An Abstract Coalgebraic Approach to Process Equivalence for Well-Behaved Operational Semantics

Bartosz Klin

PhD Dissertation

Department of Computer Science
University of Aarhus
Denmark

An Abstract Coalgebraic Approach to Process Equivalence for Well-Behaved Operational Semantics

A Dissertation
Presented to the Faculty of Science
of the University of Aarhus
in Partial Fulfilment of the Requirements for the
PhD Degree

by
Bartosz Klin
February 25, 2004

Abstract

This thesis is part of the programme aimed at finding a mathematical theory of well-behaved structural operational semantics. General and basic results shown in 1997 in a seminal paper by Turi and Plotkin are extended in two directions, aiming at greater expressivity of the framework.

The so-called bialgebraic framework of Turi and Plotkin is an abstract generalization of the well-known structural operational semantics format GSOS, and provides a theory of operational semantic rules for which bisimulation equivalence is a congruence.

The first part of this thesis aims at extending that framework to cover other operational equivalences and preorders (e.g. trace equivalence), known collectively as the van Glabbeek spectrum. To do this, a novel coalgebraic approach to relations on processes is desirable, since the usual approach to coalgebraic bisimulations as spans of coalgebras does not extend easily to other known equivalences on processes. Such an approach, based on fibrations of test suites, is presented. Based on this, an abstract characterization of congruence formats is given, parametrized by the relation on processes that is expected to be compositional. This abstract characterization is then specialized to the case of trace equivalence, completed trace equivalence and failures equivalence. In the two latter cases, novel congruence formats are obtained, extending the current state of the art in this area of research.

The second part of the thesis aims at extending the bialgebraic framework to cover a general class of recursive language constructs, defined by (possibly unguarded) recursive equations. Since unguarded equations may be a source of divergence, the entire framework is interpreted in a suitable domain category, instead of the category of sets and functions. It is shown that a class of recursive equations called regular equations can be merged seamlessly with GSOS operational rules, yielding well-behaved operational semantics for languages extended with recursive constructs.

Acknowledgements

I want to express my deepest gratitude to my supervisor, Peter D. Mosses, for all the trust he has given me during my PhD studies in Århus. The fact that I chose to focus my studies in a field not directly related to his area of research did not prevent him from giving me much needed encouragement and continuous guidance. I can only hope I have deserved the freedom and support I have had for those years.

Most results presented in my thesis were directly or indirectly inspired by enlightening ideas of Gordon Plotkin. I am grateful to him for inviting me to Edinburgh for a study visit, and for the support he offered me. In Edinburgh, Daniele Turi generously spent a lot of his time introducing me to many new ideas; the discussions about mathematics I had with him were the most exciting ones I have ever experienced.

Many thanks to Jan Rutten and Marcelo Fiore for having accepted to evaluate this thesis. Having them as examiners is certainly a great honour for me.

I would like to thank all my friends at BRICS (Alex, Branimir, Claus, Daniele, Darek, Emanuela, Frank, Gabriel, Gosia, Jesus, Jiri, Jooyong, Kirill, Maciej, Marco, Mikkel, Paulo, Saurabh and Vanda) for the wonderful time I have had with them. I am especially grateful to Paweł Sobociński, who was the first person to get excited about my work on congruence formats, and who effectively forced me to pursue this line of research.

Many fruitful discussions I have had with Paweł helped to develop the results presented in this thesis, and his numerous comments improved its presentation. I am also grateful to the anonymous referee for the *Journal of Logic and Algebraic Programming* who discovered many inaccuracies in a previous version of what is now Chapter 7 of the thesis.

My PhD studies would not have been possible without the excellent work done by Mogens Nielsen, Uffe Engberg, Lene Kjeldsteen, Karen Møller, Janne Christensen, Hanne Jensen and Ingrid Larsen at BRICS in Århus. I thank the Danish National Research Foundation for funding my PhD research in BRICS.

Last but not least, I am very grateful to my parents and to my wife Karina. Without their love and patience, I would not have become who I am now.

Bartek Klin,
Århus, February 25, 2004.

Contents

Abstract **v**

Acknowledgements **vii**

1 Introduction **1**

2 Preliminaries **13**
2.1 Labelled Transition Systems 13
2.2 Hennessy-Milner Logic and Its Fragments 13
2.3 Simulations and Bisimulations 15
2.4 Structural Operational Semantics of Processes 16
2.5 Congruence Formats 17
2.6 Basic Notions of Category Theory 17
2.6.1 Algebras and Monads 17
2.6.2 Coalgebras 20
2.7 Abstract GSOS 22

3 A Fibrational Approach to Relations on Processes **25**
3.1 Fibrations 26
3.1.1 Lifting Basic Universal Constructions 26
3.1.2 Lifting Endofunctors 28
3.1.3 Lifting Initial Algebras 28
3.1.4 Lifting Freely Generated Monads 29
3.1.5 Lifting Final Coalgebras 30
3.1.6 Lifting Abstract GSOS 31
3.2 Fibration of Relations 32
3.2.1 Bisimulations as Coalgebras in **Rel** 33
3.2.2 Limited Expressive Power of **Rel** 35
3.3 Fibration of Test Suites 35
3.3.1 Specialization Functors 37
3.3.2 Comparing Test Suites 38
3.3.3 Lifting Functors with Test Constructors 39
3.3.4 Comparing Test Constructors 40

4 Van Glabbeek Spectrum Described by Test Suites 43
4.1 (B,2)-Test Suite Constructors . . . 44
4.1.1 Relation to Modal Logics . . . 45
4.2 Simulation and Bisimulation Semantics . . . 48
4.3 Decorated Trace Semantics . . . 52
4.4 Coinduction Principle for Traces . . . 62
4.5 Comparison of Process Preorders and Equivalences . . . 64
4.6 Nested Semantics . . . 65

5 From Test Suites to Congruence Formats 71
5.1 Lifting Syntax to Test Suites . . . 71
5.2 Abstract Congruence Formats . . . 73
5.3 Trace Semantics . . . 74
5.4 Completed Trace Semantics . . . 81
5.5 Failures Semantics . . . 86

6 Test Suites for Bisimulations on CPOs 91
6.1 Preliminaries . . . 92
6.2 Coalgebra Spans and Their Limitations . . . 95
6.3 Test Suite Approach . . . 99
6.4 Preorders and Topologies on CPOs . . . 102
6.5 Compact Coalgebras . . . 105
6.6 CPO-Bisimulations . . . 107

7 Adding Recursive Constructs to Bialgebraic Semantics 111
7.1 Preliminaries . . . 113
7.2 Motivating Examples . . . 116
7.3 Unfolding Rules . . . 120
7.4 Infinite Unfolding . . . 123
7.5 Merging Unfolding Rules with Distributive Laws . . . 126
7.6 From Unfolding Rules to Models . . . 129
7.7 Regular Unfolding Rules . . . 134
7.8 Concluding Remarks . . . 138

Bibliography 141

Index 147
Concept Index . . . 147
List of Symbols . . . 150

Chapter 1

Introduction

This thesis contributes to the mathematical theory of well-structured and well-behaved operational semantics of process algebras and programming languages, basing on and extending the scope of the abstract approach to process algebra called bialgebraic semantics.

Process algebra is the area of research concerned with formal descriptions of complex computational systems, especially those with communicating, concurrently executing components. Since the 1980s it has been a well-established and intensively studied area of theoretical computer science (e.g. [10, 16, 38, 42, 59]). Traditionally, its main goal was to develop formalisms for the specification and verification of concurrent and networked computer systems, and to provide semantics for concurrent programming languages. Among the best-known traditional process algebras are ACP [15], CCS [58] and CSP [20].

More recently, methods of process algebra have been applied in other areas of computer science, in formal approaches to distributed, dynamic and mobile systems (e.g. π-calculus [27, 60], calculus of mobile ambients [21]), security of cryptographic protocols (spi-calculus [1]), web services (languages like XLANG [82]) and even computational molecular biology [24, 73]. It is likely that in our world of pervasive, distributed and mobile computer systems, specified and implemented with many different languages, the need for formal, process-algebraic techniques will steadily increase.

When describing systems and processes formally, three key aspects must be considered: their syntax, behaviour and process (also called behavioural, observational or operational) equivalence.

Syntax refers to the structure of processes, and reflects the fact that it is natural and convenient to describe various systems as composed of smaller subsystems. In the framework of process algebra, processes are represented as terms built over some set of syntactic constructs. In simple cases, processes are arbitrary terms over an algebraic signature. More sophisticated syntactic phenomena include variable binding and structural congruences, where two syntactically different terms are identified and represent the same process.

Behaviour refers to the kind of actions the processes may take. In traditional process algebras, processes are allowed to nondeterministically perform externally observable actions (carrying no specified structure) from a prescribed set. One also considers deterministic, probabilistic and/or timed behaviour, the

ability to perform unobservable actions, features related to state, input and output.

Process equivalence describes those processes whose behaviours should be considered "the same". Sometimes it is also useful to say that the behaviour of a process can be "simulated" by another process. This leads to the notion of *process preorder*. These notions clearly depend on the chosen notion of behaviour of processes, but even for a single kind of behaviour one might consider many different process equivalences and preorders, suitable for different purposes. These notions are of utmost importance for the formal description of systems, since a description obtained by considering the full behaviour of a process is often too concrete and one wants to abstract from some of its details, to describe the intended meaning of processes adequately.

The most well-established approach to the formal presentation of process algebras, covering all three aspects mentioned above, is that of *structural operational semantics*. There, the behaviour of processes is modelled by means of transition relations on processes, presented as terms over some signature. The transition relations in turn are induced by inference rules that follow the syntactic structure of processes. This idea was originally used to give formal semantics to programming languages [68, 70], and has been widely used for this purpose [39, 61]. However, the intuitive appeal of this approach and, importantly, its inherent support for modelling nondeterministic behaviour, made it a natural framework for the formal description of process algebras (see [5]), and indeed, the invention of structural operational semantics triggered the development of process algebra as a field of research. Today, structural operational semantics is considered to be the standard way to give formal semantic descriptions of concurrent programs and systems.

Inference rules in operational descriptions take the form

$$\frac{\text{premises}}{\text{conclusion}}$$

A typical example of such a rule is

$$\frac{\mathtt{x} \xrightarrow{a} \mathtt{x}'}{\mathtt{x};\mathtt{y} \xrightarrow{a} \mathtt{x}';\mathtt{y}}$$

meaning that if some process $\mathtt{p}$ can make a transition to another process $\mathtt{p}'$ performing the action a, then the process $\mathtt{p};\mathtt{q}$ can make a similar transition to the process $\mathtt{p}';\mathtt{q}$ for any process $\mathtt{q}$. In general, if all premises of an inference rule are valid under some substitution, then the conclusion is valid under the same substitution.

Generally speaking, a set of operational inference rules induces a *transition system*, i.e., a set of processes together with a transition relation on it. Depending on the form of the rules, this can be a simple labelled transition system (LTS), an LTS with unobservable steps, a probabilistic transition system, a timed transition system etc. Based on the structure of the transition relation, one can define many different equivalences and preorders on processes. The variety of possible process equivalences has been studied most intensively in

the case of LTSs, and includes bisimulation equivalence [63], simulation equivalence, trace equivalence, testing equivalence and many others. It was treated comprehensively in [35] and is now often referred to as the *van Glabbeek spectrum*. For other notions of transition system and process behaviour, the most thoroughly studied process equivalences include weak/delay/branching bisimulation equivalences (e.g. [34]), probabilistic bisimulation equivalence [53], timed bisimulation equivalence [9] and many others.

For a notion of process equivalence to be practically useful, it must be *compositional* (or, in other words, it must be a *congruence*), i.e., it must be respected by the syntactic constructs of a chosen process algebra. This is necessary for any kind of inductive reasoning about processes, for example for specification and verification of systems component by component, or for verification of software refactoring [33] and optimization techniques, where replacing a subsystem/subprogram with an equivalent one should guarantee that the behaviour of the entire system/program is equivalent to the original one.

Proofs of compositionality of chosen process equivalences with respect to particular process algebras can be quite demanding. It is therefore desirable to show general results of this kind that hold on entire classes of process algebras. In the framework of structural operational semantics, the search for such results led to the development of various *congruence formats*. A congruence format is a restriction on the syntactic form of structural inference rules that guarantees a particular process equivalence compositional.

One of the most popular congruence formats is GSOS [18]. Operational descriptions in GSOS format contain only rules of the general form

$$\frac{\left\{ \mathtt{x}_i \xrightarrow{a_{ij}} \mathtt{y}_{ij} \,:\, i \leq n,\ j \leq m_i \right\} \cup \left\{ \mathtt{x}_i \not\xrightarrow{b_{ik}} \,:\, i \leq n, k \leq n_i \right\}}{\mathtt{f}(\mathtt{x}_1, \ldots, \mathtt{x}_n) \xrightarrow{c} \mathtt{t}}$$

If all inference rules in an operational description are of this form, then bisimulation equivalence on the LTS generated from these rules is guaranteed to be compositional.

Many other congruence formats have been defined for various notions of behaviour and process equivalence (see [5] and references therein, but also [12, 49]). However, to define a general congruence format for a given notion of process equivalence is often a difficult task. Given the growing variety of disparate programming paradigms and process behaviours, it is desirable to have a general framework for constructing congruence formats for given notions of syntax, behaviour and process equivalence. To provide such a framework, one needs to employ abstract approaches to these three key aspects of process algebra.

Throughout the field of computer science, abstract approaches to various phenomena have been developed with the use of *category theory*[1]. There exist abstract, general and well-established approaches to process syntax and behaviour, based on categorical methods, and the framework presented in this thesis builds upon this work. To provide a general framework for deriving con-

[1]The reader is assumed to have basic knowledge of concepts and methods of category theory. For the basic terminology unexplained here, see [57].

gruence formats, one needs to combine these approaches with a suitable abstract approach to process equivalence.

Since the development of the initial algebra approach to semantics [36], the standard method to model *syntax* of programs and processes is based on (unsorted) algebraic signatures, terms and *algebras*. Signatures are usually represented categorically as endofunctors. More specifically, a signature Σ consisting of n language constructs $\mathtt{f}_1, \mathtt{f}_2, \ldots, \mathtt{f}_n$ with arities $k_1, k_2, \ldots, k_n$ respectively, is modelled as an endofunctor

$$\Sigma X = X^{k_1} + X^{k_2} + \cdots + X^{k_n}$$

on the category of sets and functions **Set**. Then to equip a set X with a suitable structure is just to provide a function (called a Σ-algebra)

$$h : \Sigma X \to X$$

For any Σ, all Σ-algebras with a suitable notion of algebra morphism form a category Σ-**Alg**. For Σ's constructed as above, this category has an initial object

$$\psi : \Sigma T0 \to T0$$

where $T0$ is the set of all closed terms over the signature corresponding to Σ. The construction T can be itself turned into an endofunctor, called the monad freely generated by Σ.

The abstractness of this approach ensures that it can be interpreted for many different endofunctors and in different categories.

On the other hand, the *behaviour* of processes has been successfully modelled using *coalgebras* for suitable endofunctors [46, 77, 78]. For example, a finitely branching labelled transition system $\langle X, A, \longrightarrow \rangle$ (where X is a set of processes, A a set of actions, and $\longrightarrow \subseteq X \times A \times X$ a transition relation) can be viewed as a function

$$h : X \to \mathcal{P}_{\mathrm{f}}(A \times X)$$

where $\mathcal{P}_{\mathrm{f}}$ is the (covariant) finite powerset endofunctor on **Set**. In this context, the functor $\mathcal{P}_{\mathrm{f}}(A \times -)$ is called a behaviour endofunctor, and h is a coalgebra for this functor. Varying the behaviour endofunctors, usually denoted B, one can model different kinds of transition systems, including [77] unlabelled, deterministic, with input/output, state based [37] and probabilistic [87] ones. The generality of the coalgebraic approach allows one to use different categories instead of **Set** to model transition systems with some structure imposed on the set of processes. For example, one might insist that processes form a complete partial order (cpo).

For any endofunctor B, all B-coalgebras with suitably defined coalgebra morphisms form a category B-**Coalg**. When this category has a final object, then every B-coalgebra has a canonical interpretation (called *final semantics*) in this final coalgebra. For example, the final semantics of a finitely branching LTS is equal to its unfolding to a labelled synchronization tree, with bisimulation equivalent processes identified [78, 77].

Central to the coalgebraic theory of processes is an abstract notion of B-bisimulation, based on spans of coalgebras. Given a coalgebra $h : X \to BX$, a (span) bisimulation on h is an object R with two morphisms $p_1, p_2 : R \to X$ such that for some coalgebra $r : R \to BR$ the diagram

$$\begin{array}{ccccc} X & \xleftarrow{p_1} & R & \xrightarrow{p_2} & X \\ {\scriptstyle h}\downarrow & & {\scriptstyle r}\downarrow & & \downarrow{\scriptstyle h} \\ BX & \xleftarrow[Bp_1]{} & BR & \xrightarrow[Bp_2]{} & BX \end{array}$$

commutes. If the underlying category is **Set**, then R can be viewed as a binary relation on X. For various functors B, this abstract definition specializes to well-known process equivalences, and in particular to bisimulation equivalence [63] on labelled transition systems for $B = \mathcal{P}_{\mathrm{f}}(A \times -)$.

The algebraic and coalgebraic methods were combined in the seminal work of Plotkin and Turi [85, 83], in *bialgebraic semantics* (another, similar approach to syntax and behaviour is that of transition systems with algebraic structure, e.g. [23]). There, transition systems with syntactic structure on states (processes) are modelled as bialgebras, i.e. pairs

$$\Sigma X \xrightarrow{g} X \xrightarrow{h} BX$$

for some endofunctors Σ and B. To express the fact that a bialgebra behaves well accordingly to some operational semantics, a notion of distributive law is used. Such distributive laws are induced by natural transformations

$$\lambda : \Sigma(\mathrm{Id} \times B) \to BT$$

(where T is the monad freely generated by Σ). Every natural transformation of this kind canonically induces a coalgebra

$$h_\lambda : T0 \to BT0$$

and the central result of [85], specialized to **Set** as the underlying category and Σ, T generated from some algebraic signature, says that (under a mild assumption on B) the largest (span) bisimulation on h_λ is compositional (i.e., it is a congruence on the initial Σ-algebra).

In the case of $B = \mathcal{P}_{\mathrm{f}}(A \times -)$, the natural transformations λ correspond to sets of inference rules in GSOS format, in the sense that the B-coalgebra h_λ induced from λ corresponds to the labelled transition system induced by the rules corresponding to λ. The abstract results from [85] then say that GSOS is indeed a congruence format for bisimulation equivalence.

This result can already be seen as a framework for deriving congruence formats. Indeed, it has been used for this purpose [13, 49]. However, its generality is limited by features of the coalgebra span approach to process equivalence. Indeed, instead of covering various process equivalences, the coalgebra span approach concentrates on a single, canonical process equivalence (the respective (span) bisimulation equivalence) for any notion of behaviour. Therefore, looking for a general framework for deriving congruence formats, one needs to seek a more flexible abstract approach to process equivalence.

Before we sketch the approach taken in this thesis, we briefly mention other abstract approaches to equivalences on processes known in the literature.

Almost all known equivalences on ordinary labelled transition systems can be described by appropriate *modal logics*. The general idea is to define a language of formulae that can be interpreted in any LTS, and consider two processes equivalent if and only if they satisfy exactly the same formulae. The most famous logic used for this purpose is the Hennessy-Milner logic [40], which distinguishes exactly those processes that are not bisimulation equivalent. Other process equivalences from the van Glabbeek spectrum are described by suitable fragments of the Hennessy-Milner logic (see [35] for a comprehensive treatment). Modal characterizations have been also provided for various process equivalences on other kinds of systems, including those with unobservable actions (see [34] and references therein) and probabilistic transition systems [26, 53].

The logical approach to process equivalence is very general, but it is not really a consistent, abstract framework. In particular, an abstract notion of a modal operator, applicable to various notions of behaviour, is not clearly provided. This problem was solved to large extent in *coalgebraic logic* [62, 64]. However, another difficulty is a missing link between modal descriptions of processes and their syntactic structure. Modal logics have been used to deriving congruence formats for process equivalences [17, 32], but the formats obtained were constructed by clever analysis of particular modal logics rather than with use of some general meta-theorem, parametrized by a notion of process equivalence.

Many well-known process equivalences can be described in the abstract categorical framework of *open maps* [22, 47, 26]. Despite its impressive generality, the syntactic structure of processes has been ignored in this approach. The relation of the open map framework to coalgebraic methods, although studied to some extent [54], remains rather unclear.

Similarly, the abstract framework of *quantales* [4] lacks a systematic treatment of syntactic issues or a relation to coalgebraic methods, therefore it is difficult to use it for congruence format derivation.

Among abstract coalgebraic approaches to process equivalence, the coalgebra span approach described above is the most thoroughly studied. Attempts have been made to overcome its limited generality and cover multiple notions of process equivalence for single notions of behaviour. The idea of changing the underlying category [78, 84] or the notion of coalgebra morphism [71] allowed one to cover trace equivalence for ordinary LTSs. An interpretation of LTSs as certain Moore automata allowed the authors of [19] to cover also the testing equivalence of [25] (equivalent to the failures equivalence in the van Glabbeek spectrum). However, no other process equivalences have been treated so far. Yet another modification of the definition of coalgebra morphism in [74] led to an abstract definition of weak bisimulation for a class of behaviour endofunctors. However, that approach is tailored to the specific needs of weak bisimulations.

Another coalgebraic approach to process equivalence is based on lifting coalgebras (corresponding to transition systems) to the category **Rel** of binary relations and relation-preserving functions. Such lifting, performed in a canonical way, allowed the authors of [41] to give an abstract definition of bisimulation

equivalence, corresponding to that based on coalgebra spans. This approach was extended in [45], yielding an abstract definition of simulation equivalence for any behaviour with suitable additional structure. However, no other process equivalences have been treated so far, and in this thesis it is argued that this approach cannot immediately cover the standard notion of trace equivalence on LTSs, without resorting to techniques analogous to those used in the coalgebra span approach (e.g., [44]).

In this thesis, a novel abstract coalgebraic approach to process equivalence is presented. The approach is based on simple and general notions of tests and test suites. Intuitively, two processes are considered equivalent if they cannot be distinguished by any test from a given test suite. Varying the test suites considered, one obtains different notions of process equivalence. The interesting test suites are constructed from the coalgebraic structure imposed on processes.

A *test* on an object X of processes is a morphism from X to a fixed object $\mathcal{V}$ of test values. For most purposes it is sufficient to work in the category **Set** and let $\mathcal{V}$ be the two-element set of logical values $\{\mathtt{tt}, \mathtt{ff}\}$, denoted also 2. Then a test on a set X can be viewed as a subset of X. A *test suite* on X is a set θ of tests on X. Every such test suite induces a *specialization equivalence* on X:

$$x \equiv_\theta y \iff \forall V \in \theta.\, Vx = Vy$$

(the name is chosen by the analogy to specialization orders known from general topology, as indeed a test suite resembles a topology, except that no closure conditions are imposed). Sets equipped with test suites on them form a category 2-**TS**, constructed similarly to the category of topological spaces: a function $f : \langle X, \theta\rangle \to \langle Y, \vartheta\rangle$ is a valid morphism if the inverse image of each test from ϑ along f belongs to θ.

Any functor B : **Set** $\to$ **Set** can be *lifted* to an endofunctor on 2-**TS**, by defining its action on test suites. A particularly well-structured way of such lifting is based on sets of *test constructors*, i.e., tests on the set $B2$, and *closures*, i.e., operators that given a test suite, return another test suite obtained in a structured manner. Any set W of test constructors, together with any closure Cl, induces a functor $B^{\mathsf{W}} : 2\text{-}\mathbf{TS} \to 2\text{-}\mathbf{TS}$, acting as B on underlying sets and defined by

$$B^{\mathsf{W}}\,\langle X, \theta\rangle = \langle BX, \mathrm{Cl}_{BX}\{w \circ BV \mid w \in \mathsf{W}, V \in \theta\}\rangle \qquad B^{\mathsf{W}} f = Bf$$

Moreover, for any coalgebra $h : X \to BX$ there exists the least (and thus canonical) test suite θ that lifts h, i.e., such that $h : \langle X, \theta\rangle \to B^{\mathsf{W}}\,\langle X, \theta\rangle$ is a valid morphism in 2-**TS**. It turns out that for several choices of test constructors, the specialization equivalences of these canonical test suites are exactly the various well-known process equivalences.

The test constructors chosen for particular equivalences correspond to *modal operators* used to describe these equivalences logically. For example, if $BX = \mathcal{P}_{\mathrm{f}}(A \times X)$, a test constructor $w_{\langle a\rangle} : B2 \to 2$ corresponding to the well-known "diamond" modal operator $\langle a\rangle$ is defined by

$$w_{\langle a\rangle}\beta = \mathtt{tt} \iff \langle a, \mathtt{tt}\rangle \in \beta$$

On the other hand, closures correspond to *propositional connectives* used in the respective modal logics. For example, to model simulation equivalence, one needs to close a given set of tests under all intersections, reflecting the fact that the modal logic characterizing simulation equivalence is closed under conjunction.

The test suite framework can be merged with the bialgebraic approach to operational semantics. Besides behaviour endofunctors B, also the functors Σ, T representing syntax can be lifted (in a canonical way) to functors Σ^*, T^* on 2-**TS**. An abstract result proved in this thesis states that if a natural transformation

$$\lambda : \Sigma(\mathrm{Id} \times B) \to BT$$

(where Σ and T are functors representing the syntax of a language, and B is a behaviour functor) lifts to a transformation

$$\lambda : \Sigma^*(\mathrm{Id} \times B^{\mathsf{W}}) \to B^{\mathsf{W}} T^*$$

then the process equivalence corresponding to the functor B^{W} is a congruence on the transition system generated by the operational rules corresponding to λ. Studying the conditions which λ must satisfy for this lifting to exist, one obtains a syntactic congruence format for the process equivalence considered. Thus the test suite approach to process equivalence is merged with the bialgebraic framework, yielding a general method for deriving congruence formats, parametrized by notions of syntax, behaviour and process equivalence.

The test suite approach is novel, but it was inspired by previous developments: most notably, by modal logics (tests play a rôle analogous to that of formulae in distinguishing processes, and test constructors correspond to modal operators), the coalgebra lifting approach to bisimulation (indeed, coalgebras are lifted to test suite categories similarly as in [41]), categorical logic [43] (tests do not reside in the same category as the tested entities; formally, the category 2-**TS** is fibred over **Set**), and even Chu spaces [72] (test suites are essentially extensional Chu spaces; however, these mathematical entities seem to have not been used in this context before and this correspondence will not be explored in this thesis). The relations of test suites to the fibrational coalgebraic framework of [41, 45] (essentially used also in [65], in the context of recursively defined domains) are particularly strong. In fact, both approaches are based on a general idea of enriching coalgebras with additional properties (relations in [41, 45], test suites in this thesis), technically by lifting them to total categories of suitable fibrations.

It is maybe worthwhile here to mention some apparent, but misleading, connections to some other frameworks. In spite of the same name, our tests are not related in any obvious way to those considered in the literature on testing equivalences [20, 25]. Indeed, there a test is a process itself, and it may (or must) be passed. In the test suite framework, a test is simply a predicate on processes. Also, the notion of a saturated set is often used to characterize testing equivalence. Such sets are families of subsets of some given set, and our test suites can be viewed as such families too. This observation is, however, misleading. In [20, 25] saturated families of sets of actions are considered,

and test suites are families of sets of processes, and it is hard to see any real connection between these notions.

Since the test suite framework is general, it would be reasonable to expect that the congruence formats obtained will not be as general as other formats known in the literature, carefully tailored to particular process equivalences. However, the concrete results obtained so far for three well-known equivalences on LTSs are surprisingly good. For trace equivalence, the format obtained matches exactly the state-of-the-art de Simone format [80]. The format for completed trace equivalence is the first such format ever published, and in the *Handbook of Process Algebra* [5] it had been even speculated that

> [...] one cannot really hope to formulate a general congruence format for completed trace equivalence.

The format for failures equivalence is incomparable with the most general formats known so far.

Encouraged by this, we want to seek other applications of the test suite approach. One possible choice (among many others: considering various process equivalences, behaviour endofunctors, syntax with variable binding, etc.) is to consider process languages with recursive operators, described by recursive equations such as

$$\texttt{loop}\ t = t; \texttt{loop}\ t$$

rather than by standard operational rules. To avoid problems related to fitting standard operational descriptions for such operators into the bialgebraic framework, it is convenient to treat them as abbreviations for their infinite expansions, as suggested in [69]. This causes infinite terms to come into play, and to allow inductive reasoning one works in a suitable category of domains (complete partial orders, cpos) rather than in **Set**.

Two problems related to this are considered in this thesis. Firstly, one wants to abstractly represent some useful notion of process equivalence on transition systems where processes form a cpo rather than a set. The obvious candidate is the notion of partial bisimulation equivalence considered in [2], since it is well studied and gives elegant full abstraction results. It turns out that neither the classical coalgebra span approach nor its ordered version [29, 76] fully covers this notion of process equivalence; however, the test suite approach does. This can be seen as a further confirmation of the generality of the test suite framework.

Secondly, a method to integrate recursive equations with structural operational rules is needed, to formally capture the idea of treating recursive operators as abbreviations for their infinite expansions. To this end, one considers a signature Σ (the recursion-free fragment of a language) extended to a signature Σ' (the full language). The behaviour of recursive operators is then captured as a natural transformation

$$r : T' \to TT'$$

where T and T' are the monads freely generated by Σ and Σ' respectively. If this transformation satisfies a natural condition called *regularity*, then it can be seamlessly merged (using certain fixpoint constructions, available in

the underlying category of domains) with the operational semantics for the recursion-free fragment, modelled bialgebraically as a natural transformation

$$\lambda : \Sigma(\mathrm{Id} \times B) \to BT$$

yielding a natural transformation

$$\lambda^r : \Sigma'(\mathrm{Id} \times B) \to BT'$$

corresponding to an operational semantics for the full language. This construction was conceived as an attempt to apply the test suite approach in a new setting, but it does not involve test suites directly and is of independent interest, as a method of extending the scope of bialgebraic semantics to cover a general class of recursive equations.

Further steps along this line of research — to provide a concrete description of a congruence format for the transformations λ^r, and to pull back this format along the above construction to obtain a format for λ and r — are the subject of ongoing work and are not described in this thesis.

Organization

The structure of the thesis is as follows. Chapter 2 contains some standard definitions and results from the literature, related to labelled transition systems, the van Glabbeek spectrum, structural operational semantics, algebraic and coalgebraic modelling and bialgebraic semantics.

In Chapter 3, the test suite approach to process equivalence and congruence formats is presented. The presentation is aimed at generality, usability and mathematical elegance, and not simply at covering the immediate applications we have in mind. As a result, the applications of the framework presented in the following chapters do not use it in full generality. Moreover, only the coalgebraic aspects of the test suite approach are used in Chapters 4 and 6. In Chapter 5, the algebraic (syntactic) aspects are used as well.

The structure of Chapter 3 reflects the similarities between our approach and that of [41, 45]. In Section 3.1, a very general fibrational approach is presented, which is then specialized to (a special case of) the relational approach from [41, 45] and to the test suite approach.

In Chapter 4, the abstract test suite framework is specialized to ordinary LTSs with **Set** as the underlying category, and variety of process equivalences from the van Glabbeek spectrum are characterized by suitable test suites. These characterizations are useful even when syntactic issues are not considered, as they lead to novel proof principles for various process equivalences. This application of the test suite approach is illustrated on a simple example.

Chapter 5 draws on results from Chapter 4, merged with the algebraic aspects of the test suite framework, to present congruence formats for three process equivalences from the van Glabbeek spectrum: trace equivalence, completed trace equivalence and failures equivalence.

Chapter 6 applies the test suite framework in the category of algebraic cpos, covering Abramsky's full abstraction result for coalgebras for the Plotkin powerdomain [2]. To simplify matters, unlabelled transition systems are considered.

Chapter 7 is largely independent from other chapters and does not involve any use of the test suite framework. It describes a formal approach to merging recursive equations with operational rules modelled in the bialgebraic framework.

Published contribution

Some results presented in this thesis have been published by the author or are accepted for publication.

- [52] B. Klin, P. Sobocinski. Syntactic formats for free: an abstract approach to process equivalence. In *Proc. CONCUR 2003*, volume 2671 of *Lecture Notes in Computer Science*, 2003.

  In this paper, a rudimentary version of the framework shown in Chapter 3 was described. There, the underlying category is chosen to be **Set**, with 2 as the set of test values. Also a few results (albeit simplified) from Chapter 4 were presented. The main result was the definition of congruence formats for trace equivalence, completed trace equivalence and failures equivalence on LTSs. The same formats are shown in this thesis in Chapter 5, although here they are presented in a different and, hopefully, simpler manner.

- [50] B. Klin. An abstract approach to process equivalence and a coinduction principle for traces. In *Proc. CMCS 2004*, *Electronic Notes in Theoretical Computer Science*, 2004. To appear.

  This paper is an extended abstract of Chapter 4.

- [51] B. Klin. Adding recursive constructs to bialgebraic semantics. *Journal of Logic an Algebraic Programming*, special issue on Structural Operational Semantics, 2004. To appear.

  This paper is included in this thesis as Chapter 7, with only minor changes.

Acknowledgement

Most of the results presented in this thesis are original contributions of the author, and all due references to previously published work have been made. However, it must be clearly said that results shown in Section 6.3 and the two counterexamples from Section 6.2 are entirely based on unpublished ideas of Gordon Plotkin and they should be credited to him. Obviously, the author is fully responsible for any possible mistakes in the statement, in proofs and in the presentation of these results. The original ideas of Plotkin, based on a canonical lifting of the Plotkin powerdomain to topologies of tests, were slightly modified to make them fit into the test suite framework.

From this remark, it should be clear that the entire test suite approach has been directly inspired by Plotkin's ideas. The main conceptual differences

between his original, unpublished approach and the test suite framework are that he considered:

- topologies of tests only, instead of arbitrary test suites,
- endofunctor liftings based on the canonical choice of *all* test constructors, instead of arbitrary sets of constructors,
- no closures,
- canonical, two-element sets of test values, instead of arbitrary objects of test values,
- the single notion of specialization preorder, instead of arbitrary specialization functors.

In short, Plotkin's approach was aimed at characterizing a single, canonical notion of process equivalence for every notion of behaviour, and the approach presented in this thesis aims to cover many other equivalences.

Chapter 2

Preliminaries

In this chapter, we present standard notions and results related to process algebra and categorical modelling of well-behaved operational semantics.

First, the basic notion of labelled transition system is introduced, followed by several process preorders and equivalences, known collectively as the van Glabbeek spectrum. After recalling the basic definitions of structural operational semantics, including the GSOS rule format, we proceed to present basic notations and results of category theory, used to model operational rules abstractly in the so-called abstract GSOS.

The definitions and results presented in this chapter are standard, taken mostly from [5, 35, 57, 83, 85].

2.1 Labelled Transition Systems

Definition 2.1 A *labelled transition system* (LTS) $\langle X, A, \longrightarrow \rangle$ is a set X of *processes*, a set A of *actions*, and a *transition relation* $\longrightarrow \subseteq X \times A \times X$. Usually instead of $\langle x, a, x' \rangle \in \longrightarrow$ one writes $x \xrightarrow{a} x'$.

Given an LTS $\langle X, A, \longrightarrow \rangle$, for any $Q \subseteq A$, one writes $x \xrightarrow{Q}$ meaning that $x \xrightarrow{a} x'$ for some $a \in Q$, $x' \in X$. Instead of $x \xrightarrow{A}$ one writes $x \longrightarrow$, and $x \xrightarrow{a}$ means $x \xrightarrow{\{a\}}$. Sometimes a 'negated' version of this notation is used. For example, $x \not\xrightarrow{Q}$ means that it is not the case that $x \xrightarrow{Q}$. For any $x \in X$, one also defines the set of *initials* $I(x) = \left\{ a \in A : x \xrightarrow{a} \right\}$.

An LTS $\langle X, A, \longrightarrow \rangle$ is *finitely branching* if for every process $x \in X$ there are only finitely many processes $x' \in X$ and actions $a \in A$ such that $x \xrightarrow{a} x'$.

An LTS for which its underlying graph (obtained by ignoring all actions) is a directed, rooted tree is called a *labelled synchronization tree*.

2.2 Hennessy-Milner Logic and Its Fragments

Definition 2.2 Given a set of actions A, one considers nine sets of *modal formulae* $\mathcal{F}_{\mathsf{Tr}}$, $\mathcal{F}_{\mathsf{CTr}}$, $\mathcal{F}_{\mathsf{Fl}}$, $\mathcal{F}_{\mathsf{FlTr}}$, $\mathcal{F}_{\mathsf{Rd}}$, $\mathcal{F}_{\mathsf{RdTr}}$, $\mathcal{F}_{\mathsf{S}}$, $\mathcal{F}_{\mathsf{RdS}}$ and $\mathcal{F}_{\mathsf{BS}}$, given by the

following BNF grammars:

$$\begin{array}{ll}
\mathcal{F}_{\mathsf{Tr}} & \phi ::= \top \mid \langle a\rangle\,\phi \\
\mathcal{F}_{\mathsf{CTr}} & \phi ::= \top \mid \langle a\rangle\,\phi \mid \tilde{A} \\
\mathcal{F}_{\mathsf{Fl}} & \phi ::= \top \mid \langle a\rangle\,\phi \mid \tilde{Q} \\
\mathcal{F}_{\mathsf{FlTr}} & \phi ::= \top \mid \langle a\rangle\,\phi \mid \tilde{Q} \mid \tilde{Q} \wedge \langle a\rangle\,\phi \\
\mathcal{F}_{\mathsf{Rd}} & \phi ::= \top \mid \langle a\rangle\,\phi \mid \check{Q} \\
\mathcal{F}_{\mathsf{RdTr}} & \phi ::= \top \mid \langle a\rangle\,\phi \mid \check{Q} \mid \check{Q} \wedge \langle a\rangle\,\phi \\
\mathcal{F}_{\mathsf{S}} & \phi ::= \top \mid \langle a\rangle\,\phi \mid \phi \wedge \phi \\
\mathcal{F}_{\mathsf{RdS}} & \phi ::= \top \mid \langle a\rangle\,\phi \mid \check{Q} \mid \phi \wedge \phi \\
\mathcal{F}_{\mathsf{BS}} & \phi ::= \top \mid \bot \mid \langle a\rangle\,\phi \mid [a]\phi \mid \phi \wedge \phi \mid \phi \vee \phi
\end{array}$$

where a ranges over A, and Q ranges over subsets of A. The above means that, for example, in $\mathcal{F}_{\mathsf{Rd}}$ there is a constant symbol $\check{Q}$ for every $Q \subseteq A$.

Formulae in $\mathcal{F}_{\mathsf{Tr}}$ are called (partial, finite) *traces*. Formulae in $\mathcal{F}_{\mathsf{CTr}} \setminus \mathcal{F}_{\mathsf{Tr}}$ are called (finite) *completed traces*. Formulae in $\mathcal{F}_{\mathsf{Fl}} \setminus \mathcal{F}_{\mathsf{Tr}}$ are called *failure pairs*. Formulae in $\mathcal{F}_{\mathsf{FlTr}}$ are called *failure traces*. Formulae in $\mathcal{F}_{\mathsf{Rd}} \setminus \mathcal{F}_{\mathsf{Tr}}$ are called *ready pairs*. Formulae in $\mathcal{F}_{\mathsf{RdTr}}$ are called *ready traces*.

Definition 2.3 Given an LTS $h = \langle X, A, \longrightarrow\rangle$, the satisfaction relation $\models_h$ between processes and modal formulae is defined inductively as follows:

$$\begin{array}{lcll}
x \models_h \top & & \text{always} & \\
x \models_h \bot & & \text{never} & \\
x \models_h \langle a\rangle\,\phi & \iff & x' \models_h \phi \text{ for some } x' \text{ such that } x \xrightarrow{a} x' \\
x \models_h [a]\phi & \iff & x' \models_h \phi \text{ for all } x' \text{ such that } x \xrightarrow{a} x' \\
x \models_h \tilde{Q} & \iff & x \not\xrightarrow{Q} \\
x \models_h \check{Q} & \iff & I(x) = Q \\
x \models_h \phi_1 \wedge \phi_2 & \iff & x \models_h \phi_1 \text{ and } x \models_h \phi_2 \\
x \models_h \phi_1 \vee \phi_2 & \iff & x \models_h \phi_1 \text{ or } x \models_h \phi_2
\end{array}$$

The set of formulae $\mathcal{F}_{\mathsf{BS}}$ together with the above interpretation is called the (finitary) *Hennessy-Milner logic*.

Definition 2.4 For any $\mathsf{W} \in \{\mathsf{Tr}, \mathsf{CTr}, \mathsf{Fl}, \mathsf{FlTr}, \mathsf{Rd}, \mathsf{RdTr}, \mathsf{S}, \mathsf{RdS}, \mathsf{BS}\}$ one considers the corresponding *process preorder* $\sqsubseteq_{\mathsf{W}} \subseteq X \times X$ and *process equivalence* $\cong_{\mathsf{W}} \subseteq X \times X$, defined on a given LTS h as follows:

$$\begin{array}{lcl}
x \sqsubseteq_{\mathsf{W}} x' & \iff & (\forall \phi \in \mathcal{F}_{\mathsf{W}}.\ x \models_h \phi \Longrightarrow x' \models_h \phi) \\
x \cong_{\mathsf{W}} x' & \iff & (\forall \phi \in \mathcal{F}_{\mathsf{W}}.\ x \models_h \phi \iff x' \models_h \phi)
\end{array}$$

All process preorders and equivalences mentioned above (together with some others) are known as the *van Glabbeek spectrum* and have been extensively studied and described in the literature; for a comprehensive treatment, see [35].

The preorders $\sqsubseteq_{\mathsf{Tr}}$, $\sqsubseteq_{\mathsf{CTr}}$, $\sqsubseteq_{\mathsf{Fl}}$, $\sqsubseteq_{\mathsf{FlTr}}$, $\sqsubseteq_{\mathsf{Rd}}$, and $\sqsubseteq_{\mathsf{RdTr}}$ on a given LTS are usually called *trace preorder*, *completed trace preorder*, *failures preorder*, *failure trace preorder*, *readiness preorder* and *ready trace preorder*, respectively. The corresponding equivalences are named in a similar manner. The preorders $\sqsubseteq_{\mathsf{S}}$, $\sqsubseteq_{\mathsf{RdS}}$, $\sqsubseteq_{\mathsf{BS}}$ and equivalences $\cong_{\mathsf{S}}$, $\cong_{\mathsf{RdS}}$ and $\cong_{\mathsf{BS}}$ are considered in more detail in the next section.

2.3 Simulations and Bisimulations

In the following definitions, a given LTS $\langle X, A, \longrightarrow \rangle$ is assumed.

