
Z &\in P_{\exists}^{\mathcal{G}}(q, s) \\
&\Rightarrow V_Z, \sigma(s) \Vdash_1 \theta(q) && \text{(Lemma 10.4.3)} \\
&\Rightarrow (V_Z)^c, \sigma(s) \not\Vdash_1 \delta_1(\theta(q)) && \text{(Corollary 7.4.6)} \\
&\Rightarrow (V_Z)^c, \sigma(s) \not\Vdash_1 \theta^c(q) && \text{(Definition of } \theta^c) \\
&\Rightarrow V_{Z^c}, \sigma(s) \not\Vdash_1 \theta^c(q) && (\dagger) \\
&\Rightarrow Z^c \notin [P_{\exists}^{\mathcal{G}}(q, s)] && \text{(Lemma 10.4.3)} \\
&\Rightarrow \text{there is a } Z_{\forall} \in P_{\forall}^{\mathcal{G}^c}(q, s) \text{ with } Z^c \cap Z_{\forall} = \emptyset && \text{(Proposition 9.3.8)} \\
&\Rightarrow \text{there is a } Z_{\forall} \subseteq Z \text{ with } Z_{\forall} \in P_{\forall}^{\mathcal{G}^c}(q, s) && \text{(Elementary set theory)}
\end{aligned}$$

In the above Z^c denotes $(Q \times S) \setminus Z$, and in the implication marked (\dagger) we use the fact that $(V_Z)^c = V_{Z^c}$. \square

We conclude the complementation lemma with an observation about the size of automata resulting from the complementation.

Theorem 10.4.5. *For any alternating T -automaton with n states there is a complementing alternating automaton with at most $n * 2^n$ states.*

We single out the following case, which suitably generalises tree automata, for which significantly lower bounds have been established in [Rab69, Kis07].

Theorem 10.4.6. *Let T be such that for any $\alpha \in T_{\omega}Q$, the formula $\nabla\alpha$ has a dual $\Delta\alpha \in \text{Lat}T_{\nabla}Q$. Then for any alternating T -automaton of n states there is a complementing alternating automaton with at most $n + c$ states, for some constant c .*

Chapter 11

A Pumping Lemma for Regular Languages of Coalgebras in *Set*

Languages of words recognised by state-finite automata are called regular. In order to show that a collection of words is not a regular language, one often [Sip96, HMU03] uses the following pumping lemma for regular languages, which has been introduced in [BHPS61].

Theorem 11.0.7 (Pumping Lemma for Regular Languages of Words). *The collection \mathcal{L} of words accepted by a state-finite automaton, satisfies the following pumping property: There is a natural number l , the pumping length of \mathcal{L} , such that every word in \mathcal{L} longer than l can be decomposed into segments u , v and w , such that*

1. $|v| > 0$,
2. $|uv| \leq l$, and
3. uv^+w is contained in \mathcal{L} ,

where v^+ means any of $\{v, vv, vvv, \dots\}$.

Example 11.0.8. *The pumping lemma is used to disprove regularity of the language $\mathcal{L} = \{a^n b^n \mid n \in \mathbb{N}\}$ of words consisting of a substring of a 's followed by a substring of b 's of the same length. The idea underlying the proof is informally the following. Any dissection*

of a word $a^n b^n$ into uvw as in Theorem 11.0.7 must be such that v lies completely in a^n or b^n . Otherwise pumping v yields a substring ba and thus a word not contained in the language \mathcal{L} . However pumping v in a^n or b^n yields words $a^{n+|v|}b^n$ or $a^n b^{n+|v|}$, not contained in the language \mathcal{L} either.

Any word $a^n b^n$ is a coalgebra $\mathbb{S}_n = \langle S, \sigma, s_I \rangle$ of transition type $T(-) = \{\sqrt{}\} + \{a, b\} \times (-)$ with

- $S := \{a^k b^n \mid k \leq n\} \cup \{b^k \mid k \leq n\}$,
- $\sigma(a^{k+1} b^n) := (a, a^k b^n)$ and $\sigma(b^{k+1}) := (b, b^k)$ and $\sigma(\epsilon) := \sqrt{}$, and
- $s_I := a^n b^n$.

We will revisit this example in later sections of this chapter.

In this chapter, we prove a pumping lemma for languages of state-finite T -coalgebra automata. Before we can state a theorem analogous to Theorem 11.0.7 above, we need to define analogues of pumping length and the pumping operation for coalgebras. We only consider coalgebras in *Set*. We argue that these coalgebras may be regarded as graphs with additional structure, and define pumping along paths through these graphs. Thereby we effectively reduce the pumping lemma for coalgebras in *Set* to the pumping lemma for words.

Assumption 11.0.9. *Below we assume $T : \text{Set} \rightarrow \text{Set}$*

1. *to be standard and*
2. *finitary, and*
3. *to preserve weak pullbacks.*

In order to make sense of the pumping lemma we define languages of coalgebras.

Definition 11.0.10 (Languages of Coalgebras). *Let T be a functor on a category \mathcal{C} . A language \mathcal{L} of T -coalgebras is a set of equivalence classes of pointed T -coalgebras with respect to T -bisimilarity¹.*

¹See Definition C.0.25.

11.1 Coalgebras in *Set* as Graphs

Recall that a T -coalgebra $\mathbb{S} = \langle S, \sigma, s_I \rangle$ in *Set* has a transition function $\sigma : S \rightarrow TS$. An element $\phi \in TS$ can be thought of as a T -term that “uses” a subset $Y \subseteq S$. The latter is made formal by the notion of *Base* as in Definition B.4.13. We think of $Base(\sigma(s))$ as the set of successor states of σ . For the base to be well-defined we need T to be standard as in Assumption 11.0.9.

Composing σ with *Base* yields a graph, that is a function $Base \circ \sigma : S \rightarrow \mathcal{P}S$. Unless T restricts to finite sets, $Base \circ \sigma$ may have an infinite out-degree.

Definition 11.1.1 (Graphs of Coalgebras). *Let $\mathbb{S} = \langle S, \sigma, s_I \rangle$ be a T -coalgebra, we call $Base \circ \sigma$ the graph of \mathbb{S} .*

Example 11.1.2. *In Example 11.0.8, the graph structure arises as follows.*

1. $Base(\sigma(a^{k+1}b^n)) = Base(a, a^k b^n) = \{a^k b^n\}$
2. $Base(\sigma(b^{k+1})) = Base(b, b^k) = \{b^k\}$
3. $Base(\sigma(b^0)) = Base(\surd) = \emptyset$

The intuition of coalgebras as graphs provokes the definition of concepts that are common for graphs, such as

1. reachable states,
2. subcoalgebras rooted in a state,
3. generated subcoalgebras, and
4. the depth of a coalgebra.

We explore these concepts towards a definition of pumping.

11.1.1 Reachable States

The set $Reach_{\mathbb{S}}(s) \subseteq S$ of states reachable from a state $s \in S$ in a coalgebra $\mathbb{S} = \langle S, \sigma, s_I \rangle$, can be computed by closing $Base(\sigma(s))$ under $Base \circ \sigma$ as in the following definition.

Definition 11.1.3 (Reachable States). *Let $\mathbb{S} = \langle S, \sigma, s_I \rangle$ be a T -coalgebra and $s \in S$ a state of \mathbb{S} , we define the set $\text{Reach}_{\mathbb{S}}(s)$ of states reachable from s in σ as the least set $Y \subseteq S$ which contains $\text{Base}(\sigma(s))$ and is closed under union with $\text{Base}(\sigma(y))$ for all $y \in Y$.*

Example 11.1.4. *In Example 11.0.8 we obtain a definition of $\text{Reach}_{\mathbb{S}}$ from the graph structure in Example 11.1.2.*

$$1. \text{Reach}_{\mathbb{S}}(a^{k+1}b^n) = \{a^l b^n \mid l \leq k\} \cup \{b^l \mid l \leq n\}$$

$$2. \text{Reach}_{\mathbb{S}}(b^{k+1}) = \{b^l \mid l \leq k\}$$

$$3. \text{Reach}_{\mathbb{S}}(b^0) = \text{Base}(\surd) = \emptyset$$

Lemma 11.1.5. *For all standard functors T , T -coalgebras $\mathbb{S} = \langle S, \sigma \rangle$ and states $s \in S$, the set $\text{Reach}_{\mathbb{S}}(s)$ is well-defined.*

Proof. The proof is an application of the Knaster-Tarski theorem, observing that $\mathcal{P}S$ forms a complete lattice and $Y \mapsto Y \cup \text{Base}[\sigma[Y]]$ is a monotone function on $\mathcal{P}S$. \square

As $\text{Reach}_{\mathbb{S}}(s)$ is a fix point, it is also a pre-fix point. This observation is important as it allows us to define generated subcoalgebras.

Lemma 11.1.6. *$\text{Reach}_{\mathbb{S}}(s)$ is a pre-fix point of σ in the following sense.*

$$\sigma[\text{Reach}_{\mathbb{S}}(s)] \subseteq T\text{Reach}_{\mathbb{S}}(s)$$

Proof. By definition, $\text{Base}(\sigma[\text{Reach}_{\mathbb{S}}(s)]) \subseteq \text{Reach}_{\mathbb{S}}(s)$, so that by standardness of T , $T\text{Base}(\sigma[\text{Reach}_{\mathbb{S}}(s)]) \subseteq T\text{Reach}_{\mathbb{S}}(s)$, and using the definition of Base , $\sigma[\text{Reach}_{\mathbb{S}}(s)] \subseteq T\text{Reach}_{\mathbb{S}}(s)$. \square

Reachability allows us to easily discriminate cyclic from acyclic coalgebras.

Definition 11.1.7 (Cyclic and Acyclic Coalgebras). *A coalgebra $\mathbb{S} = \langle S, \sigma \rangle$ is cyclic if there is a state $s \in S$ such that $s \in \text{Reach}_{\mathbb{S}}(s)$, and acyclic otherwise.*

Below we will need the following property, which informally means that the coalgebra is nowhere confluent.

Definition 11.1.8 (Tree-likeness). *A coalgebra $\mathbb{S} = \langle S, \sigma, s_I \rangle$ is tree-like if for all $s \in S$ and $s_0, s_1 \in \text{Reach}_{\mathbb{S}}(s)$ with $s_0 \neq s_1$, $s_0 \notin \text{Reach}_{\mathbb{S}}(s_1)$ and $s_1 \notin \text{Reach}_{\mathbb{S}}(s_0)$, it is $\text{Reach}_{\mathbb{S}}(s_0) \cap \text{Reach}_{\mathbb{S}}(s_1) = \emptyset$*

Below we will show that for every pointed coalgebra there is a bisimilar one, which is acyclic and tree-like. We will make these properties assumptions for our pumping lemma.

11.1.2 Generated Subcoalgebras

We define generated subcoalgebras analogous to generated subgraphs using the notion of reachability defined above.

Definition 11.1.9 (Least Subcoalgebra Rooted in a State). *Let $\mathbb{S} = \langle S, \sigma \rangle$ be a T -coalgebra for a standard functor T and let $s \in S$ be a point of \mathbb{S} . We define the least subcoalgebra rooted in s as the restriction of σ to $\text{Reach}_{\mathbb{S}}(s)$.*

$$\mathbb{S}_s := \langle \text{Reach}_{\mathbb{S}}(s), \sigma|_{\text{Reach}_{\mathbb{S}}(s)} \rangle \quad (11.1)$$

The definition above is well-stated. The following is a corollary of Lemma 11.1.6.

Corollary 11.1.10. *For any T -coalgebra \mathbb{S} and any state s of \mathbb{S} as above, \mathbb{S}_s exists and is a T -coalgebra.*

Proposition 11.1.11. *For any T -coalgebra \mathbb{S} and any state s of \mathbb{S} as above, \mathbb{S} and \mathbb{S}_s are bisimilar.*

Proof. For the bisimulation relation, choose the diagonal relation $\Delta_{\text{Reach}_{\mathbb{S}}(s)} := \{(s', s') \mid s' \in \text{Reach}_{\mathbb{S}}(s)\}$, then as $\text{Reach}_{\mathbb{S}}(s)$ is closed under $\text{Base} \circ \sigma$, all pairs of states in $\Delta_{\text{Reach}_{\mathbb{S}}(s)}$ are bisimilar. \square

11.1.3 Unravelling Coalgebras in Set

The unravelling of a coalgebra is defined similarly to the unravelling of graphs.

Definition 11.1.12 (Unravelling of Set-Coalgebras). *The unravelling of a pointed T -coalgebra $\mathbb{S} = \langle S, \sigma, s_I \rangle$ is defined to be the T -coalgebra $\mathbb{S}^+ = \langle S^+, \sigma^+ : S^+ \rightarrow TS^+, (s_I) \rangle^2$ with the transition function $\sigma^+(w) := T\iota_w \circ \sigma(\text{last}(w))$, where $\iota_w : S \rightarrow S^+$ takes $s \mapsto ws$ for any $w \in S^+$.*

Unravelling a coalgebra preserves its semantics.

Proposition 11.1.13. *For any T -coalgebra $\mathbb{S} = \langle S, \sigma : S \rightarrow TS \rangle$ and state $s \in S$, $(\sigma, s) \cong (\sigma^+, (s))$ ³ are bisimilar.*

Proof. It suffices to show that $Gr(\text{last}) \subseteq S^+ \times S$ is a bisimulation, which follows from the definition of σ^+ and $\bigcup_{w \in S^+} Gr(\iota_w) = (Gr(\text{last}))^{-1}$. \square

We obtain the following as a corollary of Proposition 11.1.13 and Proposition 4.7 of [Ven04], which states that if a coalgebra automaton \mathbb{A} accepts a coalgebra \mathbb{S} , then \mathbb{A} accepts any coalgebra bisimilar to \mathbb{S} .

Corollary 11.1.14. *A coalgebra automaton \mathbb{A} accepts a coalgebra \mathbb{S} if and only if \mathbb{A} accepts \mathbb{S} 's unravelling \mathbb{S}^+ .*

Unravelling a coalgebra yields an acyclic tree-like coalgebra. The latter properties are important for the proof of the pumping lemma in the next section.

Lemma 11.1.15. *Let $\mathbb{S} = \langle S, \sigma, s \rangle$ be a coalgebra and let \mathbb{S}^+ be the unravelling of \mathbb{S} . $w \in S^+$. Then for any $w, w' \in S^+$, $w' \in \text{Reach}_{\mathbb{S}^+}(w)$ if and only if w is a proper prefix of w' .*

Proof. We prove the lemma by structural induction over the definition of Reach . In the base case,

$$\begin{aligned} \text{Base}(\sigma^+(w)) &= && \text{by definition of } \sigma^+ \\ \text{Base}((T\kappa_w) \circ \sigma(\text{last}(w))) &= && \text{by Lemma B.4.16} \\ \kappa_w[\text{Base} \circ \sigma(\text{last}(w))] \end{aligned}$$

² S^+ is the set of non-empty words over S .

³ (s) denotes the one-character word consisting of s .

The induction step follows from the transitivity of being a proper prefix. \square

The following is a corollary of the previous lemma.

Corollary 11.1.16. *Let $\mathbb{S} = \langle S, \sigma \rangle$ be a T -coalgebra, the unravelling \mathbb{S}^+ of \mathbb{S} is acyclic and tree-like.*

The following is a corollary of Corollary 11.1.16.

Corollary 11.1.17. *Every T -coalgebra is bisimilar to an acyclic tree-like T -coalgebra.*

11.1.4 Pumping Length for Coalgebras in Set

The unravelling of coalgebras induces a natural notion of depth. Since we are looking to discriminate coalgebra states by finite depth, it suffices to approximate the depth from below as in the following definition.

Definition 11.1.18 (Depth of Coalgebra States). *Let $\mathbb{S} = \langle S, \sigma \rangle$ be a T -coalgebra, then the depth of states in \mathbb{S} can be approximated from below as follows.*

- Every state $s \in S$ is of depth at least 0.
- Every state $s \in S$ is of depth at least $n + 1$, if there is a state $s' \in \text{Base}(\sigma(s))$ of depth at least n .

If a state s is of depth at least n for every finite n , we say s is of infinite depth.

Example 11.1.19. *In Example 11.0.8, the depth of a state $a^k b^l$ for $(k, l) \in \{(k, n), (0, l) \mid k, l \leq n\}$ is $k + l$.*

Loops provide an immediate example of states of infinite depth.

Proposition 11.1.20. *Let $\mathbb{S} = \langle S, \sigma \rangle$ be a cyclic T -coalgebra with $s \in \text{Reach}_{\mathbb{S}}(s)$ for some state $s \in S$, then s is of infinite depth.*

Proof. That s is reachable from itself, means that there is a finite sequence s_0, \dots, s_n of length greater than 1 consisting of states with $s = s_0$, $s = s_n$, and for each $i < n$, $s_{i+1} \in \text{Base}(\sigma(s_i))$. By the definition of depth above, s is of depth at least strictly larger than s .

Using that s is of depth at least 0, it is easy to see that s is of infinite depth. \square

The following lemma tells us that T -bisimilar states have the same depth. Consequentially we can assign coherently a depth to equivalence classes of pointed T -coalgebras under T -bisimilarity, such as in languages of T -coalgebras. The latter appears in the definition of the Pumping Lemma in Theorem 11.3.2.

Lemma 11.1.21. *Let $\mathbb{S} = \langle S, \sigma \rangle$ be a T -coalgebra and $R \subseteq S \times S$ a T -bisimulation on \mathbb{S} . For any pair of states $(s_0, s_1) \in R$ and any $n \in \mathbb{N}$, s_0 is of depth at least n if and only if s_1 is of depth at least n .*

Proof. We prove one direction of the bi-implication, as the other follows through a symmetrical argument. Our proof proceeds inductively over n . In the base case, we show for any pair $(s, s') \in R$ of states, that if $\text{Base}(\sigma(s))$ is non-empty, then so is $\text{Base}(\sigma'(s'))$. Therefor we use the function $r : R \rightarrow TR$ making Diagram (11.2) commute.

$$\begin{array}{ccccc}
 TS & \xleftarrow{T\pi_S} & TR & \xrightarrow{T\pi_{S'}} & TS' \\
 \uparrow \sigma & & \uparrow r & & \uparrow \sigma' \\
 S & \xleftarrow{\pi_S} & R & \xrightarrow{\pi_{S'}} & S'
 \end{array} \tag{11.2}$$

Firstly if $\text{Base}(\sigma(s))$ is non-empty, then so is $\text{Base}((m \circ \rho)(s))$. Since $\text{Base}(\sigma(s)) \subseteq R$ it suffices to prove non-emptiness for the projection to S .

$$\pi_S \circ \text{Base} \circ r = \text{Base} \circ T\pi_S \circ r = \text{Base} \circ \sigma \circ \pi_S \tag{11.3}$$

Then the same steps as above with S' in place of S and σ' in place of σ yield the proof that $\text{Base}(\sigma'(s'))$ is non-empty from the non-emptiness of $\text{Base}(r(s, s'))$.

The induction step proceeds as follows. Combining the previous yields $(\pi_S \times \pi_{S'}) \circ \text{Base} \circ r = \text{Base} \circ \sigma \circ (\pi_S \times \pi_{S'})$. Using that $\text{Base}(r(s, s')) \subseteq R$, we see that for every $t \in \text{Base}(\sigma(s))$ there is a $t' \in \text{Base}(\sigma'(s'))$ with $(t, t') \in R$. \square

11.2 Pumping Coalgebras in Set

In the case of words, the pumping operation takes a word

$$s = u_0 \dots u_l v_0 \dots v_m w_0 \dots w_n \in Act^* \quad (11.4)$$

to the word

$$s' = u_0 \dots u_l v_0 \dots v_m v_0 \dots v_m w_0 \dots w_n \in Act^*. \quad (11.5)$$

The word s can be seen as a coalgebra $\sigma : S \rightarrow \{\sqrt{}\} + Act \times S$ where S consists of all suffixes of s and $\sigma(\emptyset) = \sqrt{}$ and $\sigma(a.t) = (a, t)$ for $a \in Act$ and $t \in S$. Pumping then replaces each suffix $v_i \dots v_m w_0 \dots w_n$ for $0 \leq i \leq m$ with a suffix $v_i \dots v_m v_0 \dots v_m w_0 \dots w_n$, and adapts the transition function accordingly. The following definition makes this algorithm precise in the broader context of coalgebras.

Definition 11.2.1 (Pumping Set-Coalgebras Once). *Let $\mathbb{S} = \langle S, \sigma, s_I \rangle$ be a pointed T -coalgebra. A pumping situation in \mathbb{S} is a pair of states $s_0, s_1 \in S$ such that $s_0 \in \{s_I\} \cup Reach_{\mathbb{S}}(s_I)$ and $s_1 \in Reach_{\mathbb{S}}(s_0)$. Pumping \mathbb{S} once at (s_0, s_1) yields a T -coalgebra $\mathbb{S}' = \langle S', \sigma', s'_I \rangle$ as follows.*

1. Put $\sigma_0 := \sigma$.

$$\longrightarrow s_0 \longrightarrow \dots \longrightarrow s_1 \longrightarrow \quad (11.6)$$

2. Define a function $\sigma_1 : S + Reach(\sigma_0, s_0) \rightarrow T(S + Reach(\sigma_0, s_0))$ joining to σ_0 the least subcoalgebra of σ_0 rooted in s_0 , $\sigma_1 := \sigma_0 + (\sigma_0)_{s_0}$.⁴

$$\longrightarrow \kappa_0 s_0 \longrightarrow \dots \longrightarrow \kappa_0 s_1 \longrightarrow \quad (11.7)$$

$$\kappa_1 s_0 \longrightarrow \dots \longrightarrow \kappa_1 s_1 \longrightarrow$$

3. Define a function $\sigma_2 : S + Reach_{\mathbb{S}}(\sigma, s) \rightarrow T(S + Reach_{\mathbb{S}}(\sigma, s))$ such that $\sigma_2(\kappa_0 s_1) :=$

⁴ $\kappa_0 : S \rightarrow S + Reach(\sigma_0, s_0)$ and $\kappa_1 : Reach(\sigma_0, s_0) \rightarrow S + Reach(\sigma_0, s_0)$ are the usual injection functions for categorical coproducts as in Definition B.2.5.

$\sigma_1(\kappa_1 s_0)$ and $\sigma_2(t) := \sigma_1(t)$ for all $t \in (S + \text{Reach}_{\mathbb{S}}(\sigma, s)) \setminus \{\kappa_0 s_1\}$.

$$\begin{array}{c}
 \longrightarrow \kappa_0 s_0 \longrightarrow \cdots \longrightarrow \kappa_0 s_1 \\
 \searrow \\
 \kappa_1 s_0 \longrightarrow \cdots \longrightarrow \kappa_1 s_1 \longrightarrow
 \end{array} \tag{11.8}$$

Finally, put $S' := S + \text{Reach}(\sigma, s)$, $\sigma' := \sigma_2$, and $s'_I = \kappa_0 s_I$ in $\mathbb{S}' = \langle S', \sigma', s'_I \rangle$.

Remark 11.2.2. The above algorithm is well-defined for standard T as $X \subseteq X + S$ for all sets X .

Example 11.2.3. We demonstrate pumping on ab . There are three ways to dissect ab into uvw according to Theorem 11.0.7 with $v \in \{a, b, ab\}$. Each yields a pumping situation in the corresponding coalgebra \mathbb{S}_1 of Example 11.0.8 as follows.

				Pumping Situation		
	u	v	w	s_0	s_1	$uvvw$
1.	ϵ	a	b	ab	b	aab
2.	a	b	ϵ	b	ϵ	abb
3.	ϵ	ab	ϵ	ab	ϵ	$abab$

Table 11.1: Example of Pumping

In each case pumping yields a T -coalgebra as follows. For the sake of a simple presentation, we display the subcoalgebras $(\mathbb{S}_{|v \in \{a, b, ab\}})_{\kappa_0 ab}$ rooted in the initial state $\kappa_0 ab$.

1. $(\mathbb{S}_a)_{\kappa_0 ab} = \langle \{\kappa_0 ab, \kappa_1 ab, \kappa_1 b, \kappa_1 \epsilon\}, \sigma_a, \kappa_0 ab \rangle$ with $\sigma_a := \{\kappa_0 ab \mapsto (a, \kappa_1 ab), \kappa_1 ab \mapsto (a, \kappa_1 b), \kappa_1 b \mapsto (b, \kappa_1 \epsilon), \kappa_1 \epsilon \mapsto \sqrt{}\}$
2. $(\mathbb{S}_b)_{\kappa_0 ab} = \langle \{\kappa_0 a, \kappa_0 b, \kappa_1 b, \kappa_1 \epsilon\}, \sigma_a, \kappa_0 ab \rangle$ with $\sigma_a := \{\kappa_0 a \mapsto (a, \kappa_0 b), \kappa_0 b \mapsto (b, \kappa_1 b), \kappa_1 b \mapsto (b, \kappa_1 \epsilon), \kappa_1 \epsilon \mapsto \sqrt{}\}$
3. $(\mathbb{S}_{ab})_{\kappa_0 ab} = \langle \{\kappa_0 ab, \kappa_0 b, \kappa_1 ab, \kappa_1 b, \kappa_1 \epsilon\}, \sigma_a, \kappa_0 ab \rangle$ with $\sigma_a := \{\kappa_0 ab \mapsto (a, \kappa_0 b), \kappa_0 b \mapsto (b, \kappa_1 ab), \kappa_1 ab \mapsto (a, \kappa_1 b), \kappa_1 b \mapsto (b, \kappa_1 \epsilon), \kappa_1 \epsilon \mapsto \sqrt{}\}$

The above coalgebras $\mathbb{S}_{|v \in \{a, b, ab\}}$ are respectively bisimilar to the T -coalgebras corresponding to the pumped words $uvvw$ as in Example 11.0.8. These calculations show by example how pumping in coalgebras restricts to pumping in words.

The following proposition shows as a corollary that pumping does not yield bisimilar coalgebras in general.

Proposition 11.2.4. *Let $\mathbb{S} = \langle S, \sigma, s_I \rangle$ be a T -coalgebra and let $\mathbb{S}' = \langle S, \sigma', s'_I \rangle$ be the result of pumping \mathbb{S} at a pumping situation (s_0, s_1) . Then \mathbb{S} and \mathbb{S}' are bisimilar if and only if s and s' are bisimilar in \mathbb{S} .*

Proof. By definition of pumping, it is $s \in \text{Reach}_{\mathbb{S}}(s_I)$. Thus $\mathbb{S} \simeq \mathbb{S}'$ only if $s' \simeq \kappa_1 s'$ which by definition of T -bisimilarity holds if and only if $s' \simeq s$, since

$$\sigma(s') \text{ Rel}_T (\simeq) \sigma'(\kappa_1 s') \text{ Rel}_T (\simeq) (T\kappa_2)\sigma(s) \text{ Rel}_T (\simeq) \sigma(s) \quad (11.9)$$

□

Pumping a T -coalgebra $\mathbb{S} = \langle S, \sigma \rangle$ once at a pumping situation (s, s') yields a coalgebra $\mathbb{S}' = \langle S', \sigma' \rangle$. The same pumping situation (s, s') can then be found as $(\kappa_1 s, \kappa_1 s')$ in \mathbb{S}' . Pumping \mathbb{S}' at $(\kappa_1 s, \kappa_1 s')$ yields a T -coalgebra $\mathbb{S}'' = \langle S'', \sigma'' \rangle$. In \mathbb{S}'' we find the same pumping situation (s, s') as $(\kappa_2 s, \kappa_2 s')$, and so forth. It is easy to see that pumping can be iterated arbitrarily often.

Definition 11.2.5 (Iterated Pumping in Set-Coalgebras). *Let $\mathbb{S} = \langle S, \sigma \rangle$ be a Set-coalgebra and (s_0, s_1) a pumping situation in σ . We define the iterated pumping $\mathbb{S}^{(n)}$ for any $n < \omega$ inductively as follows.*

1. $\mathbb{S}^{(1)}$ is defined as pumping \mathbb{S} once at (s_0, s_1) .
2. Given the n -fold pumping $\mathbb{S}^{(n)} = \langle S^{(n)}, \sigma^{(n)} \rangle$ of \mathbb{S} at (s_0, s_1) , we define the $(n+1)$ -fold pumping $\mathbb{S}^{(n+1)} = \langle S^{(n+1)}, \sigma^{(n+1)} \rangle$ of \mathbb{S} at (s_0, s_1) as pumping $\mathbb{S}^{(n)}$ once at $(\kappa_n s_0, \kappa_n s_1)$.