Definition 2.5 A relation $R \subseteq X \times X$ is a *simulation* if xRy implies that for any $a \in A$ and $x' \in X$, if $x \xrightarrow{a} x'$ then there exists $y' \in X$ such that $y \xrightarrow{a} y'$ and $x'Ry'$. If, moreover, xRy implies $I(x) = I(y)$, then R is a *ready simulation*.

Definition 2.6 A relation $R \subseteq X \times X$ is a *bisimulation* if xRy implies that for any $a \in A$,

- for any $x' \in X$ if $x \xrightarrow{a} x'$ then there exists $y' \in X$ such that $y \xrightarrow{a} y'$ and $x'Ry'$,
- for any $y' \in X$ if $y \xrightarrow{a} y'$ then there exists $x' \in X$ such that $x \xrightarrow{a} x'$ and $x'Ry'$.

Definition 2.7 Processes $x, y \in X$ are

- in *(ready) simulation preorder* if there exists a (ready) simulation R such that xRy,
- *(ready) simulation equivalent* if there exist (ready) simulations R, R' such that xRy and $yR'x$,
- *bisimulation equivalent*, or *bisimilar*, if there exists a bisimulation R such that xRy.

Proofs of the following well-known results can be found, e.g., in [35]:

Proposition 2.8 (Ready) simulation preorder is indeed a preorder (i.e., a reflexive and transitive relation) and it is the largest (ready) simulation on a given LTS. Bisimulation equivalence is indeed an equivalence relation and it is the largest bisimulation on a given LTS.

Proposition 2.9 In any LTS, the relation $\sqsubseteq_{\mathsf{S}}$ is equal to the simulation preorder, the relation $\cong_{\mathsf{S}}$ is equal to the simulation equivalence, the relation $\sqsubseteq_{\mathsf{RdS}}$ is equal to the ready simulation preorder, the relation $\cong_{\mathsf{RdS}}$ is equal to the ready simulation equivalence, and the relations $\sqsubseteq_{\mathsf{BS}}$ and $\cong_{\mathsf{BS}}$ are both equal to the bisimulation equivalence.

2.4 Structural Operational Semantics of Processes

In the context of process algebra, processes in labelled transition systems are usually closed terms over some signature.

A *signature* Σ is a set $\bar{\Sigma}$ of *language constructs*, together with an *arity function* $ar : \bar{\Sigma} \to \mathbb{N}$. For a given set X of *variables*, ΣX is the set of expressions of the form $\mathtt{f}(x_1, \ldots, x_{ar(\mathtt{f})})$, where $\mathtt{f} \in \bar{\Sigma}$ and $x_1, \ldots, x_{ar(\mathtt{f})} \in X$.

Given a signature Σ and a set X, the set $T_\Sigma X$ of *terms* over Σ with variables X is the least fixed point of the operator (monotonic with respect to set inclusion)

$$\Phi Y = X + \Sigma Y$$

where $+$ denotes disjoint union of sets. When describing terms from $T_\Sigma X$, the injections $\iota_1 : X \to T_\Sigma X$ and $\iota_2 : \Sigma T_\Sigma X \to T_\Sigma X$ will be often omitted, i.e., we will write $\mathtt{f}(x, y)$ rather than $\iota_2(\mathtt{f}(\iota_1(x), \iota_1(y)))$. Also the subscript in $T_\Sigma X$ will be omitted if Σ is irrelevant or clear from the context. Elements of $T\emptyset$ are called *closed terms* over Σ.

For a term $t \in TX$ and a function (substitution) $\sigma : X \to Y$, $t\sigma$ will denote the term in TY resulting from t by simultaneously replacing every $x \in X$ with $\sigma(x)$.

In the following, we assume a fixed, countably infinite set of variables Ξ, ranged over by $\mathtt{x}_1, \mathtt{x}_2, \ldots, \mathtt{y}_1, \mathtt{y}_2, \ldots$. To stress that some terms are built over variables from Ξ, they will be typeset $\mathtt{t}$, $\mathtt{t}'$ etc., as opposed to the normal notation t, t' etc.

Fix an arbitrary set of actions A. For a signature Σ, a *positive* Σ*-literal* is an expression $\mathtt{t} \xrightarrow{a} \mathtt{t}'$, and a *negative* Σ*-literal* is an expression $\mathtt{t} \not\xrightarrow{a}$, where $\mathtt{t}, \mathtt{t}' \in T\Xi$ and $a \in A$. An *inference rule* ρ over Σ is an expression $\frac{H}{\alpha}$, where H is a set of Σ-literals and α is a positive Σ-literal. Elements of H are then called *premises* of ρ, and α the *conclusion* of ρ. The left side and the right side of the conclusion of ρ are called the *source* and the *target* of ρ, respectively. If the source of a rule ρ is of the form $\mathtt{f}(\mathtt{x}_1, \mathtt{x}_2, \ldots, \mathtt{x}_n)$, one says that ρ is a rule *for* $\mathtt{f}$.

A *transition system specification* over Σ is a set of rules over Σ.

In the following definition assume a fixed signature Σ.

Definition 2.10 (GSOS) A transition system specification Λ is in GSOS [18] format if every rule $\rho \in \Lambda$ is of the form

$$\frac{\left\{ \mathtt{x}_i \xrightarrow{a_{ij}} \mathtt{y}_{ij} \,:\, i \le n,\ j \le m_i \right\} \cup \left\{ \mathtt{x}_i \not\xrightarrow{b_{ik}} \,:\, i \le n, k \le n_i \right\}}{\mathtt{f}(\mathtt{x}_1, \ldots, \mathtt{x}_n) \xrightarrow{c} \mathtt{t}}$$

with $\mathtt{f}$ a language construct in $\bar{\Sigma}$ and $n = ar(\mathtt{f})$, such that $\mathtt{x}_i \in \Xi$ and $\mathtt{y}_{ij} \in \Xi$ are all distinct and are the only variables that occur in ρ. If, moreover, for every $\mathtt{f} \in \bar{\Sigma}$, Λ contains only finitely many rules with $\mathtt{f}$ in the source, then Λ is *image-finite*.

Given a transition system specification Λ in GSOS format, one defines a notion of a provable positive literal in a straightforward way. The set of all provable literals forms an LTS with closed terms over Σ as processes, and with positive closed literals as transitions. This is called the LTS *induced* by Λ. If, moreover, Λ is image-finite, then the induced LTS is finitely branching (for details, see [5]).

In this thesis only image-finite GSOS specifications will be considered.

2.5 Congruence Formats

Definition 2.11 Let Σ be any signature. A preorder $R \subseteq T_\Sigma\emptyset \times T_\Sigma\emptyset$ is a *precongruence*, if for any $\mathtt{f} \in \bar{\Sigma}$ with $ar(\mathtt{f}) = n$, and for any $t_1, t_1', t_2, t_2', \ldots, t_n, t_n' \in T_\Sigma\emptyset$, whenever $t_i R t_i'$ $(i = 1, 2, \ldots, n)$ then $\mathtt{f}(t_1, t_2, \ldots, t_n) R \mathtt{f}(t_1', t_2', \ldots, t_n')$. If, additionally, R is an equivalence relation, then R is called a *congruence*.

A *congruence format* for a process preorder (equivalence) is a syntactic condition imposed on transition system specifications that guarantees the preorder (equivalence) to be a precongruence (resp. congruence) on the LTS induced by any specification satisfying the format.

The following result, proved in [18], shows that GSOS is a congruence format for bisimulation equivalence:

Proposition 2.12 For any transition system specification Λ in GSOS format, the bisimulation equivalence $\cong_{\mathsf{BS}}$ on the LTS induced by Λ is a congruence.

2.6 Basic Notions of Category Theory

To give an abstract account of labelled transition systems, operational preorders and equivalences, and transition system specifications, it is useful to apply some notions of category theory. In this section, standard categorical notions and results used throughout this thesis are recalled.

For rudimentary categorical notions of category, functor, natural transformation etc., unexplained here, see e.g. the textbook by Mac Lane [57].

2.6.1 Algebras and Monads

A common technique to represent syntax of process languages is to represent signatures as polynomial endofunctors. Then terms of a given language can be represented using the notions of initial algebra and freely generated monad.

Definition 2.13 Let Σ be an *endofunctor* on a category $\mathbb{C}$, i.e., a functor $\Sigma : \mathbb{C} \to \mathbb{C}$. A Σ-*algebra* $\langle X, h\rangle$ is an object $X \in |\mathbb{C}|$ together with a morphism $h : \Sigma X \to X$ in $\mathbb{C}$. X is then called the *carrier*, and h the *structure* of $\langle X, h\rangle$.

As a shorthand notation, a Σ-algebra $\langle X, h\rangle$ is usually denoted simply by its structure h. This never leads to confusion, since X is determined as the codomain of h.

Definition 2.14 Given an endofunctor $\Sigma : \mathbb{C} \to \mathbb{C}$, a Σ-*algebra morphism* (or Σ-*homomorphism*) from a Σ-algebra $g : \Sigma X \to X$ to a Σ-algebra $h : \Sigma Y \to Y$ is a morphism $f : X \to Y$ in $\mathbb{C}$ such that the diagram

$$\begin{array}{ccc} \Sigma X & \xrightarrow{\Sigma f} & \Sigma Y \\ {\scriptstyle g}\downarrow & & \downarrow{\scriptstyle h} \\ X & \xrightarrow[f]{} & Y \end{array}$$

commutes.

For any endofunctor Σ on $\mathbb{C}$, Σ-algebras together with Σ-algebra morphisms form a category, denoted Σ-**Alg**.

In the context of structural operational semantics, algebras are used to represent syntax of process languages. In particular, any signature Σ determines an endofunctor Σ on the category **Set** of small sets and functions, defined by

$$\begin{aligned} \Sigma X &= \coprod_{\mathtt{f} \in \bar{\Sigma}} X^{ar(\mathtt{f})} \\ \Sigma(g : X \to Y) &= \left[\iota_{\mathtt{f}} \left\langle x_1, \ldots, x_{ar(\mathtt{f})} \right\rangle \in \Sigma X \mapsto \iota_{\mathtt{f}} \left\langle g x_1, \ldots, g x_{ar(\mathtt{f})} \right\rangle \in \Sigma Y\right] \end{aligned}$$

where $\coprod$ denotes disjoint union of sets, and $\iota_{\mathtt{f}} : X^{ar(\mathtt{f})} \to \Sigma X$ is the coproduct injection corresponding to $\mathtt{f} \in \bar{\Sigma}$.

Then a Σ-algebra $h : \Sigma X \to X$ corresponds to a *model* for the signature Σ, i.e., a set X together with a function $\mathtt{f}_X : X^{ar(\mathtt{f})} \to X$ for every $\mathtt{f} \in \bar{\Sigma}$.

The functor Σ used to represent syntactic entities as algebras is usually called a *syntactic endofunctor*. This notion only reflects the context of use of Σ, and does not restrict the class of functors considered. However, in most examples (like in the definition of Σ based on a signature above) syntactic endofunctors are *polynomial functors*, i.e., they are built only from categorical products, coproducts, identity and constant functors.

Definition 2.15 Let Σ be an endofunctor. An *initial Σ-algebra* is an initial object in Σ-**Alg**, i.e., a Σ-algebra $\alpha : \Sigma A \to A$ such that for any Σ-algebra $f : \Sigma X \to X$ there exists a unique Σ-algebra morphism from α to f. This unique morphism is then called the *inductive extension* of f.

By the well-known Lambek's Lemma, for any endofunctor Σ, (structures of) initial Σ-algebras are isomorphisms.

Definition 2.16 Let Σ be an endofunctor on $\mathbb{C}$, and X be an object of $\mathbb{C}$. The *free algebra generated by Σ on X* is the initial $(X + \Sigma -)$-algebra.

For any syntactic endofunctor on **Set** determined by a signature Σ, there exists a free algebra generated by Σ on X:

$$[\eta_X, \psi_X] : X + \Sigma T X \to T X$$

with the set TX of terms over Σ with variables X as the carrier. In particular, the set of all closed terms $T0$ over Σ (here $0 = \emptyset$ is the initial object in **Set**), with the obvious Σ-algebra structure, forms an initial Σ-algebra.

The notions of precongruence and congruence from Definition 2.11 can be generalized to arbitrary Σ-algebras for endofunctors Σ obtained from signatures.

Definition 2.17 Let Σ be a signature and $h : \Sigma X \to X$ an algebra for the corresponding polynomial endofunctor on **Set**. A preorder $R \subseteq X \times X$ is a *precongruence on h* if for any coproduct injection $\mathtt{f} : X^n \to \Sigma X$, and for any $x_1, y_1, x_2, y_2, \ldots, x_n, y_n \in X$, whenever $x_i R y_i$ $(i = 1, 2, \ldots, n)$ then $h(\mathtt{f} \langle x_1, x_2, \ldots, x_n \rangle) R h(\mathtt{f} \langle y_1, y_2, \ldots, y_n \rangle)$. If, additionally, R is an equivalence relation, then R is called a *congruence* on h.

Precongruences (congruences) on the initial Σ-algebra $\psi : \Sigma T0 \to T0$ are exactly precongruences (resp. congruences) in the sense of Definition 2.11.

Free algebras generated by functors are particular examples of *monads*.

Definition 2.18 A *monad* $\langle T, \eta, \mu \rangle$ is an endofunctor T together with natural transformations $\eta : Id \to T$ and $\mu : TT \to T$ such that

$$\mu \circ T\eta = \mu \circ \eta T = id$$
$$\mu \circ T\mu = \mu \circ \mu T$$

The latter condition in the above definition is called *associativity of* μ.

Let Σ be an endofunctor on $\mathbb{C}$ such that for any object $X \in |\mathbb{C}|$, free algebras generated by Σ on X exist. Assume moreover an arbitrary choice of such free algebras $[\eta_X, \psi_X] : X + \Sigma TX \to TX$ for any X. For any morphism $f : X \to Y$, define $Tf : TX \to TY$ by initiality in $(X + \Sigma -)$-**Alg** as below:

$$\begin{array}{ccccc} X & \xrightarrow{\eta_X} & TX & \xleftarrow{\psi_X} & \Sigma TX \\ {\scriptstyle f}\downarrow & & \downarrow{\scriptstyle Tf} & & \downarrow{\scriptstyle \Sigma Tf} \\ Y & \xrightarrow{\eta_Y} & TY & \xleftarrow{\psi_Y} & \Sigma TY \end{array}$$

Moreover, for any object $X \in |\mathbb{C}|$, define a morphism $\mu_X : TTX \to TX$ by initiality in $(TX + \Sigma -)$-**Alg** as follows:

$$\begin{array}{ccccc} TX & \xrightarrow{\eta_{TX}} & TTX & \xleftarrow{\psi_{TX}} & \Sigma TTX \\ & \searrow & \downarrow{\scriptstyle \mu_X} & & \downarrow{\scriptstyle \Sigma\mu_X} \\ & & TX & \xleftarrow{\psi_X} & \Sigma TX \end{array}$$

A proof of the following standard result can be found, e.g., in [83]:

Proposition 2.19 The triple $\langle T, \eta, \mu \rangle$ forms a monad.

In particular, $\eta : Id \to T$ and $\mu : TT \to T$ are natural transformations. The monad $\langle T, \eta, \mu \rangle$ is called the *monad freely generated by* Σ.

Algebras for an endofunctor Σ are in 1-1 correspondence with those algebras for the endofunctor T (obtained as a part of the monad freely generated by Σ), which satisfy certain additional laws. More concretely:

Definition 2.20 Let $\langle T, \eta, \mu \rangle$ be a monad. A $\langle T, \eta, \mu \rangle$-*algebra* is a T-algebra $h : TX \to X$ such that

- $h \circ \eta_X = id_X$, and
- $h \circ \mu_X = h \circ Th$.

Moreover, $\langle T, \eta, \mu \rangle$-*algebra morphisms* are defined to be T-algebra morphisms.

When the natural transformations η and μ are irrelevant or clear from the context, we will speak of T-algebras instead of $\langle T, \eta, \mu \rangle$-algebras. To reconcile possible confusion, when an endofunctor T is (a part of) a monad, by T-algebras we will mean algebras for the monad, and not for the endofunctor, unless otherwise stated.

For any monad T on $\mathbb{C}$, T-algebras together with T-algebra morphisms form a category, denoted T-**Alg**.

Proposition 2.21 Let Σ be an endofunctor, and T be the monad freely generated by Σ. The categories Σ-**Alg** and T-**Alg** are isomorphic.

Proof. A full proof of this standard proposition can be found e.g. in [83]. Here we only note that to obtain a Σ-algebra from a T-algebra $f : TX \to X$, one takes the algebra $\Sigma X \xrightarrow{\Sigma\eta_X} \Sigma TX \xrightarrow{\psi_X} TX \xrightarrow{f} X$, where η_X, ψ_X come from the structure of T as the monad freely generated by Σ. □

In particular, if $f = \mu_X : TTX \to TX$, then the corresponding Σ-algebra is $\psi_X : \Sigma TX \to TX$, since the following diagram commutes:

$$\begin{array}{ccccc} \Sigma TX & \xrightarrow{\Sigma\eta_X} & \Sigma TTX & \xrightarrow{\psi_{TX}} & TTX \\ & \searrow & \downarrow{\scriptstyle \Sigma\mu_X} & & \downarrow{\scriptstyle \mu_X} \\ & & \Sigma TX & \xrightarrow[\psi_X]{} & TX \end{array}$$

(the square on the right commutes by definition of μ_X).

2.6.2 Coalgebras

In addition to *syntax*, which is usually represented by *algebras*, processes in various languages are equipped with *behaviour*, which often can be represented by another categorical entity, *coalgebras*. In this section, we recall basic definitions and results regarding coalgebras.

Definition 2.22 Let B be an endofunctor on a category $\mathbb{C}$. A B-*coalgebra* $\langle X, h \rangle$ is an object $X \in |\mathbb{C}|$ together with a morphism $h : X \to BX$ in $\mathbb{C}$. X is then called the *carrier*, and h the *structure* of $\langle X, h \rangle$.

Similarly to algebras, a B-coalgebra $\langle X, h \rangle$ is usually denoted simply by its structure h.

Definition 2.23 Given an endofunctor $B : \mathbb{C} \to \mathbb{C}$, a *$B$-coalgebra morphism* (or *B-cohomomorphism*) from a B-coalgebra $g : X \to BX$ to a B-coalgebra $h : Y \to BY$ is a morphism $f : X \to Y$ in $\mathbb{C}$ such that the diagram

$$\begin{array}{ccc} X & \xrightarrow{f} & Y \\ {\scriptstyle g}\downarrow & & \downarrow{\scriptstyle h} \\ BX & \xrightarrow[Bf]{} & BY \end{array}$$

commutes.

For any endofunctor B on $\mathbb{C}$, B-coalgebras together with B-coalgebra morphisms form a category, denoted B-**Coalg**.

The use of coalgebras in process algebra has been motivated by an easy 1-1 correspondence between $\mathcal{P}_{\mathrm{f}}(A \times X)$-coalgebras, (where A is an arbitrary set and $\mathcal{P}_{\mathrm{f}} : \mathbf{Set} \to \mathbf{Set}$ is the covariant finite powerset functor) and finitely branching labelled transition systems with A as the set of actions. Indeed, given an LTS $\langle X, A, \longrightarrow \rangle$, consider a function $h : X \to BX$ defined by

$$\langle a, x' \rangle \in hx \iff x \xrightarrow{a} x'$$

It is easy to check that this gives a 1-1 correspondence.

In the following we will often use the above correspondence silently, identifying finitely branching LTSs with their corresponding coalgebras.

Varying the endofunctor B, one obtains similar correspondences between coalgebras and deterministic automata, labelled transition systems with state predicates and many others [77].

The functor B used to represent transition systems as coalgebras is usually called a *behaviour endofunctor*. This notion only reflects the context of use of B, and does not restrict the class of functors considered.

The notion dual to that of initial algebra is that of final coalgebra:

Definition 2.24 Let B be an endofunctor. A *final B-coalgebra* is a final object in B-**Coalg**, i.e., a B-coalgebra $\phi : \Omega \to B\Omega$ such that for any B-coalgebra $h : X \to BX$ there exists a unique B-coalgebra morphism from h to ϕ. This unique morphism is then called the *coinductive extension* of h.

Again by Lambek's Lemma, for any endofunctor B, (structures of) final B-coalgebras are isomorphisms. This result implies, for example, that the full powerset functor $\mathcal{P} : \mathbf{Set} \to \mathbf{Set}$ does not admit final coalgebras. Indeed, there exists no set Ω isomorphic to $\mathcal{P}\Omega$.

However, the finite powerset functor $\mathcal{P}_{\mathrm{f}}$ admits final coalgebras, as was shown first by Barr [11, 83]. In particular:

Proposition 2.25 For any set A, the endofunctor $BX = \mathcal{P}_{\mathrm{f}}(A \times X)$ has a final coalgebra $\phi : \Omega \to B\Omega$ with Ω the set of (possibly infinitely deep) finitely branching labelled synchronization trees, with elements of A as actions, quotiented by bisimulation equivalence.

2.7 Abstract GSOS

The cornerstone of the abstract categorical modelling of transition system specifications is the following theorem by Turi and Plotkin [85]:

Theorem 2.26 There is a correspondence between transition system specifications (over a signature Σ) in the image finite GSOS format (see Definition 2.10) and natural transformations

$$\lambda : \Sigma(\mathrm{Id} \times \mathcal{P}_\mathrm{f}(A \times -)) \to \mathcal{P}_\mathrm{f}(A \times T-)$$

where T is the monad freely generated by (the endofunctor) Σ.

Proof. A full proof of this result can be found (in a marginally different version) in [83]. Here we only show how to construct a natural transformation λ from a set of rules Λ. Given a language construct $\mathtt{f} \in \bar{\Sigma}$ with arity n and a set X, a rule ρ in the GSOS format

$$\frac{\left\{ \mathtt{x}_i \xrightarrow{a_{ij}} \mathtt{y}_{ij} \,:\, i \leq n,\ j \leq m_i \right\} \cup \left\{ \mathtt{x}_i \not\xrightarrow{b_{ik}} \,:\, i \leq n, k \leq n_i \right\}}{\mathtt{f}(\mathtt{x}_1, \ldots, \mathtt{x}_n) \xrightarrow{c} \mathtt{t}}$$

defines a map $\rho_X : (X \times \mathcal{P}_\mathrm{f}(A \times X))^n \to \mathcal{P}_\mathrm{f}(A \times TX)$ as follows: $\langle c, t\rangle \in \rho_X \langle x_i, \beta_i\rangle_{i\leq n}$ iff there exists a substitution $\sigma : \Xi \to X$ satisfying

1. $\sigma(\mathtt{x_i}) = x_i$
2. $\forall i \leq n \, \forall j \leq m_i \ \langle a_{ij}, \sigma(\mathtt{y_{ij}})\rangle \in \beta_i$
3. $\forall i \leq n \, \forall k \leq n_i \, \forall x \in X \ \langle b_{ij}, x\rangle \notin \beta_i$
4. $\mathtt{t}\sigma = t$

Then given a set Λ of rules in the image finite GSOS format we can define a function $\lambda_X : \Sigma(X \times \mathcal{P}_\mathrm{f}(A \times X)) \to \mathcal{P}_\mathrm{f}(A \times TX)$ by defining for each $\mathtt{f} \in \Sigma$ a function $f_X : (X \times \mathcal{P}_\mathrm{f}(A \times X))^n \to \mathcal{P}_\mathrm{f}(A \times TX)$ as follows:

$$f_X : \langle x_i, U_i\rangle_{i\leq n} \quad \mapsto \bigcup_{\substack{\rho\in\Lambda \\ \rho \text{ a rule for } \mathtt{f}}} \rho_X \langle x_i, \beta_i\rangle_{i\leq n}$$

Image finiteness of Λ ensures that this function is well defined, i.e. that it returns only finite sets. Finally, λ_X is determined uniquely by the f_X's since it is a function from a coproduct. □

The above theorem inspired considerations about natural transformations

$$\lambda : \Sigma(\mathrm{Id} \times B) \to BT$$

for an arbitrary endofunctor B, and for $\langle T, \eta, \mu\rangle$ the monad freely generated by an arbitrary endofunctor Σ. In the following, transformations of this kind will be called *distributive laws*. Indeed, as shown in [56], they are equivalent to the distributive laws of the monad T over the copointed endofunctor $\mathrm{Id} \times B$, i.e., to transformations

$$\lambda : T(\mathrm{Id} \times B) \to BT$$

satisfying the laws

- $\lambda \circ \eta(\mathrm{Id} \times B) = B\eta \circ \pi_2 \quad : \mathrm{Id} \times B \to BT$
- $\lambda \circ \mu(\mathrm{Id} \times B) = B\mu \circ \lambda T \circ T\,\langle T\pi_1, \lambda\rangle \quad : TT(\mathrm{Id} \times B) \to BT.$

A notion closely related to distributive laws is that of bialgebra.

Definition 2.27 Let $\lambda : T(\mathrm{Id} \times B) \to BT$ be a distributive law of a monad T over $\mathrm{Id} \times B$. A λ-*bialgebra* is a pair $TX \xrightarrow{h} X \xrightarrow{g} BX$ satisfying the "pentagonal law":

$$\begin{array}{ccccc} T(X \times BX) & & \xrightarrow{\lambda} & & BTX \\ {\scriptstyle T\langle id, g\rangle}\uparrow & & & & \downarrow{\scriptstyle Bh} \\ TX & \xrightarrow[h]{} & X & \xrightarrow[g]{} & BX \end{array}$$

A λ-bialgebra morphism from $TX \xrightarrow{h} X \xrightarrow{g} BX$ to $TX' \xrightarrow{h'} X' \xrightarrow{g'} BX'$ is a morphism $f : X \to X'$ which is simultaneously a T-algebra morphism and a B-coalgebra morphism, i.e. $h' \circ Tf = f \circ h$ and $g' \circ f = Bf \circ g$.

For a given λ, the category of λ-bialgebras and λ-bialgebra morphisms is denoted λ-**Bialg**.

The following theorem is the main result of [83] and [85]. It provides a basis for the fibrational framework of Chapter 3 and for congruence formats defined in Chapter 5.

Theorem 2.28 Suppose $\mathbb{C}$ is a category with initial objects, final objects and products, Σ is an endofunctor on $\mathbb{C}$ which freely generates a monad T (in particular, Σ and T admit initial algebras), and B is an endofunctor on $\mathbb{C}$ which admits final coalgebras. Let $\lambda : T(\mathrm{Id} \times B) \to BT$ be a distributive law of T over $\mathrm{Id} \times B$. Then λ-**Bialg** has an initial object $TT0 \xrightarrow{\mu_0} T0 \xrightarrow{h_\lambda} BT0$ (where 0 is the initial object in $\mathbb{C}$) and a final object $T\Omega \xrightarrow{\delta} \Omega \xrightarrow{\phi} B\Omega$. Moreover,

1. $\mu_0 : TT0 \to T0$ is the initial object in T-**Alg**,
2. $\phi : \Omega \to B\Omega$ is the final object in B-**Coalg**,
3. h_λ is called the *intended operational model of* λ, and is defined by $h_\lambda = \lambda_0 \circ T!$, where $! : 0 \to (0 \times B0)$ is unique by initiality,
4. $\delta : T\Omega \to \Omega$ is the unique B-coalgebra morphism making the diagram

$$\begin{array}{ccccc} T\Omega & \xrightarrow{T\langle id, \phi\rangle} & T(\Omega \times B\Omega) & \xrightarrow{\lambda_\Omega} & BT\Omega \\ {\scriptstyle \delta}\downarrow & & & & \downarrow{\scriptstyle B\delta} \\ \Omega & & \xrightarrow[\phi]{} & & B\Omega \end{array}$$

commute.

In particular, if $\mathbb{C} = \mathbf{Set}$, $B = \mathcal{P}_{\mathrm{f}}(A \times -)$ and Σ is obtained from a signature, then the intended operational model of λ is the LTS generated by the image finite GSOS rules associated to λ. The proof of this was left implicit in [85], and in [83] a slightly different version was shown. A full proof can be found in [13].

Chapter 3

A Fibrational Approach to Relations on Processes

In this chapter, we present a general and abstract approach to well-structured relations on processes modelled as coalgebras. The main purpose is to introduce the test suite approach used in the following chapters. However, in order to to illustrate connections with related work, we shall begin with a rather general and abstract framework, based on lifting coalgebras and the abstract GSOS construction (see Section 2.7) to total categories of fibrations.

After explaining the basic notions of fibrations, total categories and lifting, and after some general considerations (in Section 3.1) on when such lifting can be performed, we show (in Section 3.2) how, given a finitely branching LTS, i.e., a coalgebra $h : X \to \mathcal{P}_{\mathrm{f}}(A \times X)$ in **Set**, to represent bisimulations and the bisimulation equivalence on h as coalgebras for a suitably chosen endofunctor on the total category **Rel** of a fibration of relations. This construction, a special case of the general fibrational approach, coincides with abstract constructions of Hermida, Jacobs and Hughes [41, 45], who also lifted coalgebras to the category of relations. We notice, however, that e.g. the trace preorder and the trace equivalence on finitely branching LTSs cannot be represented in the same fashion.

To overcome this problem, we propose (in Section 3.3) another special case of the general fibrational framework, based on a fibration of *test suites*, which have more structure than relations. Roughly, a *test* on a set X of processes is a function from X to a fixed set of *test values*. A test suite is simply a set of tests with the same domain and codomain. Given a test suite on X, a relation on X can be obtained by a construction analogous to that of specialisation order in general topology. By choosing appropriate functors B^{W} on the category of test suites, various relations on carriers of B-coalgebras can be represented as specialization relations of B^{W}-coalgebras.

In the next chapter, it will be shown that this approach allows to characterize most of the preorders and equivalences from the van Glabbeek spectrum.

3.1 Fibrations

In this section, we present basic definitions of fibrations and show how to lift the framework of abstract GSOS to the fibrational setting. For a simple example of the constructions presented here, refer to Section 3.2.

It must be stressed that the definitions and constructions presented in this section are not fully general. In particular, the notions of fibration and bifibration shown here are only special cases of the general notions used in fibration theory. We use the simplified definitions here, since they are general enough to perform all constructions needed for the purposes of this thesis. For a general account of fibrations and related topics, see [43].

Definition 3.1 (Grothendieck construction) Assume a category $\mathbb{C}$ (called the *base category* in this context) and a functor $(-)^* : \mathbb{C}^{op} \to \mathbf{Pos}$ (**Pos** is the category of partally ordered sets and monotonic functions). The *total category* $\mathbb{C}^*$ is defined as follows:

- objects in $\mathbb{C}^*$ are pairs $\langle X, \theta \rangle$, where X is an object in $\mathbb{C}$ and $\theta \in X^*$,
- morphisms $f : \langle X, \theta \rangle \to \langle Y, \vartheta \rangle$ are morphisms $f : X \to Y$ in $\mathbb{C}$ such that $\theta \leq f^* \vartheta$.

For a total category $\mathbb{C}^*$ obtained in this manner, the obvious forgetful functor $p : \mathbb{C}^* \to \mathbb{C}$ is called a *(split) fibration*, and the poset X^* is called the *fibre over* X. Given a morphism $f : X \to Y$ in $\mathbb{C}$, the monotonic function $f^* : Y^* \to X^*$ is called the *reindexing function along* f.

Slightly abusing the terminology, we will also call functors $(-)^* : \mathbb{C}^{op} \to \mathbf{Pos}$ fibrations.

Definition 3.2 A fibration $(-)^* : \mathbb{C}^{op} \to \mathbf{Pos}$ is a *bifibration*, if for any $f : X \to Y$ in $\mathbb{C}$, the reindexing function $f^* : Y^* \to X^*$ is a right adjoint or, equivalently, if it has a left adjoint $f_! : X^* \to Y^*$, i.e., a monotonic function such that for any $\theta \in X^*$ and $\vartheta \in Y^*$, $\theta \leq f^* \vartheta$ if and only if $f_! \theta \leq \vartheta$.

Simple examples of fibrations and bifibrations used throughout this thesis are given in Definitions 3.14 and 3.20.

In the remainder of this section it is shown how to lift the framework of abstract GSOS from the base category $\mathbb{C}$ to the total category $\mathbb{C}^*$ of any bifibration. The culmination of this section is Theorem 3.13 which, in Chapter 5, will eventually lead to the definition of congruence formats for various process equivalences.

3.1.1 Lifting Basic Universal Constructions

Theorem 3.3 Let $(-)^* : \mathbb{C}^{op} \to \mathbf{Pos}$ be a bifibration. If $\mathbb{C}$ has a final object 1, and the fibre 1^* has a top element $\top_1$, then $\langle 1, \top_1 \rangle$ is a final object in $\mathbb{C}^*$.

Proof. For any object $\langle X, \theta \rangle$ in $\mathbb{C}^*$ take the unique morphism $1 : X \to 1$ in $\mathbb{C}$. Then $1 : \langle X, \theta \rangle \to \langle 1, \top_1 \rangle$ is a valid morphism in $\mathbb{C}^*$, since $1_! \theta \leq \top_1$, and (transposing across the adjunction) $\theta \leq 1^* \top_1$. □

Theorem 3.4 Let $(-)^* : \mathbb{C}^{op} \to \mathbf{Pos}$ be a bifibration. If $\mathbb{C}$ has binary products $X \xleftarrow{\pi_1} X \times Y \xrightarrow{\pi_2} Y$, and each fibre has binary meets $\wedge$, then $\mathbb{C}^*$ has binary products $\langle X, \theta \rangle \times \langle Y, \vartheta \rangle = \langle X \times Y, \theta \boxtimes \vartheta \rangle$, where

$$\theta \boxtimes \vartheta = \pi_1^* \theta \wedge \pi_2^* \vartheta$$

Proof. The projection $\pi_1 : \langle X \times Y, \theta \boxtimes \vartheta \rangle \to \langle X, \theta \rangle$ is clearly a valid morphism in $\mathbb{C}^*$, since $\pi_1^* \theta \wedge \pi_2^* \vartheta \leq \pi_1^* \theta$. The same for π_2.

To show universality, assume two morphisms $\langle X, \theta \rangle \xleftarrow{f} \langle Z, \zeta \rangle \xrightarrow{g} \langle Y, \vartheta \rangle$ and take the unique morphism $\langle f, g \rangle : Z \to X \times Y$ in $\mathbb{C}$ arising from universality of $X \times Y$. Then $\zeta \leq f^* \theta = \langle f, g \rangle^* \pi_1^* \theta$ and $\zeta \leq g^* \theta = \langle f, g \rangle^* \pi_2^* \vartheta$. Transposing across the adjunction, one gets $\langle f, g \rangle_! \zeta \leq \pi_1^* \theta$ and $\langle f, g \rangle_! \zeta \leq \pi_2^* \vartheta$, hence $\langle f, g \rangle_! \zeta \leq \pi_1^* \theta \wedge \pi_2^* \vartheta$. Again transposing across the adjunction, one gets $\zeta \leq \langle f, g \rangle^* (\pi_1^* \theta \wedge \pi_2^* \vartheta)$ and, as a consequence,

$$\langle f, g \rangle : \langle Z, \zeta \rangle \to \langle X \times Y, \theta \boxtimes \vartheta \rangle$$

is a valid morphism in $\mathbb{C}^*$. □

Theorem 3.5 Let $(-)^* : \mathbb{C}^{op} \to \mathbf{Pos}$ be a fibration. If $\mathbb{C}$ has an initial object 0, and the fibre 0^* has a bottom element $\bot_0$, then $\langle 0, \bot_0 \rangle$ is an initial object in $\mathbb{C}^*$.

Proof. For any object $\langle X, \theta \rangle$ in $\mathbb{C}^*$, take the unique morphism $0 : 0 \to X$ in $\mathbb{C}$. Then $0 : \langle 0, \bot_0 \rangle \to \langle X, \theta \rangle$ is a valid morphism in $\mathbb{C}^*$. □

Theorem 3.6 Let $(-)^* : \mathbb{C}^{op} \to \mathbf{Pos}$ be a bifibration. If $\mathbb{C}$ has binary coproducts $X \xrightarrow{\iota_1} X + Y \xleftarrow{\iota_2} Y$, and each fibre has binary joins $\vee$, then $\mathbb{C}^*$ has binary coproducts $\langle X, \theta \rangle + \langle Y, \vartheta \rangle = \langle X + Y, \theta \boxplus \vartheta \rangle$, where

$$\theta \boxplus \vartheta = (\iota_1)_! \theta \vee (\iota_2)_! \vartheta$$

Proof. The injection $\iota_1 : \langle X, \theta \rangle \to \langle X + Y, \theta \boxplus \vartheta \rangle$ is a valid morphism in $\mathbb{C}^*$, since $\theta \leq \iota_1^* (\iota_1)_! \theta \leq \iota_1^* ((\iota_1)_! \theta \vee (\iota_2)_! \vartheta)$. The same for ι_2.

To show universality, assume two morphisms $\langle X, \theta \rangle \xrightarrow{f} \langle Z, \zeta \rangle \xleftarrow{g} \langle Y, \vartheta \rangle$ and take the unique morphism $[f, g] : X + Y \to Z$ in $\mathbb{C}$ arising from universality of $X + Y$. Then $\theta \leq f^* \zeta = \iota_1^* [f, g]^* \zeta$ and $\vartheta \leq g^* \zeta = \iota_2^* [f, g]^* \zeta$. Transposing across the adjunctions, one gets $(\iota_1)_! \theta \leq [f, g]^* \zeta$ and $(\iota_2)_! \vartheta \leq [f, g]^* \zeta$, hence $(\iota_1)_! \theta \vee (\iota_2)_! \vartheta \leq [f, g]^* \zeta$. As a consequence,

$$[f, g] : \langle X + Y, \theta \boxplus \vartheta \rangle \to \langle Z, \zeta \rangle$$

is a valid morphism in $\mathbb{C}^*$. □

3.1.2 Lifting Endofunctors

Let $(-)^* : \mathbb{C}^{op} \to \mathbf{Pos}$ be a fibration, and $p : \mathbb{C}^* \to \mathbb{C}$ the associated forgetful functor. An endofunctor $F^* : \mathbb{C}^* \to \mathbb{C}^*$ *lifts* an endofunctor $F : \mathbb{C} \to \mathbb{C}$ if $p \circ F^* = F \circ p$.

Endofunctors on $\mathbb{C}^*$ lifting a given functor $F : \mathbb{C} \to \mathbb{C}$ are determined by their *actions*, i.e., families of monotonic functions $\{ F_X : X^* \to (FX)^* \ : \ X \in |\mathbb{C}| \}$. If, for any $f : X \to Y$ in $\mathbb{C}$, the following inequality holds:

$$\begin{array}{ccc} Y^* & \xrightarrow{F_Y} & (FY)^* \\ {\scriptstyle f^*}\downarrow & \leq & \downarrow{\scriptstyle (Ff)^*} \\ X^* & \xrightarrow[F_X]{} & (FX)^* \end{array}$$

then F^* defined by

$$F^* \langle X, \theta \rangle = \langle FX, F_X \theta \rangle \qquad F^* f = Ff$$

is an endofunctor on $\mathbb{C}^*$ and it lifts F. If the above inequality holds as equality, we say that F^* is *fibred*.

In the following, given an endofunctor F^* lifting some endofunctor F, we will denote its action on an object X with F_X. This will never lead to confusion, as F^* will always be clear from the context.

Given endofunctors F^* and G^* on $\mathbb{C}^*$ lifting endofunctors F and G on $\mathbb{C}$ respectively, the composed endofunctor F^*G^* indeed lifts the endofunctor FG and is defined by the action

$$\{ F_{GX} \circ G_X \ : \ X \in |\mathbb{C}| \}$$

3.1.3 Lifting Initial Algebras

Let $(-)^* : \mathbb{C}^{op} \to \mathbf{Pos}$ be a bifibration. Consider an endofunctor F on $\mathbb{C}$ lifted to an endofunctor F^* on $\mathbb{C}^*$.

For any F-algebra $h : FX \to X$, define an operator $\Psi_h : X^* \to X^*$ by

$$\Psi_h \theta = h_! F_X \theta$$

It is clearly monotonic, since both $h_!$ and F_X are monotonic.

F^*-algebras correspond to prefixed points of such operators: an F-algebra $h : FX \to X$ can be lifted to an F^*-algebra $h : \langle FX, F_X \theta \rangle \to \langle X, \theta \rangle$ if and only if $\theta \geq \Psi_h \theta$. In this case, one says that θ *lifts* h to an F^*-algebra.

Theorem 3.7 If F has an initial algebra $\psi : FA \to A$, and Ψ_ψ has the least (pre)fixed point α, then $\psi : \langle FA, F_A \alpha \rangle \to \langle A, \alpha \rangle$ is an initial F^*-algebra.

Proof. First, ψ as above is well defined in $\mathbb{C}^*$, since $\psi_! F_A \alpha = \Psi_\psi \alpha \leq \alpha$, hence (transposing across the adjunction) $F_A \alpha \leq \psi^* \alpha$.

Consider any F^*-algebra $h : \langle FX, F_X \theta \rangle \to \langle X, \theta \rangle$ and let $k : A \to X$ be the inductive extension of $h : FX \to X$ in $\mathbb{C}$. It is enough to show that $k : \langle A, \alpha \rangle \to \langle X, \theta \rangle$ is a valid morphism in $\mathbb{C}$, i.e., that $\alpha \leq k^* \theta$. To this end, one shows that $k^* \theta$ is a prefixed point of Ψ_ψ.

Indeed, $\psi^* k^* \theta = (Fk)^* h^* \theta \geq (Fk)^* F_X \theta \geq F_A k^* \theta$ (the first inequality holds since h is an F^*-algebra, and the second since F^* is a functor, see Section 3.1.2). Transposing across the adjunction, one gets $k^* \theta \geq \psi_! F_A k^* \theta = \Psi_\psi k^* \theta$. □

Reindexing along F-algebra morphisms in the base category $\mathbb{C}$ preserves well-definedeness of F^*-algebras in the total category $\mathbb{C}^*$:

Theorem 3.8 Consider an F-algebra morphism in $\mathbb{C}$:

$$\begin{array}{ccc} FX & \xrightarrow{Fk} & FY \\ {\scriptstyle h}\downarrow & & \downarrow{\scriptstyle g} \\ X & \xrightarrow[k]{} & Y \end{array}$$

and suppose that F lifts to F^* on $\mathbb{C}^*$. Then for any lifting of g to an F^*-algebra $g : \langle FY, F_Y \theta \rangle \to \langle Y, \theta \rangle$, the morphism

$$h : \langle FX, F_X k^* \theta \rangle \to \langle X, k^* \theta \rangle$$

is a valid F^*-algebra in $\mathbb{C}^*$.

Proof. Calculate

$$F_X k^* \theta \leq (Fk)^* F_Y \theta \leq (Fk)^* g^* \theta = h^* k^* \theta$$

(the first inequality holds since F^* is a functor, see Section 3.1.2). □

3.1.4 Lifting Freely Generated Monads

Let $(-)^* : \mathbb{C}^{op} \to \mathbf{Pos}$ be a bifibration and assume that coproducts and initial algebras in $\mathbb{C}$ lift to $\mathbb{C}^*$ as described in Theorems 3.6 and 3.7.