We call pumping zero times in a T -coalgebra deleting in a T -coalgebra

Definition 11.2.6 (Deleting in a T -Coalgebra). *Let $\mathbb{S} = \langle S, \sigma \rangle$ be a T -coalgebra and (s, s') a pumping situation in \mathbb{S} . Deleting in \mathbb{S} at (s_0, s_1) yields a T -coalgebra $\mathbb{S}' = \langle S, \sigma' \rangle$ such that $\sigma'(s_0) := \sigma(s_1)$ and $\sigma'(t) := \sigma(t)$ for all $t \in S \setminus \{s\}$.*

Figure 11.1: Examples for Deleting and Pumping Once and Iteratedly in *Set*-Coalgebras

$$\dots \quad x \xrightarrow{b} y \xrightarrow{d} z \xrightarrow{\vee} \quad (11.10)$$

$$\text{---} \underset{u}{\text{---}} \text{---} \underset{v}{\text{---}} \text{---} \underset{w}{\text{---}}$$

$$\dots \quad x \xrightarrow{\vee}$$

$$\text{---} \underset{u}{\text{---}} \text{---} \underset{w}{\text{---}}$$

$$\dots \quad x \xrightarrow{b} y \xrightarrow{d} z \xrightarrow{\vee} \quad (11.11)$$

$$\text{---} \underset{u}{\text{---}} \text{---} \underset{v}{\text{---}} \text{---} \underset{w}{\text{---}}$$

$$\dots \quad \kappa_0 x \xrightarrow{b} \kappa_0 y \xrightarrow{d} \kappa_1 x \xrightarrow{b} \kappa_1 y \xrightarrow{d} \kappa_1 z \xrightarrow{\vee}$$

$$\text{---} \underset{u}{\text{---}} \text{---} \underset{v}{\text{---}} \text{---} \underset{v}{\text{---}} \text{---} \underset{w}{\text{---}}$$

$$\dots \quad x \xrightarrow{b} y \xrightarrow{d} z \xrightarrow{\vee} \quad (11.12)$$

$$\text{---} \underset{u}{\text{---}} \text{---} \underset{v}{\text{---}} \text{---} \underset{w}{\text{---}}$$

$$\dots \quad \kappa_0 x \xrightarrow{b} \kappa_1 x \xrightarrow{b} \kappa_2 x \xrightarrow{b} \kappa_2 y \xrightarrow{d} \kappa_2 z \xrightarrow{\vee}$$

$$\text{---} \underset{u}{\text{---}} \text{---} \underset{v}{\text{---}} \text{---} \underset{v}{\text{---}} \text{---} \underset{v}{\text{---}} \text{---} \underset{w}{\text{---}}$$

11.3 The Pumping Lemma

The pumping lemma states that regular languages have the pumping property.

Definition 11.3.1 (Pumping Property). *A language \mathcal{L} of pointed T -coalgebras is said to have the pumping property if there is a natural number n , such that in all T -coalgebras \mathbb{S} in \mathcal{L} of depth at least n we can find a pumping situation⁵ (s_0, s_1) in \mathbb{S} , such that deleting and pumping arbitrarily often at (s_0, s_1) yields a T -coalgebra in \mathcal{L} .*

We spend the remainder of this chapter on the proof of the following theorem.

Theorem 11.3.2. *Languages of T -coalgebras accepted by state-finite T -coalgebra automata have the pumping property.*

In the proof of the pumping lemma, we will need a property further strengthening acyclicity and tree-likeness for coalgebras $\mathbb{S} = \langle S, \sigma, s_I \rangle$. Informally, we require that (*) successor states of states $s \in S$ do not occur more than once in $\sigma(s)$. In the example of the binary tree functor $T(-) = Act \times (-) \times (-)$, we want that $\sigma(s) = (s', s'')$ as opposed to $\sigma(s) = (s', s')$. However, Property (*) is not well-stated as we characterise successor states by means of *Base*, which is defined up to isomorphism in *Set*, that is bijection. Thus *Base* does not let us distinguish occurrences of the same states in $\sigma(s)$.

For our proof it suffices to reduce Property (*) to the acceptance game $\mathcal{G}(\mathbb{A}, \mathbb{S})$ as follows. Recall that in basic positions (α, s) , \exists chooses a relation $Z \subseteq Q \times S$ such that $(\alpha, \sigma(s)) \in Rel_T(Z)$. Instead of (*) we require, that if \mathbb{A} accepts \mathbb{S} , \exists has a winning strategy, such that $f(\alpha, s) \subseteq Q \times S$ is left-unique for all positions $(\alpha, s) \in TQ \times S$ in all plays consistent with f .

Lemma 11.3.3. *For every coalgebra automaton \mathbb{A} and coalgebra \mathbb{S} such that \mathbb{A} accepts \mathbb{S} , there is a coalgebra \mathbb{T} bisimilar to \mathbb{S} such that \exists has a winning strategy f in $\mathcal{G}(\mathbb{A}, \mathbb{T})$ with $f(\alpha, s) \subseteq Q \times S$ left-unique for all positions $(\alpha, s) \in TQ \times S$ in plays of $\mathcal{G}(\mathbb{A}, \mathbb{T})$ consistent with f .*

Proof. In the following we fix a winning strategy f for \exists in $\mathcal{G}(\mathbb{A}, \mathbb{S})$. Then the proof proceeds as follows.

⁵See Definition 11.2.1.

1. There is a coalgebra $\mathbb{T} = \langle BWin_{\exists}\mathcal{G}(\mathbb{A}, \mathbb{S}), \rho, (q_I, s_I) \rangle$.
 2. \mathbb{T} is bisimilar to \mathbb{S} , so that \mathbb{A} accepts \mathbb{T} .
 3. \exists has a winning strategy f' in $\mathcal{G}(\mathbb{A}, \mathbb{T})$ such that $f'(\alpha, b) \subseteq Q \times BWin_{\exists}\mathcal{G}(\mathbb{A}, \mathbb{S})$ is left-unique at all positions $(\alpha, b) \in TQ \times BWin_{\exists}\mathcal{G}(\mathbb{A}, \mathbb{S})$ in plays consistent with f' .
1. By definition of the relation lifting $Rel_T(-)$, there is an epimorphism $e : TBWin_{\exists}\mathcal{G}(\mathbb{A}, \mathbb{S}) \rightarrow Rel_T(BWin_{\exists}\mathcal{G}(\mathbb{A}, \mathbb{S}))$. By Lemma B.4.1 e has a section $m : Rel_T(BWin_{\exists}\mathcal{G}(\mathbb{A}, \mathbb{S})) \rightarrow TBWin_{\exists}\mathcal{G}(\mathbb{A}, \mathbb{S})$ such that $e \circ m = id_{TBWin_{\exists}\mathcal{G}(\mathbb{A}, \mathbb{S})}$. Note that m is not unique, and its definition is subject to the axiom of choice. The restriction of the winning strategy of \exists to $BWin_{\exists}\mathcal{G}(\mathbb{A}, \mathbb{S})$ is a function $f|_{BWin_{\exists}\mathcal{G}(\mathbb{A}, \mathbb{S})} : BWin_{\exists}\mathcal{G}(\mathbb{A}, \mathbb{S}) \rightarrow Rel_T(BWin_{\exists}\mathcal{G}(\mathbb{A}, \mathbb{S}))$. From m and $f|_{BWin_{\exists}\mathcal{G}(\mathbb{A}, \mathbb{S})}$ we define $\rho := m \circ f|_{BWin_{\exists}\mathcal{G}(\mathbb{A}, \mathbb{S})}$. Then $\mathbb{T} = \langle BWin_{\exists}\mathcal{G}(\mathbb{A}, \mathbb{S}), \rho, (q_I, s_I) \rangle$ is a T -coalgebra.
2. We show that $R = \{(s, (q, s)) \mid (q, s) \in BWin_{\exists}\mathcal{G}(\mathbb{A}, \mathbb{S})\}$ is a bisimulation between \mathbb{S} and \mathbb{T} . Then $\pi_{TS} \circ (\sigma, (id_{TQ}, \sigma) \circ m \circ f \circ (\theta, id_S)) = \sigma \circ \pi_S$ and $\pi_{TBWin_{\exists}\mathcal{G}(\mathbb{A}, \mathbb{S})} \circ (\sigma, (id_{TQ}, \sigma) \circ m \circ f \circ (\theta, id_S)) = (id_{TQ}, \sigma) \circ m \circ f \circ (\theta, id_S) \circ \pi_{BWin_{\exists}\mathcal{G}(\mathbb{A}, \mathbb{S})}$ as in the following diagram. Thus R is a bisimulation between \mathbb{S} and \mathbb{T} .

$$\begin{array}{ccccc}
TS & \xleftarrow{\pi_{TS}} Rel_T(R) & \xrightarrow{\pi_{TBWin_{\exists}\mathcal{G}(\mathbb{A}, \mathbb{S})}} & TBWin_{\exists}\mathcal{G}(\mathbb{A}, \mathbb{S}) & \\
\uparrow \sigma & \vdots & & \begin{array}{c} \uparrow m \\ \downarrow e \end{array} & \\
S & \xleftarrow{\pi_S} R & \xrightarrow{\pi_{BWin_{\exists}\mathcal{G}(\mathbb{A}, \mathbb{S})}} & BWin_{\exists}\mathcal{G}(\mathbb{A}, \mathbb{S}) & \\
& & & \uparrow (id_{TQ}, \sigma) \circ f \circ (\theta, id_S) & \\
& & & Rel_T(BWin_{\exists}\mathcal{G}(\mathbb{A}, \mathbb{S})) & \\
& & & \uparrow (\sigma, (id_{TQ}, \sigma) \circ m \circ f \circ (\theta, id_S)) &
\end{array} \quad (11.13)$$

- 3. Define f' such that**
- $f'(q, (q, s)) := (\alpha, (q, s))$ where $(\alpha, s) = f(\theta(q), s)$ and
 - $f'(\alpha, (q, s)) := \{(q', (q', s')) \mid (q', s') \in f(\alpha, s)\}$

and let f' be defined arbitrary on all other positions of \exists .

In order to show that f' is well-defined we will use a function $k : Q \times S \rightarrow Q \times (Q \times S)$ defined such that $k(q, s) := (q, (q, s))$. k is an embedding. Since T is standard, $Rel_T(-)$

preserves embeddings and $Rel_T(k)$ is an embedding. By Lemma B.4.11 and the definition of m as the section of e , it is $Rel_T(k) = (id_{TQ}, m) \circ k'$ where $k' : TQ \times TS \rightarrow TQ \times (TQ \times TS)$ takes $k'(\alpha, \beta) := (\alpha, (\alpha, \beta))$.

1. $\alpha \in \theta(q)$ because f is well-defined.
2. That $(\alpha, m(\alpha, \sigma(s))) \in Rel_T(f'(\alpha, (q, s)))$ follows from $(\alpha, \sigma(s)) \in Rel_T(f(\alpha, s))$.

Then every play p of $\mathcal{G}(\mathbb{A}, \mathbb{T})$ consistent with f' will pass through basic positions only of the form $(q, (q, s))$, since the initial position of $\mathcal{G}(\mathbb{A}, \mathbb{T})$ is $(q_I, (q_I, s_I))$ and all elements of $f' \circ f'(q, (q, s))$ are of the form $(q', (q', s'))$.

It remains to show f' winning. By definition of f' , \exists loses with f' a finite play p' of $\mathcal{G}(\mathbb{A}, \mathbb{T})$ only if she loses a finite play p of $\mathcal{G}(\mathbb{A}, \mathbb{S})$ with f , where the basic positions in p' are stepwise the image of basic positions of p . Finally since $\Omega_{\mathcal{G}(\mathbb{A}, \mathbb{T})}(q, (q, s)) = \Omega_{\mathcal{G}(\mathbb{A}, \mathbb{S})}(q, s)$, \exists loses with f' an infinite play p' of $\mathcal{G}(\mathbb{A}, \mathbb{T})$ only if she loses an infinite play p of $\mathcal{G}(\mathbb{A}, \mathbb{S})$ with f , where the basic positions in p' are stepwise the image of basic positions of p . By contrapositivity f' is thus winning since f is. \square

Proof of Theorem 11.3.2. Because pumping can be iterated, it suffices to show that for every coalgebra automaton \mathbb{A} there is a natural number $n \in \mathbb{N}$, such that in any T -coalgebra $\mathbb{S} = \langle S, \sigma, s_I \rangle$ of depth at least n there is a pumping situation (s, s') such that deleting or pumping once at (s, s') yields a T -coalgebra \mathbb{S}' accepted by \mathbb{A} . We will show that the above holds for $n = |Q| + 1$.

Let \mathbb{S} be a T -coalgebra accepted by \mathbb{A} and of depth at least $|Q|$. By Corollary 11.1.17 we can assume without loss of generality that \mathbb{S} is tree-like. As \mathbb{A} accepts \mathbb{S} , \exists has a winning strategy f in $\mathcal{G}(\mathbb{A}, \mathbb{S})$. We distinguish the following cases.

1. In every play p of $\mathcal{G}(\mathbb{A}, \mathbb{S})$ consistent with f every automaton state q occurs at most once in basic positions.
2. There is a play p of $\mathcal{G}(\mathbb{A}, \mathbb{S})$ consistent with f and an automaton state q , which occurs twice in basic positions (q, s) and (q, s') .

In each case we select a pumping situation (s_0, s_1) as follows.

1. We choose $s_0, s_1 \in \text{Reach}_{\mathbb{S}}(s_I)$ such that s_0 and s_1 do not occur in any basic position in any play of $\mathcal{G}(\mathbb{A}, \mathbb{S})$ consistent with f and $s_1 \in \text{Reach}_{\mathbb{S}}(s_0)$.
2. Put $s_0 = s$ and $s_1 = s'$ for some pair (q, s) and (q, s') of basic positions in p with the same automaton state q .

We then prove that deleting or pumping once at (s_0, s_1) yields a coalgebra \mathbb{S}' accepted by \mathbb{A} , providing a winning strategy f' for \exists in $\mathcal{G}(\mathbb{A}, \mathbb{S}')$ in both cases.

Deleting Define $f' := f\{s_0 \mapsto f(s_1)\}$. Then f' is well-defined if in any play p of $\mathcal{G}(\mathbb{A}, \mathbb{S})$ consistent with f , s_0 occurs in at most a unique basic position. If otherwise there were basic positions (q, s_0) and (q', s_0) with $q \neq q'$, it is not necessarily $f'(q', s_0) \in (\theta(q'), s_0)$. By Lemma 11.3.3 the assumption does not restrict generality.

We then prove that f' is winning, showing that the plays of $\mathcal{G}(\mathbb{A}, \mathbb{S}')$ consistent with f' correspond to plays of $\mathcal{G}(\mathbb{A}, \mathbb{S})$ consistent with f under the forgetful function $h : S' \rightarrow S$ defined by

$$h := id_S\{s_0 \mapsto s_1\} \quad (11.14)$$

The forgetful function h extends to a function from the positions of the acceptance game $\mathcal{G}(\mathbb{A}, \mathbb{S}')$ to the positions of the acceptance game $\mathcal{G}(\mathbb{A}, \mathbb{S})$ such that

$$h(v) := \begin{cases} (q, h(s)) & \text{if } v = (q, s) \\ (q, Th(s)) & \text{if } s = (q, \alpha) \in Q \times TS \\ \langle id_Q, h \rangle(v) & \text{if } v \in \mathcal{P}(Q \times S) \end{cases} \quad (11.15)$$

and extends to initial plays of $\mathcal{G}(\mathbb{A}, \mathbb{S})$ such that

$$h(pv) := h(p)h(v). \quad (11.16)$$

Defined as above, h has the following properties.

1. h commutes with admissible moves, that is $h(E[v]) = E[h(V)]$, so that p' is total if and only if $h(p)$ is.

2. h preserves the assignment to players, such that $h(v)$ belongs to \exists in $\mathcal{G}(\mathbb{A}, \mathbb{S}')$ if and only if v does in $\mathcal{G}(\mathbb{A}, \mathbb{S})$.
3. h preserves the priority of board positions, $\Omega(h(v)) = \Omega'(v)$ for all v where h is defined.
4. h commutes with f and f' , such that $f \circ h = h \circ f'$.

Properties 1 and 2, if p' is finite, and 1, 2, and 3, if p' is infinite, entail that p' is winning for \exists in $\mathcal{G}(\mathbb{A}, \mathbb{S}')$ if $h[p']$ is winning for \exists in $\mathcal{G}(\mathbb{A}, \mathbb{S})$. Property 4 entails that f' is winning for \exists in $\mathcal{G}(\mathbb{A}, \mathbb{S}')$ since f is winning for \exists in $\mathcal{G}(\mathbb{A}, \mathbb{S})$.

Pumping Once Define $f' := (\kappa_0 f + \kappa_1 f)\{\kappa_0 s_1 \mapsto \kappa_1 f(s_0)\}$. Then f' is only well-defined if in any play of $\mathcal{G}(\mathbb{A}, \mathbb{S})$ consistent with f there is at most a unique basic position with s_1 . If otherwise there are basic positions (q, s_1) and (q', s_1) , it is not necessarily $f'(q', \kappa_0 s_1) \in (\theta(q'), \sigma'(\kappa_0 s_1))$. By Lemma 11.3.3 the assumption does not restrict generality.

Similar to the argument above we show that the plays of $\mathcal{G}(\mathbb{A}', \mathbb{S})$ consistent with f' correspond to plays of $\mathcal{G}(\mathbb{A}, \mathbb{S})$ consistent with f under a forgetful function h defined as follows.

$$h(s) := \begin{cases} s_0 & \text{if } s = \kappa_0 s_1 \\ s' & \text{if } s = \kappa_0 s' \text{ and } s' \neq s_1 \\ s' & \text{if } s = \kappa_1 s' \end{cases} \quad (11.17)$$

The forgetful function h extends to positions and initial plays of $\mathcal{G}(\mathbb{A}, \mathbb{S})$ as in (11.15) and (11.16) above. Moreover h has properties 1.-4. as above, so that by a similar argument as the one above we obtain that f' is winning, since f is. \square

Part V

Conclusions

Chapter 12

Conclusions

12.1 Summary of Contributions

With this dissertation we have made several contributions to the theory of coalgebras with branching, within the theory of coalgebras over *Set*, Generic Trace Theory of Jacobs et alii, coalgebraic logic, and the coalgebraic automata theory of Venema et alii.

12.1.1 Finite Trace Semantics

In Chapter 5 we have revisited Jacobs [Jac04, HJS06, HJS07] and presented a definition, which does not depend on the order-enrichment of the Kleisli-category of the branching type and is applicable to finitary branching types. In Chapter 5 we have seen that the order enrichment is ambiguous. Finitary branching types are important for the definition of finitary coalgebraic logics for finite traces as in Chapter 8.

12.1.2 Infinite Trace Semantics

In Chapter 6 we have adapted the basic ideas underlying the definition of finite trace semantics and have obtained an inductive definition of infinite trace semantics in the style of Böhm trees. Generalising the the inductive definition lead us to a coinductive notion of infinite trace semantics which subsumes the inductive one, as well as the acceptance behaviour of coalgebra automata.

Both, inductive definition and coinductive characterisation differ substantially from previous work of Jacobs [Jac04] and parallel work of Cirstea [Cî10]. Both aim to give a unique coinductive definition. Jacobs defined infinite trace semantics for the powerset monad as the branching type and for transition types admitting a final coalgebra. Cirstea generalises Jacobs work to affine monads assuming an order enrichment of the Kleisli-category following Generic Trace Theory of Hasuo et alii [HJS06]. We show that Jacobs' infinite trace semantics, acceptance behaviour of nondeterministic coalgebra automata are instances of our coinductive characterisation of infinite trace semantics.

12.1.3 Finitary Coalgebraic Logics

In Chapter 7 we have defined in joint work with Yde Venema the Boolean dual of Moss' modality ∇ . Thereby we showed that finitary Moss' coalgebraic logics is essentially negationfree. Our proof is largely based on the previously established [KKV08] completeness of finitary Moss' coalgebraic logic.

12.1.4 Finite Trace Logics

In Chapter 5 we have shown that finite trace semantics induces finite trace equivalence. We have defined in Chapter 8 finitary coalgebraic logics in the style of Moss' coalgebraic logics classifying states in coalgebras with branching for many finitary branching and transition types up to finite trace equivalence. Our logics are parameterised in a logic functor and its denotation. We have shown that finitary coalgebraic logics for finite traces are invariant under finite trace equivalence, and under further assumptions on the logic functor and denotation expressive. In the latter we follow the more general framework of Klin [Kli07].

12.1.5 Game Bisimulations for Parity Graph Games

Parity graph games play a role in the definition of the acceptance behaviour of classical and coalgebra automata. In Chapter 9 we have introduced in joint work with Yde Venema

a structure to parity graph games into rounds and a normalisation of the interaction pattern between the participating players, based on the power of players introduced by van Benthem [vB02]. We then define the congruence relation game bisimulation, which facilitates proofs of the equivalence between games and between automata as in Chapter 10.

12.1.6 Complementation Lemma for Coalgebra Automata

Using the complementation lemma for finitary Moss' coalgebraic logics, we define for any coalgebra automaton \mathbb{A} one accepting the complement of the language of \mathbb{A} . Together with previous results of Kupke and Venema [Ven04, KV08, KV08] we complete the correspondence between coalgebra automata and coalgebraic logics augmented with fixpoint operators following Rabin [Rab69]. We provide upper bounds on the size of the complemented automata and reconcile our observations with previous results in [Kis07].

12.1.7 A Pumping Lemma for Regular Languages of Coalgebras in *Set*

In Chapter 11 we have reviewed coalgebras in *Set* as graphs with additional structure, and shown that each such coalgebra is bisimilar to one whose graph is acyclic and tree-like. We have then reduced the pumping lemma for regular languages of coalgebras in *Set* to the one for trees.

12.2 Some Open Questions and Directions for Future Work

In this dissertation we have left several questions unanswered.

12.2.1 Monads and Categories of Algebras

It remains an open question to characterise monads and functors, which admit distributive laws.

We have defined the continuous extension of the Kleisli-lifting of *Set*-functors in Eilenberg-Moore categories. In particular for the existence of finite trace semantics and finite trace logics the size of the Kleisli-lifting is important. We leave an understanding of the latter to future work.

12.2.2 Finitary Coalgebraic Logics for Finite Traces

The definition of finite trace logics hinges on the dual adjunction induced by an ambimorphic object. We assumed that the branching type B is commutative, so that every object in $B\text{-Alg}$ is ambimorphic. We leave open, whether non-commutative algebras admit an ambimorphic object in their Eilenberg-Moore algebras. We have furthermore left open a characterisation of monads, which admit a coseparating ambimorphic object.

We have given several assumptions on the logic functor and its denotation under which finitary coalgebraic logics for finite traces are expressive. We leave it to future work as well to characterise branching and transition types allowing a logic functor meeting these assumptions. In particular, we require that the branching type B is such that there is a coseparating ambimorphic object in $B\text{-Alg}$. We leave the characterisation of such branching types open.

Appendix A

Set Theory

A.1 Basic Set Theory

This dissertation is based on Zermelo-Fraenkel set theory with the Axiom of Choice.

Definition A.1.1 (Pointed Sets). A pointed set (X, x) consists of a set X and a distinguished element $x \in X$.

Definition A.1.2 (Relations). Let X and Y be sets, a relation between X and Y is a subset $R \subseteq X \times Y$.

Definition A.1.3 (Functions). A function $f : X \rightarrow Y$ from a set X to a set Y is defined by its graph $Gr(f) \subseteq X \times Y$ such that for each $x \in X$, there is a unique $y \in Y$ such that $(x, y) \in Gr(f)$. We denote y by $f(x)$.

Definition A.1.4 (Functions With Finite Support). A function $f : X \rightarrow \mathbb{N}$ into the natural numbers is said to have finite support, if f is zero almost everywhere, that is $\{x \in X \mid f(x) \neq 0\}$ has a finite cardinality. We denote the set of such functions by $(\mathbb{N}^X)_\omega$. The definition extends to \mathbb{R} instead of \mathbb{N} and in fact all pointed sets $(Y, 0)$.

Definition A.1.5 (Upward Closure). Given a set $\mathbb{X} \subseteq \mathcal{P}X$, define

$$[\mathbb{X}] := \{Y \subseteq X \mid Y \supseteq Z \text{ for some } Z \in \mathbb{X}\}. \quad (\text{A.1})$$

A.2 Order Theory

Definition A.2.1 (Preorders). A preorder on a set X is a binary relation $(\leq) \subseteq X \times X$, which is

1. reflexive, so that $x \leq x$, and
2. transitive, so that $x \leq z$ if $x \leq y$ and $y \leq z$,

for all $x, y, z \in X$

Definition A.2.2 (Partial Orders). A partial order on a set X is a preorder $\leq \subseteq X \times X$, which is

3. antisymmetric, so that $x = y$ if $x \leq y$ and $y \leq x$, for all $x, y \in X$.

Definition A.2.3 (Directed Sets). A set X is called directed, if it comes augmented with a partial order $\leq \subseteq X \times X$, such that

1. X is non-empty, and
2. for every pair (y_1, y_2) of elements of X , with a common lower bound x , such that $x \leq y_1$ and $x \leq y_2$, there is an element $z \in X$ such that $y_1 \leq z$ and $y_2 \leq z$.

Definition A.2.4 (Partial Orders (continued)). 1. A partial order (X, \leq) is said to have a bottom element, if there is an element $\perp \in X$ such that $\perp \leq x$ for all $x \in X$.

2. A partial order (X, \leq) is called complete, if all subsets $Y \subseteq X$ have a least upper bound.
3. A partial order (X, \leq) is called directed complete, if all directed subsets $Y \subseteq X$ have an upper bound.

Appendix B

Category Theory

We review the basic definitions of category theory, we need in this dissertation. All results in this chapter are well known and we repeat proofs when they contribute to the understanding of the content of the main chapters of this dissertation. For a more concise exposition we refer to the established literature [Kel05, Mac98].

B.1 Basic Category Theory

Definition B.1.1 ((Locally Small) Categories). *A category C consists of*

1. *a collection $\text{Obj}(C)$ of objects,*
2. *for each pair objects X and Y a set $C(X, Y)$ of morphisms, the homset between X and Y ,*
3. *for each object X a canonical morphism $\text{id}_X : X \rightarrow X$,*
4. *for each pair of morphisms $f : X \rightarrow Y$ and $g : Y \rightarrow Z$ a morphism $g \circ f$ (equivalently, $f; g$) such that $\text{id}_Y \circ f = f = \text{id}_Y \circ f$ for all morphisms $f : X \rightarrow Y$*

Morphisms f and g as in 3. are called compatible.

Example B.1.2. 1. *The natural numbers, \mathbb{N} , and \leq form a category.*

2. *The ordinal ω with \in forms a category.*

3. Any preordered set $\langle X, \leq \rangle$ is a category.
4. Sets and functions form the category *Set*.
5. Sets and relations form the category *Rel*.
6. Directed complete partial orders with bottom element and order-preserving functions form the category $DCPO_{\perp}$.
7. The free algebras for a monad and their homomorphisms form the category $Kl(B)$.
8. The Eilenberg-Moore algebras for a monad and their homomorphisms form the category $B\text{-Alg}$.

Definition B.1.3 (Discrete Categories). A category C is discrete if for any pair of distinct objects X and Y , $C(X, Y) = \emptyset$. Every category C has a discrete version, which we denote by $|C|$.

Definition B.1.4 (Dual Categories). The dual C^{op} of a category C as above, is one where all morphisms are inverted, that is for every pair of objects X and Y , C^{op} has a morphism $f : X \rightarrow Y$ where C has a morphism $f : Y \rightarrow X$.

Definition B.1.5 (Covariant Functors). A functor $T : C \rightarrow \mathcal{D}$ between categories C and \mathcal{D} consists of

- a map $T : \text{Obj}(C) \rightarrow \text{Obj}(\mathcal{D})$ on the objects of C and \mathcal{D}
- a map T on the morphisms, preserving
 - the identity morphisms, such that $Tid_X = id_{TX}$
 - and composition, such that $T(g \circ f) = Tg \circ Tf$

Example B.1.6 (Covariant Functors). 1. $Id : C$ is the identity functor taking objects $X \mapsto X$ and morphisms $(f : X \rightarrow Y) \mapsto f$.

2. Let X be an object of C , we denote by X the functor taking all objects to X and morphisms to id_X .

3. $\mathcal{P} : \mathbf{Set} \rightarrow \mathbf{Set}$ is the covariant power set functor taking sets X to their powersets and functions f to their direct-image lifting $\mathcal{P}f(Y \subseteq X) := \{f(x) \mid x \in X\}$.
4. $\mathbf{Lat} : \mathbf{Set} \rightarrow \mathbf{Set}$ taking sets X to the free lattices generated from X , and functions $f : X \rightarrow Y$ to the free lattice morphism $\mathbf{Lat}X \rightarrow \mathbf{Lat}Y$.
5. $\mathbf{SLat} : \mathbf{Set} \rightarrow \mathbf{Set}$ taking sets X to the free semi-lattices generated from X , and functions $f : X \rightarrow Y$ to the free lattice morphism $\mathbf{SLat}X \rightarrow \mathbf{SLat}Y$. We will distinguish the disjunctive semi-lattice functors as \mathbf{SLat}^\vee from the conjunctive semi-lattice functor \mathbf{SLat}^\wedge , although the distinction is purely syntactic. functor

Definition B.1.7 (Contravariant Functors). A contravariant functor $T : \mathcal{C} \rightarrow \mathcal{D}$ is a covariant functor $T : \mathcal{C} \rightarrow \mathcal{D}^{op}$.

Example B.1.8 (Contravariant Functors). 1. The contravariant powerset functor takes sets X to the sets of their predicates, that is functions $m : X \rightarrow \{0, 1\}$, and functions $f : X \rightarrow Y$ to $Qf : QY \rightarrow QX$ taking predicates $n : Y \rightarrow \{0, 1\}$ to their composition with f , that is predicates $(n \circ f)(x) := n(f(x))$.

Definition B.1.9 (Endofunctors). An endofunctor is a functor $T : \mathcal{C} \rightarrow \mathcal{C}$, whose source and target category coincide.

Definition B.1.10 (Natural Transformations). Let $T, T' : \mathcal{C} \rightarrow \mathcal{D}$ be functors. A natural transformation $\alpha : T \Rightarrow T'$ is a collection of morphisms α_X for each object X of \mathcal{C} , such that for all morphisms $f : X \rightarrow Y$, $T'f \circ \alpha_X = \alpha_Y \circ Tf$ commutes. $\mathbf{Nat}(T, T')$ denotes the set of natural transformations $T \Rightarrow T'$.