Theorem 3.9 Consider an endofunctor $\Sigma : \mathbb{C} \to \mathbb{C}$ which freely generates a monad $T : \mathbb{C} \to \mathbb{C}$ and lifts to an endofunctor $\Sigma^* : \mathbb{C}^* \to \mathbb{C}^*$. Then Σ^* freely generates a monad T^* and T^* lifts T.

Proof. For any object $\langle X, \theta \rangle$ in $\mathbb{C}^*$, the functor $\langle X, \theta \rangle + \Sigma^* -$ lifts the functor $X + \Sigma -$, hence it admits initial algebras lifting the initial $(X + \Sigma -)$-algebras. By Theorem 3.7, if $[\eta_X, \psi_X] : X + \Sigma TX \to TX$ is an initial $(X + \Sigma -)$-algebra, then

$$[\eta_X, \psi_X] : \langle X + \Sigma TX, \theta \boxplus \Sigma_{TX} T_X \theta \rangle \to \langle TX, T_X \theta \rangle$$

is an initial $(\langle X, \theta \rangle + \Sigma^* -)$-algebra, where $T_X \theta$ is the least fixed point of the operator

$$\Psi_{[\eta_X, \psi_X]} \vartheta = [\eta_X, \psi_X]_! (\theta \boxplus \Sigma_{TX} \vartheta)$$

This, as shown in Section 2.6.1, gives a functor $T^* : \mathbb{C}^* \to \mathbb{C}^*$, which is the monad freely generated by Σ^*. □

When T^* is the monad freely generated by Σ^*, the categories of T^*-algebras and Σ^*-algebras are isomorphic by Proposition 2.21. Moreover, the isomorphism is the same as that between T-algebras and Σ-algebras:

Theorem 3.10 Under the assumptions of Theorem 3.9, for any T^*-algebra $h : \langle TX, T_X\theta\rangle \to \langle X, \theta\rangle$, the morphism

$$g : \langle \Sigma X, \Sigma_X \theta\rangle \to \langle X, \theta\rangle$$

is a well-defined Σ^*-algebra in $\mathbb{C}^*$, where $g : \Sigma X \to X$ is the algebra corresponding to $h : TX \to X$ along the isomorphism between T-**Alg** and Σ-**Alg**.

Proof. By Proposition 2.21 one has $g = h \circ \psi_X \circ \Sigma\eta_X$, where $\psi_X : \Sigma TX \to TX$ and $\eta_X : X \to TX$ come from the structure of the monad T. From the proof of Theorem 3.9 it is clear that morphisms

$$\eta_X : \langle X, \theta\rangle \to \langle TX, T_X\theta\rangle$$
$$\psi_X : \langle \Sigma TX, \Sigma_{TX} T_X\theta\rangle \to \langle TX, T_X\theta\rangle$$

are well-defined in $\mathbb{C}^*$, therefore $g : \langle \Sigma X, \Sigma_X\theta\rangle \to \langle X, \theta\rangle$, being a composition of three well-defined morphisms, is also well-defined. □

3.1.5 Lifting Final Coalgebras

Let $(-)^* : \mathbb{C}^{op} \to$ **Pos** be a bifibration. Consider an endofunctor F on $\mathbb{C}$ lifted to an endofunctor F^* on $\mathbb{C}^*$.

For any F-coalgebra $h : X \to FX$, define an operator $\Phi_h : X^* \to X^*$ by

$$\Phi_h\theta = h^* F_X \theta$$

It is clearly monotonic, since both h^* and F_X are monotonic.

F^*-coalgebras correspond to postfixed points of such operators: an F-coalgebra $h : X \to FX$ can be lifted to an F^*-coalgebra $h : \langle X, \theta\rangle \to \langle FX, F_X\theta\rangle$ if and only if $\theta \leq \Psi_h\theta$. In this case, one says that θ *lifts* h to an F^*-coalgebra.

Theorem 3.11 If F has a final coalgebra $\phi : \Omega \to F\Omega$, and Φ_ϕ has the greatest (post)fixed point ω, then $\phi : \langle \Omega, \omega\rangle \to \langle F\Omega, F_\Omega\omega\rangle$ is the final F^*-coalgebra.

Proof. First, ϕ as above is well defined in $\mathbb{C}$, since $\omega \leq \Phi_\phi\omega = \phi^* F_\Omega\omega$.

Consider any F^*-coalgebra $h : \langle X, \theta\rangle \to \langle FX, F_X\theta\rangle$ and let $k : X \to \Omega$ be the coinductive extension of $h : X \to FX$ in $\mathbb{C}$. It is enough to show that $k : \langle X, \theta\rangle \to \langle \Omega, \omega\rangle$ is a valid morphism in $\mathbb{C}$, i.e., that $\theta \leq k^*\omega$, or equivalently, that $k_!\theta \leq \omega$. To this end, one shows that $k_!\theta$ is a postfixed point of Φ_ϕ, or equivalently, that $k^*\Phi_\phi k_!\theta \geq \theta$.

Indeed, $k^*\Phi_\phi k_!\theta = k^*\phi^* F_\Omega k_!\theta = h^*(Fk)^* F_\Omega k_!\theta \geq h^* F_X k^* k_!\theta \geq h^* F_X\theta \geq \theta$ (the first inequality holds since F^* is a functor, the second due to the adjunction $k_! \dashv k^*$, and the third since h is an F^*-coalgebra). □

Final F^*-coalgebras in $\mathbb{C}^*$ are particularly important when F^* is fibred, as they allow to construct, for any F-coalgebra $h : X \to FX$, the greatest element $\theta \in X^*$ that lifts h to an F^*-coalgebra:

Theorem 3.12 Under the assumptions of Theorem 3.11, suppose moreover that F^* is fibred. Then for any coalgebra $h : X \to FX$, the greatest (post)fixed point of Φ_h (or, equivalently, the greatest element of X^* that lifts h to an F^*-coalgebra) exists and is equal to $k^*\omega$, where $k : X \to \Omega$ is the coinductive extension of h.

Proof. To show that $k^*\omega$ is a postfixed point of Φ_h, calculate

$$k^*\omega \leq k^*\phi^* F_\Omega \omega = h^*(Fk)^* F_\Omega \omega = h^* F_X k^* \omega = \Phi_h k^* \omega$$

(the inequality hold since ω is a postfixed point of Φ_ϕ, and the second equality holds since F^* is fibred).

Consider another postfixed point θ for Φ_h, i.e., an F^*-coalgebra $h : \langle X, \theta\rangle \to \langle FX, F_X\theta\rangle$. By finality, $k : \langle X, \theta\rangle \to \langle \Omega, \omega\rangle$ is a valid morphism in $\mathbb{C}^*$, hence $\theta \leq k^*\omega$. □

In many examples (see Section 3.2.1, Chapters 4, 6), the elements $k^*\omega$ will correspond to well-known process relations on carriers of coalgebras.

3.1.6 Lifting Abstract GSOS

Assume a bifibration where initial objects, products, coproducts, initial algebras and final coalgebras lift. Consider a functor $\Sigma : \mathbb{C} \to \mathbb{C}$ that freely generates a monad T and lifts to a functor $\Sigma^* : \mathbb{C}^* \to \mathbb{C}^*$. Assume moreover a functor $B : \mathbb{C} \to \mathbb{C}$ that has a final coalgebra $\phi : \Omega \to B\Omega$ and lifts to a fibred functor $B^* : \mathbb{C}^* \to \mathbb{C}^*$. Assume also a natural transformation

$$\lambda : \Sigma(\mathrm{Id} \times B) \to BT$$

in $\mathbb{C}$.

If λ lifts to a natural transformation in $\mathbb{C}^*$, then the entire abstract GSOS construction (see Section 2.7) can be repeated in $\mathbb{C}^*$. For our purposes, the most important result will be the following:

Theorem 3.13 Under the above notation, if λ lifts to a natural transformation

$$\lambda : \Sigma^*(\mathrm{Id} \times B^*) \to B^*T^*$$

in $\mathbb{C}^*$ (where T^* is the monad freely generated by Σ^*, see Theorem 3.9) then

$$\psi : \langle \Sigma T0, \Sigma_{T0} k^*\omega\rangle \to \langle T0, k^*\omega\rangle$$

is a valid Σ^*-algebra, where $\psi : \Sigma T0 \to T0$ is the initial Σ-algebra, ω comes from the final B^*-coalgebra $\phi : \langle \Omega, \omega\rangle \to \langle B\Omega, B_\Omega\omega\rangle$ and $k : T0 \to \Omega$ is the coinductive extension of the coalgebraic part of the initial λ-bialgebra.

Proof. By Theorem 2.28, k is the unique morphism from the initial to the final λ-bialgebra in $\mathbb{C}$:

$$\begin{array}{ccccc} TT0 & \xrightarrow{\mu_0} & T0 & \xrightarrow{\gamma} & BT0 \\ {\scriptstyle Tk}\downarrow & & {\scriptstyle k}\downarrow & & \downarrow{\scriptstyle Bk} \\ T\Omega & \xrightarrow[\delta]{} & \Omega & \xrightarrow[\phi]{} & B\Omega \end{array}$$

and δ is the unique B-coalgebra morphism making the diagram

$$\begin{array}{ccccc} T\Omega & \xrightarrow{T\langle id,\phi\rangle} & T(\Omega \times B\Omega) & \xrightarrow{\lambda_\Omega} & BT\Omega \\ {\scriptstyle \delta}\downarrow & & & & \downarrow{\scriptstyle B\delta} \\ \Omega & & \xrightarrow[\phi]{} & & B\Omega \end{array}$$

commute. Since products and the natural transformation λ in $\mathbb{C}$ lift to $\mathbb{C}^*$, both morphisms

$$\begin{array}{rcl} T\langle id,\phi\rangle & : & \langle T\Omega, T_\Omega\omega\rangle \to \langle T(\Omega \times B\Omega), T_{\Omega\times B\Omega}(\omega \boxtimes B_\Omega\omega)\rangle \\ \lambda_\Omega & : & \langle T(\Omega \times B\Omega), T_{\Omega\times B\Omega}(\omega \boxtimes B_\Omega\omega)\rangle \to \langle BT\Omega, B_{T\Omega}T_\Omega\omega\rangle \end{array}$$

are well defined in $\mathbb{C}^*$, hence $\lambda_\Omega \circ T\langle id,\phi\rangle : \langle T\Omega, T_\Omega\omega\rangle \to \langle BT\Omega, B_{T\Omega}T_\Omega\omega\rangle$ is a valid B^*-coalgebra. By finality of ϕ both in $\mathbb{C}$ and $\mathbb{C}^*$, δ is the unique B^*-coalgebra morphism as shown below:

$$\begin{array}{ccc} \langle T\Omega, T_\Omega\omega\rangle & \xrightarrow{\lambda_\Omega\circ T\langle id,\phi\rangle} & \langle BT\Omega, B_{T\Omega}T_\Omega\omega\rangle \\ {\scriptstyle \delta}\downarrow & & \downarrow{\scriptstyle B\delta} \\ \langle \Omega, \omega\rangle & \xrightarrow[\phi]{} & \langle B\Omega, B_\Omega, \omega\rangle \end{array}$$

hence $\delta : \langle T\Omega, T_\Omega\omega\rangle \to \langle \Omega, \omega\rangle$ is a well defined T^*-algebra. By Theorem 3.8 applied to this T^*-algebra and to the T-algebra morphism k, the T^*-algebra

$$\mu_0 : \langle TT0, T_{T0}k^*\omega\rangle \to \langle T0, k^*\omega\rangle$$

is well defined in $\mathbb{C}^*$. From this it is easy to infer that also

$$\psi : \langle \Sigma T0, \Sigma_{T0}k^*\omega\rangle \to \langle T0, k^*\omega\rangle$$

is a valid Σ^*-algebra. Indeed, by the remark made after Proposition 2.21, the T-coalgebra $\mu_0 : TT0 \to T0$ is mapped to $\psi : \Sigma T0 \to T0$ by the isomorphism between T**-Alg** and Σ**-Alg**. Then it is enough to apply Theorem 3.10. □

3.2 Fibration of Relations

As an example of a bifibration and an application of Theorems 3.11–3.12, we show the fibration of relations on the base category **Set**. In Section 3.2.1, we show how to represent bisimulations and bisimulation equivalences on finitely branching LTSs as coalgebras in this framework. This construction is a special case of the abstract approach by Hermida, Jacobs and Hughes [41, 45]. A similar construction was also used by Pitts [65] in the context of recursively defined domains. There, even a counterpart of the operator Φ_h was defined.

In Section 3.2.2, we show limitations of the relational approach by proving that trace equivalence cannot be easily represented in the same fashion.

Definition 3.14 Consider a functor $(-)^* : \mathbf{Set}^{op} \to \mathbf{Pos}$ defined as follows:

- $X^* = \langle \{R \subseteq X \times X\}, \subseteq \rangle$,
- $f^*R = \{ \langle x, x' \rangle \in X \times X \,:\, (fx)R(fx') \}$ for $f : X \to Y$, $R \in Y^*$.

To check functoriality, calculate for any $f : X \to Y$, $g : Y \to Z$ and $R \in Z^*$:

$$\begin{array}{rcl} (g \circ f)^* R & = & \{ \langle x, x' \rangle \in X \times X \,:\, (gfx)R(gfx') \} \\ & = & \{ \langle x, x' \rangle \in X \times X \,:\, (fx)(g^*R)(fx') \} \\ & = & f^* g^* R \end{array}$$

Monotonicity and preservation of identities is immediate.

In the total category $\mathbf{Set}^*$, a morphism $f : \langle X, R \rangle \to \langle Y, S \rangle$ is valid if and only if xRy implies $(fx)S(fy)$. Therefore $\mathbf{Set}^*$ is the category **Rel** of binary relations and relation-preserving functions.

Theorem 3.15 The functor $(-)^*$ defined as above is a bifibration.

Proof. Given a function $f : X \to Y$, consider a function $f_! : X^* \to Y^*$ defined by

$$f_! R = \{ \langle fx, fy \rangle \,:\, x, y \in X,\ xRy \}$$

For any $f : X \to Y$, $f_!$ is a left adjoint to f^*, since

$$\begin{array}{ll} R \subseteq f^* S & \iff \\ R \subseteq \{ \langle x, y \rangle \,:\, x, y \in X,\ (fx)S(fy) \} & \iff \\ \forall x, y \in X. xRy \Rightarrow (fx)S(fy) & \iff \\ \{ \langle fx, fy \rangle \,:\, x, y \in X,\ xRy \} \subseteq S & \iff \\ f_! R \subseteq S & \end{array}$$

□

Since **Set** has initial and final objects, products and coproducts, and all fibres of $(-)^*$ have all joins and meets, also the total category **Rel** has initial and final objects, products and coproducts as defined in Theorems 3.3–3.6.

3.2.1 Bisimulations as Coalgebras in Rel

For a given set A, consider the functor $B = \mathcal{P}_{\mathrm{f}}(A \times -)$ on **Set**, where $\mathcal{P}_{\mathrm{f}}$ denotes the (covariant) finite powerset functor. As was shown in Section 2.6.2, B-coalgebras in **Set** correspond to finitely branching LTSs edge-labelled with elements of A.

B can be lifted to the total category **Rel**, using the action $B_X : X^* \to (BX)^*$ defined by:

$$\begin{array}{rl} B_X R = \{ \langle \beta, \gamma \rangle \,:\, & \forall \langle a, x \rangle \in \beta.\ \exists \langle a, y \rangle \in \gamma.\ xRy \\ & \text{and } \forall \langle a, y \rangle \in \gamma.\ \exists \langle a, x \rangle \in \beta.\ xRy \} \end{array}$$

Indeed,

Theorem 3.16 The above action lifts B to a fibred endofunctor B^* on **Rel**.

Proof. To show that for every $f : X \to Y$, and any relation $R \in Y^*$, the equality $B_X f^* R = (Bf)^* B_Y R$ holds, calculate

$$\begin{aligned}
(Bf)^* B_Y R &= (Bf)^* \{\langle \beta, \gamma\rangle : \forall \langle a, x\rangle \in \beta.\ \exists \langle a, y\rangle \in \gamma.\ xRy \\
&\qquad\qquad \text{and } \forall \langle a, y\rangle \in \gamma.\ \exists \langle a, x\rangle \in \beta.\ xRy\} \\
&= \{\langle \beta, \gamma\rangle : \forall \langle a, x\rangle \in (Bf)\beta.\ \exists \langle a, y\rangle \in (Bf)\gamma.\ xRy \\
&\qquad\qquad \text{and } \forall \langle a, y\rangle \in (Bf)\gamma.\ \exists \langle a, x\rangle \in (Bf)\beta.\ xRy\} \\
&= \{\langle \beta, \gamma\rangle : \forall \langle a, x\rangle \in \beta.\ \exists \langle a, y\rangle \in \gamma.\ (fx)R(fy) \\
&\qquad\qquad \text{and } \forall \langle a, y\rangle \in \gamma.\ \exists \langle a, x\rangle \in \beta.\ (fx)R(fy)\} \\
&= \{\langle \beta, \gamma\rangle : \forall \langle a, x\rangle \in \beta.\ \exists \langle a, y\rangle \in \gamma.\ x(f^*R)y \\
&\qquad\qquad \text{and } \forall \langle a, y\rangle \in \gamma.\ \exists \langle a, x\rangle \in \beta.\ x(f^*R)y\} \\
&= B_X f^* R
\end{aligned}$$

□

The following theorem gives a correspondence between B^*-coalgebras and bisimulations, and hints that the fibrational approach can be used to describe various relations on processes modelled as coalgebras.

Theorem 3.17 Given a B-coalgebra (a finitely branching LTS) $h : X \to BX$, a binary relation R on X is a bisimulation on h if and only if

$$h : \langle X, R\rangle \to \langle BX, B_X R\rangle$$

is a valid B^*-coalgebra.

Proof. Calculate

$$\begin{aligned}
h^* B_X R &= h^* \{\langle \beta, \gamma\rangle : \forall \langle a, x\rangle \in \beta.\ \exists \langle a, y\rangle \in \gamma.\ xRy \\
&\qquad\qquad \text{and } \forall \langle a, y\rangle \in \gamma.\ \exists \langle a, x\rangle \in \beta.\ xRy\} \\
&= \{\langle x, y\rangle : \forall \langle a, x'\rangle \in hx.\ \exists \langle a, y'\rangle \in hy.\ x'Ry' \\
&\qquad\qquad \text{and } \forall \langle a, y'\rangle \in hy.\ \exists \langle a, x'\rangle \in hx.\ x'Ry'\} \\
&= \{\langle x, y\rangle : \forall x \xrightarrow{a} x'.\ \exists y \xrightarrow{a} y'.\ x'Ry' \\
&\qquad\qquad \text{and } \forall y \xrightarrow{a} y'.\ \exists x \xrightarrow{a} x'.\ x'Ry'\}
\end{aligned}$$

Therefore $h : \langle X, R\rangle \to \langle BX, B_X R\rangle$ is well defined in **Rel** if and only if

$$R \subseteq \left\{ \langle x, y\rangle \ :\ \forall x \xrightarrow{a} x'.\ \exists y \xrightarrow{a} y'.\ x'Ry' \text{ and } \forall y \xrightarrow{a} y'.\ \exists x \xrightarrow{a} x'.\ x'Ry' \right\}$$

which is just the definition of a bisimulation (Definition 2.6). □

Having characterized bisimulations as coalgebras, it is easy to characterize bisimulation equivalence:

Corollary 3.18 Given an LTS $h : X \to BX$, the largest relation on X that lifts h to a B^*-coalgebra exists and is equal to the bisimulation equivalence $\cong_{\mathsf{BS}}$ on h.

Proof. Immediate from Theorem 3.17 and Proposition 2.8. □

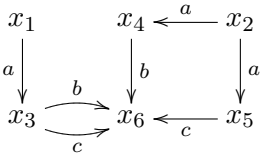
3.2.2 Limited Expressive Power of Rel

Corollary 3.18 hints that it might be possible to develop an abstract theory of process equivalences in the fibrational setting. However, as the following counterexample shows, it is not possible to represent even trace equivalence in exactly the same way as bisimulation equivalence in the fibration **Rel**.

Counterexample 3.19 There exists no functor $B^* : \mathbf{Rel} \to \mathbf{Rel}$ lifting $BX = \mathcal{P}_{\mathrm{f}}(A \times X) : \mathbf{Set} \to \mathbf{Set}$ such that for any coalgebra $h : X \to BX$, the trace equivalence $\cong_{\mathsf{Tr}}$ on h is the greatest relation that lifts h to a B^*-coalgebra.

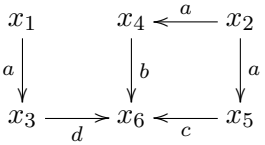
Proof. Assume that such a functor B^* exists and consider a set of processes $X = \{x_1, x_2, x_3, x_4, x_5, x_6\}$ and a relation $R = \{\langle x_1, x_2\rangle\} \cup =_X$ on X. Fix an arbitrary element $a \in A$ and consider two cases.

Case 1: $\langle \{\langle a, x_3\rangle\}, \{\langle a, x_4\rangle, \langle a, x_5\rangle\}\rangle \notin B_X R$. Then consider $h : X \to BX$ defined by the following LTS:



Here R is the trace equivalence on h, but $\langle x_1, x_2\rangle \notin h^* B_X R$, hence R does not lift h to a B^*-coalgebra.

Case 2: $\langle \{\langle a, x_3\rangle\}, \{\langle a, x_4\rangle, \langle a, x_5\rangle\}\rangle \in B_X R$. Then consider $h : X \to BX$ defined by the following LTS:



Here $=_X$ is the trace equivalence on h, but $R \subseteq h^* B_X R$, hence $=_X$ is not the largest relation that lifts h to a B^*-coalgebra. □

This counterexample hints that to characterize trace equivalence in the relational framework, one probably needs to resort to techniques analogous to those used in the coalgebra span framework [71], as was recently done in [44].

3.3 Fibration of Test Suites

Counterexample 3.19 suggests that the fibration of relations is not structured enough to represent all relations in the van Glabbeek spectrum. In this section, another fibration is proposed, based on the notion of tests and test suites.

Intuitively, any test suite on some set X induces some relation on X, depending on the distinguishing power of this test suite. As it turns out, however, test suites carry significantly more structure than relations, and using them one can represent a wide spectrum of relations on processes.

Definition 3.20 Fix any object $\mathcal{V}$ (called the object of *test values*) in a category $\mathbb{C}$ and consider a functor $\mathcal{T}_\mathcal{V} : \mathbb{C}^{op} \to \mathbf{Pos}$ defined as follows:

- $\mathcal{T}_\mathcal{V} X = \langle \mathcal{P}\mathbb{C}(X, \mathcal{V}), \supseteq \rangle$
- $(\mathcal{T}_\mathcal{V} f)\theta = \{V \circ f \mid V \in \theta\}$ for $f : X \to Y$, $\theta \in \mathcal{T}_\mathcal{V} Y$

where $\mathcal{P}$ denotes the powerset construction. A morphism from X to $\mathcal{V}$ in $\mathbb{C}$ is called a *$\mathcal{V}$-test on X*, and a set of such tests is called a *$\mathcal{V}$-test suite on X*. Then $\mathcal{T}_\mathcal{V} X$ is a set of all $\mathcal{V}$-test suites on X, ordered by reverse inclusion. To say that θ is a $\mathcal{V}$-test suite on X (i.e., $\theta \in \mathcal{T}_\mathcal{V} X$), we will write $\theta : X \rightrightarrows \mathcal{V}$.

To check that the above indeed defines a functor, calculate

$$(\mathcal{T}_\mathcal{V}(g \circ f))\theta = \{V \circ (g \circ f) \mid V \in \theta\} = \{(V \circ g) \circ f \mid V \in \theta\} = (\mathcal{T}_\mathcal{V} f)(\mathcal{T}_\mathcal{V} g)\theta$$

for any $f : X \to Y$, $g : Y \to Z$ and $\theta : Z \rightrightarrows \mathcal{V}$. Monotonicity and preservation of identities is immediate.

In the following, if $\mathcal{V}$ will be clear from the context or irrelevant, we will speak just of tests and test suites instead of $\mathcal{V}$-tests and $\mathcal{V}$-test suites. To keep the notation from Section 3.1, if $\mathcal{V}$ is irrelevant or clear from the context, and if no other fibration is around, we will often write f^* instead of $\mathcal{T}_\mathcal{V} f$ to denote reindexing functions. The omission of $\mathcal{V}$ in this shorthand notation will never lead to confusion since the definition of $\mathcal{T}_\mathcal{V} f$ looks the same independently of the choice of $\mathcal{V}$.

The total category arising from the above fibration along the lines of Definition 3.1 will be denoted $\mathcal{V}$-**TS**.

Theorem 3.21 For any object $\mathcal{V}$, the functor $\mathcal{T}_\mathcal{V}$ is a bifibration.

Proof. Given a morphism $f : X \to Y$ in $\mathbb{C}$, consider the monotonic function $f_! : \mathcal{T}_\mathcal{V} X \to \mathcal{T}_\mathcal{V} Y$ defined by

$$f_!\theta = \{\, V : Y \to \mathcal{V} \,:\, V \circ f \in \theta \,\}$$

Then $f_!$ is a left adjoint to f^*, since for any $\theta : X \rightrightarrows \mathcal{V}$ and $\vartheta : Y \rightrightarrows \mathcal{V}$,

$$\begin{array}{ll} \theta \supseteq f^*\vartheta & \iff \\ \theta \supseteq \{V \circ f \mid V \in \vartheta\} & \iff \\ \forall V \in \vartheta.\ V \circ f \in \theta & \iff \\ f_!\theta \supseteq \vartheta & \end{array}$$

□

To prove existence of initial algebras and final algebras of endofunctors on $\mathcal{V}$-**TS**, the following lemma will be useful.

Lemma 3.22 All fibres $\mathcal{T}_{\mathcal{V}}X$ in $\mathcal{V}$-**TS** are complete lattices with joins $\bigcap$ and meets $\bigcup$, and for any $f : X \to Y$, the reindexing function f^* and its left adjoint $f_!$ are continuous with respect to the complete lattice structure.

Proof. f^*, being a right adjoint, preserves arbitrary meets. From this, by general properties of complete lattices, it follows that f^* preserves also arbitrary joins. Similarly for $f_!$. □

Remark 3.23 In the context of test suite fibrations, the discussion presented in Sections 3.1.3–3.1.5 may be a little confusing, since the ordering in the fibres is the reverse of the intuitive inclusion order on test suites. For example, the least fixed points (with respect to the ordering in fibres) of certain operators discussed in Section 3.1.3 are the greatest fixed points of the same operators with respect to inclusion. To avoid confusion, we will always clearly mark whether the least (greatest) fixed points considered are with respect to the fibre ordering or the set-theoretic inclusion ordering. (and the latter, more intuitive interpretation will be normally used).

3.3.1 Specialization Functors

Fix a test suite fibration $\mathcal{T}_{\mathcal{V}}$ over a category $\mathbb{C}$, and consider another bifibration $(-)^* : \mathbb{C} \to \mathbf{Pos}$ with total category $\mathbb{C}^*$. Assume that all fibres in $\mathbb{C}^*$ have arbitrary meets.

Definition 3.24 Any object $\langle \mathcal{V}, R\rangle \in |\mathbb{C}^*|$ gives rise to a *specialization functor* $\mathrm{Spec}_R : \mathcal{V}$-**TS** $\to \mathbb{C}^*$, defined as follows:

- $\mathrm{Spec}_R \langle X, \theta\rangle = \langle X, \mathrm{Sp}_R\, \theta\rangle$, where $\mathrm{Sp}_R\, \theta = \bigwedge_{V\in\theta} V^*R$
- $\mathrm{Spec}_R\, f = f$

To check functoriality, the only thing to show is that for any morphism $f : \langle X, \theta\rangle \to \langle Y, \vartheta\rangle$ in $\mathcal{V}$-**TS**,

$$f : \langle X, \mathrm{Sp}_R\, \theta\rangle \to \langle Y, \mathrm{Sp}_R\, \vartheta\rangle$$

is a valid morphism in $\mathbb{C}^*$. To this end, assume $\theta \supseteq (\mathcal{T}_{\mathcal{V}} f)\vartheta = \{V \circ f \mid V \in \vartheta\}$ and calculate

$$\mathrm{Sp}_R\, \theta = \bigwedge_{V\in\theta} V^*R \leq \bigwedge_{V\in\vartheta} (V \circ f)^*R = \bigwedge_{V\in\vartheta} f^*V^*R = f^* \bigwedge_{V\in\vartheta} V^*R = f^* \,\mathrm{Sp}_R\, \vartheta$$

(the last but one equality holds since f^*, being a right adjoint, preserves meets).

Example 3.25 Consider $\mathbb{C} = \mathbf{Set}$, $\mathbb{C}^* = \mathbf{Rel}$ and $\mathcal{V} = 2 = \{\mathtt{tt}, \mathtt{ff}\}$. Then a 2-test V on a set X corresponds to a subset of X defined by $\{x \in X \mid Vx = \mathtt{tt}\}$. Similarly, a 2-test suite on X corresponds to a family of subsets of X. For example, every topology with carrier X can be seen as a 2-test suite on X.

Now consider the equality relation $=_2$ on 2, and the linear order relation $\leq_2 = \{\langle \mathtt{ff}, \mathtt{tt}\rangle\} \cup =_2$. Then for any $\theta : X \rightrightarrows 2$,

$$\begin{aligned} \mathrm{Sp}_{=_2}\theta &= \{\langle x, y\rangle \in X \times X \mid \forall V \in \theta.\, Vx = Vy\} \\ \mathrm{Sp}_{\leq_2}\theta &= \{\langle x, y\rangle \in X \times X \mid \forall V \in \theta.\, Vx = \mathtt{tt} \Rightarrow Vy = \mathtt{tt}\} \end{aligned}$$

In the following, the relations $\mathrm{Sp}_{=_2}\theta$ and $\mathrm{Sp}_{\leq_2}\theta$ will be denoted by $\equiv_\theta$ and $\leq_\theta$, and called the specialization equivalence and the specialization preorder of θ, respectively.

For example, if a 2-test suite θ on X is a T_0-topology, then $\leq_\theta$ is the well-known specialization order of θ.

Example 3.26 Consider any test suite category $\mathcal{V}$-**TS** over the base category $\mathbb{C}$, and take $\mathbb{C}^* = \mathcal{W}$-**TS** for some object $\mathcal{W} \in |\mathbb{C}|$. For any $\theta : \mathcal{V} \rightrightarrows \mathcal{W}$, the specialization functor $\mathrm{Spec}_\theta : \mathcal{V}$-**TS** $\to \mathcal{W}$-**TS** acts on objects as follows:

$$\begin{aligned} \mathrm{Spec}_\theta \langle X, \vartheta\rangle &= \langle X, \mathrm{Sp}_\theta \vartheta\rangle\,, \text{ where} \\ \mathrm{Sp}_\theta \vartheta &= \bigwedge_{V \in \vartheta} V^*\theta \\ &= \bigcup_{V \in \vartheta} \{W \circ V \mid W \in \theta\} \\ &= \{W \circ V \mid W \in \theta, V \in \vartheta\} \end{aligned}$$

3.3.2 Comparing Test Suites

It is sometimes possible to compare the values of a specialization functor (e.g., specialization relations) for two given test suites simply by looking at the test suites. In simple cases, the following obvious result is useful, saying that specialization functors are (reverse) monotonic:

Proposition 3.27 Consider a specialization functor $\mathrm{Spec}_R : \mathcal{V}$-**TS** $\to \mathbb{C}^*$ for an object $\langle \mathcal{V}, R\rangle$ in $\mathbb{C}^*$. For any $\mathcal{V}$-test suites θ, ϑ on the same object X, if $\theta \subseteq \vartheta$ then $\mathrm{Sp}_R\,\theta \geq \mathrm{Sp}_R\,\vartheta$.

Proof. Immediate by the definition of Sp_R. □

However, often it is possible to compare test suites based on different test values.

Indeed, consider two objects $\langle \mathcal{V}, R\rangle$ and $\langle \mathcal{W}, S\rangle$ in the total category $\mathbb{C}^*$ of some bifibration where all fibres have arbitrary meets. Along the lines of Definition 3.24, these objects give rise to two specialization functors $\mathrm{Spec}_R : \mathcal{V}$-**TS** $\to \mathbb{C}^*$ and $\mathrm{Spec}_S : \mathcal{W}$-**TS** $\to \mathbb{C}^*$, respectively. Assume also a test suite $\zeta : \mathcal{V} \rightrightarrows \mathcal{W}$, which gives rise to a specialization functor $\mathrm{Spec}_\zeta : \mathcal{V}$-**TS** $\to \mathcal{W}$-**TS**, as shown in Example 3.26. Furthermore, consider any object $X \in |\mathbb{C}|$ and two test suites $\theta : X \rightrightarrows \mathcal{V}$, $\vartheta : X \rightrightarrows \mathcal{W}$.

The following result allows to compare results of different specialization functors on θ and ϑ:

Theorem 3.28 Under the above notation,

- if $R \leq \mathrm{Sp}_S\,\zeta$ and $\vartheta \subseteq \mathrm{Sp}_\zeta\,\theta$, then $\mathrm{Sp}_R\,\theta \leq \mathrm{Sp}_S\,\vartheta$,
- if $R = \mathrm{Sp}_S\,\zeta$ and $\vartheta = \mathrm{Sp}_\zeta\,\theta$, then $\mathrm{Sp}_R\,\theta = \mathrm{Sp}_S\,\vartheta$,
- if $R \geq \mathrm{Sp}_S\,\zeta$ and $\vartheta \supseteq \mathrm{Sp}_\zeta\,\theta$, then $\mathrm{Sp}_R\,\theta \geq \mathrm{Sp}_S\,\vartheta$,

Proof. To prove the first statement, calculate

$$\begin{aligned}\mathrm{Sp}_R\,\theta &= \bigwedge_{V\in\theta} V^*R \leq \bigwedge_{V\in\theta} V^*\,\mathrm{Sp}_S\,\zeta = \bigwedge_{V\in\theta} V^* \bigwedge_{W\in\zeta} W^*S = \\ &= \bigwedge_{\substack{V\in\theta\\W\in\zeta}} V^*W^*S = \bigwedge_{U\in\mathrm{Sp}_\zeta\,\theta} U^*S \leq \bigwedge_{U\in\vartheta} U^*S = \mathrm{Sp}_S\,\vartheta\end{aligned}$$

(for the second equality in the second line, cf. Example 3.26). Proofs of the remaining statements are entirely analogous. □

Proposition 3.27 is an easy corollary from Theorem 3.28. Indeed, it is enough to consider $\mathcal{V} = \mathcal{W}$, $R = S$ and $\zeta = \{id_\mathcal{V}\}$. However, Theorem 3.28 has also other implications.

Example 3.29 Consider $\mathbb{C} = \mathbf{Set}$, $\mathbb{C}^* = \mathbf{Rel}$, $\mathcal{V} = \mathcal{W} = 2$, $R = =_2$ and $S = \leq_2$ as in Example 3.25, and let $\zeta = \{id_2\}$ contain just the identity function on 2. It is then immediate that $R \leq S = \mathrm{Sp}_S\,\zeta$. The first statement of Theorem 3.28 implies that for any $\theta, \vartheta : X \rightrightarrows 2$, if $\vartheta \subseteq \theta$ then $\equiv_\theta \subseteq \leq_\vartheta$ (see Example 3.25).

3.3.3 Lifting Functors with Test Constructors

Any functor $B : \mathbb{C} \to \mathbb{C}$ can be lifted to a fibred endofunctor on the total category of test suites $\mathcal{V}$-**TS** in possibly many ways. However, for the purpose of describing operational relations on processes (coalgebras), one particular way is especially useful. This well-structured method of lifting endofunctors is based on notions of test constructors and closures.

Definition 3.30 Let B be an endofunctor on $\mathbb{C}$, and $\mathcal{V}$ an object in $\mathbb{C}$. Morphisms from $B\mathcal{V}$ to $\mathcal{V}$ (i.e., $\mathcal{V}$-tests on $B\mathcal{V}$) are called $(B, \mathcal{V})$-*test constructors*.

Where B and $\mathcal{V}$ are clear from the context, we will sometimes speak simply of test constructors.

Definition 3.31 Let $\mathcal{V}$ be an object in $\mathbb{C}$. A $\mathcal{V}$-*closure* (or simply closure, if $\mathcal{V}$ is clear from the context) on test suites is a family (indexed by objects $X \in |\mathbb{C}|$) of monotonic functions

$$\mathrm{Cl}_X : \mathcal{P}\mathbb{C}(X, \mathcal{V}) \to \mathcal{P}\mathbb{C}(X, \mathcal{V})$$

such that for any morphism $f : X \to Y$ in $\mathbb{C}$, and for any $\theta : Y \rightrightarrows \mathcal{V}$, the equality

$$\mathrm{Cl}_X\, f^*\theta = f^*\,\mathrm{Cl}_Y\,\theta$$

holds.

A $\mathcal{V}$-closure can be viewed as an action that lifts the identity functor on $\mathbb{C}$ to a fibred functor on $\mathcal{V}$-**TS** (cf. Section 3.1.2).

Theorem 3.32 Let B be an endofunctor on $\mathbb{C}$ and $\mathcal{V}$ an object in $\mathbb{C}$. Any set $\mathsf{W} : B\mathcal{V} \rightrightarrows \mathcal{V}$ of $(B, \mathcal{V})$-test constructors with any $\mathcal{V}$-closure Cl induce a lifting of B to a fibred endofunctor B^{W} on $\mathcal{V}$-**TS**, defined by the action

$$B^{\mathsf{W}}_X\theta = \mathrm{Cl}_{BX}\{w \circ BV \mid w \in \mathsf{W}, V \in \theta\}$$

Proof. To check that the mapping B^{W} defined by

$$B^{\mathsf{W}}(\langle X, \theta\rangle) = \left\langle BX, B^{\mathsf{W}}_X\theta \right\rangle \qquad B^{\mathsf{W}}f = Bf$$

is a fibred endofunctor on $\mathcal{V}$-**TS**, calculate, for any $f : X \to Y$ in $\mathbb{C}$ and for any $\theta : Y \rightrightarrows \mathcal{V}$,

$$\begin{aligned} B_X f^*\theta &= \mathrm{Cl}_{BX}\{w \circ BV \mid w \in \mathsf{W}, V \in f^*\theta\} \\ &= \mathrm{Cl}_{BX}\{w \circ BV \circ Bf \mid w \in \mathsf{W}, V \in \theta\} \\ &= \mathrm{Cl}_{BX}(Bf)^*\{w \circ BV \mid w \in \mathsf{W}, V \in \theta\} \\ &= (Bf)^* \mathrm{Cl}_{BY}\{w \circ BV \mid w \in \mathsf{W}, V \in \theta\} \\ &= (Bf)^* B^{\mathsf{W}}_Y\theta \end{aligned}$$

□

Recall from Theorem 3.11 that if B has a final coalgebra $\phi : \Omega \to B\Omega$, then B^W has a final coalgebra $\phi : \langle \Omega, \omega\rangle \to \left\langle B\Omega, B^W_\Omega \omega\right\rangle$, where ω is the *least* (with respect to set inclusion, see Remark 3.23) (pre)fixed point of the operator Φ_ϕ. It is easy to see that the actions B^{W}_X from Theorem 3.32 are monotonic with respect to set inclusion. As closures are monotonic by definition, also operators Φ_ϕ are monotonic and, as monotonic functions on complete lattices, they have the least (pre)fixed points.

Moreover, by Theorem 3.12 (since B^W is fibred), for any coalgebra $h : X \to BX$, the test suite $k^*\omega$ (where $k : X \to \Omega$ is the coinductive extension of h) is the *least* (pre)fixed point of the operator Φ_h and the *least* test suite that lifts h to a B^W-coalgebra in $\mathcal{V}$-**TS**.

Note that every action B^{W}_X as in Theorem 3.32 in fact preserves unions of increasing chains of sets. Therefore, if a corresponding closure also preserves such unions, then the least (pre)fixed point of the operator Φ_h is characterized by

$$k^*\omega = \bigcup_{n\in\mathbb{N}} \Phi^n_h\emptyset$$

3.3.4 Comparing Test Constructors

Often it is possible to relate two endofunctors generated by different sets of test constructors, even if they involve different test set values. More specifically, given any coalgebra for one endofunctor one might try to construct a related

coalgebra for another one, provided some kind of correspondence between the two endofunctors.

Formally, consider a functor $B : \mathbb{C} \to \mathbb{C}$ and two objects $\mathcal{V}, \mathcal{V}' \in \mathbb{C}$. Assume sets of test constructors $\mathsf{W} : B\mathcal{V} \rightrightarrows \mathcal{V}$ and $\mathsf{W}' : B\mathcal{V}' \rightrightarrows \mathcal{V}'$. Fix also a $\mathcal{V}$-closure Cl and a $\mathcal{V}'$-closure Cl'. As shown in Section 3.3.3, W together with Cl induce a fibred endofunctor B^{W} on $\mathcal{V}$-**TS**. Similarly, W' together with Cl' induce a fibred endofunctor $B^{\mathsf{W}'}$ on $\mathcal{V}'$-**TS**.

Furthermore, assume a test suite $\zeta : \mathcal{V} \rightrightarrows \mathcal{V}'$, which gives rise to a specialization functor $\mathrm{Spec}_\zeta : \mathcal{V}$-**TS** $\to \mathcal{V}'$-**TS**, as shown in Example 3.26. The following theorem shows how one may relate coalgebras for B^{W} and $B^{\mathsf{W}'}$ using the test suite ζ.

Theorem 3.33 Under the above notation, assume that

- for any test suite $\theta : X \rightrightarrows \mathcal{V}$,

$$\mathrm{Cl}'_X \,\mathrm{Sp}_\zeta\, \theta \subseteq \mathrm{Sp}_\zeta(\mathrm{Cl}_X\, \theta)$$

- for every $w' \in \mathsf{W}'$ and $z' \in \zeta$ there are $w \in \mathsf{W}$ and $z \in \zeta$ such that $z \circ w = w' \circ Bz'$.

Then for any B^{W}-coalgebra

$$h : \langle X, \theta \rangle \to \left\langle BX, B^{\mathsf{W}}_X \theta \right\rangle$$

the morphism

$$h : \left\langle X, \mathrm{Sp}_\zeta\, \theta \right\rangle \to \left\langle BX, B^{\mathsf{W}'}_X \,\mathrm{Sp}_\zeta\, \theta \right\rangle$$

is a well-defined $B^{\mathsf{W}'}$-coalgebra in $\mathcal{V}'$-**TS**.

Proof. Calculate

$$\begin{array}{rcl}
h^* B^{\mathsf{W}'}_X \,\mathrm{Sp}_\zeta\, \theta & = & h^* B^{\mathsf{W}'} \left\{ z' \circ V \,:\, z' \in \zeta, V \in \theta \right\} \\
& = & h^* \,\mathrm{Cl}'_{BX} \left\{ w' \circ Bz' \circ BV \,:\, w' \in \mathsf{W}', z' \in \zeta, V \in \theta \right\} \\
& = & \mathrm{Cl}'_X \, h^* \left\{ w' \circ Bz' \circ BV \,:\, w' \in \mathsf{W}', z' \in \zeta, V \in \theta \right\} \\
& = & \mathrm{Cl}'_X \left\{ w' \circ Bz' \circ BV \circ h \,:\, w' \in \mathsf{W}', z' \in \zeta, V \in \theta \right\} \\
& \subseteq & \mathrm{Cl}'_X \left\{ z \circ w \circ BV \circ h \,:\, z \in \zeta, w \in \mathsf{W}, V \in \theta \right\} \\
& = & \mathrm{Cl}'_X \,\mathrm{Sp}_\zeta \left\{ w \circ BV \circ h \,:\, w \in \mathsf{W}, V \in \theta \right\} \\
& \subseteq & \mathrm{Sp}_\zeta \,\mathrm{Cl}_X \left\{ w \circ BV \circ h \,:\, w \in \mathsf{W}, V \in \theta \right\} \\
& = & \mathrm{Sp}_\zeta \, h^* B^{\mathsf{W}}_X \theta \\
& \subseteq & \mathrm{Sp}_\zeta \, \theta
\end{array}$$

□

Theorem 3.33 allows one to compare, similarly as in Section 3.3.2, the values of specialization functors (e.g., specialization relations) on two test suites

induced as final coalgebras from two different sets of test constructors, simply by looking at the test constructors.

Under the notation introduced before Theorem 3.33, consider two objects $\langle \mathcal{V}, R\rangle$ and $\langle \mathcal{V}', S\rangle$ in a total category $\mathbb{C}^*$ of some bifibration where all fibres have arbitrary meets. Along the lines of Definition 3.24, these objects give rise to two specialization functors $\mathrm{Spec}_R : \mathcal{V}\text{-}\mathbf{TS} \to \mathbb{C}^*$ and $\mathrm{Spec}_S : \mathcal{V}'\text{-}\mathbf{TS} \to \mathbb{C}^*$, respectively.

For a given B-coalgebra $h : X \to BX$, let $k^*\omega : X \rightrightarrows \mathcal{V}$ denote the least test suite such that $h : \langle X, k^*\omega\rangle \to \left\langle BX, B^{\mathsf{W}}_X k^*\omega\right\rangle$ is a valid B^{W}-coalgebra. Similarly, let $k^*\omega' : X \rightrightarrows \mathcal{V}'$ denote the least test suite such that $h : \langle X, k^*\omega'\rangle \to \left\langle BX, B^{\mathsf{W}'}_X k^*\omega'\right\rangle$ is a valid $B^{\mathsf{W}'}$-coalgebra. Both $k^*\omega$ and $k^*\omega'$ are assumed to exist and be characterized by Theorem 3.12.

Corollary 3.34 Under the above notation, assume that the conditions of Theorem 3.33 hold. If, moreover,

$$R \leq \mathrm{Sp}_S\, \zeta$$

then

$$\mathrm{Sp}_R\, k^*\omega \leq \mathrm{Sp}_S\, k^*\omega'$$

Proof. As known from Theorem 3.33,

$$h : \left\langle X, \mathrm{Sp}_\zeta\, k^*\omega\right\rangle \to \left\langle BX, B^{\mathsf{W}'}_X\, \mathrm{Sp}_\zeta\, k^*\omega\right\rangle$$

is a well-defined $B^{\mathsf{W}'}$-coalgebra, hence

$$k^*\omega' \subseteq \mathrm{Sp}_\zeta\, k^*\omega$$

The corollary now follows from Theorem 3.28. □

A useful special case of Theorem 3.33 and Corollary 3.34 is

Corollary 3.35 For any functor $B : \mathbb{C} \to \mathbb{C}$, an object $\mathcal{V} \in \mathbb{C}$, two sets of test constructors $\mathsf{W}, \mathsf{W}' : B\mathcal{V} \rightrightarrows \mathcal{V}$ and corresponding $\mathcal{V}$-closures $\mathrm{Cl}, \mathrm{Cl}'$, if $\mathsf{W}' \subseteq \mathsf{W}$, and if for any $\theta : X \rightrightarrows \mathcal{V}$, $\mathrm{Cl}'_X\, \theta \subseteq \mathrm{Cl}_X\, \theta$, then for any valid B^{W}-coalgebra

$$h : \langle X, \theta\rangle \to \left\langle BX, B^{\mathsf{W}}_X \theta\right\rangle$$

the morphism

$$h : \langle X, \theta\rangle \to \left\langle BX, B^{\mathsf{W}'}_X \theta\right\rangle$$

is a valid $B^{\mathsf{W}'}$-coalgebra. If, moreover, $R \leq S$, then

$$\mathrm{Sp}_R\, k^*\omega \leq \mathrm{Sp}_S\, k^*\omega'$$

where $k^*\omega$ and $k^*\omega'$ are defined as before Corollary 3.34.

Proof. Apply Theorem 3.33 and Corollary 3.34 with $\mathcal{V} = \mathcal{V}'$ and $\zeta = \{id_{\mathcal{V}}\}$. □

Several simple applications of Theorem 3.33 and Corollaries 3.34–3.35 are shown in Section 4.5.

Chapter 4

Van Glabbeek Spectrum Described by Test Suites

In this chapter, we specialize the general test suite framework described in Chapter 3 to describe various notions of process equivalences and preorders from the van Glabbeek spectrum (see Chapter 2).

For this purpose, fix the category $\mathbb{C} = \mathbf{Set}$, and the behaviour functor $BX = \mathcal{P}_{\mathrm{f}}(A \times X)$ for a fixed set A.

In most of this chapter, we will only consider the test value set $\mathcal{V} = 2 = \{\mathtt{tt}, \mathtt{ff}\}$ together with the test suite category 2-**TS** and its associated fibration. We will also consider two specialization functors: $\mathrm{Spec}_{=_2}, \mathrm{Spec}_{\leq_2} : 2\text{-}\mathbf{TS} \to \mathbf{Rel}$. Recall from Example 3.25 that for any 2-test suite θ, the relations $\mathrm{Sp}_{=_2}\theta$ and $\mathrm{Sp}_{\leq_2}\theta$ are denoted $\equiv_\theta$ and $\leq_\theta$, respectively.

As mentioned in Example 3.25, every 2-test on X can be identified with a subset of X. However, to avoid confusion, we introduce special notation for this correspondence.

Notation 4.1 Given a 2-test $V : X \to 2$ on X, the set $\{x \in X \mid Vx = \mathtt{tt}\}$ will be denoted $\overline{V}$. Similarly, given a set $Y \subseteq X$, the test $\overrightarrow{Y} : X \to 2$ is defined by

$$\overrightarrow{Y}x = \begin{cases} \mathtt{tt} & \text{if } x \in Y \\ \mathtt{ff} & \text{otherwise} \end{cases}$$

We will also speak of *unions* and *intersections* of 2-tests, denoted and defined as follows:

$$\begin{aligned} V \vee V' &= \overrightarrow{\overline{V} \cup \overline{V'}} \\ V \wedge V' &= \overrightarrow{\overline{V} \cap \overline{V'}} \end{aligned}$$

This notation extends in the obvious way to unions and intersections of arbitrary families of 2-tests.

In the following sections, various liftings B^{W} of B to 2-**TS** will be proposed, based on different choices of $\mathsf{W} : B2 \rightrightarrows 2$. For various choices of W, B^{W}-coalgebras will be related, by means of specialization functors, to process equivalences and preorders described in Definition 2.4.

4.1 (B,2)-Test Suite Constructors

We begin by defining some $(B, 2)$-test constructors, useful to represent most of relations from the van Glabbeek spectrum.

Definition 4.2 For any $a \in A$, $Q \subseteq A$, define functions

$$w_{\langle a\rangle}, w_{[a]}, \tilde{w}_Q, \tilde{w}_{aQ}, \check{w}_Q, \check{w}_{aQ} : B2 \to 2$$

as follows:

$$w_{\langle a\rangle}\beta = \begin{cases} \mathtt{tt} & \text{if } \langle a, \mathtt{tt}\rangle \in \beta \\ \mathtt{ff} & \text{otherwise} \end{cases}$$

$$w_{[a]}\beta = \begin{cases} \mathtt{ff} & \text{if } \langle a, \mathtt{ff}\rangle \in \beta \\ \mathtt{tt} & \text{otherwise} \end{cases}$$

$$\tilde{w}_Q\beta = \begin{cases} \mathtt{tt} & \text{if } \beta \cap (Q \times 2) = \emptyset \\ \mathtt{ff} & \text{otherwise} \end{cases}$$

$$\tilde{w}_{aQ}\beta = \begin{cases} \mathtt{tt} & \text{if } \tilde{w}_Q\beta = \mathtt{tt} \text{ and } w_{\langle a\rangle}\beta = \mathtt{tt} \\ \mathtt{ff} & \text{otherwise} \end{cases}$$

$$\check{w}_Q\beta = \begin{cases} \mathtt{tt} & \text{if } \{\, a \in A \,:\, \langle a, \mathtt{tt}\rangle \in \beta \text{ or } \langle a, \mathtt{ff}\rangle \in \beta \,\} = Q \\ \mathtt{ff} & \text{otherwise} \end{cases}$$

$$\check{w}_{aQ}\beta = \begin{cases} \mathtt{tt} & \text{if } \check{w}_Q\beta = \mathtt{tt} \text{ and } w_{\langle a\rangle}\beta = \mathtt{tt} \\ \mathtt{ff} & \text{otherwise} \end{cases}$$

The following sets of test constructors will be useful:

Definition 4.3 Define

$$\begin{aligned} \mathsf{Tr} &= \{\, w_{\langle a\rangle} \,:\, a \in A \,\} \\ \mathsf{CTr} &= \mathsf{Tr} \cup \{\tilde{w}_A\} \\ \mathsf{Fl} &= \mathsf{Tr} \cup \{\, \tilde{w}_Q \,:\, Q \subseteq A \,\} \\ \mathsf{FlTr} &= \mathsf{Fl} \cup \{\, \tilde{w}_{aQ} \,:\, a \in A, Q \subseteq A \,\} \\ \mathsf{Rd} &= \mathsf{Tr} \cup \{\, \check{w}_Q \,:\, \subseteq A \,\} \\ \mathsf{RdTr} &= \mathsf{Rd} \cup \{\, \check{w}_{aQ} \,:\, a \in A, Q \subseteq A \,\} \\ \mathsf{BS} &= \mathsf{Tr} \cup \{\, w_{[a]} \,:\, a \in A \,\} \end{aligned}$$

We will also consider three different 2-closures:

Definition 4.4 2-closures $\mathrm{Cl}^\top$, $\mathrm{Cl}^\wedge$ and $\mathrm{Cl}^{\vee\wedge}$ are defined by

$$\begin{aligned} \mathrm{Cl}^\top_X\,\theta &= \theta \cup \{\overrightarrow{X}\} \\ \mathrm{Cl}^\wedge_X\,\theta &= \left\{ \bigwedge_{i=1}^{n} V_i \,:\, n \in \mathbb{N}, V_i \in \theta \right\} \\ \mathrm{Cl}^{\vee\wedge}_X\,\theta &= \left\{ \bigvee_{i=1}^{n} \bigwedge_{j=1}^{m} V_{ij} \,:\, n, m \in \mathbb{N}, V_{ij} \in \theta \right\} \end{aligned}$$

To check that $\mathrm{Cl}^{\top}$, $\mathrm{Cl}^{\wedge}$ and $\mathrm{Cl}^{\vee\wedge}$ indeed are closures (cf. Definition 3.31), take any function $f : X \to Y$ and any test suite $\theta : Y \rightrightarrows 2$, observe that

$$\begin{array}{rcl} \overrightarrow{Y} \circ f & = & \overrightarrow{X} \\ (V_1 \wedge V_2) \circ f & = & (V_1 \circ f) \wedge (V_2 \circ f) \\ (V_1 \vee V_2) \circ f & = & (V_1 \circ f) \vee (V_2 \circ f) \end{array}$$

for any $V_1, V_2 : Y \to 2$, and calculate

$$\begin{array}{rcl} \mathrm{Cl}^{\top}_X f^*\theta & = & \{\, V \circ f \,:\, V \in \theta \,\} \cup \{\, \overrightarrow{Y} \circ f \} = f^* \,\mathrm{Cl}^{\top}_Y \theta \\ \mathrm{Cl}^{\wedge}_X f^*\theta & = & \{\, \bigwedge_{i=1}^{n} (V_i \circ f) \,:\, n \in \mathbb{N}, V_i \in \theta \,\} = \\ & = & \{\, (\bigwedge_{i=1}^{n} V_i) \circ f \,:\, n \in \mathbb{N}, V_i \in \theta \,\} = f^* \,\mathrm{Cl}^{\wedge}_Y \theta \\ \mathrm{Cl}^{\vee\wedge}_X f^*\theta & = & \left\{ \bigvee_{i=1}^{n} \bigwedge_{j=1}^{m} (V_{ij} \circ f) \,:\, n, m \in \mathbb{N}, V_{ij} \in \theta \right\} = \\ & = & \left\{ (\bigvee_{i=1}^{n} \bigwedge_{j=1}^{m} V_{ij}) \circ f \,:\, n, m \in \mathbb{N}, V_{ij} \in \theta \right\} = f^* \,\mathrm{Cl}^{\vee\wedge}_Y \theta \end{array}$$

Note that for any test suite θ on X, one has $\overrightarrow{\emptyset} = \bigvee_{i=1}^{0} V \in \mathrm{Cl}^{\vee\wedge} \theta$. Also $\mathrm{Cl}^{\top} \theta \subseteq \mathrm{Cl}^{\wedge} \theta \subseteq \mathrm{Cl}^{\vee\wedge} \theta$, and $\mathrm{Cl}^{\vee\wedge}$ is simply the closure under finite unions and intersections of tests.

Finally, sets of $(B, 2)$-test constructors from Definition 4.3, together with suitably chosen closures, induce liftings of B to fibred endofunctors on 2-**TS** along the lines of Theorem 3.32:

Definition 4.5 For $\mathsf{W} \in \{\mathsf{Tr}, \mathsf{CTr}, \mathsf{Fl}, \mathsf{FlTr}, \mathsf{Rd}, \mathsf{RdTr}\}$, the endofunctor on 2-**TS** induced by the set of $(B, 2)$-test constructors W and by the 2-closure $\mathrm{Cl}^{\top}$ is denoted B^{W}. The endofunctors on 2-**TS** induced by sets of $(B, 2)$-test constructors Tr and Rd and by the 2-closure $\mathrm{Cl}^{\wedge}$ are denoted B^{S} and B^{RdS}, respectively. The endofunctor on 2-**TS** induced by the set of $(B, 2)$-test constructors BS and by the 2-closure $\mathrm{Cl}^{\vee\wedge}$ is denoted B^{BS}.

Given a B-coalgebra $h : X \to BX$, the operator on $\mathcal{T}_2 X$ corresponding to B^{W} as defined in Section 3.1.5 is denoted Φ^{W}_h. Note that the closures $\mathrm{Cl}^{\top}, \mathrm{Cl}^{\wedge}, \mathrm{Cl}^{\vee\wedge}$ preserve unions of increasing chains of test suites, therefore the least fixed points of the operators Φ^{W}_h can be characterized as remarked in Section 3.3.3.

4.1.1 Relation to Modal Logics

It is not difficult to notice that test constructors shown in Definitions 4.2 and 4.3 are related to syntactic constructors for the BNF grammars shown in Definition 2.2. This is not a coincidence, and indeed modal logics defined in Section 2.2 inspired Definition 4.3. This subsection is devoted to give a formal correspondence between the two definitions.

Definition 4.6 For any $\mathsf{W} \in \{\mathsf{Tr}, \mathsf{CTr}, \mathsf{Fl}, \mathsf{FlTr}, \mathsf{Rd}, \mathsf{RdTr}, \mathsf{S}, \mathsf{RdS}, \mathsf{BS}\}$, and any B-coalgebra $h : X \to BX$, define a function $[\![-]\!]_h : \mathcal{F}_{\mathsf{W}} \to (X \to 2)$ inductively

as follows:

$$\begin{aligned}
[\![\top]\!]_h &= \overrightarrow{X} \\
[\![\bot]\!]_h &= \overrightarrow{\emptyset} \\
[\![\langle a\rangle\,\phi]\!]_h &= w_{\langle a\rangle} \circ B[\![\phi]\!]_h \circ h \\
[\![[a]\phi]\!]_h &= w_{[a]} \circ B[\![\phi]\!]_h \circ h \\
[\![\tilde{Q}]\!]_h &= \tilde{w}_Q \circ B[\![\top]\!]_h \circ h \\
[\![\check{Q}]\!]_h &= \check{w}_Q \circ B[\![\top]\!]_h \circ h \\
[\![\phi_1 \wedge \phi_2]\!]_h &= [\![\phi_1]\!]_h \wedge [\![\phi_2]\!]_h \\
[\![\phi_1 \vee \phi_2]\!]_h &= [\![\phi_1]\!]_h \vee [\![\phi_2]\!]_h
\end{aligned}$$

The correspondence between modal formulae and 2-tests is given in:

Theorem 4.7 Let $\mathsf{W} \in \{\mathsf{Tr}, \mathsf{CTr}, \mathsf{Fl}, \mathsf{FlTr}, \mathsf{Rd}, \mathsf{RdTr}, \mathsf{S}, \mathsf{RdS}, \mathsf{BS}\}$ and let $h : X \to BX$ be a B-coalgebra. For any formula $\phi \in \mathcal{F}_{\mathsf{W}}$ and any $x \in X$,

$$[\![\phi]\!]_h x = \mathtt{tt} \iff x \models_h \phi$$

Proof. By straightforward structural induction on modal formulae. Indeed,

$$\begin{aligned}
[\![\top]\!]_h x = \mathtt{tt} &\iff x \models_h \top \text{ (both sides hold for all } x) \\
[\![\bot]\!]_h x = \mathtt{tt} &\iff x \models_h \bot \text{ (both sides hold for no } x) \\
[\![\langle a\rangle\,\phi]\!]_h x = \mathtt{tt} &\iff w_{\langle a\rangle}(B[\![\phi]\!]_h(hx)) = \mathtt{tt} \iff \langle a, \mathtt{tt}\rangle \in B[\![\phi]\!]_h(hx) \iff \\
&\iff \exists \left\langle a, x'\right\rangle \in hx.\ [\![\phi]\!]_h x' = \mathtt{tt} \iff \\
&\iff \exists \left\langle a, x'\right\rangle \in hx.\ x' \models_h \phi \iff x \models_h \langle a\rangle\,\phi \\
[\![[a]\phi]\!]_h x = \mathtt{ff} &\iff w_{[a]}(B[\![\phi]\!]_h(hx)) = \mathtt{ff} \iff \langle a, \mathtt{ff}\rangle \in B[\![\phi]\!]_h(hx) \iff \\
&\iff \exists \left\langle a, x'\right\rangle \in hx.\ [\![\phi]\!]_h x' = \mathtt{ff} \iff \\
&\iff \exists \left\langle a, x'\right\rangle \in hx.\ x' \not\models_h \phi \iff x \not\models_h [a]\phi \\
[\![\tilde{Q}]\!]_h x = \mathtt{tt} &\iff \tilde{w}_Q(B[\![\top]\!]_h(hx)) = \mathtt{tt} \iff B[\![\top]\!]_h(hx) \cap (Q \times 2) = \emptyset \\
&\iff I(x) \cap Q = \emptyset \iff \\
&\iff x \models_h \tilde{Q} \\
[\![\check{Q}]\!]_h x = \mathtt{tt} &\iff \check{w}_Q(B[\![\top]\!]_h(hx)) = \mathtt{tt} \iff \\
&\iff \{\, a \in A : \langle a, \mathtt{tt}\rangle \in B[\![\top]\!]_h(hx) \text{ or } \langle a, \mathtt{ff}\rangle \in B[\![\top]\!]_h(hx) \,\} = Q \\
&\iff I(x) = Q \iff x \models_h \check{Q} \\
[\![\phi_1 \wedge \phi_2]\!]_h x = \mathtt{tt} &\iff [\![\phi_1]\!]_h x = \mathtt{tt} \text{ and } [\![\phi_2]\!]_h x = \mathtt{tt} \iff \\
&\iff x \models_h \phi_1 \text{ and } x \models_h \phi_1 \iff x \models_h \phi_1 \wedge \phi_2 \\
[\![\phi_1 \vee \phi_2]\!]_h x = \mathtt{tt} &\iff [\![\phi_1]\!]_h x = \mathtt{tt} \text{ or } [\![\phi_2]\!]_h x = \mathtt{tt} \iff \\
&\iff x \models_h \phi_1 \text{ or } x \models_h \phi_1 \iff x \models_h \phi_1 \vee \phi_2
\end{aligned}$$

□

The above correspondence maps modal logics to the least test suites lifting coalgebras to endofunctors B^{W}:

Theorem 4.8 Let $h : X \to BX$ be a B-coalgebra and $k : X \to \Omega$ the coinductive extension of h (see Proposition 2.25). For $\mathsf{W} \in \{\mathsf{Tr}, \mathsf{CTr}, \mathsf{Fl}, \mathsf{FlTr}, \mathsf{Rd}, \mathsf{RdTr}, \mathsf{S}, \mathsf{RdS}, \mathsf{BS}\}$,

$$\{\, [\![\phi]\!]_h \,:\, \phi \in \mathcal{F}_{\mathsf{W}} \,\} = k^* \omega^{\mathsf{W}}$$

where ω^{W} is taken from the final B^{W}-coalgebra $\phi : \langle \Omega, \omega^{\mathsf{W}} \rangle \to \langle B\Omega, B^{\mathsf{W}}_{\Omega} \omega^{\mathsf{W}} \rangle$.

Note that until now we have been a little sloppy with the use of the symbol W. In Definition 2.4, it ranged over a fixed set of notions of process equivalence, and in Definition 4.5 it ranged over sets of test constructors. This theorem, and its Corollary 4.9, mean that this cannot really lead to any confusion.

Proof. We prove the theorem for $\mathsf{W} = \mathsf{Tr}$, the other cases are similar. By Theorem 3.11, $k^* \omega^{\mathsf{Tr}}$ is the least (with respect to set inclusion, see Remark 3.23) (pre)fixed point of the operator

$$\Phi^{\mathsf{Tr}}_X \theta = h^* B^{\mathsf{Tr}}_X \theta = \{w \circ BV \circ h \mid w \in \mathsf{Tr}, V \in \theta\} \cup \overrightarrow{X}$$

First, $\{\, [\![\phi]\!]_h \,:\, \phi \in \mathcal{F}_{\mathsf{Tr}} \,\}$ is a prefixed point of Φ^{Tr}_X, since by Definition 4.6

$$\overrightarrow{X} = [\![\top]\!]_h$$

and for any $\phi \in \mathcal{F}_{\mathsf{Tr}}$, $a \in A$,

$$w_{\langle a \rangle} \circ B[\![\phi]\!]_h \circ h = [\![\langle a \rangle\, \phi]\!]_h$$

To prove that any prefixed point of Φ^{Tr}_X contains $\{\, [\![\phi]\!]_h \,:\, \phi \in \mathcal{F}_{\mathsf{Tr}} \,\}$, proceed by straightforward structural induction on $\mathcal{F}_{\mathsf{Tr}}$. □

Theorem 4.8 allows one to describe operational preorders and equivalences on a given coalgebra $h : X \to BX$ as specialization relations of the least test suites which lift h to coalgebras of various endofunctors on 2-**TS**, as the following easy corollary shows.

Corollary 4.9 Let $h : X \to BX$ be a B-coalgebra and $k : X \to \Omega$ the coinductive extension of h. For $\mathsf{W} \in \{\mathsf{Tr}, \mathsf{CTr}, \mathsf{Fl}, \mathsf{FlTr}, \mathsf{Rd}, \mathsf{RdTr}, \mathsf{S}, \mathsf{RdS}, \mathsf{BS}\}$,

$$\sqsubseteq_{\mathsf{W}} = \leq_{k^* \omega^{\mathsf{W}}} \qquad \text{and} \qquad \cong_{\mathsf{W}} = \equiv_{k^* \omega^{\mathsf{W}}}$$

where ω^{W} is as in Theorem 4.8.

Proof. Compare Definition 2.4 with Example 3.25 and use Theorems 4.8 and 4.7. □

This important corollary gives a coalgebraic characterization of preorders and equivalences from the van Glabbeek spectrum. It is also a basis for the search of congruence formats for those preorders and equivalences, as will be shown in Chapter 5. However, an even stronger correspondence between B^{W}-coalgebras and process relations can be shown, providing a full characterization of B^{W}-coalgebras. In Sections 4.2–4.3 we present this correspondence, followed by an application of it to deriving coinductive proof principles in Section 4.4.

4.2 Simulation and Bisimulation Semantics

We begin by characterizing the operational preorders and equivalences for B^{S}, B^{RdS} and B^{BS}-coalgebras. The remaining functors B^{W} from Definition 4.5 are treated in the next section.

The following two theorems show that the specialization preorders of B^{S}-coalgebras are exactly reflexive and transitive simulations (see Definition 2.5).

Theorem 4.10 For any B^{S}-coalgebra $h : \langle X, \theta \rangle \to \langle BX, B^{\mathsf{S}}_X \theta \rangle$, the specialization preorder $\leq_\theta$ is a reflexive, transitive simulation on $h : X \to BX$.

Proof. To show that $\leq_\theta$ is a simulation on h, assume any $a \in A$, $x, x', y \in X$ such that $x \xrightarrow{a} x'$ and $x' \not\leq_\theta y'$ for all $y' \in X$ such that $y \xrightarrow{a} y'$. This means that for every such y' there exists a test $V_{y'} \in \theta$ such that $V_{y'}x' = \mathtt{tt}$ but $V_{y'}y' = \mathtt{ff}$.

Consider a test

$$V = w_{\langle a \rangle} \circ B(\bigwedge_{y'} V_{y'}) \circ h$$

where the intersection occurs over all $y' \in X$ such that $y \xrightarrow{a} y'$. Since θ lifts h to a B^{S}-coalgebra, and h is finitely branching, $V \in \theta$. However, it is easy to check that $Vx = \mathtt{tt}$ and $Vy = \mathtt{ff}$, hence $x \not\leq_\theta y$ and $\leq_\theta$ is a simulation.

Reflexivity and transitivity of $\leq_\theta$ is immediate by definition. □

Theorem 4.11 For any coalgebra $h : X \to BX$ in **Set**, and any reflexive, transitive simulation R on h, there exists a test suite $\theta_R : X \rightrightarrows 2$ such that $\leq_{\theta_R} = R$ and $h : \langle X, \theta_R \rangle \to \langle BX, B^{\mathsf{S}}_X \theta_R \rangle$ is a valid B^{S}-coalgebra.

Proof. Assume any reflexive, transitive simulation R on $h : X \to BX$ and consider the 2-test suite on X:

$$\theta_R = \left\{ V : X \to 2 \;:\; \overline{V} \text{ is } R\text{-upper} \right\}$$

where $\overline{V}$ is R-upper means that for any $x, y \in X$, if xRy and $x \in \overline{V}$ then $y \in \overline{V}$.

To show that $R \subseteq \leq_{\theta_R}$, assume any $x, y \in X$ such that xRy and take any $V \in \theta_R$ such that $Vx = \mathtt{tt}$. Since $\overline{V}$ is R-upper, also $Vy = \mathtt{tt}$. Since V was chosen arbitrarily, $x \leq_{\theta_R} y$.

To show that $R \supseteq \leq_{\theta_R}$, assume any $x, y \in X$ such that $x \not R y$ and consider the test $V = \overrightarrow{\{ z \in X \;:\; xRz \}}$. It is easy to check (using transitivity of R) that $\overline{V}$ is R-upper, hence $V \in \theta_R$. However, $Vy = \mathtt{ff}$ and (by reflexivity of R) $Vx = \mathtt{tt}$, hence $x \not\leq_{\theta_R} y$.

To prove that θ_R lifts h to a B^{S}-coalgebra, one has to show that

- $\overrightarrow{X} \in \theta_R$,
- for any $V \in \theta_R$, also $w_{\langle a \rangle} \circ BV \circ h \in \theta_R$,
- for any $V_1, \ldots, V_n \in \theta_R$, also $\bigwedge_{i=1}^n V_i \in \theta_R$.

The first condition holds obviously, since X is R-upper for any relation R.

For the second condition, assume any $V \in \theta_R$, $x, y \in X$ such that xRy and $(w_{\langle a \rangle} \circ BV \circ h)x = \mathtt{tt}$. The latter assumption means that there exists an $x' \in X$ such that $x \xrightarrow{a} x'$ and $Vx' = \mathtt{tt}$. Since R is a simulation, there exists a $y' \in X$ such that $y \xrightarrow{a} y'$ and $x'Ry'$, hence $Vy' = \mathtt{tt}$. This means that $(w_{\langle a \rangle} \circ BV \circ h)y = \mathtt{tt}$, and $\overline{w_{\langle a \rangle} \circ BV \circ h}$ is R-upper.

For the third condition, it is easily checked that for any relation R, intersection of any family of R-upper sets is again R-upper. □

Corollary 4.12 Specialization preorders $\leq_\theta$ for B^{S}-coalgebras $h : \langle X, \theta \rangle \to \langle BX, B^{\mathsf{S}}_X \theta \rangle$ are exactly reflexive, transitive simulations on $h : X \to BX$. The simulation preorder $\sqsubseteq_{\mathsf{S}}$ is the largest such relation.

Proof. Immediate from Theorems 4.10 and 4.11, and from Proposition 2.8. □

Corollary 4.13 Specialization equivalences $\equiv_\theta$ for B^{S}-coalgebras $h : \langle X, \theta \rangle \to \langle BX, B^{\mathsf{S}}_X \theta \rangle$ are exactly the equivalence relations $R \cap R^{-1}$ associated to reflexive, transitive simulations R on $h : X \to BX$. The simulation equivalence $\cong_{\mathsf{S}}$ is the largest such relation.

Proof. See Example 3.25 and Definition 2.4 to see that $\equiv_\theta = \leq_\theta \cap (\leq_\theta)^{-1}$ and $\cong_{\mathsf{S}} = \sqsubseteq_{\mathsf{S}} \cap (\sqsubseteq_{\mathsf{S}})^{-1}$. □

We proceed to give a similar characterization of B^{RdS}-coalgebras.

Theorem 4.14 For any B^{RdS}-coalgebra $h : \langle X, \theta \rangle \to \langle BX, B^{\mathsf{RdS}}_X \theta \rangle$, the specialization relation $\leq_\theta$ is a reflexive, transitive ready simulation on $h : X \to BX$.

Proof. By Theorem 4.10, $\leq_\theta$ is a reflexive, transitive simulation, since if θ lifts h to a B^{RdS}-coalgebra then it also lifts h to a B^{S}-coalgebra, by Corollary 3.35. Thus it is enough to assume any $x, y \in X$ and prove that $x \leq_\theta y$ implies $I(x) = I(y)$.

To this end, assume $I(x) \neq I(y)$ and consider a test

$$V = \check{w}_{I(x)} \circ B(\overrightarrow{X}) \circ h$$

Since θ lifts h to a B^{RdS}-coalgebra, $V \in \theta$. However, it is easy to check that $Vx = \mathtt{tt}$ and $Vy = \mathtt{ff}$, hence $x \not\leq_\theta y$ and $\leq_\theta$ is a ready simulation. □

Theorem 4.15 For any coalgebra $h : X \to BX$ in **Set**, and any reflexive, transitive ready simulation R on h, there exists a test suite $\theta_R : X \rightrightarrows 2$ such that $\leq_{\theta_R} = R$ and $h : \langle X, \theta_R \rangle \to \langle BX, B^{\mathsf{RdS}}_X \theta_R \rangle$ is a valid B^{RdS}-coalgebra.

Proof. Assume any reflexive, transitive ready simulation R on $h : X \to BX$ and consider the 2-test suite on X:

$$\theta_R = \left\{ V : X \to 2 \;:\; \overline{V} \text{ is } R\text{-upper} \right\}$$

For a proof that $\leq_{\theta_R} = R$, see the proof of Theorem 4.11.

To prove that θ_R lifts h to a B^{RdS}-coalgebra, one has to show that

- θ_R lifts h to a B^{S}-coalgebra,
- for any $Q \subseteq A$, $V \in \theta_R$, there is $\check{w}_Q \circ BV \circ h \in \theta_R$.

The first condition holds by Theorem 4.11, since R, being a ready simulation, is also a simulation.

For the second condition, assume any $Q \subseteq A$, $V \in \theta_R$, $x, y \in X$ such that xRy and $(\check{w}_Q \circ BV \circ h)x = \mathtt{tt}$. The latter assumption means $I(x) = Q$. Since R is a ready simulation, also $I(y) = Q$, hence $(\check{w}_Q \circ BV \circ h)y = \mathtt{tt}$ and $\overline{\check{w}_Q \circ BV \circ h}$ is R-upper. □

Corollary 4.16 Specialization preorders $\leq_\theta$ for B^{RdS}-coalgebras $h : \langle X, \theta\rangle \to \langle BX, B_X^{\mathsf{RdS}}\theta\rangle$ are exactly reflexive, transitive ready simulations on $h : X \to BX$. The ready simulation preorder $\sqsubseteq_{\mathsf{RdS}}$ is the largest such relation.

Proof. Immediate from Theorems 4.14 and 4.15, and from Proposition 2.8. □

Corollary 4.17 Specialization equivalences $\equiv_\theta$ for B^{RdS}-coalgebras $h : \langle X, \theta\rangle \to \langle BX, B_X^{\mathsf{RdS}}\theta\rangle$ are exactly the equivalence relations $R \cap R^{-1}$ associated to reflexive, transitive ready simulations R on $h : X \to BX$. The ready simulation equivalence $\cong_{\mathsf{RdS}}$ is the largest such relation.

Proof. See Example 3.25 and Definition 2.4 to see that $\equiv_\theta = \leq_\theta \cap (\leq_\theta)^{-1}$ and $\cong_{\mathsf{RdS}} = \leq_{\mathsf{RdS}} \cap (\leq_{\mathsf{RdS}})^{-1}$. □

Finally, we give a characterization of B^{BS}-coalgebras.

Theorem 4.18 For any B^{BS}-coalgebra $h : \langle X, \theta\rangle \to \langle BX, B_X^{\mathsf{BS}}\theta\rangle$, the specialization relation $\leq_\theta$ is a reflexive, transitive bisimulation on $h : X \to BX$.

Proof. Reflexivity, transitivity and the first condition in the definition of bisimulation (Definition 2.6) hold for $\leq_\theta$ by Theorem 4.10, since if θ lifts h to a B^{BS}-coalgebra then it also lifts h to a B^{S}-coalgebra, by Corollary 3.35.

To show the second condition, assume any $a \in A$, $x, y, y' \in X$ such that $y \xrightarrow{a} y'$ and $x' \not\leq_\theta y'$ for all $x' \in X$ such that $x \xrightarrow{a} x'$. This means that for every such x' there exists a test $V_{x'} \in \theta$ such that $V_{x'}x' = \mathtt{tt}$ but $V_{x'}y' = \mathtt{ff}$.

Consider a test

$$V = w_{[a]} \circ B(\bigvee_{x'} V_{x'}) \circ h$$

where the union occurs over all $x' \in X$ such that $x \xrightarrow{a} x'$. Since θ lifts h to a B^{BS}-coalgebra, and h is finitely branching, $V \in \theta$. However, it is easy to check that $Vx = \mathtt{tt}$ and $Vy = \mathtt{ff}$, hence $x \not\leq_\theta y$ and $\leq_\theta$ is a bisimulation. □

Theorem 4.19 For any coalgebra $h : X \to BX$ in **Set**, and any reflexive, transitive bisimulation R on h, there exists a test suite $\theta_R : X \rightrightarrows 2$ such that $\leq_{\theta_R} = R$ and $h : \langle X, \theta_R\rangle \to \langle BX, B_X^{\mathsf{BS}}\theta_R\rangle$ is a valid B^{BS}-coalgebra.

Proof. Assume any reflexive, transitive bisimulation R on $h : X \to BX$ and consider the 2-test suite on X:

$$\theta_R = \left\{ V : X \to 2 \;:\; \overline{V} \text{ is } R\text{-upper} \right\}$$

For a proof that $\leq_{\theta_R} = R$, see the proof of Theorem 4.11.

To prove that θ_R lifts h to a B^{BS}-coalgebra, one has to show that

- θ_R lifts h to a B^{S}-coalgebra,
- $\overrightarrow{\emptyset} \in \theta_R$,
- for any $V \in \theta_R$, also $w_{[a]} \circ BV \circ h \in \theta_R$,
- for any $V_1, \ldots, V_n \in \theta_R$, also $\bigvee_{i=1}^{n} V_i \in \theta_R$.

The first condition holds by Theorem 4.11, since R, being a bisimulation, is also a simulation.

The second condition holds obviously, since $\emptyset$ is R-upper for any relation R.

For the third condition, assume any $V \in \theta_R$, $x, y \in X$ such that xRy and $(w_{[a]} \circ BV \circ h)y = \mathtt{ff}$. The latter assumption means that there exists $y' \in X$ such that $y \xrightarrow{a} y'$ and $Vy' = \mathtt{ff}$. Since R is a bisimulation, there exists $x' \in X$ such that $x \xrightarrow{a} x'$ and $x'Ry'$, hence $Vx' = \mathtt{ff}$. This means that $(w_{[a]} \circ BV \circ h)x = \mathtt{ff}$ and $\overline{w_{[a]} \circ BV \circ h}$ is R-upper.

For the fourth condition, it is easily checked that for any relation R, union of any family of R-upper sets is again R-upper. □

Corollary 4.20 Specialization preorders $\leq_\theta$ for B^{BS}-coalgebras $h : \langle X, \theta \rangle \to \langle BX, B^{\mathsf{BS}}_X \theta \rangle$ are exactly reflexive, transitive bisimulations on $h : X \to BX$. The bisimulation preorder $\sqsubseteq_{\mathsf{BS}}$ is the largest such relation.

Proof. Immediate from Theorems 4.18 and 4.19, and by Proposition 2.8. □

Corollary 4.21 Specialization equivalences $\equiv_\theta$ for B^{BS}-coalgebras $h : \langle X, \theta \rangle \to \langle BX, B^{\mathsf{BS}}_X \theta \rangle$ are exactly the equivalence relations $R \cap R^{-1}$ associated to reflexive, transitive bisimulations R on $h : X \to BX$. The bisimulation equivalence $\cong_{\mathsf{BS}}$ is the largest such relation.

Proof. See Example 3.25 and Definition 2.4 to see that $\equiv_\theta = \leq_\theta \cap (\leq_\theta)^{-1}$ and $\cong_{\mathsf{BS}} = \leq_{\mathsf{BS}} \cap (\leq_{\mathsf{BS}})^{-1}$. □

Recall from Proposition 2.8 that bisimulation preorder (hence, by Proposition 2.9, also bisimulation equivalence) is characterized as the largest bisimulation. This gives rise to the so-called coinduction proof principle used to show that certain operations behave well with respect to bisimulation equivalence. Indeed, to show that two processes in an LTS are bisimulation equivalent, it is enough to show any bisimulation that relates them. Many examples of proofs using this principle are shown, e.g., in [77, Section 12]. Similar characterizations of simulation and ready simulation preorders are given in Proposition 2.8,

but the remaining preorders from the van Glabbeek spectrum have not been characterized in this fashion.

As is shown in Corollary 4.9, bisimulation preorder on a finitely branching LTS $h : X \to BX$ is the specialization preorder of the least test suite on X that lifts h to a B^{W}-coalgebra. More generally however, specialization preorders on B^{W}-coalgebras are bisimulations on the underlying B-coalgebras. Therefore, the characterization of bisimulation equivalence by bisimulations corresponds to the characterization of the corresponding B^{W}-coalgebra as the least B^{W}-coalgebra lifting the underlying B-coalgebra.

This, together the characterization of various preorders given in Corollary 4.9, hints that a characterization of specialization preorders of B^{W}-coalgebras for various W might lead to proof principles for other equivalences in the van Glabbeek spectrum. Such a characterization is presented in the next section, and an example of the resulting proof principle is shown in Section 4.4.

4.3 Decorated Trace Semantics

In this section, we characterize operational preorders and equivalences of B^{W}-coalgebras for $\mathsf{W} \in \{\mathsf{Tr}, \mathsf{CTr}, \mathsf{Fl}, \mathsf{FlTr}, \mathsf{Rd}, \mathsf{RdTr}\}$. This fragment of the van Glabbeek spectrum is called collectively *decorated trace semantics*.

To give a full characterization of (the specialization relations $\leq_\theta$ and $\equiv_\theta$ of) B^{W}-coalgebras, we first need a few technical definitions and results.

Definition 4.22 A relation $S \subseteq X \times \mathcal{P}X$ is called a *quasi-preorder* on X if

- for any $x \in X$, $xS\{x\}$ and $x \not S \emptyset$,
- for any $x, y \in X$, $\xi, \chi \subseteq X$, if $xS\xi$, $y \in \xi$ and $yS\chi$ then $xS((\xi \setminus \{y\}) \cup \chi)$.

Definition 4.23 Let S be a quasi-preorder on X. A set $V \subseteq X$ is called *quasi-S-upper*, if for any $x \in V$, $\xi \subseteq X$ such that $xS\xi$, the intersection $V \cap \xi$ is not empty.

Lemma 4.24 Let S be a quasi-preorder on X, and fix arbitrary elements $x, y \in X$. If for every quasi-S-upper set $Y \subseteq X$, $x \in Y$ implies $y \in Y$, then $xS\{y\}$.

Proof. Assume $y \in Y$ for every quasi-S-upper Y such that $x \in Y$. Define $y\downarrow = \{\, z \in X \,:\, zS\{y\} \,\}$, and consider

$$Y_0 = X \setminus y\downarrow$$

To show that Y_0 is quasi-S-upper, take any $z \in Y_0$, $\xi \subseteq X$ such that $zS\xi$.

Assume that $\xi \subseteq y\downarrow$. Since S is a quasi-preorder, this implies that $z \in y\downarrow$, which contradicts the assumption that $z \in Y_0$. Therefore $\xi \not\subseteq y\downarrow$ and one may consider an element $v \in \xi$ such that $v \notin y\downarrow$, hence Y_0 is quasi-S-upper.

Since S is a quasi-preorder, $yS\{y\}$, hence $y \notin Y_0$ and (by the assumption, since Y_0 is quasi-S-upper) $x \notin Y_0$. As a consequence, $xS\{y\}$. □

Now we proceed to define various kinds of *trace-aware relations*, that are well-structured with respect to decorated trace semantics. They are based on auxiliary relations called *one-by-many simulations*.