Definition B.1.11 (Diagrams). A diagram in a category \mathcal{D} is a functor $D : \mathcal{C} \rightarrow \mathcal{D}$, where \mathcal{C} is called the indexing category. A diagram is called discrete if \mathcal{C} is a discrete category.

Definition B.1.12 (Images of Diagrams). Let $D : \mathcal{C} \rightarrow \mathcal{D}$ be a diagram. The image of D is the subcategory of \mathcal{D} consisting of

- objects DX for each object X of \mathcal{C} and

- morphisms $Df : DX \rightarrow DY$ for each pair of objects X and Y in C and morphisms $f : X \rightarrow Y$ in C .

Functoriality of D asserts, that the image of D satisfies identity and compositionality laws of categories.

In this thesis we meet frequently the chain diagrams Seq_T and Seq^T for functors T . The following definitions make the explanations in the introduction formal.

Definition B.1.13 (Seq_T). Given a functor T on a category C with an initial object 0 with initial object morphism i_X For any object X of C , we define the ω -chain Seq_T , which is a diagram $Seq_T : \omega \rightarrow C$, where ω is the category in 2. of Example B.1.2. Recall, that objects of ω are elements of ω and morphisms are given by \in . Then Seq_T is generated from

$$Seq_T(n) := T^n 0 \text{ and } Seq_T(n \in n+1) := T^n i_{T^0} \text{ for all } n < \omega \quad (B.1)$$

as in the following diagram

$$0 \xrightarrow{i_{T^0}} T^0 \longrightarrow \dots \longrightarrow T^n 0 \xrightarrow{T^n i_{T^0}} T^{n+1} 0 \longrightarrow \dots \quad (B.2)$$

Definition B.1.14 (Seq^T). Given a functor T on a category C with an initial object 0 with initial object morphism i_X For any object X of C , we define the ω^{op} -chain Seq^T , which is a diagram $Seq^T : \omega^{op} \rightarrow C$, where ω^{op} is the dual of category ω as in 2. of Example B.1.2. Then Seq^T is generated from

$$Seq^T(n) := T^n 1 \text{ and } Seq^T(n \in n+1) := T^n !_{T^1} \text{ for all } n < \omega \quad (B.3)$$

as in the following diagram

$$1 \xleftarrow{!_{T^1}} T^1 \longleftarrow \dots \longleftarrow T^n 1 \xleftarrow{T^n !_{T^1}} T^{n+1} 1 \longleftarrow \dots \quad (B.4)$$

Definition B.1.15 (n -Multi-Endofunctors). Let $n \in \mathbb{N}$ be a natural number. An n -multi-endofunctor is a functor T , whose source consists of an n -tuple of categories. T consists

of

- a map $T : \text{Obj}(C)_1 \times \cdots \times \text{Obj}(C)_n \rightarrow \mathcal{D}$, and
- a map T on morphisms, preserving collectively
 - identity morphisms, such that $T : (id_{X_1}, \dots, id_{X_n}) \mapsto id_{T(X_1, \dots, X_n)}$, and
 - composition component wise, such that $T : (g_1 \circ f_1, \dots, g_n \circ f_n) \mapsto T(g_1, \dots, g_n) \circ T(f_1, \dots, f_n)$.

Definition B.1.16 (Natural Transformations). A natural transformation $\alpha : S \Rightarrow T$ between functors $S, T : C \rightarrow \mathcal{D}$ is a collection of morphisms $(\alpha_X)_{X \in \text{Obj}(C)}$, such that $\alpha_Y \circ S f = T f \circ \alpha_X$ commutes for all pairs of objects X and Y of C and morphisms $f : X \rightarrow Y$ between them.

Definition B.1.17 (Adjunctions). Functors $S : C \rightarrow \mathcal{D}$ and $T : \mathcal{D} \rightarrow C$ are said to form an adjunction such that S is left adjoint to T , written $S \dashv T$, or equivalently T is right adjoint to S , if $\mathcal{D}(S(-), +) \cong C(-, T(+))$ are in bijection. We denote the bijection by $(-)^{\dagger}$ without stating the direction, as this is usually clear from the context.

1. An adjunction $S \dashv T$ has a unit, that is a natural transformation $\eta : Id \Rightarrow TS$ in C , and a counit, that is a natural transformation $\epsilon : ST \Rightarrow Id$ in \mathcal{D} , defined such that

- $\eta := (id_S)^{\dagger}$
- $\epsilon := (id_T)^{\dagger}$

2. Conversely we can define $(-)^{\dagger}$ from η and ϵ . Let $f : X \rightarrow TY$ and $g : SX \rightarrow Y$, then we define

- $f^{\dagger} := \epsilon_Y \circ S f$
- $g^{\dagger} := T g \circ \eta_X$

Directions 1. and 2. are converse to each other.

Definition B.1.18 (Comma Categories). Let C and \mathcal{D} be category, let D be an object of \mathcal{D} , and let $T : C \rightarrow \mathcal{D}$ be a functor. $(T \downarrow D)$ denotes the category consisting of

morphism $h : X \times Y \rightarrow Z$ making the following diagram commute.

$$\begin{array}{ccccc}
 X & \xleftarrow{\pi_X} & X \times Y & \xrightarrow{\pi_Y} & Y \\
 & \searrow \pi'_X & \uparrow h & \nearrow \pi'_Y & \\
 & & Z & &
 \end{array} \tag{B.6}$$

Coproducts are dual to products.

Definition B.2.5 (Coproducts). *The coproduct $X + Y$ of objects X and Y of a category C is an object with morphisms $\kappa_X : X \rightarrow X + Y$ and $\kappa_Y : Y \rightarrow X + Y$ such that for every object Z with morphisms $\kappa'_X : X \rightarrow Z$ and $\kappa'_Y : Y \rightarrow Z$, there is a unique morphism $h : Z \rightarrow X + Y$ making the following diagram commute.*

$$\begin{array}{ccccc}
 X & \xrightarrow{\kappa_X} & X + Y & \xleftarrow{\kappa_Y} & Y \\
 & \searrow \kappa'_X & \downarrow h & \nearrow \kappa'_Y & \\
 & & Z & &
 \end{array} \tag{B.7}$$

Powers and copowers generalise the notion of products and coproducts. In this thesis, we only use the following notion copowers in sets.

Definition B.2.6 (Copowers in Sets). *Let Act be a set and let X be an object of a category C , the copower $Act \cdot X$ is defined to be an object such that $C(Act \cdot X, Y) \cong C(X, Y)^{Act}$ natural in Y for all objects Y in C .*

Colimits subsume epimorphisms, coproducts and copowers.

Definition B.2.7 (Colimits). *Let $D : I \rightarrow C$ be a diagram in C , then the colimit $colim D$ of D is the object satisfying*

$$Nat(D, D') \cong C(colim D, D') \tag{B.8}$$

for any object D' in C .

Note that when we instantiate $D' := colim D$ in Equation B.8 and insert the identity morphism $id_{colim D} : colim D \rightarrow colim D$ on the right hand side, we obtain a uniquely defined

natural transformation $c : D \Rightarrow \text{colim}D$ on the left hand side. We call this natural transformation the cocone of $\text{colim}D$ over D .

Proposition B.2.8. *Let D be a diagram in C and let $c : D \Rightarrow \text{colim}D$ be the colimiting cocone over D , then c is jointly epic.*

Proof. Epicness follows from the universal property of colimits, that $\text{Nat}(D, X) = C(\text{colim}D, X)$ are in bijection for all X . \square

B.2.1 Filtered Colimits and Finitary Functors

For more details on filtered colimits and finitary functors we refer to [AR94] and [ARV10].

In this thesis, we only need Definition B.2.13.

Definition B.2.9 (Filtered Categories). 1. C is non-empty, that is C has an object.

2. For any pair of objects $X, Y \in C$, there exists an object $Z \in C$ and morphisms $X \rightarrow Z$ and $Y \rightarrow Z$.

3. For any pair of parallel morphisms $f, g : X \rightarrow Y$ in C , there exists a morphism $h : Y \rightarrow Z$ making $h \circ f = h \circ g$ commute.

Definition B.2.10 (Filtered Diagrams). A diagram $D : C \rightarrow \mathcal{D}$ is filtered, if C is a filtered category.

Lemma B.2.11. Seq_L is a filtered diagram.

Proof. Seq_L is a chain diagram and thus filtered. See also Example 2.3 of [ARV10]. \square

Definition B.2.12 (Filtered Colimits). A colimit $\text{colim}D$ of a diagram D is called filtered, if D is filtered.

Definition B.2.13 (Finitary Functors). A functor $T : C \rightarrow C$ is finitary if T preserves finite filtered colimits.

The following proposition follows immediately from Lemma B.2.11 and Definition B.2.13.

Proposition B.2.14. Finitary functors T preserve the colimit of Seq_T , so that the initial T -sequence terminates after ω steps.

B.2.2 Equalisers and Coequalisers

Definition B.2.15 (Equalisers). *The equaliser of a pair of morphisms $f_0, f_1 : X \rightarrow Y$ is a morphism $e : Z \rightarrow X$ such that $f_0 \circ e = f_1 \circ e$ such that for all objects Z' and morphisms $e' : Z' \rightarrow X$ with $f_0 \circ e' = f_1 \circ e'$, there is a unique morphism $h : Z' \rightarrow Z$ such that $e \circ h = e'$.*

$$\begin{array}{ccccc} Y & \xrightleftharpoons[f_0]{f_1} & X & \xleftarrow{e} & Z \\ & & \searrow e' & \uparrow h & \downarrow \\ & & & & Z' \end{array} \quad (\text{B.9})$$

A proof of the following can be found for instance in [Mac98].

Lemma B.2.16. *Every equaliser is monic.*

Coequalisers are dual to equalisers.

Definition B.2.17 (Coequalisers). *The coequaliser of a pair of morphisms $f_0, f_1 : X \rightarrow Y$ is a morphism $c : Y \rightarrow Z$ such that $c \circ f_0 = c \circ f_1$ such that for all objects Z' and morphisms $c' : Y \rightarrow Z'$ with $c' \circ f_0 = c' \circ f_1$, there is a unique morphism $h : Z \rightarrow Z'$ such that $c = c' \circ h$.*

$$\begin{array}{ccccc} X & \xrightleftharpoons[f_1]{f_0} & Y & \xrightarrow{c} & Z \\ & & \searrow c' & \downarrow h & \downarrow \\ & & & & Z' \end{array} \quad (\text{B.10})$$

The following lemma is the dual statement of Lemma B.2.16.

Lemma B.2.18. *Every coequaliser is epic.*

Lemma B.2.19. *Let the following be coequaliser diagrams:*

$$\begin{array}{ccccc} A & \xrightleftharpoons[f_2]{f_1} & B & \xrightarrow{g} & C \\ \downarrow k & & \downarrow l & & \downarrow h \\ A' & \xrightleftharpoons[f'_2]{f'_1} & B' & \xrightarrow{g'} & C' \end{array} \quad (\text{B.11})$$

such that

1. $A \rightarrow B \rightarrow C$ and $A' \rightarrow B' \rightarrow C'$ are coequaliser diagrams
2. $l \circ f_i = f'_i \circ k$ for both $i \in \{1, 2\}$

3. h is the unique arrow determined given by the lower coequaliser

Then h is monic.

Proof. Suppose h were not mono, then we could construct an object D into which C' embeds such that there is not a unique morphism $C' \rightarrow D$ contradicting the universal property of the coequaliser C' . \square

B.2.3 Pointwise Construction of Kan Extensions

Definition B.2.20 (Left Kan-Extension). Consider the following diagram, where \mathcal{A} , \mathcal{B} , and C are categories, and $K : C \rightarrow \mathcal{A}$, $T : C \rightarrow \mathcal{B}$ and $L : \mathcal{A} \rightarrow \mathcal{B}$ are functors.

$$\begin{array}{ccc} \mathcal{A} & \xrightarrow{L} & \mathcal{B} \\ K \uparrow & \nearrow T & \\ C & & \end{array} \quad (\text{B.12})$$

The left Kan-extension of T along K , denoted $\text{Lan}_K T$, is a functor L with a natural transformation $\lambda : T \Rightarrow LK$ universal such that for any other functor $L' : \mathcal{A} \rightarrow \mathcal{B}$ and natural transformation $\lambda' : T \Rightarrow L'K$ there is a unique natural transformation $\delta : L \Rightarrow L'$ such that $\delta_K \circ \lambda = \lambda'$.

The right Kan-extension is the defined dually.

Mac Lane [Mac98] gave a construction of the right Kan-extension as limit. Next we give the dual construction for reference.

Definition B.2.21 (Point wise Left Kan-Extension). Let $(K \downarrow A)$ be the comma category of arrows $f : KC \rightarrow A$ in \mathcal{A} . Define the functor $O_A : (K \downarrow A) \rightarrow \mathcal{B}$ as $O_A(f) := TC$. Then define $LC := \text{colim} O_A$, given the colimit exists.

Definition B.2.21 is sound. The following is the dual of Theorem X.3.1 of [Mac98].

Theorem B.2.22. The functor defined in Definition B.2.21 is the left Kan-extension of T along K .

B.3 Monoidal Categories

Definition B.3.1 ((Strong) Monoidal Categories). A (strong) monoidal category $\langle C, \otimes, I \rangle$ consists of

- a category C ,
- a two-multi-endofunctor $C \otimes C \rightarrow C$, and
- a distinguished object $I \in \text{Obj}(C)$

such that

- $I \otimes X \stackrel{\lambda}{=} X \stackrel{\rho^{-1}}{=} X \otimes I$ are isomorphic, natural in X
- $(X \otimes Y) \otimes Z \stackrel{\alpha}{=} X \otimes (Y \otimes Z)$ are isomorphic, natural in X , Y , and Z

Definition B.3.2 ((Strong) Symmetric Monoidal Categories). A (strong) monoidal category C as above is symmetric, if $X \otimes Y$ and $Y \otimes X$ are isomorphic.

Definition B.3.3 (Closed Category). A category C is closed, if for each pair of objects X and Y , $C(X, Y)$ is an object of C .

Definition B.3.4 ((Strong) Symmetric Monoidal Closed Categories). A (strong) symmetric monoidal closed category is a closed (strong) symmetric monoidal category $\langle C, \otimes, I \rangle$ such that $[X \otimes Y, Z] = [X, [Y, Z]]$ are isomorphic natural in X , Y , and Z .

Example B.3.5. $\langle \text{Set}, \times, \{*\} \rangle$ is a strong symmetric monoidal closed category.

Definition B.3.6 (Faithful Functors). A functor $T : C \rightarrow \mathcal{D}$ is faithful if T is injective on $C(X, Y)$ for all objects X and Y of C .