Definition 4.25 Consider a finitely branching LTS $h : X \to BX$. A relation $S \subseteq X \times \mathcal{P}X$ is called a *one-by-many simulation* on h if for any $a \in A$, if $xS\xi$ and $x \xrightarrow{a} x'$, then $x'S\{y' \in X : \exists y \in \xi.\ y \xrightarrow{a} y'\}$. A relation $R \subseteq X \times X$ is called *trace-aware* on h if there exists a one-by-many simulation S on h such that $xRy \iff xS\{y\}$. If, moreover, S is a quasi-preorder, then R is called a *trace-aware preorder*.

Note that any trace-aware preorder is indeed a preorder. Reflexivity and transitivity immediately follow from Definition 4.22.

Definition 4.26 Consider a finitely branching LTS $h : X \to BX$. A relation $S \subseteq X \times \mathcal{P}X$ is called a *one-by-many completed simulation* on h if

- S is a one-by-many simulation on h, and
- if $xS\xi$ and $x \not\longrightarrow$, then for some $y \in \xi$, $y \not\longrightarrow$.

A relation $R \subseteq X \times X$ is called *completed trace-aware* on h if there exists a one-by-many completed simulation Q on h such that $xRy \iff xS\{y\}$. If, moreover, S is a quasi-preorder, then R is called a *completed trace-aware preorder*.

Definition 4.27 Consider a finitely branching LTS $h : X \to BX$. A relation $S \subseteq X \times \mathcal{P}X$ is called a *one-by-many failure simulation* on h if

- S is a one-by-many simulation on h, and
- for any $Q \subseteq A$, if $xS\xi$ and $x \not\xrightarrow{Q}$, then for some $y \in \xi$, $y \not\xrightarrow{Q}$.

A relation $R \subseteq X \times X$ is called *failures-aware* on h if there exists a one-by-many failure simulation S on h such that $xRy \iff xS\{y\}$. If, moreover, S is a quasi-preorder, then R is called a *failures-aware preorder*.

Definition 4.28 Consider a finitely branching LTS $h : X \to BX$. A relation $S \subseteq X \times \mathcal{P}X$ is called a *one-by-many failure trace simulation* on h if

- S is a one-by-many failure simulation on h, and
- for any $a \in A$, $Q \subseteq A$, if $xS\xi$, $x \xrightarrow{a} x'$ and $x \not\xrightarrow{Q}$, then $x'S\left\{ y' \in X : \exists y \in \xi.\ y \xrightarrow{a} y', y \not\xrightarrow{Q} \right\}$.

A relation $R \subseteq X \times X$ is called *failure trace-aware* on h if there exists a one-by-many failure trace simulation S on h such that $xRy \iff xS\{y\}$. If, moreover, S is a quasi-preorder, then R is called a *failure trace-aware preorder*.

Definition 4.29 Consider an finitely branching LTS $h : X \to BX$. A relation $S \subseteq X \times \mathcal{P}X$ is called a *one-by-many ready simulation* on h if

- S is a one-by-many simulation on h, and
- if $xS\xi$ then for some $y \in \xi$, $I(y) = I(x)$.

A relation $R \subseteq X \times X$ is called *readiness-aware* on h if there exists a one-by-many ready simulation S on h such that $xRy \iff xS\{y\}$. If, moreover, S is a quasi-preorder, then R is called a *readiness-aware preorder*.

Definition 4.30 Consider a finitely branching LTS $h : X \to BX$. A relation $S \subseteq X \times \mathcal{P}X$ is called a *one-by-many ready trace simulation* on h if

- S is a one-by-many readiness simulation on h, and
- for any $a \in A$, if $xS\xi$ and $x \xrightarrow{a} x'$,
  then $x'S\left\{ y' \in X \,:\, \exists y \in \xi.\ y \xrightarrow{a} y', I(y) = I(x) \right\}$.

A relation $R \subseteq X \times X$ is called *ready trace-aware* on h if there exists a one-by-many ready trace simulation S on h such that $xRy \iff xS\{y\}$. If, moreover, S is a quasi-preorder, then R is called a *ready trace-aware preorder*.

The following series of theorems and corollaries characterize the specialization preorders of B^{W}-coalgebras (where $\mathsf{W} \in \{\mathsf{Tr}, \mathsf{CTr}, \mathsf{Fl}, \mathsf{FlTr}, \mathsf{Rd}, \mathsf{RdTr}\}$) as corresponding trace-aware preorders.

Theorem 4.31 For any B^{Tr}-coalgebra $h : \langle X, \theta\rangle \to \left\langle BX, B^{\mathsf{Tr}}_X\theta\right\rangle$, the specialization relation $\leq_\theta$ is a trace-aware preorder on $h : X \to BX$.

Proof. First, recall that saying that $h : \langle X, \theta\rangle \to \left\langle BX, B^{\mathsf{Tr}}_X\theta\right\rangle$ is a B^{Tr}-coalgebra is equivalent to saying that $\theta \supseteq h^* B^{\mathsf{Tr}}_X\theta$. Equivalently, $\overrightarrow{X} \in \theta$ and for any $V \in \theta$ and $a \in A$, also $\langle a\rangle \circ BV \circ h \in \theta$.

Define $S \subseteq X \times \mathcal{P}X$ as follows:

$$xS\xi \text{ iff } \forall V \in \theta.\ Vx = \mathtt{tt} \Rightarrow \exists y \in \xi.\ Vy = \mathtt{tt}$$

Obviously $x \leq_\theta y$ if and only if $xS\{y\}$, hence it is enough to show that S is a quasi-preorder and a one-by-many simulation.

To check that S is a quasi-preorder, consider the conditions in Definition 4.22 in turn.

- Assume $xS\xi$. Since $\overrightarrow{X} \in \theta$ and $\overrightarrow{X}x = \mathtt{tt}$, by definition of S, $\xi \neq \emptyset$. The fact that $xS\{x\}$ is immediate by definition of S.
- Assume $xS\xi$, $y \in \xi$ and $yS\chi$. Consider any test $V : X \to 2$ such that $Vx = \mathtt{tt}$. Then, by definition of S, there exists an element $z \in \xi$ such that $Vz = \mathtt{tt}$.

  If $z \neq y$ then $z \in ((\xi \setminus \{y\}) \cup \chi)$. On the other hand, if $z = y$, then (since $yS\chi$) there exists an element $u \in \chi$ such that $Vu = \mathtt{tt}$, and obviously $u \in ((\xi \setminus \{y\}) \cup \chi)$.

  As a result, since V was chosen arbitrarily, $xS((\xi \setminus \{y\}) \cup \chi)$.

To show that S is a one-by-many simulation, consider any $x \in X$, $\xi \subseteq X$. Assume $x \xrightarrow{a} x'$ and $x' \not S \{y' \in X \mid \exists y \in \xi.\ y \xrightarrow{a} y'\}$. By definition of S, there exists a test $V \in \theta$ such that

- $Vx' = \mathtt{tt}$, and
- $Vy' = \mathtt{ff}$ for all $y \in \xi$, $y \xrightarrow{a} y'$.

Consider a test $V' = w_{\langle a \rangle} \circ BV \circ h$. Obviously $V'x = \mathtt{tt}$, but for every $y \in \xi$, $V'y = \mathtt{ff}$. Since $V \in \theta$, also $V' \in \theta$, hence $x \not\mathrel{S} \xi$. □

Theorem 4.32 For any coalgebra $h : X \to BX$ in **Set**, and any trace-aware preorder R on h, there exists a test suite $\theta_R : X \rightrightarrows 2$ such that $\leq_{\theta_R} = R$ and $h : \langle X, \theta_R \rangle \to \left\langle BX, B_X^{\mathsf{Tr}} \theta_R \right\rangle$ is a valid B^{Tr}-coalgebra.

Proof. Assume a quasi-preorder and one-by-many simulation S on h such that xRy if and only if $xS\{y\}$. Define

$$\theta_R = \left\{ V : X \to 2 \;:\; \overline{V} \text{ is quasi-}S\text{-upper} \right\}$$

To check that $R \subseteq \leq_{\theta_R}$, assume xRy, or equivalently, $xS\{y\}$. Consider any $V \in \theta_R$ such that $Vx = \mathtt{tt}$. Since $\overline{V}$ is quasi-S-upper, also $Vy = \mathtt{tt}$.

To check that $\leq_{\theta_R} \subseteq R$, assume that for every $V : X \to 2$ such that $\overline{V}$ is quasi-S-upper, if $Vx = \mathtt{tt}$ then $Vy = \mathtt{tt}$. By Lemma 4.24, $xS\{y\}$, hence xRy.

To check that $h : \langle X, \theta \rangle \to \left\langle BX, B_X^{\mathsf{Tr}} \theta_R \right\rangle$ is a valid B^{Tr}-coalgebra, it is enough to check that

- $\overrightarrow{X} \in \theta_R$, and
- for any $V \in \theta_R$, also $w_{\langle a \rangle} \circ BV \circ h \in \theta_R$.

The first condition is easy, since X is quasi-S-upper for any quasi-preorder S on X.

For the second condition, assume any $V \in \theta_R$ and denote $V' = w_{\langle a \rangle} \circ BV \circ h$. Take any $x \in X$, $\xi \subseteq X$ such that $xS\xi$ and $V'x = \mathtt{tt}$. The latter assumption means that there exists an $x' \in X$ such that $x \xrightarrow{a} x'$ and $Vx' = \mathtt{tt}$.

Since S is a one-by-many simulation, this means that

$$x' S \left\{ y' \in X \;:\; \exists y \in \xi.\; y \xrightarrow{a} y' \right\}$$

and there exist $y, y' \in X$ such that $y \in \xi$, $y \xrightarrow{a} y'$ and $Vy' = \mathtt{tt}$. Then $V'y = \mathtt{tt}$. Since $xS\xi$ were chosen arbitrarily, $\overline{V'}$ is quasi-S-upper. □

Corollary 4.33 Specialization preorders $\leq_\theta$ for B^{Tr}-coalgebras $h : \langle X, \theta \rangle \to \left\langle BX, B_X^{\mathsf{Tr}} \theta \right\rangle$ are exactly trace-aware preorders on $h : X \to BX$.

The trace preorder $\sqsubseteq_{\mathsf{Tr}}$ on h is the largest trace-aware relation on h.

Proof. The first statement follows immediately from Theorems 4.27 and 4.28. For the second statement, use Corollary 4.9, Theorem 3.12 and Proposition 3.27 to notice that $\sqsubseteq_{\mathsf{Tr}}$ is the largest trace-aware *preorder* on h. Then observe that the reflexive and transitive closure of any trace-aware relation is again trace-aware. □

Corollary 4.34 Specialization equivalences $\equiv_\theta$ for B^{Tr}-coalgebras $h : \langle X, \theta\rangle \to \langle BX, B^{\mathsf{Tr}}_X\theta\rangle$ are exactly the equivalence relations $R \cap R^{-1}$ associated to trace-aware preorders R on $h : X \to BX$.

The trace equivalence $\cong_{\mathsf{Tr}}$ on h is the largest such relation on h.

Proof. See Example 3.25 and Definition 2.4 to see that $\equiv_\theta = \leq_\theta \cap (\leq_\theta)^{-1}$ and $\cong_{\mathsf{Tr}} = \leq_{\mathsf{Tr}} \cap (\leq_{\mathsf{Tr}})^{-1}$. □

We proceed to consider the case of B^{CTr}-coalgebras, extending Theorems and Corollaries 4.31-4.34.

Theorem 4.35 For any B^{CTr}-coalgebra $h : \langle X, \theta\rangle \to \langle BX, B^{\mathsf{CTr}}_X\theta\rangle$, the specialization relation $\leq_\theta$ is a completed trace-aware preorder on $h : X \to BX$.

Proof. First, observe that $h : \langle X, \theta\rangle \to \langle BX, B^{\mathsf{CTr}}_X\theta\rangle$ is a valid B^{CTr}-coalgebra if and only if $h : \langle X, \theta\rangle \to \langle BX, B^{\mathsf{Tr}}_X\theta\rangle$ is a valid B^{Tr}-coalgebra and for any $V \in \theta$, also $\tilde{w}_A \circ BV \circ h \in \theta$.

To check that $\leq_\theta$ is a completed trace-aware preorder, consider $S \subseteq X \times \mathcal{P}X$ defined as in the proof of Theorem 4.31:

$$xS\xi \text{ iff } \forall V \in \theta.\ Vx = \mathtt{tt} \Rightarrow \exists y \in \xi.\ Vy = \mathtt{tt}$$

It is enough to check that S is a one-by-many completed simulation and a quasi-preorder. Since θ lifts h to a B^{Tr}-algebra, by Theorem 4.31 S is a one-by-many simulation and a quasi-preorder, hence it is enough to assume arbitrary $x \in X$, $\xi \subseteq X$ such that $xS\xi$ and $hx = \emptyset$ and prove that $hy = \emptyset$ for some $y \in \xi$.

To this end, take $V = \tilde{w}_A \circ B\overrightarrow{X} \circ h \in \theta$. Since $hx = \emptyset$, one has $Vx = \mathtt{tt}$, hence (by definition of S) for some $y \in \xi$, $Vy = \mathtt{tt}$ and $hy = \emptyset$. □

Theorem 4.36 For any coalgebra $h : X \to BX$ in **Set**, and any completed trace-aware preorder R on h, there exists a test suite $\theta_R : X \rightrightarrows 2$ such that $\leq_{\theta_R} = R$ and $h : \langle X, \theta_R\rangle \to \langle BX, B^{\mathsf{CTr}}_X\theta_R\rangle$ is a valid B^{CTr}-coalgebra.

Proof. Assume a quasi-preorder and one-by-many completed simulation S on h such that xRy if and only if $xS\{y\}$. Define, as in the proof of Theorem 4.32,

$$\theta_R = \left\{\, V : X \to 2 \;:\; \overline{V} \text{ is quasi-}S\text{-upper} \,\right\}$$

Since R is completed trace-aware, it is also trace-aware and by Theorem 4.32, $\leq_{\theta_R} = R$ and $h : \langle X, \theta_R\rangle \to \langle BX, B^{\mathsf{Tr}}_X\theta_R\rangle$ is a valid B^{Tr}-coalgebra. It is therefore enough to assume arbitrary $V \in \theta_R$ and prove that $\tilde{w}_A \circ BV \circ h \in \theta_R$.

To this end, denote $V' = \tilde{w}_A \circ BV \circ h$ and take any $x \in X$, $\xi \subseteq X$ such that $xS\xi$ and $V'x = \mathtt{tt}$. The latter assumption means that $x \not\longrightarrow$.

Since S is a one-by-many completed simulation, this means that $y \not\longrightarrow$ for some $y \in \xi$. Then $V'y = \mathtt{tt}$. Since $xS\xi$ were chosen arbitrarily, $\overline{V'}$ is quasi-S-upper. □

Corollary 4.37 Specialization preorders $\leq_\theta$ for B^{CTr}-coalgebras $h : \langle X, \theta\rangle \to \langle BX, B_X^{\mathsf{CTr}}\theta\rangle$ are exactly completed trace-aware preorders on $h : X \to BX$.

The completed trace preorder $\sqsubseteq_{\mathsf{CTr}}$ on h is the largest completed trace-aware relation on h.

Specialization equivalences $\equiv_\theta$ for B^{CTr}-coalgebras $h : \langle X, \theta\rangle \to \langle BX, B_X^{\mathsf{CTr}}\theta\rangle$ are exactly the equivalence relations $R \cap R^{-1}$ associated to completed trace-aware preorders R on $h : X \to BX$.

The completed trace equivalence $\cong_{\mathsf{Tr}}$ on h is the largest such relation on h.

Proof. Analogous to Corollaries 4.33-4.34, using Theorems 4.35-4.36. □

We proceed to consider the case of B^{Fl}-coalgebras.

Theorem 4.38 For any B^{Fl}-coalgebra $h : \langle X, \theta\rangle \to \langle BX, B_X^{\mathsf{Fl}}\theta\rangle$, the specialization relation $\leq_\theta$ is a failures-aware preorder on $h : X \to BX$.

Proof. First, observe that $h : \langle X, \theta\rangle \to \langle BX, B_X^{\mathsf{Fl}}\theta\rangle$ is a valid B^{Fl}-coalgebra if and only if $h : \langle X, \theta\rangle \to \langle BX, B_X^{\mathsf{Tr}}\theta\rangle$ is a valid B^{Tr}-coalgebra and for any $Q \subseteq A$, $V \in \theta$, also $\tilde{w}_Q \circ BV \circ h \in \theta$.

To check that $\leq_\theta$ is a failures-aware preorder, consider $S \subseteq X \times \mathcal{P}X$ defined as in the proof of Theorem 4.31:

$$xS\xi \text{ iff } \forall V \in \theta.\ Vx = \mathtt{tt} \Rightarrow \exists y \in \xi.\ Vy = \mathtt{tt}$$

It is enough to check that S is a quasi-preorder and a one-by-many failure simulation. Since θ lifts h to a B^{Tr}-algebra, by Theorem 4.31 S is a quasi-preorder and a one-by-many simulation, hence it is enough to assume arbitrary $Q \subseteq A$, $x \in X$, $\xi \subseteq X$ such that $xS\xi$ and $x \overset{Q}{\not\longrightarrow}$ and prove that $y \overset{Q}{\not\longrightarrow}$ for some $y \in \xi$.

To this end, take $V = \tilde{w}_Q \circ B\overrightarrow{X} \circ h \in \theta$. Since $x \overset{Q}{\not\longrightarrow}$, one has $Vx = \mathtt{tt}$, hence (by definition of S) for some $y \in \xi$, $Vy = \mathtt{tt}$ and $x \overset{Q}{\not\longrightarrow}$. □

Theorem 4.39 For any coalgebra $h : X \to BX$ in **Set**, and any failures-aware preorder R on h, there exists a test suite $\theta_R : X \rightrightarrows 2$ such that $\leq_{\theta_R} = R$ and $h : \langle X, \theta_R\rangle \to \langle BX, B_X^{\mathsf{Fl}}\theta_R\rangle$ is a valid B^{Fl}-coalgebra.

Proof. Assume a quasi-preorder and one-by-many failure simulation S on h such that xRy if and only if $xS\{y\}$. Define, as in the proof of Theorem 4.32,

$$\theta_R = \left\{\, V : X \to 2 \;:\; \overline{V} \text{ is quasi-}S\text{-upper} \,\right\}$$

Since R is failures-aware, it is also trace-aware and by Theorem 4.32, $\leq_{\theta_R} = R$ and $h : \langle X, \theta_R\rangle \to \langle BX, B_X^{\mathsf{Tr}}\theta_R\rangle$ is a valid B^{Tr}-coalgebra. It is therefore enough to assume arbitrary $Q \subseteq A$, $V \in \theta_R$ and prove that $\tilde{w}_Q \circ BV \circ h \in \theta_R$.

To this end, denote $V' = \tilde{w}_Q \circ BV \circ h$ and take any $x \in X$, $\xi \subseteq X$ such that $xS\xi$ and $V'x = \mathtt{tt}$. The latter assumption means that $x \overset{Q}{\not\longrightarrow}$.

Since S is a one-by-many failure simulation, this means that $y \overset{Q}{\not\longrightarrow}$ for some $y \in \xi$. Then $V'y = \mathtt{tt}$. Since $xS\xi$ were chosen arbitrarily, $\overline{V'}$ is quasi-S-upper. □

Corollary 4.40 Specialization preorders $\leq_\theta$ for B^{Fl}-coalgebras $h : \langle X, \theta\rangle \to \langle BX, B^{\mathsf{Fl}}_X\theta\rangle$ are exactly failures-aware preorders on $h : X \to BX$.

The failures preorder $\sqsubseteq_{\mathsf{Fl}}$ on h is the largest failures-aware relation on h.

Specialization equivalences $\equiv_\theta$ for B^{Fl}-coalgebras $h : \langle X, \theta\rangle \to \langle BX, B^{\mathsf{Fl}}_X\theta\rangle$ are exactly the equivalence relations $R \cap R^{-1}$ associated to failures-aware preorders R on $h : X \to BX$.

The failures equivalence $\cong_{\mathsf{Fl}}$ on h is the largest such relation on h.

Proof. Analogous to Corollaries 4.33-4.34, using Theorems 4.38-4.39. □

We proceed to consider the case of B^{FlTr}-coalgebras.

Theorem 4.41 For any B^{FlTr}-coalgebra $h : \langle X, \theta\rangle \to \langle BX, B^{\mathsf{FlTr}}_X\theta\rangle$, the specialization relation $\leq_\theta$ is a failure trace-aware preorder on $h : X \to BX$.

Proof. First, observe that $h : \langle X, \theta\rangle \to \langle BX, B^{\mathsf{FlTr}}_X\theta\rangle$ is a valid B^{FlTr}-coalgebra if and only if $h : \langle X, \theta\rangle \to \langle BX, B^{\mathsf{Fl}}_X\theta\rangle$ is a valid B^{Fl}-coalgebra and for any $Q \subseteq A$, $a \in A$, $V \in \theta$, also $\tilde{w}_{aQ} \circ BV \circ h \in \theta$.

To check that $\leq_\theta$ is a failure trace-aware preorder, consider $S \subseteq X \times \mathcal{P}X$ defined as in the proof of Theorem 4.31:

$$xS\xi \text{ iff } \forall V \in \theta.\ Vx = \mathtt{tt} \Rightarrow \exists y \in \xi.\ Vy = \mathtt{tt}$$

It is enough to check that S is a quasi-preorder and a one-by-many failure trace simulation. Since θ lifts h to a B^{Fl}-algebra, by Theorem 4.38 S is a quasi-preorder and a one-by-many failure simulation, hence it is enough to assume arbitrary $Q \subseteq A$, $a \in A$, $x, x' \in X$, $\xi \subseteq X$ such that $x \xrightarrow{a} x'$ and $x \overset{Q}{\not\longrightarrow}$ and prove that if $xS\xi$ then $x'S\left\{ y' \in X : \exists y \in \xi.\ y \xrightarrow{a} y', y \overset{Q}{\not\longrightarrow} \right\}$.

To this end, assume the opposite $x' \not S \left\{ y' \in X : \exists y \in \xi.\ y \xrightarrow{a} y', y \overset{Q}{\not\longrightarrow} \right\}$. By definition of S, there exists a test $V \in \theta$ such that

- $Vx' = \mathtt{tt}$, and
- $Vy' = \mathtt{ff}$ for all $y \in \xi$, $y \xrightarrow{a} y'$ such that $y \overset{Q}{\not\longrightarrow}$.

Consider three tests

$$V' = \tilde{w}_{aQ} \circ BV \circ h$$
$$V'' = \tilde{w}_{Q} \circ BV \circ h$$
$$V''' = w_{\langle a\rangle} \circ BV \circ h$$

Since $x \overset{Q}{\not\longrightarrow}$, one has $V''x = \mathtt{tt}$. Also $Vx' = \mathtt{tt}$ and $x \xrightarrow{a} x'$, hence $V'''x = \mathtt{tt}$. This, by definition of $\tilde{w}_{aQ}$, means that $V'x = \mathtt{tt}$.

On the other hand, by properties of V, for each $y \in \xi$ such that $V''y = \mathtt{tt}$ one has $V'''y = \mathtt{ff}$, hence $V'y = \mathtt{ff}$.

Since $V \in \theta$, also $V' \in \theta$, hence $x \not S \xi$. □

Theorem 4.42 For any coalgebra $h : X \to BX$ in **Set**, and any failure trace-aware preorder R on h, there exists a test suite $\theta_R : X \rightrightarrows 2$ such that $\leq_{\theta_R} = R$ and $h : \langle X, \theta_R\rangle \to \langle BX, B^{\mathsf{FlTr}}_X\theta_R\rangle$ is a valid B^{FlTr}-coalgebra.

Proof. Assume a quasi-preorder and one-by-many failure trace simulation S on h such that xRy if and only if $xS\{y\}$. Define, as in the proof of Theorem 4.32,

$$\theta_R = \left\{ V : X \to 2 \;:\; \overline{V} \text{ is quasi-}S\text{-upper} \right\}$$

Since R is failure trace-aware, it is also failures-aware and by Theorem 4.39, $\leq_{\theta_R} = R$ and $h : \langle X, \theta_R \rangle \to \left\langle BX, B_X^{\mathsf{Fl}}\theta_R \right\rangle$ is a valid B^{Fl}-coalgebra. It is therefore enough to assume arbitrary $Q \subseteq A$, $a \in A$, $V \in \theta_R$ and prove that $\tilde{w}_{aQ} \circ BV \circ h \in \theta_R$.

To this end, denote $V' = \tilde{w}_{aQ} \circ BV \circ h$ and take any $x \in X$, $\xi \subseteq X$ such that $xS\xi$ and $V'x = \mathtt{tt}$. By definition of $\tilde{w}_{aQ}$, the latter assumption means that $x \stackrel{Q}{\not\longrightarrow}$ and $x \stackrel{a}{\longrightarrow} x'$ for some $x' \in X$ such that $Vx' = \mathtt{tt}$.

Since S is a one-by-many failure trace simulation, this means that

$$x' S \left\{ y' \in X \;:\; \exists y \in \xi.\; y \stackrel{a}{\longrightarrow} y', y \stackrel{Q}{\not\longrightarrow} \right\}$$

and there exist $y, y' \in X$ such that $y \in \xi$, $y \stackrel{a}{\longrightarrow} y'$, $y \stackrel{Q}{\not\longrightarrow}$ and $Vy' = \mathtt{tt}$. Then $V'y = \mathtt{tt}$. Since $xS\xi$ were chosen arbitrarily, $\overline{V'}$ is quasi-S-upper. □

Corollary 4.43 Specialization preorders $\leq_\theta$ for B^{FITr}-coalgebras $h : \langle X, \theta \rangle \to \left\langle BX, B_X^{\mathsf{FITr}}\theta \right\rangle$ are exactly failure trace-aware preorders on $h : X \to BX$.

The failure trace preorder $\sqsubseteq_{\mathsf{FITr}}$ on h is the largest failure trace-aware relation on h.

Specialization equivalences $\equiv_\theta$ for B^{FITr}-coalgebras $h : \langle X, \theta \rangle \to \left\langle BX, B_X^{\mathsf{FITr}}\theta \right\rangle$ are exactly the equivalence relations $R \cap R^{-1}$ associated to failure trace-aware preorders R on $h : X \to BX$.

The failure trace equivalence $\cong_{\mathsf{Fl}}$ on h is the largest such relation on h.

Proof. Analogous to Corollaries 4.33-4.34, using Theorems 4.41-4.42. □

We proceed to consider the case of B^{Rd}-coalgebras.

Theorem 4.44 For any B^{Rd}-coalgebra $h : \langle X, \theta \rangle \to \left\langle BX, B_X^{\mathsf{Rd}}\theta \right\rangle$, the specialization relation $\leq_\theta$ is a readiness-aware preorder on $h : X \to BX$.

Proof. First, observe that $h : \langle X, \theta \rangle \to \left\langle BX, B_X^{\mathsf{Rd}}\theta \right\rangle$ is a valid B^{Rd}-coalgebra if and only if $h : \langle X, \theta \rangle \to \left\langle BX, B_X^{\mathsf{Tr}}\theta \right\rangle$ is a valid B^{Tr}-coalgebra and for any $Q \subseteq A$, $V \in \theta$, also $\check{w}_Q \circ BV \circ h \in \theta$.

To check that $\leq_\theta$ is a readiness-aware preorder, consider $S \subseteq X \times \mathcal{P}X$ defined as in the proof of Theorem 4.31:

$$xS\xi \text{ iff } \forall V \in \theta.\; Vx = \mathtt{tt} \Rightarrow \exists y \in \xi.\; Vy = \mathtt{tt}$$

It is enough to check that S is a quasi-preorder and a one-by-many ready simulation. Since θ lifts h to a B^{Tr}-algebra, by Theorem 4.31 S is a quasi-preorder and a one-by-many simulation and it is enough to assume arbitrary $x \in X$, $\xi \subseteq X$ such that $xS\xi$ and prove that $I(y) = I(x)$ for some $y \in \xi$.

To this end, take $V = \check{w}_{I(x)} \circ B\overrightarrow{X} \circ h \in \theta$. By definition of $\check{w}_{I(x)}$ one has $Vx = \mathtt{tt}$, hence (by definition of S) for some $y \in \xi$, $Vy = \mathtt{tt}$ and $I(y) = I(x)$. □

Theorem 4.45 For any coalgebra $h : X \to BX$ in **Set**, and any readiness-aware preorder R on h, there exists a test suite $\theta_R : X \rightrightarrows 2$ such that $\leq_{\theta_R} = R$ and $h : \langle X, \theta_R \rangle \to \langle BX, B^{\mathsf{Rd}}_X \theta_R \rangle$ is a valid B^{Rd}-coalgebra.

Proof. Assume a quasi-preorder and one-by-many ready simulation S on h such that xRy if and only if $xS\{y\}$. Define, as in the proof of Theorem 4.32,

$$\theta_R = \left\{ V : X \to 2 \;:\; \overline{V} \text{ is quasi-}S\text{-upper} \right\}$$

Since R is readiness-aware, it is also trace-aware and by Theorem 4.32, $\leq_{\theta_R} = R$ and $h : \langle X, \theta_R \rangle \to \langle BX, B^{\mathsf{Tr}}_X \theta_R \rangle$ is a valid B^{Tr}-coalgebra. It is therefore enough to assume arbitrary $Q \subseteq A$, $V \in \theta_R$ and prove that $\check{w}_Q \circ BV \circ h \in \theta_R$.

To this end, denote $V' = \tilde{w}_Q \circ BV \circ h$ and take any $x \in X$, $\xi \subseteq X$ such that $xS\xi$ and $V'x = \mathtt{tt}$. The latter assumption means that $I(x) = Q$.

Since S is a one-by-many ready simulation, this means that $I(y) = Q$ for some $y \in \xi$. Then $V'y = \mathtt{tt}$. Since $xS\xi$ were chosen arbitrarily, $\overline{V'}$ is quasi-S-upper. □

Corollary 4.46 Specialization preorders $\leq_\theta$ for B^{Rd}-coalgebras $h : \langle X, \theta \rangle \to \langle BX, B^{\mathsf{Rd}}_X \theta \rangle$ are exactly readiness-aware preorders on $h : X \to BX$.

The readiness preorder $\sqsubseteq_{\mathsf{Rd}}$ on h is the largest readiness-aware relation on h.

Specialization equivalences $\equiv_\theta$ for B^{Rd}-coalgebras $h : \langle X, \theta \rangle \to \langle BX, B^{\mathsf{Rd}}_X \theta \rangle$ are exactly the equivalence relations $R \cap R^{-1}$ associated to readiness preorders R on $h : X \to BX$.

The readiness equivalence $\cong_{\mathsf{Rd}}$ on h is the largest such relation on h.

Proof. Analogous to Corollaries 4.33-4.34, using Theorems 4.44-4.45. □

Finally, we consider the case of B^{RdTr}-coalgebras.

Theorem 4.47 For any B^{RdTr}-coalgebra $h : \langle X, \theta \rangle \to \langle BX, B^{\mathsf{RdTr}}_X \theta \rangle$, the specialization relation $\leq_\theta$ is a ready trace-aware preorder on $h : X \to BX$.

Proof. First, observe that $h : \langle X, \theta \rangle \to \langle BX, B^{\mathsf{RdTr}}_X \theta \rangle$ is a valid B^{RdTr}-coalgebra if and only if $h : \langle X, \theta \rangle \to \langle BX, B^{\mathsf{Rd}}_X \theta \rangle$ is a valid B^{Rd}-coalgebra and for any $Q \subseteq A$, $a \in A$, $V \in \theta$, also $\check{w}_{aQ} \circ BV \circ h \in \theta$.

To check that $\leq_\theta$ is a ready trace-aware preorder, consider $S \subseteq X \times \mathcal{P}X$ defined as in the proof of Theorem 4.31:

$$xS\xi \text{ iff } \forall V \in \theta.\ Vx = \mathtt{tt} \Rightarrow \exists y \in \xi.\ Vy = \mathtt{tt}$$

It is enough to check that S is a quasi-preorder and a one-by-many ready trace simulation. Since θ lifts h to a B^{Rd}-algebra, by Theorem 4.44 S is a quasi-preorder and a one-by-many readiness simulation and it is enough to assume

arbitrary $a \in A$, $x, x' \in X$, $\xi \subseteq X$ such that $x \xrightarrow{a} x'$ and prove that if $xS\xi$ then

$$x' S \left\{ y' \in X \,:\, \exists y \in \xi.\ y \xrightarrow{a} y', I(y) = I(x) \right\}$$

To this end, assume the opposite

$$x' \not S \left\{ y' \in X \,:\, \exists y \in \xi.\ y \xrightarrow{a} y', I(y) = I(x) \right\}$$

By definition of S, there exists a test $V \in \theta$ such that

- $Vx' = \mathtt{tt}$, and
- $Vy' = \mathtt{ff}$ for all $y \in \xi$, $y \xrightarrow{a} y'$ such that $I(y) = I(x)$.

Consider three tests

$$\begin{aligned} V' &= \check{w}_{aI(x)} \circ BV \circ h \\ V'' &= \check{w}_{I(x)} \circ BV \circ h \\ V''' &= w_{\langle a \rangle} \circ BV \circ h \end{aligned}$$

By definition of $\check{w}_{I(x)}$, one has $V''x = \mathtt{tt}$. Also $Vx' = \mathtt{tt}$ and $x \xrightarrow{a} x'$, hence $V'''x = \mathtt{tt}$. This, by definition of $\check{w}_{aI(x)}$, means that $V'x = \mathtt{tt}$.

On the other hand, by properties of V, for each $y \in \xi$ such that $V''y = \mathtt{tt}$ one has $V'''y = \mathtt{ff}$, hence $V'y = \mathtt{ff}$.

Since $V \in \theta$, also $V' \in \theta$, hence $x \not S \xi$. □

Theorem 4.48 For any coalgebra $h : X \to BX$ in **Set**, and any ready trace-aware preorder R on h, there exists a test suite $\theta_R : X \rightrightarrows 2$ such that $\leq_{\theta_R} = R$ and $h : \langle X, \theta_R \rangle \to \left\langle BX, B_X^{\mathsf{RdTr}} \theta_R \right\rangle$ is a valid B^{RdTr}-coalgebra.

Proof. Assume a one-by-many ready trace simulation S on h such that xRy if and only if $xS\{y\}$. Define, as in the proof of Theorem 4.32,

$$\theta_R = \left\{ V : X \to 2 \,:\, \overline{V} \text{ is quasi-}S\text{-upper} \right\}$$

Since R is ready trace-aware, it is also readiness-aware and by Theorem 4.45, $\leq_{\theta_R} = R$ and $h : \langle X, \theta_R \rangle \to \left\langle BX, B_X^{\mathsf{Rd}} \theta_R \right\rangle$ is a valid B^{Rd}-coalgebra. It is therefore enough to assume arbitrary $Q \subseteq A$, $a \in A$, $V \in \theta_R$ and prove that $\check{w}_{aQ} \circ BV \circ h \in \theta_R$.

To this end, denote $V' = \check{w}_{aQ} \circ BV \circ h$ and take any $x \in X$, $\xi \subseteq X$ such that $xS\xi$ and $V'x = \mathtt{tt}$. By definition of $\check{w}_{aQ}$, the latter assumption means that $I(x) = Q$ and $x \xrightarrow{a} x'$ for some $x' \in X$ such that $Vx' = \mathtt{tt}$.

Since S is a one-by-many simulation, this means that

$$x' S \left\{ y' \in X \,:\, \exists y \in \xi.\ y \xrightarrow{a} y, I(y) = I(y) \right\}$$

and there exist $y, y' \in X$ such that $y \in \xi$, $y \xrightarrow{a} y'$, $I(y) = I(x)$ and $Vy' = \mathtt{tt}$. Then $V'y = \mathtt{tt}$. Since $xS\xi$ were chosen arbitrarily, $\overline{V'}$ is quasi-S-upper. □

Corollary 4.49 Specialization preorders $\leq_\theta$ for B^{RdTr}-coalgebras $h : \langle X, \theta\rangle \to \langle BX, B^{\mathsf{RdTr}}_X \theta\rangle$ are exactly ready trace-aware preorders on $h : X \to BX$.

The ready trace preorder $\sqsubseteq_{\mathsf{RdTr}}$ on h is the largest ready trace-aware relation on h.

Specialization equivalences $\equiv_\theta$ for B^{RdTr}-coalgebras $h : \langle X, \theta\rangle \to \langle BX, B^{\mathsf{RdTr}}_X \theta\rangle$ are exactly the equivalence relations $R \cap R^{-1}$ associated to ready trace-aware preorders R on $h : X \to BX$.

The ready trace equivalence $\cong_{\mathsf{RdTr}}$ on h is the largest such relation on h.

Proof. Analogous to Corollaries 4.33-4.34, using Theorems 4.46-4.47. □

4.4 Coinduction Principle for Traces

One of the most useful applications of coalgebraic semantics of processes is the coinduction proof principle, based on the fact that the bisimulation equivalence on any LTS is the largest bisimulation on it (Proposition 2.8). Therefore to prove that two processes are bisimulation equivalent, it is enough to provide any bisimulation that relates them. The rich structure of bisimulation relations allows to use this proof principle in a very convenient fashion. Many examples of its use can be found e.g. in [77, Section 12].

Results shown in Section 4.3 allow to apply similar reasoning to other equivalences from the van Glabbeek spectrum. Corollaries 4.33, 4.34, 4.37, 4.40, 4.43, 4.46 and 4.49 characterize decorated trace preorders and equivalences as the largest trace-aware relations of a suitable kind. As it turns out, the trace-aware relations have enough structure to play in reasoning about decorated trace equivalences a similar rôle to that of bisimulations in reasoning about bisimulation equivalence.

We show an example of use of such a "coinduction principle for traces" for the case of trace equivalence.

Example 4.50 Consider a final B-coalgebra $\phi : \Omega \to B\Omega$ (recall Proposition 2.25). On the set Ω, define the associative, idempotent and commutative binary operation $+$ by

$$p + q = \phi^{-1}(\phi p \cup \phi q)$$

Now define a function $\mathtt{glue} : \Omega \to \Omega$ as the coinductive extension of the B-coalgebra $\alpha : \Omega \to B\Omega$ defined by

$$\alpha p = \left\{ \left\langle a, \coprod_{\langle a,p'\rangle \in \phi p'} p' \right\rangle \,:\, a \in I(p) \right\}$$

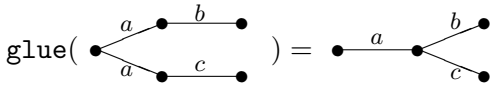
where $\coprod$ denotes the obvious extension of $+$ to finite subsets of Ω.

Using the "operational rule" notation for coinductive definitions as introduced in [77, Section 11], one may write alternatively

$$\frac{p_1, \ldots, p_n (n > 0) \text{ are exactly the processes for which } p \xrightarrow{a} p_i}{\mathtt{glue}(p) \xrightarrow{a} \mathtt{glue}(p_1 + \cdots + p_n)}$$

for any $a \in A$.

For example, if Ω is the set of all finitely branching labelled synchronisation trees quotiented by bisimulation equivalence, one has



Theorem 4.51 The operation `glue` preserves and respects traces. In other words, for any process $p \in \Omega$, p and $\mathtt{glue}(p)$ are trace equivalent.

Proof. A standard way of proving this theorem is to use induction on the length of traces. Instead, we use the coinduction proof principle for traces, as expressed in Corollaries 4.33 and 4.34.

First, we show that the relation $\{ \langle \mathtt{glue}(p), p \rangle : p \in \Omega \}$ is a trace-aware relation. Consider $S \subseteq \Omega \times \mathcal{P}\Omega$ defined by

$$pS\{q_1, \ldots, q_n\} \iff \mathtt{glue}\left(\coprod_{i \in I} q_i\right) = p \text{ for some } I \subseteq \{1, \ldots, n\}.$$

To show that S is a one-by-many simulation, consider any $a \in A$, $p, p' \in \Omega$, $\xi \in \Omega$ such that $pS\xi$ and $p \xrightarrow{a} p'$. By definition of S, $p = \mathtt{glue}\left(\coprod_{i \in I} q_i\right)$ for some index set I such that $q_i \in \xi$ for every $i \in I$. By definitions of `glue` and $\coprod$, there is

$$p' = \mathtt{glue}\left(\coprod_{i \in I, q_i \xrightarrow{a} q_i'} q_i'\right)$$

hence

$$p' S \left\{ q' \in \Omega : \exists q \in \xi.\ q \xrightarrow{a} q' \right\}$$

This concludes the first part of the proof.

Next, we show that the relation $\{ \langle p, \mathtt{glue}(p) \rangle : p \in \Omega \}$ is contained in a trace-aware relation. Consider $S \subseteq \Omega \times \mathcal{P}\Omega$ defined by

$$pS\xi \iff \mathtt{glue}(p + q) \in \xi \text{ for some } q \in \Omega.$$

To show that S is a one-by-many simulation, consider any $a \in A$, $p, p' \in \Omega$, $\xi \in \Omega$ such that $pS\xi$ and $p \xrightarrow{a} p'$. This means that also $p + q \xrightarrow{a} p'$, where $\mathtt{glue}(p + q) \in \xi$. Then, by definition of `glue`, $\mathtt{glue}(p + q) \xrightarrow{a} \mathtt{glue}(p' + q')$ for some $q' \in \Omega$. From this it follows that

$$\mathtt{glue}(p' + q') \in \left\{ r' \in \Omega : \exists r \in \xi.\ r \xrightarrow{a} r' \right\}$$

hence

$$p' S \left\{ r' \in \Omega : \exists r \in \xi.\ r \xrightarrow{a} r' \right\}$$

and S is a one-by-many simulation, which concludes the proof.

□

4.5 Comparison of Process Preorders and Equivalences

This section contains some applications of results from Section 3.3.4, allowing one to compare specialization preorders and equivalences induced from different liftings of the same behaviour endofunctor.

It must be stressed that the results presented here are not novel. In fact, they were all proved in e.g. [35]. The purpose of this section is to show that Theorem 3.33 and its corollaries are powerful enough to prove many simple comparisons between process preorders and equivalences, even though they involve only test constructors, and not any more explicit representation of these preorders and equivalences. On the other hand, as is shown below, these results have some limitations: they do not allow one to prove all known containment relations between process preorders in the van Glabbeek spectrum.

Theorem 4.52 For $\mathsf{W} \in \{\mathsf{Tr}, \mathsf{CTr}, \mathsf{Fl}, \mathsf{FlTr}, \mathsf{Rd}, \mathsf{RdTr}, \mathsf{S}, \mathsf{RdS}, \mathsf{BS}\}$, for any $h : X \to BX$,

$$\cong_{\mathsf{W}} \subseteq \sqsubseteq_{\mathsf{W}}$$

Proof. A simple consequence of Corollary 3.35 (see also Example 3.29 and Corollary 4.9). □

Theorem 4.53 The following relations hold for any coalgebra $h : X \to BX$:

- $\sqsubseteq_{\mathsf{FlTr}} \subseteq \sqsubseteq_{\mathsf{Fl}} \subseteq \sqsubseteq_{\mathsf{CTr}} \subseteq \sqsubseteq_{\mathsf{Tr}}$,
- $\sqsubseteq_{\mathsf{RdTr}} \subseteq \sqsubseteq_{\mathsf{Rd}} \subseteq \sqsubseteq_{\mathsf{CTr}}$,
- $\cong_{\mathsf{FlTr}} \subseteq \cong_{\mathsf{Fl}} \subseteq \cong_{\mathsf{CTr}} \subseteq \cong_{\mathsf{Tr}}$,
- $\cong_{\mathsf{RdTr}} \subseteq \cong_{\mathsf{Rd}} \subseteq \cong_{\mathsf{CTr}}$.