B.4 Category Theory of Set

Lemma B.4.1. In the presence of the axiom of choice, every epimorphism $e : X \rightarrow Y$ has a monic section $m : Y \rightarrow X$, such that $e \circ m = \text{id}_Y$.

Proof. Let c be a choice function $c : \{e^{-1}[y] \mid y \in Y\} \rightarrow X$, then define $m(y) := c(e^{-1}[y])$. Then $m(y) \in e^{-1}[y]$ and thus $e(m(y)) = y$. \square

Definition B.4.2 (Finitarisation of *Set*-Functors). *Let $T : \text{Set} \rightarrow \text{Set}$ be a functor, we define the finitarisation $T_\omega : \text{Set} \rightarrow \text{Set}$ of T such that $T_\omega(X) := \bigcup \{TY \mid Y \subseteq_\omega X\}$.*

B.4.1 Standard and Weak-Pullback Preserving Functors

We adopt the following definition of standardness for *Set*-functors, which has been used in [Mos99, Ven04] for instance.

Definition B.4.3 (Standard Functors). *A set functor T is standard, if T preserves inclusion.*

$$\text{if } X \subseteq Y \text{ then } TX \subseteq TY \quad (\text{B.13})$$

The above definition differs from the definition of standardness in [AT90]. The latter is based on choice functors $C_1 : \text{Set} \rightarrow \text{Set}$ and $C_{01} : \text{Set} \rightarrow \text{Set}$ defined such that

$$C_1 : X \mapsto \{*\} \quad C_1 : (f : X \rightarrow Y) \mapsto id_{\{*\}} \quad (\text{B.14})$$

and

$$C_{01} : X \mapsto \begin{cases} \emptyset & \text{if } X = \emptyset \\ \{*\} & \text{otherwise} \end{cases} \quad C_1 : (f : X \rightarrow Y) \mapsto \begin{cases} \emptyset & \text{if } X = \emptyset \\ id_{\{*\}} & \text{otherwise} \end{cases} \quad (\text{B.15})$$

Definition B.4.4 (\emptyset -Standard Functors). *A functor $T : \text{Set} \rightarrow \text{Set}$ is called \emptyset -standard if T is standard and every natural transformation $C_{01} \Rightarrow T$ can be extended to a natural transformation $C_1 \Rightarrow T$.*

The following lemma appears as Lemma A.2.12 in [Kup06].

Lemma B.4.5. *Every standard weak-pullback preserving functor $T : \text{Set} \rightarrow \text{Set}$ is \emptyset -standard.*

Using Lemma B.4.5, Theorem III.4.5 in [AT90] implies the following theorem.

Theorem B.4.6. *Every weak-pullback preserving functor is naturally isomorphic to standard weak-pullback preserving one.*

The following is Proposition III.4.6 of [AT90], and is necessary for Definition B.4.13 to be sound.

Proposition B.4.7. *Any standard functor on \mathbf{Set} preserves finite intersections.*

B.4.2 Relations and Relation Liftings in \mathbf{Set}

Definition B.4.8 (Relations). *A relation R is given by a span*

$$\begin{array}{ccc} & R & \\ \pi_X \swarrow & & \searrow \pi_Y \\ X & & Y \end{array} \quad (\text{B.16})$$

which we denote by $(\pi_X, \pi_Y) : R \rightarrow X \times Y$. The composition $R; R'$ of relations is a weak pullback $(\pi_R, \pi_{R'}) : R; R' \rightarrow R \times R'$ of π_Y and π'_Y as in the following diagram, defined such that $R; R' := \{(x, z) \mid \exists y \in Y. (x, y) \in R \text{ and } (y, z) \in R'\}$.

$$\begin{array}{ccccc} & & R; R' & & \\ & \swarrow \pi_R & & \searrow \pi_{R'} & \\ & R & & R' & \\ \pi_X \swarrow & & \pi_Y & \swarrow \pi'_Y & \searrow \pi'_Z \\ X & & Y & & Z \end{array} \quad (\text{B.17})$$

The relation lifting for a \mathbf{Set} -functor T is a functor $\text{Rel}_T(-)$ on the category \mathbf{Rel} of sets and relations between them. The following definition makes this precise.

Definition B.4.9 (Binary Relation Lifting in \mathbf{Set}). *Let T be a weak pullback preserving functor on \mathbf{Set} , the relation lifting for T is a functor $\text{Rel}_T(-)$ on \mathbf{Rel} defined on relations $(\pi_X, \pi_Y) : R \rightarrow X \times Y$ by the following epi-mono-factorisation.*

$$\begin{array}{ccc} TR & \xrightarrow{(T\pi_X, T\pi_Y)} & TX \times TY \\ \downarrow & \dashrightarrow & \uparrow \\ \text{Rel}_T(R) & \xrightarrow{(\pi_{TX}, \pi_{TY})} & TX \times TY \end{array} \quad (\text{B.18})$$

Concretely, we obtain the following definition.

$$Rel_T(R) := \{(x, y) \in TX \times TY \mid \exists z \in TR. (T\pi_X)(z) = x \text{ and } (T\pi_Y)(z) = y\} \quad (B.19)$$

The preservation of weak pullbacks is necessary to make $Rel_T(-)$ functorial, in that it commutes with relation composition as shown in Lemma B.4.12

The definition of binary relation lifting by factorisation generalises to n -ary relations as follows.

Definition B.4.10 (n -ary Relation Lifting in *Set*). *Let T be a weak pullback preserving functor on *Set*, the relation lifting for T is a functor $Rel_T(-)$ on *Rel* defined on relations $(\pi_{X_0}, \dots, \pi_{X_{n-1}}) : R \rightarrow \prod_{i \in n} X_i$ by the following epi-mono-factorisation.*

$$\begin{array}{ccc} TR & \xrightarrow{(T\pi_{X_0}, \dots, T\pi_{X_{n-1}})} & \prod_{i \in n} X_i \\ \downarrow & \searrow \text{---} (\pi_{TX_0}, \dots, \pi_{TX_{n-1}}) & \\ Rel_T(R) & \hookrightarrow & \end{array} \quad (B.20)$$

Concretely, $Rel_T(R) := \{(x_0, \dots, x_{n-1}) \in \prod_{i < n} TX_i \mid \exists z \in TR. (T\pi_{X_i})(z) = x_i \text{ for all } i < n\}$.

The preservation of weak pullbacks is necessary to make $Rel_T(-)$ functorial, in that it commutes with relation composition.

The definition of n -ary relation lifting for finite n does not add to the generality of the definition of binary relation liftings. In general, $(n + 1)$ -ary relation liftings reduce to n -ary relation liftings. We show only the case for $n = 2$, as this is relevant in Chapter 11.

Lemma B.4.11. *Let $(\pi_X, \pi_Y, \pi_Z) : R \rightarrow X \times Y \times Z$ be a ternary relation, then R is in bijection with a binary relation $(\pi_X, \pi_{R_{YZ}}) : R_{XYZ} \rightarrow X \times R_{YZ}$ where $(\pi'_Y, \pi'_Z) : R_{YZ} \rightarrow Y \times Z$ with $R_{YZ} := \{(y, z) \mid (x, y, z) \in R\}$, such that $R_{XYZ} := \{(x, (y, z)) \mid (x, y, z) \in R\}$. Then $Rel_T(R_{XYZ}) = (id_X, e_{YZ})(Rel_T(R))$ where $e_{YZ} : Rel_T(R_{YZ}) \rightarrow TR_{YZ}$.*

Proof. It is a fact of set theory that $\pi_Y = \pi'_Y \circ \pi_{R_{YZ}}$ and $\pi_Z = \pi'_Z \circ \pi_{R_{YZ}}$ commute, so that $T\pi_Y = T\pi'_Y \circ T\pi_{R_{YZ}}$ and $T\pi_Z = T\pi'_Z \circ T\pi_{R_{YZ}}$. Then the lemma reduces to the commutativity of all triangles in the following diagram, which follows from the definition of relation

liftings.

$$\begin{array}{ccccc}
 & TR & \xrightarrow{e} & Rel_T(R) & \\
 & \searrow^{T\pi_X} & \searrow^{T\pi_{R_{YZ}}} & \searrow^{\pi_{TR_{YZ}}} & \\
 & TX & & & \\
 & \nearrow^{\pi_{TX}} & \nearrow^{T\pi'_Y} & \nearrow^{\pi'_{TY}} & \\
 & TY & & & \\
 & \searrow^{\pi'_{TY}} & \searrow^{T\pi'_Z} & \searrow^{T\pi'_{TZ}} & \\
 & TZ & & & \\
 & \nearrow^{\pi_{TY}} & \nearrow^{e'} & \nearrow^{T\pi'_{TZ}} & \\
 & & & Rel_T(R_{YZ}) &
 \end{array}
 \tag{B.21}$$

□

The following lemma can be found for instance in [Kup06].

Lemma B.4.12. *Let T be a standard functor preserving weak pullbacks, let X , Y and Z be sets, and let R and R' be relations as below.*

1. $Rel_T(\Delta)_X = \Delta_{TX}$.
2. Let $(\pi_X, \pi_Y) : R \rightarrow X \times Y$ and $(\pi'_Y, \pi'_Z) : R' \rightarrow Y \times Z$, then $Rel_T(R; R') = Rel_T(R); Rel_T(R)'$.
3. $Rel_T(-)$ is monotone, so that $Rel_T(R) \subseteq Rel_T(R')$ if $R \subseteq R'$ for $(\pi_X, \pi_Y) : R \rightarrow X \times Y$ and $(\pi'_X, \pi'_Y) : R' \rightarrow X \times Y$.
4. $Rel_T(-)$ commutes with taking converse $Rel_T(R) = Rel_T(R)^\smile$, where $(\pi_X, \pi_Y) : R \rightarrow X \times Y$.

Proof. 1. Δ_X is a relation $(\pi_X, \pi'_X) : \Delta_X \rightarrow X \times X$, where $\pi_X = \pi'_X$ and thus $T\pi_X = T\pi'_X$.

Thus by definition of relation liftings, $(\pi_{TX}, \pi'_{TX}) : Rel_T(\Delta_X) \rightarrow X \times X$ is given by

$\pi_{TX} = \pi'_{TX}$, so that $Rel_T(\Delta_X) = \Delta_{TX}$.

2. $Rel_T(R; R') = Rel_T(R); Rel_T(R')$ follows from the commutativity of the following diagram.

$$\begin{array}{ccccccc}
 & & Rel_T(R; R') & \xleftarrow{e^*} & T(R; R') & & \\
 & & \downarrow \pi_{TR} & & \downarrow T\pi_{R'} & & \\
 Rel_T(R) & \xleftarrow{e} & TR & \xleftarrow{T\pi_R} & TR' & \xrightarrow{e'} & Rel_T(R)' \\
 \downarrow \pi_{TX} & & \downarrow T\pi_X & & \downarrow T\pi'_Y & & \downarrow \pi'_{TZ} \\
 TX & & TY & & TY & & TZ
 \end{array}
 \tag{B.22}$$

3. Monotonicity of $Rel_T(-)$ follows from the commutativity of the following diagram and the definition of relation liftings, that is (B.19).

$$\begin{array}{ccc}
 Rel_T(R) & \xleftarrow{e} & TR \\
 & \searrow (\pi_{TX}, \pi_{TY}) & \swarrow (T\pi_X, T\pi_Y) \\
 & TX \times TY & \\
 & \swarrow (\pi'_{TX}, \pi'_{TY}) & \searrow (T\pi'_X, T\pi'_Y) \\
 Rel_T(R') & \xleftarrow{e'} & TR'
 \end{array}
 \quad \begin{array}{c} \subseteq \\ \downarrow \end{array}
 \quad (B.23)$$

4. The commutativity of $Rel_T(-)$ with taking converses follows from the commutativity of the following diagram and the definition of relation liftings, that is (B.19).

$$\begin{array}{ccc}
 Rel_T(R) & \xleftarrow{e} & TR \\
 & \searrow (\pi_{TX}, \pi_{TY}) & \swarrow (T\pi_X, T\pi_Y) \\
 & TX \times TY & \\
 & \downarrow \sigma & \\
 & TY \times TX & \\
 & \swarrow (\pi_{TY}, \pi_{TX}) & \searrow (T\pi_Y, T\pi_X) \\
 Rel_T(R) & \xleftarrow{e'} & TR'
 \end{array}
 \quad \begin{array}{c} (-)^{\smile} \downarrow \\ T(-)^{\smile} \downarrow \end{array}
 \quad (B.24)$$

where $\sigma : TX \times TY \rightarrow TY \times TX$ takes $(\alpha, \beta) \mapsto (\beta, \alpha)$.

□

B.4.3 Bases and Redistributions

Definition B.4.13 (Base). *Let T be a standard functor on Set , and let X be a set and $\alpha \in T_\omega X$, we define the base of α , such that*

$$Base(\alpha) := \bigcap \{Y \subseteq_\omega X \mid \alpha \in TY\} \quad (B.25)$$

The following lemma is Lemma 2.3.6 of [Kup06].

Lemma B.4.14. *Let X and α be as in Definition B.4.13, then $Base(\alpha)$ is the smallest finite subset Y of X such that $\alpha \in TY$.*

Example B.4.15. Let $T := (-) \times (-) \times (-)$, the base of $(1, 2, 1) \in T_\omega \mathbb{N}$ is $\{1, 2\}$.

The following lemma is known.

Lemma B.4.16. Base is a strict natural transformation $T \Rightarrow \mathcal{P}$ for all standard functors T , in particular $\text{Base}(Tf) = (\mathcal{P}f)\text{Base}$ for any function f .

Proof. Let X and X' be sets, $f : X \rightarrow X'$ a function, and let $\alpha \in TX$.

$$\begin{aligned} (\mathcal{P}f)\text{Base}(\alpha) &= (\mathcal{P}f) \bigcap \{Y \mid Y \subseteq X, \alpha \in TY\} = \\ &= \bigcap \{(\mathcal{P}f)Y \mid Y \subseteq X, \alpha \in TY\} = \\ &= \bigcap \{Y \mid Y \subseteq (\mathcal{P}f)X, (Tf)\alpha \in TY\} = \\ &= \bigcap \{Y \mid Y \subseteq X', (Tf)\alpha \in TY\} = \text{Base}((Tf)\alpha) \end{aligned}$$

□

Definition B.4.17 (Redistributions). A set $\Phi \in T\mathcal{P}X$ is a redistribution of a set $A \in \mathcal{P}TX$ if $(\alpha, \Phi) \in \text{Rel}_T(\in)$ for all $\alpha \in A$. A redistribution $\Phi \in \mathcal{P}_\omega T_\omega X$ is slim, if $\Phi \in T_\omega \mathcal{P}_\omega (\bigcup_{\alpha \in A} \text{Base}(\alpha))$. We denote the set of slim redistributions of a set A as $\text{SRD}(A)$.