Proof. For the former two statements, consider Corollary 3.35 with $\mathcal{V} = 2$, $S = R = \leq_2$, $\mathrm{Cl} = \mathrm{Cl}' = \mathrm{Cl}^{\top}$ and $\zeta = \{id_2\}$. In all cases mentioned above the condition $\mathsf{W}' \subseteq \mathsf{W}$ holds (see Definitions 4.3, 4.5). In particular, for the case $\sqsubseteq_{\mathsf{Rd}} \subseteq \sqsubseteq_{\mathsf{CTr}}$, observe that $\tilde{w}_A = \tilde{w}_{\emptyset}$.

For the latter two statements, proceed analogously with $S = R = =_2$. □

Theorem 4.54 For any coalgebra $h : X \to BX$, one has $\sqsubseteq_{\mathsf{S}} \subseteq \sqsubseteq_{\mathsf{Tr}}$ and $\cong_{\mathsf{S}} \subseteq \cong_{\mathsf{Tr}}$.

Proof. Recall Definition 4.5 and consider Corollary 3.35 with $\mathcal{V} = 2$, $S = R = \leq_2, =_2$, $\zeta = \{id_2\}$, $\mathsf{W} = \mathsf{W}' = \mathsf{Tr}$, $\mathrm{Cl} = \mathrm{Cl}^{\wedge}$, $\mathrm{Cl}' = \mathrm{Cl}^{\top}$. It is easy to check that all conditions in Corollary 3.35 hold. □

Theorem 4.55 For any coalgebra $h : X \to BX$, one has $\sqsubseteq_{\mathsf{BS}} \subseteq \sqsubseteq_{\mathsf{S}}$ and $\cong_{\mathsf{BS}} \subseteq \cong_{\mathsf{S}}$.

Proof. Recall Definition 4.5 and consider Corollary 3.35 with $\mathcal{V} = \mathcal{V}' = 2$, $S = R = \leq_2, =_2$, $\zeta = \{id_2\}$, $\mathsf{W} = \mathsf{BS}$, $\mathsf{W}' = \mathsf{Tr}$, $\mathrm{Cl} = \mathrm{Cl}^{\vee\wedge}$, $\mathrm{Cl}' = \mathrm{Cl}^{\wedge}$. It is easy to check that all conditions in Corollary 3.34 hold. □

Note that we have not proved all known relations between operational preorders and equivalences from the van Glabbeek spectrum. In particular, from Theorem 3.33 one cannot easily conclude that $\cong_{\mathsf{Rd}} \subseteq \cong_{\mathsf{Fl}}$, which is a well known result [35].

It is likely possible to extend Theorem 3.33 to an abstract result which, when specialized to the special case considered in this chapter, would allow to prove this and other missing relations. It is not our purpose, however, since it is highly unlikely that such an abstract result would lead to novel corollaries, impossible or difficult to obtain by syntactic manipulations on modal formulae. The purpose of this section is rather to hint at the usage and the scope of Theorem 3.33, for potential use in other settings.

4.6 Nested Semantics

In previous sections, not all preorders and equivalences from the van Glabbeek spectrum as described in [35] were treated. The equivalences left aside are 2-nested simulation equivalence, possible futures equivalence, and possible worlds equivalence. Indeed, it turns out that these three equivalences (and corresponding preorders) cannot be easily described in the test suite framework as shown in this chapter so far, with the test value set $\mathcal{V} = 2$.

This is related to the fact that the modal formulae used to characterize the three equivalences cannot be described by BNF grammars with single nonterminal symbols. Indeed, if one insists that the $(B, 2)$-test constructors chosen to lift the endofunctor B to 2-**TS** correspond to the modal operators in the corresponding modal logic (as we indeed insisted in Definition 4.3), then the tests constructed by means of these constructors correspond to completely arbitrary formulae built of the modal operators. However, BNF grammars with multiple nonterminal symbols can impose syntactic restrictions on the formulae considered, not reflected in the set of tests induced by test constructors.

A general solution to this problem would be to extend the framework presented in Chapter 3, allowing tests to be described syntactically as terms over some multi-sorted algebraic signature, as used in theory of algebraic specifications (e.g. [79]), and only then mapped to proper tests (i.e., functions to a set of test values) by means of some semantic function. This, however, would complicate the entire framework considerably.

Fortunately, the lack of syntactic restrictions on tests can be circumvented even in the simple framework shown in Chapter 3, by choosing a different set of test values rather that 2. Roughly, instead of the simple set of logical values $\{\mathtt{tt}, \mathtt{ff}\}$, one considers a set containing multiple *copies* of the logical values, corresponding to different 'sorts' of tests. This way, information about the 'sort' of a given test is stored in its values when applied to processes, and it can be used by the test constructors to recognize the 'syntactically ill-formed' tests. Technically, one also needs a special 'error' test value, to represent the

ill-formed tests. An appropriate notion of specialization relation then ensures that the ill-formed tests, i.e. those that take the 'error' test value when applied to a process, are not given any distinguishing power.

In this section this approach is illustrated on the example of 2-nested simulation semantics. The other two problematic examples can be treated in analogous fashion.

Definition 4.56 Given a set of actions A, the set of modal formulae $\mathcal{F}_{\mathsf{2S}}$ is defined by the BNF grammar (with the starting nonterminal symbol ϕ)

$$\begin{array}{rcl} \phi & ::= & \top \mid \langle a \rangle\, \phi \mid \phi \wedge \phi \mid \boxdot \phi_\neg \\ \phi_\neg & ::= & \bot \mid [a]\phi_\neg \mid \phi_\neg \vee \phi_\neg \end{array}$$

For convenience, the set of formulae generated by the nonterminal symbol $\phi_\neg$ is also considered, and denoted $\mathcal{F}_{\neg\mathsf{S}}$.

Given a finitely branching LTS $h = \langle X, A, \rightarrow \rangle$ (equivalently, a B-coalgebra $h : X \rightarrow BX$), the satisfaction relation $\models_h \subseteq (\mathcal{F}_{\mathsf{2S}} \cup \mathcal{F}_{\neg\mathsf{S}}) \times X$ is defined inductively as in Definition 2.3, with the additional clause

$$x \models_h \boxdot \phi \iff x \models_h \phi$$

Clearly the operator $\boxdot$ does not contribute much to the semantics of formulae. It is introduced here to clarify the presentation by making a clear syntactic distinction between formulae in $\mathcal{F}_{\mathsf{2S}}$ and $\mathcal{F}_{\neg\mathsf{S}}$.

As in Definition 2.4, one considers a preorder $\sqsubseteq_{\mathsf{2S}}$ and an equivalence $\cong_{\mathsf{2S}}$, defined on a given LTS h as follows:

$$\begin{array}{rcl} x \sqsubseteq_{\mathsf{2S}} x' & \iff & (\forall \phi \in \mathcal{F}_{\mathsf{2S}}.\ x \models_h \phi \Longrightarrow x' \models_h \phi) \\ x \cong_{\mathsf{2S}} x' & \iff & (\forall \phi \in \mathcal{F}_{\mathsf{2S}}.\ x \models_h \phi \iff x' \models_h \phi) \end{array}$$

Assuming a given finitely branching LTS $\langle X, A, \rightarrow \rangle$, the relations $\sqsubseteq_{\mathsf{2S}}$ and $\cong_{\mathsf{2S}}$ are characterized by a notion of 2-nested simulation, similar to those of simulation and bisimulation (Definitions 2.5–2.7):

Definition 4.57 A relation $R \subseteq X \times X$ is a *2-nested simulation* if it is a simulation contained in the simulation equivalence $\cong_{\mathsf{S}}$. Processes $x, y \in X$ are

- in *2-nested simulation preorder* if there exists a 2-nested simulation R such that xRy,
- *2-nested simulation equivalent* if there exist 2-nested simulations R, R' such that xRy and $yR'x$.

As shown in [35], formulae from $\mathcal{F}_{\mathsf{2S}}$ characterize 2-nested simulation preorder and equivalence:

Proposition 4.58 In any finitely branching LTS, the relation $\sqsubseteq_{\mathsf{2S}}$ is equal to 2-nested simulation preorder, and the relation $\cong_{\mathsf{2S}}$ is equal to 2-nested simulation equivalence.

To describe the 2-nested simulation semantics coalgebraically, we specialize the framework described in Chapter 3 with the category $\mathbb{C} = \mathbf{Set}$ and the behaviour functor $BX = \mathcal{P}_{\mathrm{f}}(A \times X)$ for a fixed set A, as before in this chapter. However, instead of $\mathcal{V} = 2$, we choose

$$\mathcal{V} = 5 = \{\mathtt{tt}, \mathtt{ff}, \mathtt{tt}_\neg, \mathtt{ff}_\neg, \mathtt{err}\}$$

Note that 5-tests on X cannot be identified with subsets of X, hence the convenient Notation 4.1 does not apply here.

On the set 5, define a binary relation R by

$$xRy \iff (x = \mathtt{tt} \Longrightarrow y = \mathtt{tt})$$

The pair $\langle 5, R\rangle$ is an object in the category **Rel** and induces a specialization functor $\mathrm{Spec}_R : 5\text{-}\mathbf{TS} \to \mathbf{Rel}$ along the lines of Definition 3.24. For any test suite $\theta : X \rightrightarrows 5$, the relation $\mathrm{Sp}_R\,\theta \subseteq X \times X$ can be described by

$$x(\mathrm{Sp}_R\,\theta)y \iff \forall V \in \theta.(Vx = \mathtt{tt} \Longrightarrow Vy = \mathtt{tt})$$

Note that $\mathrm{Sp}_R\,\theta$ is always a preorder.

To lift the endofunctor B to the category 5-**TS**, we define an appropriate set of $(B, 5)$-test constructors.

Definition 4.59 For any $a \in A$, define functions

$$u_{\langle a\rangle}, u_{[a]} : B5 \to 5$$

as follows (compare Definition 4.2):

$$u_{\langle a\rangle}\beta = \begin{cases} \mathtt{tt} & \text{if } \langle a, \mathtt{tt}\rangle \in \beta \\ \mathtt{err} & \text{otherwise, if for some } a \in A, v \in \{\mathtt{err}, \mathtt{tt}_\neg, \mathtt{ff}_\neg\}, \langle a, v\rangle \in \beta \\ \mathtt{ff} & \text{otherwise} \end{cases}$$

$$u_{[a]}\beta = \begin{cases} \mathtt{ff}_\neg & \text{if } \langle a, \mathtt{ff}_\neg\rangle \in \beta \\ \mathtt{err} & \text{otherwise, if for some } a \in A, v \in \{\mathtt{err}, \mathtt{tt}, \mathtt{ff}\}, \langle a, v\rangle \in \beta \\ \mathtt{tt}_\neg & \text{otherwise} \end{cases}$$

We also define a test suite closure, based (for a given set X) on the following two test constants:

$$\begin{aligned} \mathtt{T}x &= \mathtt{tt} \\ \mathtt{F}x &= \mathtt{ff}_\neg \end{aligned}$$

the unary test operator

$$(\boxdot V)x = \begin{cases} \mathtt{tt} & \text{if } Vx = \mathtt{tt}_\neg \\ \mathtt{ff} & \text{if } Vx = \mathtt{ff}_\neg \\ \mathtt{err} & \text{otherwise} \end{cases}$$

and two binary test operators

$$(V \sqcap V')x = \begin{cases} \mathtt{tt} & \text{if } Vx = \mathtt{tt} \text{ and } V'x = \mathtt{tt} \\ \mathtt{ff} & \text{otherwise, if } Vx, V'x \in \{\mathtt{tt}, \mathtt{ff}\} \\ \mathtt{err} & \text{otherwise} \end{cases}$$

$$(V \sqcup V')x = \begin{cases} \mathtt{ff}_\neg & \text{if } Vx = \mathtt{ff}_\neg \text{ and } V'x = \mathtt{ff}_\neg \\ \mathtt{tt}_\neg & \text{otherwise, if } Vx, V'x \in \{\mathtt{tt}_\neg, \mathtt{ff}_\neg\} \\ \mathtt{err} & \text{otherwise} \end{cases}$$

Definition 4.60 Let $\theta : X \rightrightarrows 5$ be a test suite. The test suite $\mathrm{Cl}_X^{\mathsf{2S}}\,\theta$ is the smallest test suite ϑ that contains θ and such that

- $\mathtt{T}, \mathtt{F} \in \vartheta$,
- if $V \in \vartheta$ then $\boxdot V \in \vartheta$,
- if $V, V' \in \vartheta$ then $V \sqcup V', V \sqcap V' \in \vartheta$.

It is straightforward to check that the operations $\mathrm{Cl}_X^{\mathsf{2S}}$ for all sets X form a test suite closure (henceforth denoted $\mathrm{Cl}^{\mathsf{2S}}$) according to Definition 3.31. The proof is entirely analogous to that for closures $\mathrm{Cl}^{\wedge}, \mathrm{Cl}^{\vee\wedge}$ in Section 4.1.

Definition 4.61 The endofunctor on 5-**TS** induced by the set of $(B, 5)$-test suite constructors

$$\left\{ u_{\langle a \rangle} : a \in A \right\} \cup \left\{ u_{[a]} : a \in A \right\}$$

together with the closure $\mathrm{Cl}^{\mathsf{2S}}$, is denoted B^{2S}. Given a B-coalgebra $h : X \to BX$, the operator corresponding to B^{2S} as defined in Section 3.1.5 is denoted Φ_h^{2S}.

Analogously to Definition 4.6, one inductively defines a function $[\![-]\!]_h : (\mathcal{F}_{\mathsf{2S}} \cup \mathcal{F}_{\neg\mathsf{S}}) \to (X \to 5)$ as follows:

$$\begin{aligned} [\![\top]\!]_h &= \mathsf{T} \\ [\![\bot]\!]_h &= \mathsf{F} \\ [\![\langle a \rangle\, \phi]\!]_h &= u_{\langle a \rangle} \circ B[\![\phi]\!]_h \circ h \\ [\![[a]\phi]\!]_h &= u_{[a]} \circ B[\![\phi]\!]_h \circ h \\ [\![\boxdot \phi]\!]_h &= \boxdot [\![\phi]\!]_h \\ [\![\phi_1 \wedge \phi_2]\!]_h &= [\![\phi_1]\!]_h \sqcap [\![\phi_2]\!]_h \\ [\![\phi_1 \vee \phi_2]\!]_h &= [\![\phi_1]\!]_h \sqcap [\![\phi_2]\!]_h \end{aligned}$$

All 5-tests obtained this way are particularly well-behaved. Indeed,

Lemma 4.62 For any formula $\phi \in \mathcal{F}_{\neg\mathsf{S}}$, and for any $x \in X$, there is $[\![\phi]\!]_h x \in \{\mathtt{tt}_\neg, \mathtt{ff}_\neg\}$. For any formula $\phi \in \mathcal{F}_{\mathsf{2S}}$, and for any $x \in X$, there is $[\![\phi]\!]_h x \in \{\mathtt{tt}, \mathtt{ff}\}$.

Proof. Straightforward induction on formulae. □

Then, analogously to Theorem 4.7, one obtains a correspondence between formulae and 5-tests.

Theorem 4.63 Let $h : X \to BX$ be a coalgebra. For any formula $\phi \in \mathcal{F}_{\neg\mathsf{S}}$ and any $x \in X$,

$$[\![\phi]\!]_h x = \mathtt{tt}_\neg \iff x \models_h \phi$$

For any formula $\phi \in \mathcal{F}_{\mathsf{2S}}$ and any $x \in X$,

$$[\![\phi]\!]_h x = \mathtt{tt} \iff x \models_h \phi$$

Proof. Again, straightforward induction on formulae, much the same as in the proof of Theorem 4.7, using Lemma 4.62. □

The last missing item in the coalgebraic characterization of the 2-nested simulation preorder is a counterpart of Theorem 4.8. Technical reasons force us to restrict attention only to LTSs with no stuck processes:

Theorem 4.64 Let $h : X \to BX$ be a coalgebra with $hx \neq \emptyset$ for all $x \in X$. The test suite

$$\theta_{\mathsf{2S}} = \{\, [\![\phi]\!]_h \,:\, \phi \in \mathcal{F}_{\mathsf{2S}} \,\} \cup \{\, [\![\phi]\!]_h \,:\, \phi \in \mathcal{F}_{\neg\mathsf{S}} \,\} \cup \{\mathsf{E}\}$$

(where E is the test constantly equal $\mathtt{err}$) is the least fixed point of the operator Φ_h^{2S}.

Proof. To show that θ_{2S} is a prefixed point of Φ_h^{2S}, one needs to check that it is closed under all operations mentioned in Definitions 4.59–4.60. The only problematic cases are those of $u_{\langle a\rangle}$ and $u_{[a]}$. For any test $V : X \to 5$, assume $V \in \theta_{\mathsf{2S}}$ and denote $V' = u_{\langle a\rangle} \circ BV \circ h$. If $V = [\![\phi]\!]_h$ for some $\phi \in \mathcal{F}_{\mathsf{2S}}$, then $V' = [\![\langle a\rangle\, \phi]\!]_h$. If $V = \mathsf{E}$ or $V = [\![\phi]\!]_h$ for some $\phi \in \mathcal{F}_{\neg\mathsf{S}}$, then it is straightforward to check, by definition of $u_{\langle a\rangle}$ and by Lemma 4.62, that $V' = \mathsf{E}$ (here the assumption that always $hx \neq \emptyset$ is used). The proof for $u_{[a]}$ is entirely analogous.

To check that θ_{2S} is the least fixed point of Φ^{2S}, proceed by straightforward structural induction on formulae. □

Finally, analogously to Corollary 4.9,

Corollary 4.65 Let $h : X \to BX$ be a B-coalgebra where $hx \neq \emptyset$ for all $x \in X$, and let $k : X \to \Omega$ be the coinductive extension of h. Consider $k^*\omega^{\mathsf{2S}} : X \rightrightarrows 5$, the least 5-test suite lifting h to a B^{2S}-coalgebra. Then

$$\sqsubseteq_{\mathsf{2S}} = \mathrm{Sp}_R(k^*\omega^{\mathsf{2S}})$$

Proof. By Theorem 3.12, $k^*\omega^{\mathsf{2S}}$ is equal to θ_{2S} from Theorem 4.64. Now use Lemma 4.62, compare the definition of Sp_R with the definition of $\sqsubseteq_{\mathsf{2S}}$ and use Theorem 4.63. □

The required restriction to LTSs with no stuck processes admittedly gives the entire construction shown in this section a rather unnatural feeling. Nevertheless, the construction characterizes 2-nested simulation preorder coalgebraically, and shows that test suites based on test value sets different from 2 can be of use.

A full characterization of $B^{2\mathsf{S}}$-coalgebras analogous to those given in Sections 4.3 and 4.4 is missing, but it is worthwhile to prove a partial result, as it is an example of the use of Theorem 3.33 in its full generality, unlike the simple examples from Section 4.4.

Theorem 4.66 For every $B^{2\mathsf{S}}$-coalgebra

$$h : \langle X, \theta \rangle \to \left\langle BX, B^{2\mathsf{S}}_X \theta \right\rangle$$

the relation $\mathrm{Sp}_R\, \theta$ is a reflexive, transitive simulation on $h : X \to BX$.

Proof. Instead of characterizing $\mathrm{Sp}_R\, \theta$ directly, we use Theorem 3.33 with $\mathcal{V} = 5$, $\mathcal{V}' = 2$, $\mathsf{W} = 2\mathsf{S}$, $\mathsf{W}' = \mathsf{Tr}$, $\mathrm{Cl} = \mathrm{Cl}^{2\mathsf{S}}$, and $\mathrm{Cl}' = \mathrm{Cl}^{\wedge}$. To this end, consider the function $z : 5 \to 2$ defined by

$$z(x) = \mathtt{tt} \iff x = \mathtt{tt}$$

and let $\zeta : 5 \rightrightarrows 2$ be $\{z\}$.

Two conditions from the statement of Theorem 3.33 must be checked. First, consider any test suite $\theta : X \rightrightarrows 5$. Elements of $\mathrm{Cl}^{\wedge}\, \mathrm{Sp}_\zeta\, \theta$ are of the form

$$V = (z \circ V_1) \wedge \ldots \wedge (z \circ V_k) : X \to 2$$

where $V_1, \ldots, V_k \in \theta$. Given such a test V, consider the test $V' : X \to 2$ defined by

$$V' = z \circ (V_1 \sqcap \ldots \sqcap V_k)$$

and check that $Vx = \mathtt{tt}$ if and only if $V'x = \mathtt{tt}$, i.e., $V = V'$. Therefore, $\mathrm{Cl}^{\wedge}\, \mathrm{Sp}_\zeta \subseteq \mathrm{Sp}_\zeta\, \mathrm{Cl}^{2\mathsf{S}}\, \theta$.

For the second condition, take any $w_{\langle a \rangle} \in \mathsf{Tr}$. It is straightforward to check that

$$z \circ u_{\langle a \rangle} = w_{\langle a \rangle} \circ Bz$$

This, by Theorem 3.33, means that

$$h : \left\langle X, \mathrm{Sp}_\zeta\, \theta \right\rangle \to \left\langle BX, B^{\mathsf{S}}_X\, \mathrm{Sp}_\zeta\, \theta \right\rangle$$

is a valid B^{S}-coalgebra (see Definition 4.5), therefore, by Theorem 4.10 $\leq_{\mathrm{Sp}_\zeta\, \theta}$ is a reflexive, transitive simulation on $h : X \to BX$.

On the other hand, it is easy to check that for the relation $R \subseteq 5 \times 5$ defined before, one has $R = \leq_\zeta = \mathrm{Sp}_{\leq_2} \zeta$, hence, by the second statement of Theorem 3.28,

$$\mathrm{Sp}_R\, \theta = \mathrm{Sp}_{\leq_2}\, \mathrm{Sp}_\zeta\, \theta = \leq_{\mathrm{Sp}_\zeta\, \theta}$$

which concludes the proof. □

Chapter 5

From Test Suites to Congruence Formats

In Chapter 4, it was described how to represent various known process preorders and equivalences using coalgebras for suitably chosen endofunctors on the test suite category 2-**TS**. To achieve that, only the coalgebraic aspects of the framework presented in Chapter 3 were used.

In this chapter, solutions presented in Chapter 4 are combined with the algebraic aspects of the fibrational framework to obtain syntactic formats for GSOS specifications that guarantee various preorders and equivalences to be (pre)congruences. The main tool used for this purpose is Theorem 3.13.

In Section 5.1, we show how to lift polynomial syntactic endofunctors to the category 2-**TS** in a structured way. This, together with a given lifting of the behaviour endofunctor, allows to lift the entire GSOS framework along the lines of Section 3.1.6. In Section 5.2, Theorem 3.13 is rephrased for the special case of specialization preorders and equivalences for coalgebras in 2-**TS**, showing a way to define syntactic formats for various notions of equivalence of processes. Finally, in Sections 5.3-5.5, this technique is used to derive congruence formats for the trace, completed trace and failures preorders and equivalences from the van Glabbeek spectrum.

5.1 Lifting Syntax to Test Suites

To lift the bialgebraic abstract GSOS framework to the total category 2-**TS** along the lines presented in Section 3.1.6, one needs a way to lift the syntactic and the behaviour endofunctors Σ and B together with the natural transformation λ. Various ways to lift the behaviour $BX = \mathcal{P}_{\mathrm{f}}(A \times X)$ were shown in Chapter 4, now we proceed to show a lifting of polynomial endofunctors, usually used to model syntax of processes.

We begin by giving concrete descriptions of products and coproducts in 2-**TS** (see Theorems 3.4 and 3.6).

Lemma 5.1 For any test suites $\theta : X \rightrightarrows 2$, $\vartheta : Y \rightrightarrows 2$,

$$\begin{aligned} \theta \boxplus \vartheta &= \left\{ [V, V'] : V \in \theta, V' \in \vartheta \right\} \\ \theta \boxtimes \vartheta &= \left\{ V \circ \pi_1 : V \in \theta \right\} \cup \left\{ V' \circ \pi_2 : V' \in \vartheta \right\} \end{aligned}$$

Proof. Calculate

$$\begin{aligned}\theta \boxtimes \vartheta &= \pi_1^* \theta \cup \pi_2^* \vartheta = \{\, V \circ \pi_1 \,:\, V \in \theta \,\} \cup \{\, V' \circ \pi_2 \,:\, V' \in \vartheta \,\} \\ \theta \boxplus \vartheta &= (\iota_1)_! \theta \cap (\iota_2)_! \vartheta = \{\, V : X + Y \to 2 \,:\, V \circ \iota_1 \in \theta, V \circ \iota_2 \in \vartheta \,\} = \\ &= \{\, [V, V'] \,:\, V \in \theta, V' \in \vartheta \,\}\end{aligned}$$

□

In the following, it will also be useful to consider another construction on test suites:

Definition 5.2 Consider any test suites $\theta : X \rightrightarrows 2$, $\vartheta : Y \rightrightarrows 2$. The test suite $\theta \bowtie \vartheta : X \times Y \rightrightarrows 2$ is defined by

$$\theta \bowtie \vartheta = \{\, \wedge \circ (V \times V') \,:\, V \in \theta, V' \in \vartheta \,\}$$

where $\wedge : 2 \times 2 \to 2$ is the logical-and operator.

It is easy to see that $\bowtie$ is associative, therefore we will omit parentheses around its use when appropriate.

Intuitively, given test suites $\theta : X \rightrightarrows 2$ and $\vartheta : Y \rightrightarrows 2$, $\theta \boxplus \vartheta$ is the test suite on $X + Y$ obtained by taking (disjoint) unions of tests from θ on X and ϑ on Y, $\theta \boxtimes \vartheta$ contains the tests on $X \times Y$ which consist of *either* a test from θ on X *or* a test from θ on Y; finally, $\theta \bowtie \vartheta$ is the test suite on $X \times Y$ consisting of tests built by performing a test from θ on X and simultaneously performing a test from ϑ on Y and accepting when *both* tests accept.

Now we show a structured way of lifting polynomial syntactic endofunctors to 2-**TS**.

Definition 5.3 Given a polynomial endofunctor Σ on **Set**:

$$\Sigma X = X^{n_1} + \cdots + X^{n_k}$$

define an endofunctor Σ^* on 2-**TS** by the following action $\Sigma_X : \mathcal{T}_2 X \to \mathcal{T}_2 \Sigma X$ for any set X (see Section 3.1.2):

$$\Sigma_X \theta = \mathrm{Cl}^\vee_{\Sigma X}(\theta^{\bowtie n_i} \boxplus \cdots \boxplus \theta^{\bowtie n_k})$$

where $\mathrm{Cl}^\vee$ is the closure under *all* unions of tests (compare Definition 4.4) and $\theta^{\bowtie n_i}$ denotes $\underbrace{\theta \bowtie \theta \bowtie \cdots \bowtie \theta}_{n_i \text{ times}}$; in particular $\theta^{\bowtie 0} = \mathbf{Set}(1, 2) = \{\, \overrightarrow{1}, \overrightarrow{\emptyset} \,\}$.

It is easy to check that $\mathrm{Cl}^\vee$ is indeed a closure according to Definition 3.31. To check functoriality of Σ^* (see Section 3.1.2), since $\mathrm{Cl}^\vee$ is a closure and $\boxplus$ the action of the categorical coproduct on 2-**TS**, it is enough to check that the action $(-)^{\bowtie 2}$ induces an endofunctor, i.e., that for any $f : X \in Y$ in **Set**, and

any 2-test suite θ on Y, one has $(f^*\theta) \bowtie (f^*\theta) \supseteq (f \times f)^*(\theta \bowtie \theta)$. Indeed, calculate

$$\begin{aligned}
(f \times f)^*(\theta \bowtie \theta) &= (f \times f)^* \left\{ \wedge \circ (V \times V') \,:\, V, V' \in \theta \right\} = \\
&= \left\{ \wedge \circ (V \times V') \circ (f \times f) \,:\, V, V' \in \theta \right\} = \\
&= \left\{ \wedge \circ ((V \circ f) \times (V' \circ f)) \,:\, V, V' \in \theta \right\} = \\
&= \left\{ \wedge \circ (V \times V') \,:\, V, V' \in f^*\theta \right\} = \\
&= (f^*\theta) \bowtie (f^*\theta)
\end{aligned}$$

The following theorem is a crucial property of Σ^* defined above, and it will allow to define various congruence formats in later sections. Indeed, varying the definition of Σ^* in our framework would lead to the definition of various formats, but only as long as the following property holds.

Theorem 5.4 For any Σ^*-algebra $h : \langle \Sigma X, \Sigma_X \theta \rangle \to \langle X, \theta \rangle$, the specialization preorder $\leq_\theta$ is a precongruence, and the specialization equivalence $\equiv_\theta$ a congruence on $h : \Sigma X \to X$ (see Definition 2.17).

Proof. To prove the first statement, let $\mathtt{f} : X^n \to \Sigma X$ be one of the coproduct injections to ΣX, and consider elements $x_1, y_1, \ldots, x_n, y_n \in X$ such that there is $x_i \leq_\theta y_i$ for $i = 1, \ldots, n$. This means that

$$\langle x_1, \ldots, x_n \rangle \leq_{\theta^{\bowtie n}} \langle y_1, \ldots, y_n \rangle$$

Observe that the closure operator $\mathrm{Cl}^\vee$ does not change specialization preorders: for any test suite θ, one has $\leq_{\mathrm{Cl}^\vee \theta} = \leq_\theta$. Hence

$$\mathtt{f}\,\langle x_1, \ldots, x_n \rangle \leq_{\Sigma_X \theta} \mathtt{f}\,\langle y_1, \ldots, y_n \rangle$$

Now since h is a morphism 2-**TS**, we know that $h^*\theta \subseteq \Sigma_X \theta$, hence

$$\mathtt{f}\,\langle x_1, \ldots, x_n \rangle \leq_{h^*\theta} \mathtt{f}\,\langle y_1, \ldots, y_n \rangle$$

and

$$h(\mathtt{f}\,\langle x_1, \ldots, x_n \rangle) \leq_\theta h(\mathtt{f}\,\langle y_1, \ldots, y_n \rangle)$$

To prove the second statement, proceed identically, replacing $\leq$ by $\equiv$ throughout. □

Note that by Theorem 3.9, the monad T freely generated by Σ lifts to a monad T^* freely generated by Σ^*.

5.2 Abstract Congruence Formats

The following corollary, based on several results shown previously in this thesis, expresses abstract conditions on GSOS specifications that guarantee various operational preorders and equivalences from the van Glabbeek spectrum to be precongruences (resp. congruences).

Corollary 5.5 Let $\mathsf{W} \in \{\mathsf{Tr}, \mathsf{CTr}, \mathsf{Fl}, \mathsf{FlTr}, \mathsf{Rd}, \mathsf{RdTr}, \mathsf{S}, \mathsf{RdS}, \mathsf{BS}\}$ as in Definition 4.5. Let Λ be an image finite GSOS specification and $\lambda : \Sigma(\mathrm{Id} \times B) \to BT$ (where $B = \mathcal{P}_\mathrm{f}(A \times -)$) the natural transformation corresponding to Λ along the lines of Theorem 2.25. If λ lifts to a well-defined natural transformation

$$\lambda : \Sigma^*(\mathrm{Id} \times B^\mathsf{W}) \to B^\mathsf{W} T^*$$

then the operational preorder $\sqsubseteq_\mathsf{W}$ (the operational equivalence $\cong_\mathsf{W}$) on the LTS generated by Λ is a precongruence (resp. congruence, see Definition 2.11).

Proof. By Theorem 3.13,

$$\psi : \left\langle \Sigma T0, \Sigma_{T0} k^* \omega^\mathsf{W} \right\rangle \to \left\langle T0, k^* \omega^\mathsf{W} \right\rangle$$

is a valid Σ^*-algebra, where $\psi : \Sigma T0 \to T0$ is the initial Σ-algebra, $k : T0 \to \Omega$ is the coinductive extension of the coalgebraic part of the initial λ-bialgebra (i.e., by Theorem 2.28, the LTS generated by Λ), and ω^W is taken from the final B^W-coalgebra

$$\phi : \left\langle \Omega, \omega^\mathsf{W} \right\rangle \to \left\langle B\Omega, B^\mathsf{W}_\Omega \omega^\mathsf{W} \right\rangle$$

As a consequence, by Theorem 5.4, the specialization preorder $\leq_{k^*\omega^\mathsf{W}}$ (the specialization equivalence $\equiv_{k^*\omega^\mathsf{W}}$) is a precongruence (resp. congruence) on $\psi : \Sigma T0 \to T0$ (see Definition 2.17). The theorem follows from Corollary 4.9. □

This result expresses congruence formats for relations from the van Glabbeek spectrum in an abstract fashion. In the following sections, we provide concrete syntactic restrictions on GSOS specifications that ensure these abstract requirements for $\mathsf{W} = \{\mathsf{Tr}, \mathsf{CTr}, \mathsf{Fl}\}$.

5.3 Trace Semantics

In this section, based on Corollary 5.5 specialized to the case of $\mathsf{W} = \mathsf{Tr}$, we show a congruence format for trace preorder and equivalence.

Format 5.6 (Tr-format) An image finite set of GSOS rules Λ is in Tr-*format*, if for each $\rho \in \Lambda$:

- all premises of ρ are positive,
- no variable occurs more than once in the left-hand sides of premises and in the target.

It is easy to see that this format coincides with the well-known de Simone format [80]. The fact that this syntactic format ensures the trace preorder to be a precongruence was first proved in [86].

The following standard example, on which all examples in the following sections will be based, is taken from [5].

Example 5.7 Assuming a finite set A of actions, the syntax Σ of the *basic process algebra* **BPA** is defined by the BNF grammar

$$t ::= \mathtt{0} \mid \alpha t \mid t{+}t$$

and the transition system specification **BPA** over Σ is a collection of rules

$$\frac{}{\alpha \mathtt{x} \xrightarrow{\alpha} \mathtt{x}} \qquad \frac{\mathtt{x} \xrightarrow{\alpha} \mathtt{x}'}{\mathtt{x}+\mathtt{y} \xrightarrow{\alpha} \mathtt{x}'} \qquad \frac{\mathtt{y} \xrightarrow{\alpha} \mathtt{y}'}{\mathtt{x}+\mathtt{y} \xrightarrow{\alpha} \mathtt{y}'}$$

where α ranges over A. When presenting terms over the above syntax, the trailing $\mathtt{0}$'s are omitted. It is easy to see that **BPA** is in the image finite GSOS format and in the Tr-format.

The following theorem, together with Corollary 5.5, shows that Tr-format is a precongruence format for trace preorder and a congruence format for trace equivalence.

Theorem 5.8 Let Λ be an image finite GSOS specification and $\lambda : \Sigma(\mathrm{Id}\times B) \to BT$ the corresponding natural transformation. If Λ is in Tr-format, then

$$\lambda : \Sigma^*(\mathrm{Id} \times B^{\mathsf{Tr}}) \to B^{\mathsf{Tr}} T^*$$

is a natural transformation in 2-**TS**.

Proof. It is enough to ensure that given an object $\langle X, \theta\rangle$ in 2-**TS**,

$$\lambda_X : \Sigma^*(\langle X, \theta\rangle \times B^{\mathsf{Tr}}\,\langle X, \theta\rangle) \to B^{\mathsf{Tr}} T^*\,\langle X, \theta\rangle$$

is a morphism in 2-**TS**; in other words, that for every test $V \in B^{\mathsf{Tr}}_{TX} T_X \theta$, the test $V \circ \lambda_X$ is an element of the test suite $\Sigma_{X\times BX}(\theta \boxtimes B^{\mathsf{Tr}}_X \theta)$.

To do this, it will be useful to understand the nature of tests in $\Sigma_X\theta$, $T_X\theta$ and $B^{\mathsf{Tr}}_{TX} T_X \theta$, for a given $\langle X, \theta\rangle$.

Definition 5.9 For a polynomial endofunctor Σ, a set X and a test suite $\theta : X \rightrightarrows 2$, a *basic flat θ-check* on ΣX is a term

$$\gamma \in \Sigma\theta$$

A term $t \in \Sigma X$ *passes* a basic flat θ-check γ, if t can be obtained from γ by replacing every $V \in \theta$ by some $x \in V$.

The test $\upsilon(\gamma) : \Sigma X \to 2$ corresponding to a basic flat θ-check γ is defined by

$$\upsilon(\gamma)t = \begin{cases} \mathtt{tt} & \text{if } t \text{ passes } \gamma \\ \mathtt{ff} & \text{otherwise} \end{cases}$$

Tests of this kind will be called *basic flat θ-tests* on ΣX.

Lemma 5.10 For any Σ, $\langle X, \theta\rangle$, tests from $\Sigma_X\theta$ are exactly all unions of basic flat θ-tests on ΣX.

Proof. Recall Definition 5.3 and take any test

$$V \in \theta^{\bowtie n_1} \boxplus \cdots \boxplus \theta^{\bowtie n_k}$$

By Lemma 5.1, $V = [V_1, \cdots, V_k]$, where $V_i \in \theta^{\bowtie n_i}$. Then for every $1 \leq i \leq k$,

$$V_i = \wedge \circ (U_1 \times \cdots \times U_{n_i}) \text{ where } U_j \in \theta$$

Note that $\gamma_i = \mathtt{f}_i \langle U_1, \ldots, U_{n_i} \rangle$ is a basic flat τ-check, and

$$\upsilon(\gamma_i) = [\overrightarrow{\emptyset}, \overrightarrow{\emptyset}, \ldots, V_i, \ldots, \overrightarrow{\emptyset}] \qquad (V_i \text{ on the } i\text{-th place})$$

where $\mathtt{f}_i : X^{n_i} \to \Sigma X$ is the i-th coproduct injection into ΣX. Then

$$V = [V_1, V_2, \ldots, V_k] = \upsilon(\gamma_1) \vee \upsilon(\gamma_2) \vee \cdots \vee \upsilon(\gamma_k)$$

which completes the presentation of V as a union of basic flat θ-tests.

Also any union of basic flat θ-tests belongs to $\Sigma_X \theta$. Indeed, for any basic flat θ-check $\gamma = \mathtt{f}_i \langle U_1, \ldots, U_{n_i} \rangle$ one has

$$\upsilon(\gamma) = [\overrightarrow{\emptyset}, \overrightarrow{\emptyset}, \ldots, \wedge \circ (U_1 \times \cdots \times U_{n_i}), \ldots, \overrightarrow{\emptyset}] \in \Sigma_X \theta$$

and closure under unions is guaranteed by the definition of $\Sigma_X \theta$. □

Definition 5.11 For a polynomial endofunctor Σ with the freely generated monad T, a set X and a test suite $\theta : X \rightrightarrows 2$, a *basic term θ-check* on TX is a term

$$\gamma \in T\theta$$

A term $t \in TX$ *passes* a basic term θ-check γ, if t can be obtained from γ by replacing every $V \in \theta$ by some $x \in V$.

The 2-test $\upsilon(\gamma)$ on TX corresponding to a basic term θ-check γ is defined by

$$\upsilon(\gamma)t = \begin{cases} \mathtt{tt} & \text{if } t \text{ passes } \gamma \\ \mathtt{ff} & \text{otherwise} \end{cases}$$

Tests of this kind will be called *basic term θ-tests* on ΣX.

Lemma 5.12 For any Σ, $\langle X, \theta \rangle$, every test in $T_X \theta$ is a union of basic term θ-tests on TX.

Proof. Recall from Theorem 3.9 that $T_X \theta$ is the *greatest* (with respect to set inclusion, see Remark 3.23) fixed point of the operator Ψ:

$$\Psi\nu = [\psi_X, \eta_X]_!(\Sigma_{TX}\nu \boxplus \theta)$$

where $\psi_X : \Sigma TX \to TX$ and $\eta_X : X \to TX$ arise from the free monad structure of T.

Note that in general, the closure $\mathrm{Cl}^\vee$ in the definition of Σ^* does *not* preserve intersections of decreasing ω-chains of test suites. Therefore, it is not immediately clear whether $T_X\theta$ can be characterized as

$$T_X\theta = \bigcap_{n\in\mathbb{N}} \Phi^n(\mathbf{Set}(TX, 2))$$

However, this characterization is not needed here. For our purposes, it is enough to observe that

$$T_X\theta \subseteq \bigcap_{n\in\mathbb{N}} \Phi^n(\mathbf{Set}(TX, 2))$$

and this is a general property of the greatest fixed points of monotonic functions on complete lattices.

The proof proceeds by constructing for all $n \in \mathbb{N}$ and for any test $V \in \Psi^n(\mathbf{Set}(TX, 2))$, a family $\Gamma_n(V)$ of basic term θ-checks from $T_n\theta$ such that

$$V \wedge \overrightarrow{T_nX} = \bigvee_{\gamma\in\Gamma_n(V)} \upsilon(\gamma)$$

where T_nX denotes the set of Σ-terms of depth at most n.

Given this construction, for any test $V \in T_X\theta \subseteq \bigcap_{n\in\mathbb{N}} \Psi^n(\mathbf{Set}(TX, 2))$ one has

$$V = V \wedge (\bigvee_{n\in\mathbb{N}} \overrightarrow{T_nX}) = \bigvee_{n\in\mathbb{N}} (V \wedge \overrightarrow{T_nX}) = \bigvee_{n\in\mathbb{N}} \bigvee_{\gamma\in\Gamma_n(V)} \upsilon(\gamma)$$

which will complete the proof.

The families $\Gamma_n(V)$ are constructed by induction on n. For the base case, take $\Gamma_0(V) = \emptyset$ for any V. Indeed,

$$V \wedge \overrightarrow{T_0X} = V \wedge \overrightarrow{\emptyset} = \overrightarrow{\emptyset} = \bigvee_{\gamma\in\emptyset} \upsilon(\gamma)$$

For the induction step, take a test $V \in \Psi^n(\mathbf{Set}(TX, 2))$. By definition of Ψ and $[\psi_X, \eta_X]_!$,

$$V \circ [\psi_X, \eta_X] \in \Sigma_{TX}\Psi^{n-1}(\mathbf{Set}(TX, 2)) \boxplus \theta$$

therefore, by definition of $\boxplus$, one has $V\circ\eta_X \in \theta$ and $V\circ\psi_X \in \Sigma_{TX}\Psi^{n-1}(\mathbf{Set}(TX, 2))$.

Note that $\eta_\theta(V \circ \eta_X) \in T\theta$ is a basic term θ-check on X, and that

$$\upsilon(\eta_\theta(V \circ \eta_X)) = [\overrightarrow{\emptyset}, V \circ \eta_X]$$

By Lemma 5.10, for some indexing set I,

$$V \circ \psi_X = \bigvee_{i\in I} \upsilon(\delta_i)$$

where each δ_i is a basic flat $\Psi^{n-1}(\mathbf{Set}(TX, 2))$-check on ΣTX, i.e.

$$\delta_i \in \Sigma(\Psi^{n-1}(\mathbf{Set}(TX, 2)))$$

For every $i \in I$, assume without any loss of generality that $\delta_i = \mathtt{f}\,\langle U_1, \ldots, U_m\rangle$ for some $\mathtt{f} : X^m \to \Sigma X$, $m \in \mathbb{N}$ and $U_j \in \Psi^{n-1}(\mathbf{Set}(TX, 2))$ for $j = 1, \ldots, m$.

By the inductive assumption, for each U_j there exist a family $\Gamma_{n-1}(U_j) \subseteq T_{n-1}\theta$ of basic term θ-checks such that

$$\upsilon(U_j) \wedge \overrightarrow{T_{n-1}X} = \bigvee_{u \in \Gamma_{n-1}(U_j)} \upsilon(u)$$

for $j = 1, \ldots, m$.

Consider the family of basic term θ-checks

$$\Theta_i = \{\, \mathtt{f}\,\langle u_1, \ldots, u_m\rangle \in T\theta \,:\, u_j \in \Gamma_{n-1}(U_j) \text{ for } j = 1, \ldots, m \,\} \subseteq T_n\theta$$

and calculate

$$\begin{aligned}
\bigcup_{\gamma \in \Theta_i} \overline{\upsilon(\gamma)} &= \Big\{ \mathtt{f}\,\langle t_1, \ldots, t_m\rangle \in T\theta \,:\, \text{for } j = 1, \ldots, m, \\
&\qquad\qquad t_j \in \overline{\upsilon(u_j)} \text{ for some } u_j \in \Gamma_{n-1}(U_j) \Big\} \\
&= \left\{ \mathtt{f}\,\langle t_1, \ldots, t_m\rangle \in T\theta \,:\, \text{for } j = 1, \ldots, m, \quad t_j \in \bigcup_{u \in \Gamma_{n-1}(U_j)} \overline{\upsilon(u)} \right\} \\
&= \left\{ \mathtt{f}\,\langle t_1, \ldots, t_m\rangle \in T\theta \,:\, \text{for } j = 1, \ldots, m, \quad t_j \in \overline{\upsilon(U_j)} \cap T_{n-1}X \right\} \\
&= \left\{ \mathtt{f}\,\langle t_1, \ldots, t_m\rangle \in T\theta \,:\, \text{for } j = 1, \ldots, m, \quad t_j \in \overline{\upsilon(U_j)} \right\} \cap T_nX \\
&= \overline{[\upsilon(\delta_i),\, \overrightarrow{\emptyset}\,]} \cap T_nX
\end{aligned}$$

Now take

$$\Gamma_n(V) = \bigcup_{i \in I} \Theta_i \cup \{\eta_\theta(V \circ \eta_X)\}$$

and check that

$$\begin{aligned}
V \wedge \overrightarrow{T_nX} &= [V \circ \psi_X, V \circ \eta_X] \wedge \overrightarrow{T_nX} \\
&= ([V \circ \psi_X,\, \overrightarrow{\emptyset}\,] \wedge \overrightarrow{T_nX}) \vee [\,\overrightarrow{\emptyset}\,, V \circ \eta_X] \\
&= \Big(\bigvee_{i \in I} [\upsilon(\delta_i),\, \overrightarrow{\emptyset}\,] \wedge \overrightarrow{T_nX} \Big) \vee \upsilon(\eta_\theta(V \circ \eta_X)) \\
&= \Big(\bigvee_{i \in I} \bigvee_{\gamma \in \Theta_i} \upsilon(\gamma) \Big) \vee \upsilon(\eta_\theta(V \circ \eta_X)) \\
&= \bigvee_{\gamma \in \Gamma_n(V)} \upsilon(\gamma)
\end{aligned}$$

This completes the induction step, and the proof of Lemma 5.12. □

Definition 5.13 For a test suite $\theta : X \rightrightarrows 2$, a *positive θ-check* on BX is an expression of the form $\xrightarrow{a} V$, where $a \in A$ and $V \in \theta$. A set $\beta \in BX$ *passes* such a check, if there is some $\langle a, x\rangle \in \beta$ such that $x \in V$. As in Definition 5.9,

the 2-test on BX associated to a positive θ-check c is denoted $\upsilon(c)$ and is called a *positive θ-test*.

Similarly, for a given test suite θ on X, a *positive term θ-check* on BTX is an expression of the form $\xrightarrow{a} \gamma$, where $a \in A$ and γ is a basic term θ-check on TX. The definition of passing and of the *positive term θ-test* $\upsilon(\xrightarrow{a} \gamma)$ is as above.

Lemma 5.14 A test in $B^{\mathsf{Tr}}_{TX} T_X \theta$ is either the always true test $\overrightarrow{BTX}$, or a union of positive term θ-tests.

Proof. By Definition of B^{Tr}, if a test $V \in B^{\mathsf{Tr}}_{TX} T_X \theta$ is not equal to $\overrightarrow{BTX}$, then

$$\overline{V} = \overline{w_{\langle a \rangle} \circ BV'} = \left\{ \beta \in BTX \,:\, \langle a, t \rangle \in \beta \text{ for some } t \in \overline{V'} \right\}$$

for some $a \in A$ and $V' \in T_X \theta$. But then, by Lemma 5.12, V' is a union of basic term θ-tests:

$$V' = \bigvee_{i \in I} \upsilon(\gamma_i)$$

hence

$$\overline{V} = \bigcup_{i \in I} \left\{ \beta \in BTX \,:\, \langle a, t \rangle \in \beta, t \in \overline{\upsilon(\gamma_i)} \right\} = \bigcup_{i \in I} \overline{\upsilon(\xrightarrow{a} \gamma_i)}$$

□

We are now ready to prove Theorem 5.8. By Lemmas 5.14 and 5.10, it is enough to show that for any set X and test suite $\theta : X \rightrightarrows 2$, and for any positive term θ-check $\xrightarrow{a} \gamma$ on BTX, there exists a family $\{ \delta_i : i \in I \}$ of basic flat $(\theta \boxtimes B^{\mathsf{Tr}}_X \theta)$-checks such that

$$\upsilon(\xrightarrow{a} \gamma) \circ \lambda_X = \bigvee_{i \in I} \upsilon(\delta_i)$$

To achieve this, fix a positive term θ-check $\xrightarrow{a} \gamma$. As shown in the proof of Theorem 2.26, λ_X is obtained from functions $f_X : (X \times BX)^n \to BTX$ corresponding to coproduct injections $\mathtt{f} : X^n \to \Sigma X$. A function f_X for a given $\mathtt{f}$ is in turn obtained from functions $\rho_X : (X \times BX)^n \to BTX$ induced from rules $\rho \in \Lambda$ for $\mathtt{f}$.

For any $\mathtt{f} \in \bar{\Sigma}$, and for any rule $\rho \in \Lambda$ with the conclusion of the form

$$\mathtt{f}(\mathtt{x}_1, \dots \mathtt{x}_n) \xrightarrow{c} \mathtt{t}$$

(where $c \in A$, $\mathtt{t} \in T\Xi$ and n is the arity of $\mathtt{f}$), we will construct a family of basic flat $(\theta \boxtimes B^{\mathsf{Tr}}_X \theta)$-checks δ_i such that

$$\upsilon(\xrightarrow{a} \gamma) \circ [\varnothing, \varnothing, \dots, \rho_X, \dots, \varnothing] = \bigvee_{i \in I} \upsilon(\delta_i)$$

where $\varnothing$ denotes the function to BTX constantly equal $\emptyset$. By construction of λ_X from Theorem 2.26, this will complete the proof of Theorem 5.8.

Consider two cases:

a) $c \neq a$ or $\mathtt{t}$ cannot be obtained from γ by replacing each $V \in \theta$ with some $x \in \Xi$. This means that the set

$$\rho_X(\langle\langle x_1, \beta_1\rangle, \ldots, \langle x_n, \beta_n\rangle\rangle)$$

does *not* pass the check $\xrightarrow{a}\blacktriangleright t$ for any arguments $x_1, \ldots, x_n \in X$ and $\beta_1, \ldots, \beta_n \in BX$. Therefore it is enough to take $I = \emptyset$ and the empty family of checks.

b) $c = a$ and $\mathtt{t}$ can be obtained from γ by replacing each $V \in \theta$ with some $x \in \Xi$. Since by definition of Tr-format no variable occurs in $\mathtt{t}$ more than once, this gives a function $\varsigma : \Xi \to \theta$ such that $\gamma = \mathtt{t}\varsigma$. Without loss of generality, assume that $\varsigma(\mathtt{x}) = \overrightarrow{X}$ if $\mathtt{x}$ does not occur in $\mathtt{t}$.

Now for each $1 \leq i \leq n$ construct a test $V_i \in \theta \boxtimes B_X\theta$ as follows:

- If $\mathtt{x}_i$ does not occur in any premise of ρ, then take $V_i = \varsigma(\mathtt{x}_i) \circ \pi_1$.
- If $\mathtt{x}_i$ occurs on the left side of a positive premise $\mathtt{x}_i \xrightarrow{b_i} \mathtt{y}_i$, then take $V_i = \upsilon(\xrightarrow{b_i}\blacktriangleright \varsigma(\mathtt{y}_i)) \circ \pi_2$.

Note that the syntactic restrictions of the Tr-format ensure that the above definition is complete and unambiguous.

Having defined the tests V_i, consider a basic flat $\theta \boxtimes B_X\theta$-check

$$\delta_\rho = \mathtt{f}\,\langle V_1, \ldots, V_n\rangle$$

We will show that

$$\upsilon(\xrightarrow{a}\blacktriangleright \gamma) \circ [\varnothing, \varnothing, \ldots, \rho_X, \ldots, \varnothing] = \upsilon(\delta_\rho)$$

To this end, take any

$$r = \langle\langle x_1, \beta_1\rangle, \ldots, \langle x_n, \beta_n\rangle\rangle \in (X \times BX)^n$$

and check that the following are equivalent:

$\upsilon(\delta_\rho)(\mathtt{f}r) = \mathtt{tt}$

$\iff$ $V_i\,\langle x_i, \beta_i\rangle = \mathtt{tt}$ for all $1 \leq i \leq n$

$\iff$ For each $1 \leq i \leq n$, either $\mathtt{x}_i$ does not occur in any premise and $x_i \in \varsigma(\mathtt{x}_i)$, or $\mathtt{x}_i$ occurs in a premise $\mathtt{x}_i \xrightarrow{b_i} \mathtt{y}_i$ and $\langle b_i, y_i\rangle \in \beta_i$ for some $y_i \in \varsigma(\mathtt{y}_i)$

$\iff$ The function $\sigma : \Xi \to X$ mapping each $\mathtt{x}_i$ to x_i, and $\mathtt{y}_i$ to y_i (note that this definition is unambiguous, due to syntactic restrictions of GSOS format) satisfies the first three of the four conditions described in the proof of Theorem 2.26 (in particular, due to syntactic restrictions of Tr-format, all $n_i = 0$ and all $m_i \in \{0, 1\}$). Moreover, $\sigma(\mathtt{x}) \in \varsigma(\mathtt{x})$ for all $\mathtt{x} \in \Xi$.

$\iff$ $\langle a, \mathtt{t}\sigma\rangle \in \rho_X r$. Moreover, $\upsilon(\gamma)(\mathtt{t}\sigma) = \mathtt{tt}$.

$\iff$ $\upsilon(\xrightarrow{a}\blacktriangleright \gamma)(\rho_X r) = \mathtt{tt}$.

This completes the proof of Theorem 5.8. □

5.4 Completed Trace Semantics

In this section, based on Corollary 5.5 specialized to the case of $\mathsf{W} = \mathsf{CTr}$, we show a congruence format for trace preorder and equivalence. Most proofs are extended versions of analogous proofs from Section 5.3.

First, some useful technical definitions.

Definition 5.15 Let Λ be a set of GSOS rules with the same source $\mathtt{f}(\mathtt{x}_1, \ldots, \mathtt{x}_n)$. A *minimal blocking set* for Λ is a set $\mathbb{B}$ of literals obtained by choosing a single premise from each rule in Λ.

Note that if some rules in Λ have no premises then Λ has no minimal blocking sets.

Definition 5.16 A set $\mathbb{B}$ of literals is a CTr-*blocking set* if either:

- for some $\mathtt{x}, \mathtt{y} \in \Xi$, $a \in A$, both $\mathtt{x} \xrightarrow{a} \mathtt{y}$ and $\mathtt{x} \not\xrightarrow{a}$ belong to $\mathbb{B}$, or
- both conditions below hold:
  - for every $\mathtt{x}, \mathtt{y} \in \Xi$, $a \in A$, if $\mathtt{x} \xrightarrow{a} \mathtt{y} \in \mathbb{B}$ then for every $b \in A$ there is some $\mathtt{y}' \in \Xi$ such that $\mathtt{x} \xrightarrow{b} \mathtt{y}' \in \mathbb{B}$, and
  - for every $\mathtt{x} \in \Xi$, $a \in A$, if $x \not\xrightarrow{a} \in \mathbb{B}$ then for every $b \in A$ different from a, $x \not\xrightarrow{b} \notin \mathbb{B}$.

If the first of the above conditions does not hold, the blocking set is called *satisfiable*.

In the remainder of this chapter, $\Lambda_{\mathtt{f}}$ denotes the set of all rules from Λ (Λ will be always understood from context) for the construct $\mathtt{f}$, with possibly renamed variables so that they all have the same source. Similarly, for a set $Q \subseteq A$, by $\Lambda_{\mathtt{f}Q}$ we denote those rules in $\Lambda_{\mathtt{f}}$ which have an element of Q as the action in the rule conclusion. Obviously $\Lambda_{\mathtt{f}A} = \Lambda_{\mathtt{f}}$.

Format 5.17 A set of image finite GSOS rules Λ is in CTr-*format*, if:

1. For each rule $\rho \in \Lambda$:
   - if ρ has a negative premise $\mathtt{x} \not\xrightarrow{a}$, then for every label $b \in A$, ρ has also the negative premise $\mathtt{x} \not\xrightarrow{b}$,
   - no variable occurs more than once in the target of ρ,
   - no variable occurs simultaneously in the left-hand side of a premise and in the target of ρ,
   - no variable occurs simultaneously in the left-hand side of a positive premise and in the left-hand side of any other premise of ρ.
2. For each construct $\mathtt{f}$ of the language, every minimal blocking set for $\Lambda_{\mathtt{f}}$ is a CTr-blocking set.

Proposition 5.18 **BPA** (see Example 5.7) is in CTr-format.

Proof. It is clear that all rules of **BPA** satisfy condition 1 of CTr-format. For condition 2, consider a language construct $a-$ corresponding to $a \in A$. Since the only rule for $a-$:

$$\frac{}{a\mathtt{x} \xrightarrow{a} \mathtt{x}}$$

has no premises, it does not have any minimal blocking set. For the binary construct $+$, the only minimal blocking set for Λ_+ is

$$\left\{ \mathtt{x} \xrightarrow{a} \mathtt{x}' \,:\, a \in A \right\} \cup \left\{ \mathtt{y} \xrightarrow{a} \mathtt{y}' \,:\, a \in A \right\}$$

and it is a CTr-blocking set. □

The following example is taken from [5].

Example 5.19 Assume $A = \{a, b\}$, and extend **BPA** with an operational rule for the *encapsulation operator* $\partial_{\{b\}}$:

$$\frac{\mathtt{x} \xrightarrow{a} \mathtt{y}}{\partial_{\{b\}}(\mathtt{x}) \xrightarrow{a} \partial_{\{b\}}(\mathtt{y})}$$

It is easy to check that the above extension of **BPA** does not respect completed traces. Indeed, $aa + ab \cong_{\mathsf{CTr}} a(a + b)$ but that $\partial_{\{b\}}(aa + ab) \not\sqsubseteq_{\mathsf{CTr}} \partial_{\{b\}}(a(a + b))$, since $\langle a \rangle \tilde{A}$ is a completed trace of $\partial_{\{b\}}(aa + ab)$, but not of $\partial_{\{b\}}(a(a + b))$.

Proposition 5.20 The semantics for the encapsulation operator ∂ is not in CTr-format.

Proof. The rule for the encapsulation operator fails to satisfy condition 2 of CTr-format. Indeed, the set $\{\mathtt{x} \xrightarrow{a} \mathtt{y}\}$ is a minimal blocking set for $\Lambda_{\partial_{\{b\}}}$, but it is not a CTr-blocking set. □

Another example of simple GSOS rules that do not respect completed traces was described in [86]. Together with Example 5.19, it led the authors of [5] to speculate that one cannot hope for a general syntactic congruence format for completed trace equivalence:

Example 5.21 Assume $A = \{a, b\}$, and extend **BPA** with a collection of rules for binary *synchronous composition* $\times$:

$$\frac{\mathtt{x} \xrightarrow{\alpha} \mathtt{x}' \quad \mathtt{y} \xrightarrow{\alpha} \mathtt{y}'}{\mathtt{x} \times \mathtt{y} \xrightarrow{\alpha} \mathtt{x}' \times \mathtt{y}'}$$

where α ranges over A.

Here it is easy to see that $aa \times (aa + ab) \not\sqsubseteq_{\mathsf{CTr}} aa \times a(a + b)$, since $\langle a \rangle \tilde{A}$ is a completed trace of $aa \times (aa + ab)$, but not of $aa \times (a(a + b))$.

Proposition 5.22 The semantics for the synchronous composition is not in CTr-format.

Proof. The rules for synchronous composition fail to satisfy condition 2 of CTr-format. Indeed, the set $\{\mathtt{x} \xrightarrow{a} \mathtt{x}', \mathtt{y} \xrightarrow{b} \mathtt{y}'\}$ is a minimal blocking set for $\Lambda_\times$, but it is not a CTr-blocking set. □

A non-trivial example of a transition system specification in CTr-format is that of sequential composition.

Example 5.23 Extend **BPA** with a collection of rules for binary *sequential composition* ;:

$$\frac{\mathtt{x} \xrightarrow{\alpha} \mathtt{x}'}{\mathtt{x};\mathtt{y} \xrightarrow{\alpha} \mathtt{x}';\mathtt{y}} \qquad \frac{\mathtt{x} \not\xrightarrow{a} \text{ for all } a \in A \quad \mathtt{y} \xrightarrow{\alpha} \mathtt{y}'}{\mathtt{x};\mathtt{y} \xrightarrow{\alpha} \mathtt{y}'}$$

where α ranges over A.

Proposition 5.24 BPA extended with sequential composition is in CTr-format.

Proof. Condition 1 of CTr-format is checked easily. For condition 2, by Proposition 5.18, it is enough to check it for the sequential composition operator. First, observe that for any minimal blocking set $\mathbb{B}$ for $\Lambda_;$ one has

$$\left\{ \mathtt{x} \xrightarrow{a} \mathtt{x}' \,:\, a \in A \right\} \subseteq \mathbb{B}$$

Then realize that the only minimal blocking set $\mathbb{B}$ that does not contain $\mathtt{x} \not\xrightarrow{a}$ for any $a \in A$ (and if it does, it is necessarily CTr-blocking) is

$$\left\{ \mathtt{x} \xrightarrow{a} \mathtt{x}' \,:\, a \in A \right\} \cup \left\{ \mathtt{y} \xrightarrow{a} \mathtt{y}' \,:\, a \in A \right\}$$

which is also CTr-blocking. □

CTr-format cannot be compared with any other congruence format for completed trace equivalence, since it is, to the author's best knowledge, the first such format published. Note, however, that there are natural examples of GSOS specifications that behave well with respect to completed trace semantics, but are not in CTr-format, as the following example (pointed out by Rob van Glabbeek) shows.

Example 5.25 Extend Example 5.23 with a collection of rules for binary *Kleene star* $\star$:

$$\frac{\mathtt{x} \xrightarrow{\alpha} \mathtt{x}'}{\mathtt{x} \star \mathtt{y} \xrightarrow{\alpha} \mathtt{x}';(\mathtt{x} \star \mathtt{y})} \qquad \frac{\mathtt{y} \xrightarrow{\alpha} \mathtt{y}'}{\mathtt{x} \star \mathtt{y} \xrightarrow{\alpha} \mathtt{y}'}$$

It is straightforward to check that completed trace equivalence is a congruence for the above rules. However,

Proposition 5.26 BPA extended with the Kleene star is not in CTr-format.

Proof. The first rule in Example 5.25 does not satisfy the third part of condition 1 of CTr-format. Indeed, the variable $\mathtt{x}$ occurs both on the left side of the premise and in the target of the rule. □

The following theorem, together with Corollary 5.5, shows that CTr-format is a precongruence format for completed trace preorder and a congruence format for completed trace equivalence.

Theorem 5.27 Let Λ be a GSOS specification and $\lambda : \Sigma(\mathrm{Id} \times B) \to BT$ the corresponding natural transformation. If Λ is in CTr-format, then

$$\lambda : \Sigma^*(\mathrm{Id} \times B^{\mathsf{CTr}}) \to B^{\mathsf{CTr}} T^*$$

is a natural transformation in 2-**TS**.

Proof. Similarly as in the proof of Theorem 5.8, we show that for any 2-test suite θ on a set X, for every test $V \in B_{TX}^{\mathsf{CTr}} T_X \theta$ there exists a test $V' \in \Sigma_X(\theta \bowtie B_X^{\mathsf{CTr}}\theta)$ such that $V \circ \lambda_X = V'$.

To this end, recall the characterization of tests in $\Sigma_X\theta$ and $T_X\theta$ given in Lemmas 5.10 and 5.12.

Definition 5.28 For any set $Q \subseteq A$, the *Q-failure check* is denoted $\xrightarrow{Q}\!\!\!\!\!\!\!\!\diagup\!\!\blacktriangleright$. Instead of $\xrightarrow{A}\!\!\!\!\!\!\!\!\diagup\!\!\blacktriangleright$, we will often write $\longrightarrow\!\!\!\!\!\!\!\!\diagup\!\!\blacktriangleright$. For any set X, a set $\beta \in BX$ *passes* the Q-failure check if

$$\beta \cap \{\, \langle a, x\rangle \,:\, a \in Q, x \in X \,\} = \emptyset$$

For any X, the test on BX associated to the Q-failure check will be denoted $\upsilon(\xrightarrow{Q}\!\!\!\!\!\!\!\!\diagup\!\!\blacktriangleright)$, and will be called the *Q-failure test*.

Lemma 5.29 A test $V \in B_{TX}^{\mathsf{CTr}} T_X \theta$ is either the always true test $\overrightarrow{BTX}$, or the A-failure test $\upsilon(\longrightarrow\!\!\!\!\!\!\!\!\diagup\!\!\blacktriangleright)$, or a union of positive term θ-tests.

Proof. By definition of B^{CTr} (see Definition 4.5), if a test $V \in B_{TX}^{\mathsf{CTr}} T_X \theta$ is not equal to $\overrightarrow{BTX}$, then either $V = w_{\langle a\rangle} \circ BV'$ for some $V' \in T_X\theta$, or $V = \tilde{w}_A \circ BV'$ for some $V' \in T_X\theta$. In the former case, V is a union of positive term θ-tests, as shown in Lemma 5.14. In the latter case,

$$\tilde{w}_A \circ BV' = \overrightarrow{\{\emptyset\}} = \upsilon(\longrightarrow\!\!\!\!\!\!\!\!\diagup\!\!\blacktriangleright)$$

□

To prove Theorem 5.27, it is enough to consider single positive term θ-tests on BTX, and the A-failure test on BTX. For the positive term tests, proceed as in the proof of Theorem 5.8, except that when constructing the tests V_i in case b) consider one additional case:

- If $\mathtt{x}_i$ occurs in ρ in a negative premise $\mathtt{x}_i \not\xrightarrow{a}$ (and hence, it occurs in a premise $\mathtt{x}_i \not\xrightarrow{b}$ for any $b \in A$), then take $V_i = \upsilon(\longrightarrow\!\!\!\!\!\!\!\!\diagup\!\!\blacktriangleright) \circ \pi_2$.

The syntactic restrictions of condition 1 of CTr-format ensure that the definition of V_i's extended this way is complete and unambiguous.

The final reasoning following the definition of the V_i's is changed accordingly:

$$\upsilon(\delta_\rho)(\mathtt{f}r) = \mathtt{tt}$$

$\iff$ $V_i \langle x_i, \beta_i \rangle = \mathtt{tt}$ for all $1 \leq i \leq n$

$\iff$ For each $1 \leq i \leq n$, either $\mathtt{x}_i$ does not occur in any premise and $x_i \in \varsigma(\mathtt{x}_i)$, or $\mathtt{x}_i$ occurs in a premise $\mathtt{x}_i \xrightarrow{b_i} \mathtt{y}_i$ and $\langle b_i, y_i \rangle \in \beta_i$ for some $y_i \in \varsigma(\mathtt{y}_i)$, or $\mathtt{x}_i$ occurs in negative premises $\mathtt{x_i} \not\xrightarrow{a}$ for all $a \in A$ and $\beta_i = \emptyset$

$\iff$ The function $\sigma : \Xi \to X$ mapping each $\mathtt{x}_i$ to x_i, and $\mathtt{y}_i$ to y_i (note that this definition is unambiguous, due to syntactic restrictions of the GSOS format) satisfies the first three of the four conditions described in the proof of Theorem 2.26 (in particular, due to syntactic restrictions of CTr-format, all $n_i \in \{0, |A|\}$ and all $m_i \in \{0, 1\}$). Moreover, $\sigma(\mathtt{x}) \in \varsigma(\mathtt{x})$ for all $\mathtt{x} \in \Xi$.

$\iff$ $\langle a, \mathtt{t}\sigma \rangle \in \rho_X r$. Moreover, $\upsilon(\gamma)(\mathtt{t}\sigma) = \mathtt{tt}$.

$\iff$ $\upsilon(\xrightarrow{a}\!\!\blacktriangleright\, \gamma)(\rho_X r) = \mathtt{tt}$.

The rest of the argument remains as in the case of Tr-format.

For the A-failure test $\upsilon(\not\blacktriangleright)$ on BTX, for any language construct $\mathtt{f}(\mathtt{x}_1, \ldots, \mathtt{x}_n)$ take any minimal blocking set $\mathbb{B}$ for $\Lambda_\mathtt{f}$. By condition 2 of the CTr-format, $\mathbb{B}$ is a CTr-blocking set. Assume that $\mathbb{B}$ is satisfiable. For each $i = 1, \ldots, n$, define a test $V_i \in \theta \boxtimes B_X^{\mathsf{CTr}}\theta$ as follows:

- If for no $a \in A$, $\mathtt{x}_i \xrightarrow{a} \mathtt{y}$ nor $\mathtt{x}_i \not\xrightarrow{a}$ belong to $\mathbb{B}$, then take $V_i = \overrightarrow{BX} \circ \pi_2$.
- If, for all $a \in A$, $\mathtt{x}_i \xrightarrow{a} \mathtt{y} \in \mathbb{B}$, and if for all $b \in A$, $\mathtt{x}_i \not\xrightarrow{b} \notin \mathbb{B}$, then take $V_i = \upsilon(\not\blacktriangleright) \circ \pi_2$.
- If, for all $a \in A$, $\mathtt{x}_i \xrightarrow{a} \mathtt{y} \notin \mathbb{B}$, and if for some $b \in A$, $\mathtt{x}_i \not\xrightarrow{b} \in \mathbb{B}$, then take $V_i = \upsilon(\xrightarrow{b}\!\!\blacktriangleright\, \overrightarrow{X}) \circ \pi_2$.

Note that the definition of a CTr-blocking set ensures that the above definition is complete and unambiguous.

Now consider the basic flat $\theta \boxtimes B_X^{\mathsf{CTr}}\theta$-check

$$\delta_{\mathtt{f}\mathbb{B}} = \mathtt{f} \langle V_i, \ldots, V_i \rangle$$

Recall the definition of the function f_X in the proof of Theorem 2.26. We will now show that

$$\bigvee_{\mathbb{B}} \upsilon(\delta_{\mathtt{f}\mathbb{B}}) = \upsilon(\not\blacktriangleright) \circ f_X$$

where the union occurs over all satisfiable minimal blocking sets for $\Lambda_\mathtt{f}$.

To this end, consider any $r \in \Sigma(X \times BX)$ of the form

$$r = \mathtt{f}\left\langle \langle x_1, \beta_1 \rangle, \ldots, \langle x_n, \beta_n \rangle \right\rangle$$

and check that

$\upsilon(\not\blacktriangleright) f_X r = \mathtt{tt}$

$\iff$ $f_X r = \emptyset$

$\iff$ There is no substitution $\sigma : \Xi \to X$ such that for some rule $\rho \in \Lambda_{\mathtt{f}}$,

$$\begin{array}{l} \sigma \mathtt{x}_i = x_i \\ \forall i \leq n \forall j \leq m_i . \langle a_{ij}, \sigma(\mathtt{y}_{ij}) \rangle \in \beta_i \\ \forall i \leq n \forall j \leq n_i \forall x \in X . \langle b_{ij}, x \rangle \notin \beta_i \end{array}$$

where $m_i, n_i, a_{ij}, b_{ij}, \mathtt{y}_{ij}$ are taken from ρ.

$\iff$ There exists a minimal blocking set $\mathbb{B}$ for $\Lambda_{\mathtt{f}}$ for which there is no substitution $\sigma : \Xi \to X$ such that

$$\begin{array}{l} \sigma \mathtt{x}_i = x_i, \text{ and either} \\ \text{for some } \mathtt{x}_i \xrightarrow{a} \mathtt{y} \in \mathbb{B}, \quad \langle a, \sigma(\mathtt{y}) \rangle \in \beta_i, \text{ or} \\ \text{for some } \mathtt{x}_i \not\xrightarrow{a} \in \mathbb{B}, \quad \forall x \in X . \langle a, x \rangle \notin \beta_i \end{array}$$

Note that such $\mathbb{B}$ is necessarily satisfiable.

$\iff$ There exists a satisfiable minimal blocking set $\mathbb{B}$ for $\Lambda_{\mathtt{f}}$ such that for each $i = 1, \ldots, n$ there is $V_i(x_i, \beta_i) = \mathtt{tt}$.

$\iff$ There exists a satisfiable minimal blocking set $\mathbb{B}$ for $\Lambda_{\mathtt{f}}$ such that $\upsilon(\delta_{\mathtt{f}\mathbb{B}}) = \mathtt{tt}$.

From this, by definition of λ as in the proof of Theorem 2.25,

$$\bigvee_{\mathtt{f} \in \bar{\Sigma}} \bigvee_{\mathbb{B}} \upsilon(d_{\mathtt{f}i}) = \upsilon(\not\blacktriangleright) \circ \lambda_X$$

where the inner unions occur over all satisfiable minimal blocking sets for $\Lambda_{\mathtt{f}}$.

This completes the proof of Theorem 5.25. □

5.5 Failures Semantics

In this section, based on Corollary 5.5 specialized to the case of $\mathsf{W} = \mathsf{Fl}$, we show a congruence format for failures preorder and equivalence. Most proofs are extended versions of analogous proofs from Sections 5.3 and 5.4.

First, we modify Definition 5.16.

Definition 5.30 A set $\mathbb{B}$ of literals is an Fl-*blocking set* if either

- for some $\mathtt{x}, \mathtt{y} \in \Xi$, $a \in A$, both $\mathtt{x} \xrightarrow{a} \mathtt{y}$ and $\mathtt{x} \not\xrightarrow{a}$ belong to $\mathbb{B}$, or

- for every $\mathtt{x} \in \Xi$, $a \in A$, if $\mathtt{x} \stackrel{a}{\not\longrightarrow} \in \mathbb{B}$ then no other literal with $\mathtt{x}$ on the left side belongs to $\mathbb{B}$.

If the first of the above conditions does not hold, the blocking set is called *satisfiable*.

It is straightforward to check that every Fl-blocking set is a CTr-blocking set.

Format 5.31 An image finite set of GSOS rules Λ is in Fl-*format*, if

1. For each rule $\rho \in \Lambda$:
   - no variable occurs more than once in the target of ρ,
   - no variable occurs simultaneously in the left-hand side of a premise and in the target of ρ,
   - no variable occurs simultaneously in the left-hand side of a positive premise and in the left-hand side of any other premise of ρ.
2. For each construct $\mathtt{f}$ of the language and for every set $Q \subseteq A$, every minimal blocking set for $\Lambda_{\mathtt{f}Q}$ is an Fl-blocking set.

Proposition 5.32 **BPA** is in Fl-format.

Proof. Again, clearly all rules of **BPA** (see Example 5.7) satisfy condition 1 of Fl-format. For condition 2, consider a language construct $a-$ corresponding to $a \in A$. For any $Q \subseteq A$, if $a \in Q$ then take Λ_{a-Q} has no minimal blocking set. If $a \notin Q$, then $\Lambda_{a-Q} = \emptyset$ and it has the unique empty minimal blocking set, which is an Fl-blocking set. For the binary construct $+$, for any $Q \subseteq A$, the only minimal blocking set for Λ_{+Q} is

$$\left\{ \mathtt{x} \stackrel{a}{\longrightarrow} \mathtt{x}' \,:\, a \in Q \right\} \cup \left\{ \mathtt{y} \stackrel{a}{\longrightarrow} \mathtt{y}' \,:\, a \in Q \right\}$$

and it is a Fl-blocking set. □

The similar structure of CTr-format and Fl-format might be misleading. In fact, these formats are quite different, as the following two propositions show (compare Propositions 5.22 and 5.24).

Proposition 5.33 For $A = \{a, b\}$, the semantics for synchronous composition (see Example 5.21) is in Fl-format.

Proof. Condition 1 of Fl-format is obviously satisfied, and it is enough to check Condition 2 for the synchronous composition operator $\times$. Simple analysis of all minimal blocking sets for any $\Lambda_{\times Q}$ shows that they are all Fl-blocking sets. For example, if $Q = \{a\}$ then there are two minimal blocking sets: $\{\mathtt{x} \stackrel{a}{\longrightarrow} \mathtt{x}'\}$ and $\{\mathtt{y} \stackrel{a}{\longrightarrow} \mathtt{y}'\}$, both Fl-blocking. □

In [17] it was shown that the failures preorder is not a precongruence for **BPA** extended with sequential composition (Example 5.23).

Proposition 5.34 If A contains at least two different labels a, b, then **BPA** extended with sequential composition is not in Fl-format.

Proof. Condition 2 of Fl-format fails for the sequential composition operator. Indeed, take $Q = \{a\}$. Then $\Lambda_{;Q}$ contains two rules and one of its minimal blocking sets:

$$\{\mathtt{x} \xrightarrow{a} \mathtt{x}', \mathtt{x} \not\xrightarrow{b}\}$$

is not Fl-blocking. □

The Fl-format excludes many examples of transition system specifications that behave well with respect to the failures preorder. Many of these examples are covered by the 'failure trace format' introduced in [17]. However, the latter format excludes also some examples covered by Fl-format. Indeed, assume $a, b \in A$ and extend **BPA** with two unary constructs $\mathtt{g}, \mathtt{h}$ and operational rules

$$\frac{\mathtt{x} \xrightarrow{\alpha} \mathtt{x}'}{\mathtt{g}(\mathtt{x}) \xrightarrow{\alpha} \mathtt{h}(\mathtt{x}')} \qquad \frac{\mathtt{x} \not\xrightarrow{a}}{\mathtt{h}(\mathtt{x}) \xrightarrow{b} \mathtt{0}}$$

where α ranges over A.

Proposition 5.35 **BPA** extended with $\mathtt{g}$ and $\mathtt{h}$ as above, is in Fl-format.

Proof. Condition 1 of Fl-format is obviously satisfied, and it is enough to check condition 2 for constructs $\mathtt{g}$ and $\mathtt{h}$ only.

For $\mathtt{g}$, for any $Q \subseteq A$ the only minimal blocking set for $\Lambda_{\mathtt{g}Q}$ is the set $\left\{\mathtt{x} \xrightarrow{a} \mathtt{x}' : a \in Q\right\}$, which is an Fl-blocking set. For $\mathtt{h}$, for any $Q \subseteq A$, if $b \in Q$ then take the only minimal blocking set for $\Lambda_{\mathtt{h}Q}$ is and $\{\mathtt{x} \not\xrightarrow{a}\}$, which is an Fl-blocking set. If $b \notin Q$, then $\Lambda_{\mathtt{h}Q}$ is empty and its unique empty minimal blocking set is Fl-blocking. □

This means that Fl-format is incomparable with the 'failure trace format' shown in [17].

The following theorem, together with Corollary 5.5, shows that Fl-format is a precongruence format for failures preorder and a congruence format for failures equivalence.

Theorem 5.36 Let Λ be a GSOS specification and $\lambda : \Sigma(\mathrm{Id} \times B) \to BT$ the corresponding natural transformation. If Λ is in Fl-format, then

$$\lambda : \Sigma^*(\mathrm{Id} \times B^{\mathsf{Fl}}) \to B^{\mathsf{Fl}}T^*$$

is a natural transformation in 2-**TS**.

Proof. As in Theorems 5.8 and 5.25, we show that for any 2-test suite θ on a set X, for every test $V \in B^{\mathsf{Fl}}_{TX}T_X\theta$ there exists a test $V' \in \Sigma_X(\theta \bowtie B^{\mathsf{Fl}}_X\theta)$ such that $V \circ \lambda = V'$.

As before, we begin with a characterisation of tests in $B^{\mathsf{Fl}}_{TX}T_X\theta$:

Lemma 5.37 A test $V \in B^{\mathsf{Fl}}_{TX} T_X\theta$ is either the always true test $\overrightarrow{BTX}$, or the Q-failure test $\upsilon(\overset{Q}{\not\longrightarrow})$ for some $Q \subseteq A$, or a union of positive term θ-tests.

Proof. As in the proof of Lemma 5.27. □

The proof proceeds much the same as in Theorem 5.25. For the positive term tests V on BTX, we proceed as in the proof of Theorem 8.8, except that when constructing the tests V_i we consider one additional case:

- If $\mathtt{x}_i$ occurs in ρ in some negative premises, then take $V_i = \upsilon(\overset{Q_i}{\not\longrightarrow}) \circ \pi_2$, where $Q_i = \left\{ a \in A : \mathtt{x}_i \overset{a}{\not\longrightarrow} \text{ is a premise in } \rho \right\}$.

Again, the syntactic restrictions of condition 1 of Fl-format ensure that the definition of V_i's extended this way is complete and unambiguous.

The rest of the argument remains as in the case of Tr-format.

For the Q-failure test $\upsilon(\overset{Q}{\not\longrightarrow})$ for some $Q \subseteq A$, we proceed as in the proof of Theorem 5.25, except that we construct the tests V_i in a slightly different manner:

- If for no $a \in A$ $\mathtt{x}_i \xrightarrow{a} \mathtt{y}$ nor $\mathtt{x}_i \overset{a}{\not\longrightarrow}$ belong to $\mathbb{B}$, then take $V_i = \overrightarrow{BX} \circ \pi_2$.
- If, for some $a \in A$, $\mathtt{x}_i \xrightarrow{a} \mathtt{y} \in \mathbb{B}$, and if for all $b \in A$, $\mathtt{x}_i \overset{b}{\not\longrightarrow} \notin \mathbb{B}$, then take $V_i = \upsilon(\overset{Q_i}{\not\longrightarrow}) \circ \pi_2$, where $Q_i = \left\{ a \in A : \exists \mathtt{y} \in \Xi.\ \mathtt{x}_i \xrightarrow{a} \mathtt{y} \in \mathbb{B} \right\}$.
- If, for all $a \in A$, $\mathtt{x}_i \xrightarrow{a} \mathtt{y} \notin \mathbb{B}$, and if for some $b \in A$, $\mathtt{x}_i \overset{b}{\not\longrightarrow} \in \mathbb{B}$, then take $V_i = \upsilon(\xrightarrow{b} \overrightarrow{X}) \circ \pi_2$.

(note that the first and the third case are as in the case of CTr-format).

The definition of Fl-blocking set ensures that the above definition is complete and unambiguous.

The rest of the argument remains the same as in the proof of Theorem 5.25, except that $\Lambda_{\mathtt{f}}$ is replaced with $\Lambda_{\mathtt{f}Q}$ throughout. □

Chapter 6

Test Suites for Bisimulations on CPOs

In this chapter, the test suite approach is realized in a category of complete partial orders (cpos) to give a coalgebraic characterization of a version of bisimulation. The use of cpos instead of sets is motivated by semantic considerations involving recursive operators in process algebras and programming languages. To give a formal operational description of recursive operators, a simple and natural idea [69] is not to add them to the language syntax formally, but rather consider them as abbreviations for their infinite expansions. For example, in presence of the sequential composition operator `;` in the language, the term `loop a` might be considered as an abbreviation for the infinite term $\mathtt{a};\mathtt{a};\mathtt{a};\ldots$ This simplifies the semantic description of the language in question, since one does not need to give any other semantics to the construct `loop` besides its definition in terms of the sequential composition operator. However, this comes for a price: infinite syntactic terms must be considered, and to allow any kind of inductive reasoning on terms (for example, any kind of denotational semantics), one needs to consider them as elements of a cpo rather than of a set.

In Chapter 7, we shall investigate how recursive operators can be formally defined by recursive equations and combined with bialgebraic semantics for the recursion-free fragment of a language to obtain bialgebraic semantics for the full language. In this chapter, we focus on applying the test suite approach to transition systems based on cpos, in order to obtain a coalgebraic characterization of a canonical notion of process equivalence. To simplify matters, we consider unlabelled transition systems.

More specifically, we consider coalgebras for the functor $(\mathcal{P}^0 -)_\perp$ on the category $\mathbf{Acpo}_\perp$ of algebraic, pointed cpos, where $\mathcal{P}^0$ is the Plotkin powerdomain with empty set. The final coalgebra S for this endofunctor is a simplified (with labels removed) version of Abramsky's "domain for bisimulation". In [2], Abramsky showed how to interpret any *transition system with divergence* in S. He also defined a notion of *partial bisimulation* and showed that the interpretation in S is fully abstract with respect to the largest partial bisimulation, i.e. that for any transition system with divergence h with set of processes X, two processes $x, y \in X$ are related by a partial bisimulation if and only if they have the same interpretation in S. $(\mathcal{P}^0 -)_\perp$-coalgebras are a particular case of transition systems with divergence, and Abramsky's interpretation of them coincides with their coinductive extensions. Therefore, Abramsky's notion of

partial bisimulation provides full abstraction for the final semantics of them.

In this chapter, we shall show that the test suite approach is expressive enough to abstractly represent the full abstraction result. More specifically, in Section 6.3 we shall show a lifting of the endofunctor $(\mathcal{P}^0-)_\perp$ to an endofunctor $\mathcal{P}^{\mathsf{BS}}$ on an appropriate test suite category $\mathbb{O}$-$\mathbf{TS}$, such that for any coalgebra $h : D \to (\mathcal{P}^0 D)_\perp$, processes $x, y \in D$ are related by an appropriate specialization preorder of the least test suite lifting h to a $\mathcal{P}^{\mathsf{BS}}$-coalgebra if and only if $k(x) \leq k(y)$, where $k : D \to S$ is the coinductive extension of h.