Appendix C

Coalgebras

Definition C.0.18 (Coalgebras and Their Morphisms). *Let T be a functor on a category C and S an object of C . We call a morphism $\sigma : S \rightarrow TS$ in C a T -coalgebra and T the transition type of σ . A T -coalgebra morphism between T -coalgebras $\sigma : S \rightarrow TS$ and $\delta : Y \rightarrow TY$ is a morphism $f : S \rightarrow S'$ making the following commute.*

$$\begin{array}{ccc} S & \xrightarrow{\sigma} & TS \\ f \downarrow & & \downarrow Tf \\ S' & \xrightarrow{\delta} & TS' \end{array} \quad (\text{C.1})$$

T -coalgebras and their morphisms form a category, which we denote by $\text{Coalg}_C T$.

Definition C.0.19 (Final Coalgebras). *The final object $\langle Z, \xi \rangle$ of $\text{Coalg}_C T$ is called the final T -coalgebra.*

Definition C.0.20 (Coproducts of Coalgebras). *Let $\mathbb{S} = \langle S, \sigma \rangle$ and $\mathbb{S}' = \langle S', \sigma' \rangle$ be T -coalgebras in Set for a standard functor T , their coproduct $\mathbb{S} + \mathbb{S}'$ in $\text{Coalg}_{\text{Set}} T$ is the T -coalgebra, $\langle S + S', \sigma + \sigma' \rangle$ defined such that $(\sigma + \sigma')(s) = \sigma(s)$ and $(\sigma + \sigma')(s') = \sigma'(s')$ for all $s \in S$ and $s' \in S'$.*

Pointed coalgebras are coalgebras in which we distinguish a point. Pointed coalgebras are not necessarily rooted in the sense that every point is reachable from the distinguished point as in Section 11.1.1. In this dissertation pointed coalgebras play mainly a role as the structures recognised by coalgebra automata.

Definition C.0.21 (Pointed Coalgebras). A pointed T -coalgebra is a structure $\langle \sigma : S \rightarrow TS, s \rangle$ where $s \in S$ is a point of σ . A coalgebra morphism between pointed T -coalgebras $\mathbb{S} = \langle S, \sigma, s_I \rangle$ and $\mathbb{S}' = \langle S', \sigma', s'_I \rangle$ is a T -coalgebra morphism f between $\langle S, \sigma \rangle$ and $\langle S', \sigma' \rangle$ preserving the distinguished point, that is $f(s_I) = s'_I$.

In the following we review a generic notion of bisimilarity valid in the category *Set*. For an overview of other definitions of bisimilarity see Staton [Sta11].

Definition C.0.22 (Observational Equivalence). States s and s' in T -coalgebras $\mathbb{S} = \langle S, \sigma \rangle$ and $\mathbb{S}' = \langle S', \sigma' \rangle$, respectively, are said to be observationally equivalent, if there is a T -coalgebra $\mathbb{Z} = \langle Z, \xi \rangle$ with T -coalgebra morphisms $f : \mathbb{S} \rightarrow \mathbb{Z}$ and $g : \mathbb{S}' \rightarrow \mathbb{Z}$ with $f(s) = g(s')$.

If T preserves weak pullbacks, observational equivalence as above is equivalent to the bisimilarity as the largest T -bisimulation as follows. The definition is due to Aczel and Mendler [AM89].

Definition C.0.23 (T -Bisimilarity and T -Bisimulation between T -Coalgebras in *Set*). A T -bisimulation between T -coalgebras $\mathbb{S} = \langle S, \sigma \rangle$ and $\mathbb{S}' = \langle S', \sigma' \rangle$ in *Set* is a relation $R \subseteq S \times S'$ such that there is a morphism $r : R \rightarrow TR$ making both sides of the following commute.

$$\begin{array}{ccccc}
 TS & \xleftarrow{T\pi_1} & TR & \xrightarrow{T\pi_2} & TS' \\
 \uparrow \sigma & & \uparrow r & & \uparrow \sigma' \\
 S & \xleftarrow{\pi_1} & R & \xrightarrow{\pi_2} & S'
 \end{array} \tag{C.2}$$

T -Bisimilarity between \mathbb{S} and \mathbb{S}' is the largest T -bisimulation between \mathbb{S} and \mathbb{S}' .

Hermido and Jacobs [HJ97] observed that the previous definition can be reformulated in terms of relation liftings as follows.

Definition C.0.24 (T -Bisimilarity and T -Bisimulation between T -Coalgebras in *Set*). A T -bisimulation between T -coalgebras $\mathbb{S} = \langle S, \sigma \rangle$ and $\mathbb{S}' = \langle S', \sigma' \rangle$ in *Set* is a relation $R \subseteq S \times S'$ such that there is a morphism $\rho : R \rightarrow \text{Rel}_T(R)$ making both sides of the

following commute.

$$\begin{array}{ccccc}
 TS & \xleftarrow{\pi_{TS}} & Rel_T(R) & \xrightarrow{\pi_{TS'}} & TS' \\
 \uparrow \sigma & & \uparrow r & & \uparrow \sigma' \\
 S & \xleftarrow{\pi_S} & R & \xrightarrow{\pi_{S'}} & S'
 \end{array} \tag{C.3}$$

T -Bisimilarity between \mathbb{S} and \mathbb{S}' is the largest T -bisimulation between \mathbb{S} and \mathbb{S}' .

Definition C.0.23 coincides with Definition 1.5.1 of T -bisimulation and T -bisimilarity for standard functors T , as a T -bisimulation R between T -coalgebras \mathbb{S} and \mathbb{S}' as above is a T -bisimulation on $\mathbb{S} + \mathbb{S}'$. We need Definition C.0.23 for the following definition of T -bisimulation between pointed T -coalgebras.

Definition C.0.25 (T -Bisimilarity and T -Bisimulation for Pointed T -Coalgebras in Set). A T -bisimulation between pointed T -coalgebras $\mathbb{S} = \langle S, \sigma, s_I \rangle$ and $\mathbb{S}' = \langle S', \sigma', s'_I \rangle$ is a T -bisimulation R between $\langle S, \sigma \rangle$ and $\langle S', \sigma' \rangle$ such that $(s_I, s'_I) \in R$. T -bisimilarity between \mathbb{S} and \mathbb{S}' is the largest T -bisimulation between \mathbb{S} and \mathbb{S}' .

In this dissertation we the presentation of T -bisimilarity in terms of a cone of \mathbb{S} over Seq^T .

Lemma C.0.26. For any T -coalgebra $\mathbb{S} = \langle S, \sigma \rangle$, any T -bisimulation R on \mathbb{S} is contained in the kernel of the cone $f : S \Rightarrow Seq^T$ defined inductively such that

$$f_0 := !_S \text{ and } f_{n+1} := T f_n \circ \sigma. \tag{C.4}$$

Proof. Similar to \mathbb{S} , R has a cone over the Seq^T , such that

$$g_0 := !_S \text{ and } g_{n+1} := T g_n \circ r \tag{C.5}$$

as in the following diagram.

$$\begin{array}{ccccccc}
 & & R & \xrightarrow{r} & TR & \xrightarrow{Tg_n} & T^{n+1}\{*\} \\
 g_0 \swarrow & & \searrow g_n & & \searrow g_{n+1} & & \searrow \\
 \{*\} & \leftarrow \dots \leftarrow & T^n\{*\} & \xleftarrow{T^n !_S} & T^{n+1}\{*\} & \leftarrow \dots
 \end{array} \tag{C.6}$$

We use the commutativity of Diagram 1.11 to prove that $R \subseteq \ker f_n$ for all $n < \omega$ for all n .

$$\begin{array}{ccc}
 S & \xrightarrow{\sigma} & TS \\
 \pi_1 \swarrow & & \nearrow T\pi_1 \\
 R & \xrightarrow{r} & TR \\
 \pi_2 \swarrow & & \nearrow T\pi_2 \\
 f_n \searrow & & \nearrow Tf_n \\
 & T^n\{*\} & \xleftarrow{T^{n+1}\{*\}} T^{n+1}\{*\} \\
 & & \downarrow Tg_n \\
 & & T^n\{*\}
 \end{array}
 \quad (C.7)$$

In the base case $n = 0$, $g_0 = f_0 \circ \pi_0 = f_0 \circ \pi_1$ by finality of 1. The induction step we obtain as follows.

$$\ker f_{n+1} = \ker(Tf_n \circ \sigma) \supseteq r[TR] \supseteq R \quad (C.8)$$

□

List of Tables

1.1	Concepts from Algebra and Coalgebra	2
1.2	Examples of Coalgebras	3
1.3	Examples of Transition Functions for Nondeterministic Automata	6
1.4	Examples of Branching Types for Word Automata	6
3.1	Examples of Monads	30
7.1	Notational Convention for Coalgebraic Logic	77
8.1	Comparison of Moss' Coalgebraic Logics and Finite Trace Logics	88
10.1	Acceptance Games for Nondeterministic Coalgebra Automata	127
10.2	Branching Types of Coalgebra Automata	127
10.3	Acceptance Games for Alternating Automata	128
10.4	Acceptance Games for Semi-Transalternating Automata	129
10.5	Acceptance Games for Transalternating Automata	130
11.1	Example of Pumping	146

List of Figures

7.1	Axiom System M for Finitary Coalgebraic Logics	79
10.1	Transformation of Semi-Transalternating into Alternating Automata . . .	133
11.1	Examples for Deleting and Pumping Once and Iteratedly in <i>Set</i> -Coalgebras	148

Bibliography

- [Abr90] Samson Abramsky. *The lazy lambda calculus*, pages 65–116. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1990.
- [Acz88] Peter Aczel. *Non-Well-Founded Sets*. CSLI, Stanford, 1988.
- [AGT09] Jiří Adámek, Heinz-Peter Gumm, and Vera Trnkova. Presentation of Set Functors: A Coalgebraic Perspective. *J Logic Computation*, pages exn090+, February 2009.
- [AK79] Jiří Adámek and Václav Koubek. Least Fixed Point of a Functor. *J. Comput. Syst. Sci.*, 19(2):163–178, 1979.
- [AM89] Peter Aczel and Nax P. Mendler. A Final Coalgebra Theorem. In David H. Pitt, David E. Rydeheard, Peter Dybjer, Andrew M. Pitts, and Axel Poigné, editors, *Category Theory and Computer Science*, volume 389 of *Lecture Notes in Computer Science*, pages 357–365. Springer, 1989.
- [AR94] Jiří Adámek and Jiří Rosický. *Locally Presentable and Accessible Categories*, volume 189 of *London Mathematical Society Lecture Notes Series*. Cambridge University Press, 1st edition edition, 1994.
- [ARV10] Jiří Adámek, Jiří Rosický, and Enrico M. Vitale. *Algebraic Theories: A Categorical Introduction to General Algebra (Cambridge Tracts in Mathematics)*. Cambridge University Press, December 2010.
- [AT90] Jiří Adámek and Vera Trnkova. *Automata and Algebras in Categories*. Kluwer Academic Publishers, Norwell, MA, USA, 1990.

- [Bar81] Henk Barendregt. *The Lambda Calculus - Its Syntax and Semantics*, volume 103 of *Studies in Logic*. North-Holland, 1981.
- [BdRV01] Patrick Blackburn, Maarten de Rijke, and Yde Venema. *Modal Logic*, volume 53 of *Cambridge Tracts in Theoretical Computer Science*. Cambridge University Press, 2001.
- [Bec69] Jon Beck. Distributive laws. In B. Eckmann, editor, *Seminar on Triples and Categorical Homology Theory*, volume 80 of *Lecture Notes in Mathematics*, chapter 6, pages 119–140. Springer Berlin / Heidelberg, 1969.
- [BHPS61] Yehoshua Bar-Hillel, Micha A. Perles, and Eli Shamir. On formal properties of simple phrase structure grammars. *Zeitschrift für Phonetik, Sprachwissenschaft und Kommunikationsforschung*, (14):143–172, 1961.
- [BIM95] Bard Bloom, Sorin Istrail, and Albert R. Meyer. Bisimulation can’t be traced. *J. ACM*, 42(1):232–268, January 1995.
- [BM04] Jon Barwise and Lawrence S. Moss. *Vicious Circles (Center for the Study of Language and Information - Lecture Notes)*. Center for the Study of Language and Inf, August 2004.
- [BMR97] Stefano Bistarelli, Ugo Montanari, and Francesca Rossi. Semiring-based constraint satisfaction and optimization. *J. ACM*, 44(2):201–236, 1997.
- [BPV08] Marta Bílková, Alessandra Palmigiano, and Yde Venema. Proof systems for the coalgebraic cover modality. In Carlos Areces and Robert Goldblatt, editors, *Advances in Modal Logic*, pages 1–21. College Publications, 2008.
- [CÎ0] Corina Cîrstea. Generic Infinite Traces and Path-Based Coalgebraic Temporal Logics. In Bart P. F. Jacobs, Milad Niqui, Jan J. M. M. Rutten, and Alexandra Silva, editors, *Coalgebraic Methods in Computer Science 2010*, volume 264, pages 83–103. Elsevier, 2010.

- [DR95] Volker Diekert and Grzegorz Rozenberg. *The Book of Traces*. World Scientific Publishing Co., Inc., River Edge, NJ, USA, 1995.
- [EJ91] E. Allen Emerson and Charanijt S. Jutla. Tree automata, mu-Calculus and Determinacy. In *Proceedings of the 32nd annual symposium on Foundations of computer science*, pages 368–377, Los Alamitos, CA, USA, 1991. IEEE Computer Society Press.
- [Gol10] Jonathan S. Golan. *Semirings and their Applications*. Springer, softcover reprint of hardcover 1st ed. 1999 edition, December 2010.
- [GTW02] Erich Grädel, Wolfgang Thomas, and Thomas Wilke, editors. *Automata, Logics, and Infinite Games: A Guide to Current Research*. Springer-Verlag New York, Inc., New York, NY, USA, 2002.
- [Gum99] Heinz P. Gumm. *Elements Of The General Theory of Coalgebras*, 1999.
- [Has08] Ichiro Hasuo. *Tracing Anonymity with Coalgebras*. PhD thesis, Radboud Universiteit Nijmegen, March 2008.
- [HJ97] Claudio Hermida and Bart Jacobs. Structural Induction and Coinduction in a Fibrational Setting. *Information and Computation*, 145:107–152, 1997.
- [HJS06] Ichiro Hasuo, Bart P. F. Jacobs, and Ana Sokolova. Generic Trace Theory. In *International Workshop on Coalgebraic Methods in Computer Science (CMCS 2006)*, volume 164 of *Elect. Notes in Theor. Comp. Sci.*, pages 47–65. Elsevier, 2006.
- [HJS07] Ichiro Hasuo, Bart P. F. Jacobs, and Ana Sokolova. Generic Trace Semantics via Coinduction. *ArXiv e-prints*, October 2007.
- [HK07] Ichiro Hasuo and Yoshinobu Kawabe. Probabilistic Anonymity Via Coalgebraic Simulations. In *ESOP’07 Proceedings of the 16th European conference on Programming*, pages 379–394. Springer-Verlag Berlin, Heidelberg, 2007.

- [HMu03] John E. Hopcroft, Rajeev Motwani, and Jeffrey D. Ullman. *Introduction to automata theory, languages, and computation - international edition (2. ed)*. Addison-Wesley, 2003.
- [HTS02] Michael Hohmuth, Hendrik Tews, and Shane G. Stephens. Applying source-code verification to a microkernel – The VFiasco project. In *EW 10 Proceedings of the 10th workshop on ACM SIGOPS European workshop*. ACM, 2002.
- [Jac94] Bart P. F. Jacobs. Semantics of Weakening and Contraction. *Ann. Pure Appl. Logic*, 69(1):73–106, 1994.
- [Jac04] Bart P. F. Jacobs. Trace Semantics for Coalgebras. *Electronic Notes in Theoretical Computer Science*, 106:167–184, December 2004.
- [Jac05] Bart P. F. Jacobs. Introduction to Coalgebra. Towards Mathematics of States and Observations. 2005.
- [Jay95] C. Barry Jay. A Semantics for Shape. *Science of Computer Programming*, 25:25–251, 1995.
- [Jec06] Thomas Jech. *Set Theory*. Springer, 3rd edition, April 2006.
- [Joh86] Peter T. Johnstone. *Stone Spaces (Cambridge Studies in Advanced Mathematics)*. Cambridge University Press, August 1986.
- [Kel05] Gregory M. Kelly. Basic Concepts in Enriched Category Theory, 2005.
- [Kis05] Christian Kissig. Introducing Modal Fixed-Point Operators into CCSL, February 2005.
- [Kis07] Christian Kissig. Satisfiability of S2S. Master’s thesis, Universiteit van Amsterdam, August 2007.
- [KK10] Christian Kissig and Alexander Kurz. Generic Trace Logics. 2010.
- [KKV04] Clemens Kupke, Alexander Kurz, and Yde Venema. Stone coalgebras. *Theor. Comput. Sci.*, 327:109–134, October 2004.

- [KKV08] Clemens Kupke, Alexander Kurz, and Yde Venema. Completeness of the finitary Moss logic. In Carlos Areces and Rob Goldblatt, editors, *Advances in Modal Logic*, 2008.
- [KL09] Clemens Kupke and Raul A. Leal. Characterising Behavioural Equivalence: Three Sides of One Coin. In Alexander Kurz, Marina Lenisa, and Andrzej Tarlecki, editors, *CALCO*, volume 5728 of *Lecture Notes in Computer Science*, pages 97–112. Springer, 2009.
- [Kli07] Bartek Klin. Coalgebraic modal logic beyond Sets. In *In MFPS XXIII*, volume 173, pages 177–201, 2007.
- [Koc70] Anders Kock. Monads on symmetric monoidal closed categories. *Archiv der Mathematik*, 21(1):1–10, December 1970.
- [Kup06] Clemens Kupke. *Finitary Coalgebraic Logic*. PhD thesis, Universiteit van Amsterdam, 2006.
- [Kur01a] Alexander Kurz. Course on Coalgebras and Modal Logic, October 2001.
- [Kur01b] Alexander Kurz. *Logics for Coalgebras and Applications to Computer Science*. PhD thesis, Ludwig-Maximilians-Universität München, May 2001.
- [KV05] Clemens Kupke and Yde Venema. Closure Properties of Coalgebra Automata. In *LICS '05: Proceedings of the 20th Annual IEEE Symposium on Logic in Computer Science (LICS' 05)*, pages 199–208, Washington, DC, USA, 2005. IEEE Computer Society.
- [KV08] Clemens Kupke and Yde Venema. Coalgebraic Automata Theory: Basic Results. *Computing Research Repository*, November 2008.
- [KV09] Christian Kissig and Yde Venema. Complementation of Coalgebra Automata. In Alexander Kurz, Marina Lenisa, and Andrzej Tarlecki, editors, *Calco 2009*, *Lecture Notes in Computer Science*, pages 81–96. Springer, September 2009.

- [Lea07] Raul A. Leal. Expressivity of Coalgebraic Modal Languages. Master's thesis, Universiteit van Amsterdam, August 2007.
- [Lea08] Raul A. Leal. Predicate Liftings Versus Nabla Modalities. *Electr. Notes Theor. Comput. Sci.*, 203(5):195–220, 2008.
- [Mac98] Saunders MacLane. *Categories for the Working Mathematician*. Graduate Texts in Mathematics. Springer, New York, 2nd edition edition, 1998.
- [MM07] Ernie Manes and Philip Mulry. Monad Compositions I: General Constructions and Distributive Laws. *Theory and Applications of Categories*, 18(7):172–208, 2007.
- [Mos91] Andrzej Mostowski. Games with forbidden positions. Technical Report 78, University of Gdansk, 1991.
- [Mos99] Lawrence S. Moss. Coalgebraic Logic. *Annals of Pure and Applied Logic*, 96(1-3):277–317, 1999.
- [Pat01a] Dirk Pattinson. *Expressivity Results in the Modal Logic of Coalgebras*. PhD thesis, Universität München, 2001.
- [Pat01b] Dirk Pattinson. Semantical Principles in the Modal Logic of Coalgebras. In *STACS '01: Proceedings of the 18th Annual Symposium on Theoretical Aspects of Computer Science*, pages 514–526, London, UK, 2001. Springer-Verlag.
- [Pat03a] Dirk Pattinson. An Introduction to the Theory of Coalgebras, 2003.
- [Pat03b] Dirk Pattinson. Coalgebraic Modal Logic: Soundness, Completeness and Decidability of Local Consequence. *Theoretical Computer Science*, 309, 2003.
- [Pat04] Dirk Pattinson. Expressive Logics for Coalgebras via Terminal Sequence Induction. *Notre Dame Journal of Formal Logic*, 45(1):19–33, 2004.

- [PT99] John Power and Daniele Turi. A Coalgebraic Foundation for Linear Time Semantics. In *In Category Theory and Computer Science*, volume 29, pages 2–9, 1999.
- [Rab69] Michael O. Rabin. Decidability of Second Order Theories and Automata on Infinite Trees. *Transactions of the American Mathematical Society*, 141:1–35, 1969.
- [Rem00] Didier Remy. Using, Understanding, and Unraveling - The Ocaml Language - From Practice to Theory and vice versa, 2000.
- [RJ97] Jan J. M. M. Rutten and Bart P. F. Jacobs. A Tutorial on (Co)Algebras and (Co)Induction. *Bulletin of the EATCS*, 62:222–259, 1997.
- [RTJ01] Jan Rothe, Hendrik Tews, and Bart P. F. Jacobs. The Coalgebraic Class Specification Language CCSL. *Journal of Universal Computer Science*, 7:175–193, 2001.
- [Rut96] Jan J. M. M. Rutten. *Universal Coalgebra: A Theory of Systems*, 1996.
- [Rut02] Jan J. M. M. Rutten. Coinductive Counting With Weighted Automata. *J. Autom. Lang. Comb.*, 2002.
- [Saf88] Shmuel Safra. On the Complexity of omega-Automata. In *Proceedings of the 29th Annual Symposium on Foundations of Computer Science FoCS '88*, pages 319–327. IEEE Computer Society Press, 1988.
- [Sch05] Lutz Schröder. Expressivity of Coalgebraic Modal Logic: The Limits and Beyond. In Vladimiro Sassone, editor, *Foundations of Software Science And Computation Structures*, volume 3441 of *Lecture Notes in Computer Science*, pages 440–454. Springer; Berlin; <http://www.springer.de>, 2005.
- [Seg95] Roberto Segala. A compositional trace-based semantics for probabilistic automata. In *6th Intl. Conf. on Concurrency Theory (CONCUR '95)*, volume 962 of *Lecture Notes in Computer Science*. Springer, 1995.

- [Sip96] Michael Sipser. *Introduction to the Theory of Computation*. PWS Pub. Co., 1 edition, December 1996.
- [SP82] Michael B. Smyth and Gordon D. Plotkin. The category theoretic solution of recursive domain equations. *SIAM Journal of Computing*, 11:761–783, 1982.
- [Sta11] Sam Staton. Relating coalgebraic notions of bisimulation. *CoRR*, abs/1101.4223, 2011.
- [vB77] Johan F. A. K. van Benthem. *Modal Correspondence Theory*. PhD thesis, Universiteit van Amsterdam, 1977.
- [vB02] Johan F. A. K. van Benthem. Extensive Games as Process Models. *Journal of Logic, Language and Information*, 11(3):289–313, June 2002.
- [Ven04] Yde Venema. Automata and Fixed Point Logics for Coalgebras. *Electronic Notes in Computer Science*, 106:355–375, 2004.
- [vG90] Rob J. van Glabbeek. The linear time - branching time spectrum. pages 278–297. 1990.
- [Wor05] James Worrell. On the final sequence of a finitary set functor. *Theoretical Computer Science*, 338(1-3):184–199, June 2005.
- [Zie98] Wiesław Zielonka. Infinite games on finitely coloured graphs with applications to automata on infinite trees. *Theoretical Computer Science*, 200(1-2):135–183, June 1998.

Index

- DCPO*, 122
- DCPO*_⊥, 122
- SRD*, 134
- Set*-Coalgebras
 - Pumping, 108
 - Pumping Property, 110
 - Pumping Situation, 108
 - Unravelling, 106
- DCPO*_⊥-Enrichment of Kleisli-Categories, 42
- ∅-Standard Functors, 130
- Seq*^T, 125
- Seq*_T, 125
- (Strong) Symmetric Monoidal Categories, 130
- (Strong) Symmetric Monoidal Closed Categories, 130
- Acceptance Games
 - Basic Positions, 96
- Acyclic Coalgebras, 105
- Adjunctions, 126
- Algebras, 2
- Ambimorphic Objects, 68
- Automata
 - Coalgebra Automata, 95
- Base, 133
- Base Dualisation Map, 64
- Basic Positions in Acceptance Games, 96
- Behavioural Equivalence, 2
- Bisimilarity, 6, 136
- Bisimilarity in *Set*, 7
- Bisimulation, 136
- Bisimulation in *Set*, 7
- Categories, 123
 - Comma Categories, 126
- Closed Categories, 130
- Coalgebra Automata, 95
- Coalgebra Automata, 55, 96
 - Acceptance Behaviour, 55
 - Acceptance Games, 94, 96
 - Algebraic Definition, 95
 - Alternating Coalgebra Automata, 95
 - Equivalence of Coalgebra Automata, 94
 - Languages of Coalgebra Automata, 94
 - Logical Form, 94
 - Semi-Transalternating Coalgebra Automata, 95
 - Transalternating Coalgebra Automata, 96
- Coalgebra Semantics, 6

- Coalgebraic Modal Logics, 11
- Coalgebras, 1, 2, 135
 - Pointed Coalgebras, 135
- Coequalisers, 128
- Coinduction, 2
- Comma Categories, 126
- Commutative Monads, 30
- Comparison Functors K , 22
- Compatible Morphisms, 123
- Complete Partial Orders, 121
- Congruence, 2
- Contravariant Functors, 124
- Copowers in Sets, 127
- Coseparators, 69
- Counits of Adjunctions, 126
- Covariant Functors, 124
- Cyclic Coalgebras, 105
- Deleting in Coalgebras in *Set*, 110
- Depth of States in Coalgebras in *Set*, 107
- Depth One Formulas, 63
- Diagrams, 124
- Directed Complete Partial Orders, 121
- Directed Sets, 122
- Discrete Categories, 124
- Discrete Diagrams, 124
- Distributive Laws, 33, 34
- Dual Categories, 124
- Eilenberg-Moore-Algebras, 21, 22
- Endofunctors, 124
- Epimorphisms, 126
- Equalisers, 128
- Equivalence of Coalgebra Automata, 94
- Faithful Functors, 130
- Final Coalgebra, 2
- Final Coalgebra Semantics, 6
- Final Coalgebras, 7
- Finitarisation of *Set*-Functors, 130
- Finitary Functors, 128
- Finite Trace Equivalence, 6, 46
- Finite Trace Logics, 70
- Finite Trace Semantics, 6, 45
- Free Logic Functors, 70
- Functions, 121
- Functors
 - Faithful Functors, 130
 - Multi-Endofunctors, 125
- Game Bisimulations, 89
- Generic Trace Theory, 41
- Graphs of Coalgebras, 104
- Images of Diagrams, 125
- Induction, 2
- Infinite Trace Equivalence, 6
- Infinite Trace Semantics, 6
- Initial Algebra, 2
- Kleisli Category, 22
- Kleisli Construction, 22
- Languages of Coalgebra Automata, 94

- Left Kan-Extension, 129
- Logic Functors, 70
 - Denotation, 70
- Modal Correspondence Theory, 11
- Modal Logics, 11
- Monads, 21
 - Commutative Monads, 30
 - Double Strength Laws, 30
 - Strength Laws, 30
 - Strong Monads, 30
- Monic Natural Transformations, 126
- Monomorphisms, 126
- Multi-Endofunctors, 125
- Multihomomorphic Extensions, 31
- Natural Transformations, 125
- Nondeterministic Coalgebra Automata, 55
- Normal Parity Graph Games, 89
- Normalised Parity Graph Games, 88
- Observations, 2
- Operations, 2
- Parity Graph Games, 83–90
 - Arenas, 83
 - Basic Positions, 85, 90
 - Equivalence of Parity Graph Games, 84
 - Game Bisimulation, 89, 90
 - History-free Determinacy, 84
 - Initial Plays, 83
 - Local Game Trees, 86
 - Local Games, 87
 - Local Strategies, 87
 - Normal Parity Graph Games, 89
 - Normalised Parity Graph Games, 88
 - Plays, 83
 - Power of Players, 86
 - Strategies, 84
 - Terminal Positions, 83
 - Total Plays, 83
 - Unravelling, 85
 - Winning Condition, 83
 - Winning Positions, 84
 - Winning Regions, 84
 - Winning Strategies, 84
- Partial Orders, 121
- Pointed Coalgebras, 1
- Preorders, 121
- Pullbacks, 126
- Pumping, 108
- Pumping Lemma, 103
- Pumping Property, 103
- Pumping Situation, 108
- Redistributions, 134
- Regular Languages, 103
- Relation Liftings, 131, 132
- Relations, 121, 131
- Semi-Transalternating Coalgebra Automata, 95
- Set Coalgebras

- Reachable States, 105
- Shapely Functors, 34
- Slim Redistributions, 134
- Standard Functors, 130
- Strong Monads, 30
- Strong Monoidal Categories, 129
- Transition Types, 1
- Tree-like Coalgebras, 105
- Units of Adjunctions, 126
- Universal Coalgebra, 135
- Upward Closure, 121
- Weak Pullbacks, 126