This full abstraction result confirms the expressivity of the test suite framework, and in Section 6.2, we argue that the most popular abstract approach to coalgebraic bisimulation, the coalgebra span approach, fails to cover full abstraction for $(\mathcal{P}^0-)_\perp$-coalgebras.

Encouraged by this, in Sections 6.4–6.6 we show a relational characterization of specialization preorders for $\mathcal{P}^{\mathsf{BS}}$-coalgebras as *cpo-bisimulations*. The characterization is partial, i.e. it does not work for all $\mathcal{P}^{\mathsf{BS}}$-coalgebras, but the full abstraction result can be phrased in terms of this characterization provided the underlying $(\mathcal{P}^0-)_\perp$-coalgebra is *compact*.

6.1 Preliminaries

In this section, we recall some standard notions related to complete partial orders (cpos) and transition systems with divergence, taken mostly from [2, 67].

Definition 6.1 An *(ω-)complete partial order* (cpo) is a partial order $\langle D, \leq \rangle$ (often denoted just D) where all increasing ω-chains $x_0 \leq x_1 \leq x_2 \leq \cdots$ have least upper bounds (lubs), denoted $\bigsqcup_D x_n$ (the subscript is omitted where D is understood). A *continuous function* between cpos is a monotonic function that preserves lubs. A cpo with a least element (denoted $\perp$) is called *pointed*. A continuous function is called *strict* if it preserves the least element, and *very strict* if it preserves and reflects the least element.

Alternative definition of cpo requires lubs of all *directed sets*, and not only ω-chains, to exist (e.g. [3]). In the case of countable partial orders, the two definitions coincide. In the following, we will only consider countable, pointed cpos.

Definition 6.2 Let D be a cpo. The *Scott topology* σ_D contains, as open sets, subsets $X \subseteq D$ such that

- for any $x, y \in D$, if $x \in X$ and $x \leq y$ then $y \in X$, and
- for any ω-chain $x_1 \leq x_2 \leq \cdots$, if $\bigsqcup x_n \in X$ then $x_n \in X$ for some $n \in \mathbb{N}$.

A simpler topology can be defined on any preordered set:

Definition 6.3 Let $\langle D, \leq \rangle$ be a preorder. The *Alexandrov topology* $A_\leq$ contains, as open sets, subsets $X \subseteq D$ such that for any $x, y \in D$, if $x \in X$ and $x \leq y$ then $y \in X$.

It is easy to check that both of the above indeed define topologies.

Definition 6.4 Let D be a cpo. An element $a \in D$ is called *finite* if for any increasing chain $x_1 \leq x_2 \leq \cdots$, if $a \leq \bigsqcup x_n$ then $a \leq x_n$ for some $n \in \mathbb{N}$. The cpo D is *algebraic* if every element $x \in D$ is a lub of some increasing chain of finite elements.

The set of all finite elements of a cpo D is denoted $K(D)$.

Proposition 6.5 Algebraic, pointed cpos together with continuous, strict functions form a category, in the following denoted $\mathbf{Acpo}_\perp$.

Algebraic cpos are closed under many useful constructions [67]. For the purposes of this chapter, however, it is enough to consider lifting and the Plotkin powerdomain with empty set [66, 2].

Definition 6.6 Let D be a cpo. The cpo $D_\perp$ is the set $\{\, up(d) \,:\, d \in D \,\} \cup \{\perp\}$ with the ordering

$$x \leq y \iff (x = \perp) \text{ or } (\exists u, v \in D.\ u \leq v \wedge x = up(u) \wedge y = up(v))$$

It is easy to see how this definition extends to a functor $(-)_\perp : \mathbf{Acpo}_\perp \to \mathbf{Acpo}_\perp$.

We now define the Plotkin powerdomain [66], which is a counterpart of the powerset construction on sets.

First, for any algebraic cpo D, define a closure operation $(-)^\star$ on subsets of D:

$$X^\star = \{\, y \in D \,:\, (\exists x \in X.\ x \leq y) \wedge (\forall a \in K(D).\ a \leq y \Longrightarrow \exists x \in X.\ a \leq x) \,\}$$

Subsets of D can be ordered by a modified version of the Egli-Milner ordering as follows: $X \lesssim_{\mathrm{EM}} Y$ if and only if

$$\forall x \in X.\ \forall a \in K(D).\ (a \leq x \Longrightarrow \exists y \in Y.\ a \leq y) \quad \text{and}$$
$$\forall y \in Y.\ \exists x \in X. x \leq y$$

Note that the empty set is not related to anything except itself by the ordering relation $\lesssim_{\mathrm{EM}}$.

Finally, we define the Plotkin powerdomain with empty set [2]:

Definition 6.7 Let D be an algebraic cpo. The cpo $\mathcal{P}^0(D)$ has as elements all subsets $X \subseteq D$ compact in the Scott topology σ_D and such that $X^\star = X$, with the ordering relation

$$X \leq Y \iff X = \{\perp\} \text{ or } X \lesssim_{\mathrm{EM}} Y$$

In, e.g., [67] it is proved that $\mathcal{P}^0(D)$ is indeed an algebraic and pointed cpo (in particular, $\{\perp\}$ is the least element). It can also be extended to a functor $\mathcal{P}^0 : \mathbf{Acpo}_\perp \to \mathbf{Acpo}_\perp$ by putting, for a strict, continuous $f : D \to E$,

$$(\mathcal{P}^0 f)(X) = \{\, f(x) \,:\, x \in X \,\}^\star$$

When speaking about elements of cpos of the form $(\mathcal{P}^0 D)_\perp$, to simplify the notation, we will skip the constructor *up* coming from the lifting functor. This will not lead to any confusion, as all elements of $\mathcal{P}^0 D$ are presented as sets and thus are different from $\perp$.

From general properties of $\mathbf{Acpo}_\perp$ (more specifically, from the fact that $\mathbf{Acpo}_\perp$ is $\mathbf{Cppo}_\perp$-enriched and $\mathbf{Cpo}$-algebraically compact (see [28]), from the limit/colimit coincidence theorem [81], and from local continuity of $\mathcal{P}^0$ [67]) it follows that the endofunctor $(\mathcal{P}^0 -)_\perp$ admits a final coalgebra $\phi : S \to (\mathcal{P}^0 S)_\perp$, which is the initial solution to the "domain equation" $X \simeq (\mathcal{P}^0 X)_\perp$ in the spirit of [2]. As follows from [67, Chapter 5], the finite elements of S can be characterized inductively as follows (see also [2, Proposition 3.10]):

Proposition 6.8 The set $K(S)$ of finite elements of the final $(\mathcal{P}^0 -)_\perp$-coalgebra $\phi : S \to (\mathcal{P}^0 S)_\perp$ is defined inductively by

- $\perp = \phi^{-1}(\perp) \in K(S)$
- $\phi^{-1}(\emptyset) \in K(S)$
- if $a_1, \ldots, a_n \in K(S)$ then $\phi^{-1}(\{a_1, \ldots, a_n\}^\star) \in K(S)$

In the following, we will consider very strict coalgebras for the endofunctor $(\mathcal{P}^0 -)_\perp$. Such coalgebras are special instances of the following general notion:

Definition 6.9 A *transition system with divergence* $\langle X, \longrightarrow, \uparrow \rangle$ is a set X of *processes*, a *transition relation* $\longrightarrow \subseteq X \times X$ and a *divergence predicate* $\uparrow \subseteq X$. If for a process $x \in X$ it is not the case that $x \in \uparrow$ (denoted $x \uparrow$), we write $x \downarrow$.

Any very strict coalgebra $h : D \to (\mathcal{P}^0 D)_\perp$ in $\mathbf{Acpo}_\perp$ can be viewed as a transition system with divergence, by taking $D \setminus \{\perp\}$ as the set of processes (with ordering on D ignored), the transition relation defined by

$$x \longrightarrow y \iff y \in h(x), y \neq \perp$$

and the divergence predicate

$$x \uparrow \iff \perp \in h(x)$$

Along the lines of [2], every transition system with divergence can be canonically interpreted in the final coalgebra $\phi : S \to (\mathcal{P}^0 S)_\perp$. For a transition system obtained from a $(\mathcal{P}^0 -)_\perp$-coalgebra h as above, this interpretation coincides with the coinductive extension of h. In [2], Abramsky showed a general full abstraction result, based on the notion of partial bisimulation.

Definition 6.10 A *partial bisimulation* on a transition system with divergence $\langle X, \longrightarrow, \uparrow \rangle$ is a relation $R \subseteq X \times X$ such that for every $x, y \in X$, xRy implies:

- $\forall x \longrightarrow x'.\ \exists y \longrightarrow y'.\ x'Ry'$, and
- $p \downarrow \Longrightarrow q \downarrow \wedge \forall y \longrightarrow y'.\ \exists x \longrightarrow x'.\ x'Ry'$.

The full abstraction theorem proved in [2] says essentially

Proposition 6.11 Take $h : D \to (\mathcal{P}^0 D)_\perp$ as a transition system with divergence. If h is *finitary*, then for any $x, y \in D$, $kx \leq ky$ (where $k : D \to S$ is the coinductive extension of h) if and only if xRy for some partial bisimulation R on h.

This characterization works only under the mild condition of *finitarity*, which we leave unexplained here. Without this assumption (and not all coalgebras for the Plotkin powerdomain satisfy it), the characterization is a bit more complicated (see [2] for details).

The remainder of this chapter attempts to cover this full abstraction result by a few abstract approaches to bisimulation.

6.2 Coalgebra Spans and Their Limitations

The coalgebra span framework [8, 77] and its ordered version [29, 76] are the classical abstract coalgebraic approaches to bisimulation. It is therefore tempting to give an abstract account of the full abstraction result by Abramsky by analysis of this abstract approach. In this section we briefly recall the coalgebra span approach (for a comprehensive treatment, see [77, 29]) and observe that it is not entirely appropriate for our purposes.

Definition 6.12 Let F be any endofunctor on a category $\mathbb{C}$, and let $h : X \to FX$ be a coalgebra. A *(span) bisimulation* on h is an object R together with two morphisms $p, q : R \to X$ such that there exists a coalgebra structure $r : R \to FR$ that makes p and q into coalgebra morphisms:

$$\begin{array}{ccccc} X & \xleftarrow{p} & R & \xrightarrow{q} & X \\ {\scriptstyle h}\downarrow & & \downarrow{\scriptstyle r} & & \downarrow{\scriptstyle h} \\ FX & \xleftarrow[Fp]{} & FR & \xrightarrow[Fq]{} & FX \end{array}$$

With the assumption that F admits final coalgebras, general results about (span) bisimulations say that the final semantics (the coinductive extension) is always *sound* with respect to the above notion of bisimulation. Speaking more concretely, in the case of $\mathbb{C} = \mathbf{Acpo}_\perp$, bisimulations R on a coalgebra $h : X \to BX$ can be seen as binary relations on X, with p, q being projections on X. The soundness result then means that if two elements $x, y \in X$ are related by any bisimulation, then $kx = ky$, where $k : X \to \Omega$ is the coinductive extension of h.

If, moreover, $\mathbb{C}$ has kernel pairs (which is the case for $\mathbb{C} = \mathbf{Acpo}_\perp$) and if weak kernel pairs are preserved by F, then the kernel pair of k is a bisimulation and the final semantics is *fully abstract*. For $\mathbb{C} = \mathbf{Acpo}_\perp$, this means that for any $x, y \in X$, if $kx = ky$ then x and y are related by some (span) bisimulation (indeed, by the kernel pair of k).

Unfortunately, it turns out that endofunctors $\mathcal{P}^0, (P^0 -)_\perp : \mathbf{Acpo}_\perp \to \mathbf{Acpo}_\perp$ do *not* preserve weak kernel pairs due to convexity phenomena. Indeed, as was probably discovered by Plotkin,

Counterexample 6.13 There exists a coalgebra $h : X \to (\mathcal{P}^0 X)_\perp$ such that the kernel pair of the coinductive extension $k : X \to S$ of h is not a span bisimulation.

Proof. To simplify the notation, define the following (finite) elements of S:

$$\begin{aligned} x_1 &= \phi^{-1}(\{\perp\}) \\ x_2 &= \phi^{-1}(\{x_1\}) \\ x_3 &= \phi^{-1}(\{x_2\}) \\ z &= \phi^{-1}(\{x_1, x_2, x_3\}^\star) \end{aligned}$$

Note that $x_1 \leq x_2 \leq x_3$ in S, hence $\{x_1, x_3\}^\star = \{x_1, x_2, x_3\}^\star$.

Now define a cpo D as the set

$$\{\perp, a, b, a_1, a_2, a_3, b_1, b_3\}$$

with the ordering

$$x \leq y \iff x = \perp \vee x = y$$

(i.e., D is *flat*).

Consider the (very strict) coalgebra structure $h : D \to (\mathcal{P}^0 D)_\perp$ defined by

$$\begin{array}{rcl} \perp & \mapsto & \perp \\ a & \mapsto & \{a_1, a_2, a_3\} \\ b & \mapsto & \{b_1, b_3\} \\ a_1 & \mapsto & \{\perp\} \\ a_2 & \mapsto & \{a_1\} \\ a_3 & \mapsto & \{a_2\} \\ b_1 & \mapsto & \{\perp\} \\ b_3 & \mapsto & \{a_2\} \end{array}$$

Let $k : D \to S$ be the coinductive extension of h. It is easy to check that $k(a_1) = k(b_1) = x_1$, $k(a_3) = k(b_3) = x_3$ and $k(a_2) = x_2$. Also $k(a) = \phi^{-1}(\{k(a_1), k(a_2), k(a_3)\}^\star) = z$ and $k(b) = \phi^{-1}(\{k(b_1), k(b_3)\}^\star) = z$ (note how convexity comes into play here).

Consider the kernel relation R of k and assume a coalgebra structure $r : R \to (\mathcal{P}^0 R)_\perp$ such that makes the projections $\pi_1, \pi_2 : R \to D$ into coalgebra morphisms:

$$\begin{array}{ccccc} R & \overset{\pi_1}{\underset{\pi_2}{\rightrightarrows}} & D & \xrightarrow{k} & S \\ {\scriptstyle r}\downarrow & & \downarrow{\scriptstyle h} & & \downarrow{\scriptstyle \phi} \\ (\mathcal{P}^0 R)_\perp & \overset{(\mathcal{P}^0\pi_1)_\perp}{\underset{(\mathcal{P}^0\pi_2)_\perp}{\rightrightarrows}} & (\mathcal{P}^0 D)_\perp & \xrightarrow[(\mathcal{P}^0 k)_\perp]{} & (\mathcal{P}^0 S)_\perp \end{array}$$

Note that $\langle a, b\rangle \in R$ and

$$\begin{aligned} (\mathcal{P}^0\pi_1)_\perp \theta \langle a, b\rangle &= h(\pi_1 \langle a, b\rangle) = \{a_1, a_2, a_3\} \\ (\mathcal{P}^0\pi_2)_\perp \theta \langle a, b\rangle &= h(\pi_2 \langle a, b\rangle) = \{b_1, b_3\} \end{aligned}$$

This implies that $\langle a_2, b_1\rangle \in \theta \langle a, b\rangle$ or $\langle a_2, b_3\rangle \in \theta \langle a, b\rangle$. On the other hand, $\theta \langle a, b\rangle \in (\mathcal{P}^0 R)_\perp$, hence $\theta \langle a, b\rangle \subseteq R$. This means that $\langle a_2, b_1\rangle \in R$ or $\langle a_2, b_3\rangle \in R$, which gives a contradiction, since $k(a_2) \neq k(b_1), k(b_3)$. □

This counterexample shows that the original coalgebra span approach does not lead to the full abstraction result for the final semantics of $(\mathcal{P}^0 -)_\perp$-coalgebras.

In the ordered setting, another version of the coalgebra span approach has been considered, based on the notion of ordered bisimulation [78, 29, 76]. There, one works in a **Pos**-enriched category, i.e., a category where all homsets carry a partial order structure and where composition is monotonic (for a general treatment of enriched category theory, see [48]). Obviously $\mathbf{Acpo}_\perp$ is such a category.

In **Pos**-enriched categories, the standard notion of kernel pair can be generalized to the notion of ordered kernel pair. We define this notion after [29]:

Definition 6.14 In a **Pos**-enriched category $\mathbb{C}$, an object C with two morphisms $p, q : C \to D$ is a *ordered kernel pair* of a morphism $k : D \to E$ if $k \circ p \leq k \circ q$ and if for any $p', q' : C' \to D$ such that $k \circ p' \leq k \circ q$ there exists a unique $r : C' \to C$ such that $p' \circ r \geq p$ and $q' \circ r = q$. If the uniqueness condition is dropped, then C, p, q is a *weak ordered kernel pair* of k.

In $\mathbf{Acpo}_\perp$ all ordered kernel pairs exist, and the ordered kernel pair of a continuous $k : D \to E$ is the set of pairs $\langle x, y \rangle \in D \times D$ such that $kx \leq ky$ in E, ordered by the componentwise order inherited from D.

The notion of span bisimulation from Definition 6.12 can be generalized, in the ordered setting, to

Definition 6.15 Let F be a locally monotonic (i.e., monotonic on homsets) endofunctor on a category $\mathbb{C}$, and let $h : X \to FX$ be a coalgebra. An *ordered bisimulation* on h is an object R together with two morphisms $p, q : R \to X$ such that there exists a coalgebra structure $r : R \to FR$ that makes the following diagram commute:

$$\begin{array}{ccccc} X & \xleftarrow{p} & R & \xrightarrow{q} & X \\ {\scriptstyle h}\downarrow & \leq & \downarrow{\scriptstyle r} & & \downarrow{\scriptstyle h} \\ FX & \xleftarrow[Fp]{} & FR & \xrightarrow[Fq]{} & FX \end{array}$$

An abstract theorem from [76] states that if $\mathbb{C}$ has ordered kernel pairs and if F preserves weak ordered kernel pairs, than the ordered kernel pair of the inductive extension k of any F-coalgebra $h : D \to FD$ is an ordered bisimulation on h. For $\mathbb{C} = \mathbf{Acpo}_\perp$ this means that for any $x, y \in D$, $kx \leq ky$ if and only if x and y are related by some ordered bisimulation.

This result seems quite similar to the Abramsky's full abstraction theorem (Proposition 6.11). Unfortunately, contrary to what was speculated in [76], the Plotkin powerdomain functor does *not* preserve weak ordered kernel pairs due to a limiting phenomenon, and indeed:

Counterexample 6.16 There exists a coalgebra $h : X \to (\mathcal{P}^0 X)_\perp$ such that the ordered kernel pair of the coinductive extension $k : X \to S$ of h is not an ordered bisimulation.

Proof. To simplify the notation, define the following infinite sequence of elements $x_1, x_2, x_3, \ldots$ of S:

$$\begin{aligned} x_1 &= \phi^{-1}(\{\bot\}) \\ x_2 &= \phi^{-1}(\{x_1\}) \\ x_3 &= \phi^{-1}(\{x_2\}) \\ &\ldots \end{aligned}$$

These elements form an increasing chain: $x_1 \leq x_2 \leq x_3 \leq \cdots$. Moreover, denote

$$\begin{aligned} x_\omega &= \textstyle\bigsqcup_{n\in\mathbb{N}} x_n \\ z &= \phi^{-1}(\{\bot, x_1, x_2, x_3, \ldots, x_\omega\}^\star) \end{aligned}$$

Note that $\{\bot, x_1, x_2, x_3, \ldots\}^\star = \{\bot, x_1, x_2, x_3, \ldots, x_\omega\}^\star$.

Now define a cpo D as the set

$$\{\bot, a, b, c_1, c_2, c_3, \ldots, c_\omega\}$$

with the ordering

$$x \leq y \iff x = \bot \vee x = y$$

(i.e., D is again *flat*).

Consider the (very strict) coalgebra structure $h : D \to (\mathcal{P}^0 D)_\bot$ defined by

$$\begin{array}{rcll} \bot & \mapsto & \bot & \\ a & \mapsto & \{\bot, c_1, c_2, c_3, \ldots\} & \\ b & \mapsto & \{\bot, c_1, c_2, c_3, \ldots, c_\omega\} & \\ c_1 & \mapsto & \{\bot\} & \\ c_n & \mapsto & \{c_{n-1}\} & \text{for } n > 1 \\ c_\omega & \mapsto & \{c_\omega\} & \end{array}$$

Let $k : D \to S$ be the coinductive extension of h, and let R denote the relation on D corresponding to the ordered kernel pair of k (i.e., xRy iff $k(x) \leq k(y)$.)

As observed in [29], ordered bisimulations on $\mathcal{P}^0$-coalgebras with flat carriers are partial bisimulations on the respective transition systems with divergence. However, here R is not a partial bisimulation. Indeed, it is easy to see that:

$$\begin{array}{rcll} k(c_n) & = & x_n & \text{for } n \in \mathbb{N} \\ k(c_\omega) & = & x_\omega & \\ k(a) & = & z & \\ k(b) & = & z & \end{array}$$

In particular, bRa. However, $c_\omega \in h(b)$ and there is no element $x \in h(a)$ such that xRc_ω, therefore the first condition in Definition 6.10 fails for R.

As a side remark, this means that the above coalgebra h is not finitary in the sense of [2], since the simple characterization from Proposition 6.11 does not work for it. □

This counterexample shows that the ordered coalgebra span approach does not lead to a full abstraction result for the final semantics of $(\mathcal{P}^0 -)_\bot$-coalgebras

either. Note that the counterexample does not contradict the abstract characterizations given by Pitts in [65], as they only concern the "internal full abstraction" of the final coalgebra, with no regard to the final semantics of other coalgebras.

In the remainder of this chapter, we show how the test suite approach described in Chapter 3 can be used for the abstract treatment of bisimulations on cpos.

6.3 Test Suite Approach

To define an appropriate notion of bisimulation on cpos abstractly, we use the test suite framework described in Chapter 3, specialized to the case $\mathbb{C} = \mathbf{Acpo}_\perp$. We let the test value object $\mathcal{V}$ be the *Sierpinski space* $\mathbb{O}$, defined by

$$\mathbb{O} = \{\mathtt{ff}, \mathtt{tt}\} \text{ with the ordering } \mathtt{ff} \leq \mathtt{tt}$$

This gives rise to the category of $\mathbb{O}$-test suites $\mathbb{O}$-**TS**, along the lines of Definition 3.1.

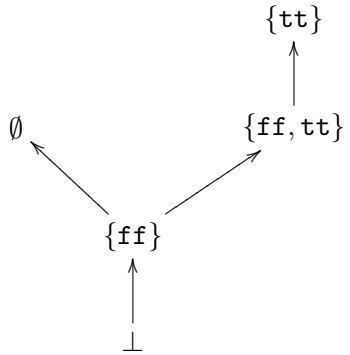
For a specialization functor construction, consider the partial order R on $\mathbb{O}$ equal to the ordering relation of $\mathbb{O}$. When considered as an object of a suitable relation category, this yields a specialization preorder construction along the lines of Theorem 3.24. In elementary terms, the specialization preorder $\leq_\theta$ for a test suite $\theta : D \rightrightarrows \mathbb{O}$ is defined by

$$x \leq_\theta y \iff \forall V \in \theta.\, Vx = \mathtt{tt} \Longrightarrow Vy = \mathtt{tt}$$

(compare Example 3.25).

To lift the endofunctor $(\mathcal{P}^0 -)_\perp$ to the category $\mathbb{O}$-**TS** we provide a set of $((\mathcal{P}^0 -)_\perp, \mathbb{O})$-test constructors (i.e. continuous, strict functions from $(\mathcal{P}^0\mathbb{O})_\perp$ to $\mathbb{O}$), as shown in Section 3.3.3.

First, recall that $(\mathcal{P}^0\mathbb{O})_\perp$ is the cpo



For any pointed cpo D, $\mathbb{O}$-tests on D correspond to Scott-open subsets of the (not necessarily pointed) cpo $D \setminus \{\perp\}$ (note that strictness of $\mathbb{O}$-tests prevents the entire cpo D from being identified with any test). As in Notation 4.1, we introduce a special notation for this correspondence:

$$\begin{array}{rcll} \overline{V} & = & \{\, x \in D : Vx = \mathtt{tt} \,\} & \text{for } V : D \to \mathbb{O} \text{ strict} \\ \overrightarrow{X}x & = & \begin{cases} \mathtt{tt} & \text{if } x \in X \\ \mathtt{ff} & \text{otherwise} \end{cases} & \text{for } X \subsetneq D \text{ Scott-open} \end{array}$$

We will be, however, more sloppy when speaking about entire test suites, and we will not distinguish between test suites and families of the corresponding sets. In particular, we will sometimes say that a set (rather than the corresponding test) belongs to a test suite. This should not lead to any confusion.

Consider the following test constructors, i.e. strict functions from $(\mathcal{P}^0\mathbb{O})_\perp$ to $\mathbb{O}$:

$$w_\Box = \overrightarrow{\{\emptyset, \{\mathtt{tt}\}\}}$$
$$w_\Diamond = \overrightarrow{\{\{\mathtt{ff}, \mathtt{tt}\}, \{\mathtt{tt}\}\}}$$

Since $\mathbb{O}$-tests on D correspond to Scott-open subsets of $D \setminus \{\perp\}$, it is valid to consider for any D the closure operator Cl^t_D, that given an $\mathbb{O}$-test suite θ on D, returns the smallest topology on $D \setminus \{\perp\}$ that contains θ. Since elements of θ are guaranteed to be Scott-open, the topology $\mathrm{Cl}^t_D\,\theta$ is always a subtopology of the Scott topology $\sigma_{D\setminus\{\perp\}}$.

It is easy to verify that the operators Cl^t_D form a test suite closure Cl^t in the sense of Definition 3.31. Indeed, the proof proceeds exactly as in the case of $\mathrm{Cl}^{\vee\wedge}$ on sets, in Section 4.1. Now we are ready to lift the endofunctor $(\mathcal{P}^0-)_\perp$ to the category $\mathbb{O}$-**TS**:

Definition 6.17 The endofunctor on $\mathbb{O}$-**TS** induced by the set of $((\mathcal{P}^0-)_\perp, \mathbb{O})$-test constructors $\{w_\Box, w_\Diamond\}$ and by the closure Cl^t is denoted $\mathcal{P}^{\mathsf{BS}}$.

This particular lifting of $(\mathcal{P}^0-)_\perp$ to $\mathbb{O}$-**TS** might seem a bit arbitrary, but is well motivated by the following results. From Theorem 3.11 it follows that the final $(\mathcal{P}^0-)_\perp$-coalgebra $\phi : S \to (P^0S)_\perp$ can be lifted to a final $\mathcal{P}^{\mathsf{BS}}$-coalgebra $\phi : \langle S, \omega\rangle \to \big\langle (P^0S)_\perp, \mathcal{P}^{\mathsf{BS}}_S\omega\big\rangle$, where ω is the least (with respect to inclusion) fixed point of the operator $\Phi_\phi\theta = \phi^*\mathcal{P}^{\mathsf{BS}}_S\theta$.

Theorem 6.18 Under the above notation, $\omega = \sigma_{S\setminus\{\perp\}}$.

Proof. The inclusion $\omega \subseteq \sigma_{S\setminus\{\perp\}}$ is trivial. For the other inclusion use the fact that sets of the form $a{\uparrow} = \{\, x \in S : a \leq x \,\}$ for $a \in K(S)$ form a basis for the Scott topology σ_S. It is therefore enough to show that all such sets (except S itself) are contained in ω. To this end, we proceed by structural induction on finite elements of S, using Proposition 6.8.

For $a = \phi^{-1}(\emptyset)$, consider the test

$$W = w_\Box \circ (\mathcal{P}^0\overrightarrow{\emptyset})_\perp \circ \phi$$

Note that $(w_\Box \circ (\mathcal{P}^0\overrightarrow{\emptyset})_\perp)z = \mathtt{tt}$ if and only if $z = \emptyset$, hence $\phi^{-1}(\emptyset) = \overline{W}$. Clearly $W \in \omega$.

For $a = \phi^{-1}\{\perp, a_1, \ldots, a_n\}^\star$, assume tests $V_1, \ldots, V_n \in \omega$ such that $V_i = a_i{\uparrow}$ for $i = 1, \ldots, n$ and consider the test

$$W = \bigwedge_{i=1}^{n} (w_\Diamond \circ (\mathcal{P}^0V_i)_\perp \circ \phi)$$

Observe that $Wx = \mathtt{tt}$ if and only if for every $i = 1, \ldots, n$ there is some $x' \in \phi(x)$ such that $a_i \leq x'$. Since all a_i's are finite, this amounts exactly to saying that $\{\bot, a_1, \ldots, a_n\}^\star \leq \phi(x)$ in $(\mathcal{P}^0 S)_\bot$.

For $a = \phi^{-1}\{a_1, \ldots, a_n\}^\star$, where $a_i \neq \bot$, $V_1, \ldots, V_n \in \omega$ such that $V_i = a_i{\uparrow}$ for $i = 1, \ldots, n$ and consider the test

$$W = \left(\bigwedge_{i=1}^{n} (w_\Diamond \circ (\mathcal{P}^0 V_i)_\bot \circ \phi) \right) \wedge \left(w_\Box \circ (\mathcal{P}^0 \bigvee_{i=1}^{n} V_i)_\bot \circ \phi \right)$$

Here, $Wx = \mathtt{tt}$ if and only if

- for every $i = 1, \ldots, n$ there is some $x' \in \phi(x)$ such that $a_i \leq x'$, and
- for every $x' \in \phi(x)$ there is some $i = 1, \ldots, n$ such that $a_i \leq x'$.

Again, this is equivalent to saying that $\{a_1, \ldots, a_n\}^\star \leq \phi(x)$ in $(\mathcal{P}^0 S)_\bot$. This concludes the inductive step. □

Note how the constructions used in this proof resemble the logical constructions used in the Definability Theorem 4.9 in [2].

Recall from Theorem 3.12 that for any coalgebra $h : D \to (P^0 D)_\bot$ with the coinductive extension $k : D \to S$, the test suite $k^*\omega$ is the least test suite that lifts h to a $\mathcal{P}^{\mathsf{BS}}$-coalgebra. On the other hand,

Lemma 6.19 For any elements $x, y \in D$, $x \leq_{k^*\sigma_{S\setminus\{\bot\}}} y$ if and only if $k(x) \leq k(y)$.

Proof. It is routine to show that for (any) cpo S, the specialization preorder $\leq_{\sigma_S}$ is equal to the ordering relation on S. Since $S \neq \{\bot\}$, this also applies to $\leq_{\sigma_{S\setminus\{\bot\}}}$. Now

$$\begin{array}{ll}
x \leq_{k^*\sigma_{S\setminus\{\bot\}}} y & \iff \\
\forall V \in k^*\sigma_{S\setminus\{\bot\}}.\ Vx = \mathtt{tt} \Longrightarrow Vy = \mathtt{tt} & \iff \\
\forall V' \in \sigma_{S\setminus\{\bot\}}.\ V(k(x)) = \mathtt{tt} \Longrightarrow V(k(y)) = \mathtt{tt} & \iff \\
k(x) \leq_{\sigma_{S\setminus\{\bot\}}} k(y) & \iff \\
k(y) \leq k(y) &
\end{array}$$

□

Theorem 6.12 and Lemma 6.13 show that final semantics is indeed fully abstract with respect to the abstract notion of "bisimulation" defined as the specialization preorder of a $\mathcal{P}^{\mathsf{BS}}$-coalgebra. This chapter might well end here; it has been shown that the test suite approach is expressive enough to cover a canonical (i.e. leading to full abstraction of the final semantics) process equivalence on transition systems based on cpos.

However, for a better understanding of our abstract notion of process equivalence we look for a concrete, relational description of $\mathcal{P}^{\mathsf{BS}}$-coalgebras. The remainder of this chapter is devoted to finding such a characterization.

6.4 Preorders and Topologies on CPOs

We begin with a general characterization of those relations on cpos that are specialization preorders of subtopologies of Scott topologies.

We will consider preorders R on D, where $\langle D, \leq \rangle$ is an algebraic cpo. We will always assume that $\leq \subseteq R$.

Some basic notation will be useful:

$$\begin{aligned} x{\uparrow^R} &= \{y \in D \mid xRy\} \\ x{\downarrow_R} &= \{y \in D \mid yRx\} \end{aligned}$$

In the following definition always $x, y \in D$ and $a, b \in K(D)$.

Definition 6.20 A preorder R is called *finitary*, if

$$\forall x, y.(\forall a.aRx \Rightarrow aRy) \Longrightarrow xRy$$

It is called *strongly finitary*, if

$$\forall x, y.(\forall a \leq x.aRy) \Longrightarrow xRy$$

It is called *weakly algebraic*, if

$$\forall a, y.aRy \Longrightarrow \exists b \leq y.aRb$$

It is called *algebraic*, if

$$\forall x, y.(xRy \Longleftrightarrow \forall a \leq x.\exists b \leq y.aRb)$$

Lemma 6.21 The properties of finitarity (F), strong finitarity (SF), weak algebraicity (WA) and algebraicity (A) are related by the following implications:

$$\mathrm{F} \Longleftarrow \mathrm{SF} \Longleftarrow \mathrm{F} \wedge \mathrm{WA} \Longleftrightarrow \mathrm{A}$$

Proof. The implication SF $\Longrightarrow$ F is obvious, since $\leq \subseteq R$. To show that F $\wedge$ WA $\Longrightarrow$ SF, fix $x, y \in D$ and assume that for all finite $a \leq x$ there is aRy. Take any finite b such that bRx. By weak algebraicity, there is a finite $a \leq x$ such that bRa, hence by the above assumption and by transitivity of R, bRy. This works for arbitrary finite b such that bRx, hence by finitarity, xRy.

For the equivalence F $\wedge$ WA $\Longleftrightarrow$ A, note that by the previous implications, in presence of weak algebraicity, finitarity is equivalent to strong finitarity. It therefore enough to prove that SF $\wedge$ WA $\Longleftrightarrow$ A.

For the right-to-left implication, assume R algebraic. It is then obviously weakly algebraic (take $x = a$ in the definition of algebraicity). To prove strong finitarity, assume $\forall a \leq x.aRy$. Applying algebraicity for every finite $a \leq x$, one gets

$$\forall a \leq x.\exists b \leq y.aRb$$

hence by algebraicity, xRy.

For the left-to-right implication, assume R strongly finitary and weakly algebraic, and for all $x, y \in D$ show both directions of

$$xRy \Leftrightarrow \forall a \leq x.\exists b \leq y.aRb)$$

$\Rightarrow$: Assume xRy. Take any $a \leq x$. Since $\leq \subseteq \theta$, by transitivity aRy. By weak algebraicity there exists $b \leq y$ such that aRb.

$\Leftarrow$: Assume $\forall a \leq x.\exists b \leq y.aRb$. Since $\leq \subseteq \theta$, by transitivity of R there is

$$\forall a \leq x.aRy$$

which, by strong finitarity, gives xRy. □

Any preorder R on D can be canonically extended to the least strongly finitary preorder that contains R, as follows from

Lemma 6.22 For any family $\{R_i\}_{i \in I}$ of strongly finitary preorders on D, the relation

$$R = \bigcap_{i \in I} R_i$$

is a strongly finitary preorder.

Proof. Reflexivity and transitivity of R is straightforward. So is strong finitarity:

$$\begin{array}{l} \forall a \in K(D).(a \leq x \Rightarrow aRy) \Longrightarrow \\ \forall i \in I.\forall a \in K(D).(a \leq x \Rightarrow aR_iy) \Longrightarrow (R_i \text{ strongly finitary}) \\ \forall i \in I.xR_iy \Longrightarrow \\ xRy \end{array}$$

□

From this it easily follows that for any preorder R on D, the intersection of all strongly finitary preorders R' such that $R \subseteq R'$ is the least strongly finitary extension of R.

We proceed to characterize those preorders on cpos that are specialization preorders of subtopologies of the Scott topology, as strongly finitary preorders. Also for any preorder R on a cpo D, we find a topology θ_R for which the specialization preorder is somehow related to R (more precisely, it is the least strongly finitary extension of R).

In the following, by a topology we mean a subtopology of the Scott topology on a given algebraic cpo.

Lemma 6.23 For any topology θ on D, the specialization preorder $\leq_\theta$ is strongly finitary.

Proof. Take any $x, y \in D$ and assume for all finite $a \leq x$, $a \leq_\theta y$. Take any $V \in \theta$ such that $x \in V$. Since V is Scott-open (and D is algebraic), it contains some finite $a \leq x$, so by the assumption it also contains y. □

Definition 6.24 For any preorder R on D, let θ_R denote the intersection of the Scott topology on $D \setminus \{\bot\}$ and the Alexandrov topology of R.

Obviously θ_R is a topology in our sense, i.e., a subtopology of the Scott topology on D.

Lemma 6.25 If R on D is strongly finitary then $\leq_{\theta_R} = R$.

Proof. $\supseteq$: Assume xRy and take any $V \in \theta_R$ such that $x \in V$. Since V is by definition R-upper, also $y \in V$. Note that strong finitarity of R is not used here.

$\subseteq$: First note that strong finitarity of R amounts to saying that for any $y \in D$, the set $y{\downarrow_R}$ is Scott-closed.

Now take $x, y \in D$ such that $x \not R y$. It is enough to show a Scott-open, R-upper set V such that $x \in V$ and $y \notin V$. Take $V = D \setminus y{\downarrow_R}$. It is Scott-open, because $y{\downarrow_R}$ is Scott-closed. It is also obviously R-upper, and $y \notin V$. Since $x \not R y$, also $x \in V$, hence $x \not\leq_{\theta_R} y$. □

Lemmas 6.23 and 6.24 give a correspondence between topologies on D and strongly finitary preorders on D. This correspondence is in fact even stronger:

Theorem 6.26 For any preorder R on D, $\leq_{\theta_R}$ is the least strongly finitary extension of R.

Proof. First, note that the construction of θ_R from θ is reverse monotonic, i.e., if $R \subseteq R$ then $\theta_R \supseteq \theta_{R'}$. (It is enough to observe that if $R \subseteq R$ then every R'-upper set is R-upper.) Also the construction of specialization preorder $\leq_\theta$ from a topology θ is reverse monotonic, as is easily checked. As a conclusion, the construction of $\leq_{\theta_R}$ from R is monotonic.

Now for any R, the preorder $\leq_{\theta_R}$ is strongly finitary by Lemma 6.23 and extends R (see the first half of the proof of Lemma 6.25). Now let S be the least strongly finitary extension of R (it exists by Lemma 6.22). Obviously $S \subseteq \leq_{\theta_R}$. On the other hand, since $R \subseteq S$, by the above monotonicity observations $\leq_{\theta_R} \subseteq \leq_{\theta_S} = S$ (the last equality holds by Lemma 6.25). □

If R is weakly algebraic, $\leq_{\theta_R}$ can be characterized without explicit use of topologies:

Theorem 6.27 For any weakly algebraic preorder R on D, the preorder R^F defined by

$$x R^F y \iff (\forall a \in K(D).\ aRx \Rightarrow aRy)$$

is the least strongly finitary extension of R and is algebraic.

Proof. Reflexivity and transitivity of R^F is obvious. To show that $R \subseteq R^F$, check

$$xRy \Rightarrow (\forall a \in K(D).\ aRx \Rightarrow aRy) \iff x R^F y$$

To show that R^F is algebraic (and strongly finitary), check that it is finitary and weakly algebraic and use Lemma 6.21. For finitarity, consider any $x, y \in D$

and assume that $aR^F x \Rightarrow aR^F y$ for all $a \in K(D)$. Then if bRx for some $b \in K(D)$, also $bR^F x$ (see above) and, by the assumption, $bR^F y$. Now let $x = a = b$ in the definition of R^F to see that bRy. This works for arbitrary $b \in K(D)$, hence by definition, $xR^F y$ and R^F is finitary.

For weak algebraicity of R^F, consider any $a \in K(D)$ and $y \in D$ such that $aR^F y$. In particular, as a is finite, one has aRy (take $x = a$ in the definition of R^F). Since R is weakly algebraic, there exists a finite $b \leq y$ such that aRb, and since $R \subseteq R^F$, also $aR^F b$.

It remains to be checked R^F is the *least* strongly finitary extension of R. To this end, consider any strongly finitary preorder S such that $R \subseteq S$ and calculate, for any $x, y \in D$,

$$xR^F y \iff (\forall a. aRx \Rightarrow aRy) \Longrightarrow (\forall a. a \leq x \Rightarrow aSy) \Longrightarrow xSy$$

using the assumption that $\leq \subseteq R \subseteq S$. □

6.5 Compact Coalgebras

In the previous section, we obtained a characterization of those preorders on algebraic cpos that are specialization preorders of subtopologies of the Scott topology. This is the first step of a characterization of the specialization preorders of test suites taken from $\mathcal{P}^{\mathsf{BS}}$-coalgebras (these test suites are necessarily subtopologies of the Scott topology).

When looking for such a characterization in the next section, we will decide to restrict our attention only to those topologies for which the specialization preorders are weakly algebraic, leaving other topologies not treated. This leads to an important question: what conditions should be imposed on a coalgebra h that would guarantee the specialization preorder of the topology $k^*\sigma_S$ (where S is the final coalgebra and k is the coinductive extension of h) to be weakly algebraic? This question must be tackled, as the topology $k^*\sigma_S$ is the source of full abstraction for the final semantics (see Lemma 6.19). To this end, we give some simple definitions.

Definition 6.28 A coalgebra $h : D \to (\mathcal{P}^0 D)_\perp$ is *compact* if the coinductive extension k of h preserves finite elements.

The name *compact* was borrowed from the study [6] on cpo models for GSOS languages.

Definition 6.29 A topology θ on an algebraic cpo D is *closed under finitary intersections*, if for any finite $a \in D$, there is $a{\uparrow^{\leq_\theta}} \in \theta$.

Note that, by definition, $a{\uparrow^{\leq_\theta}}$ is the intersection of all $V \in \theta$ such that $a \in V$. This justifies the name "closed under finitary intersections". The two notions defined above are linked by

Lemma 6.30 If $h : D \to (\mathcal{P}^0 D)_\perp$ is compact then $k^*\sigma$ is closed under finitary intersections, where σ is the Scott topology on S and $k : D \to S$ is the coinductive extension of h.

Proof. Recall that $h^*\sigma$ can be viewed as an $\mathbb{O}$-test suite on D. Now

$$\begin{aligned} a{\uparrow}^{\leq_{h^*\sigma}} &= \{x \in D \mid a \leq_{h^*\sigma} x\} \\ &= \{x \in D \mid \forall V \in h^*\sigma.V(a) \leq V(x)\} \\ &= \{x \in D \mid \forall V' \in \sigma.V'(h(a)) \leq V'(h(x))\} \\ &= \{x \in D \mid h(a) \leq h(x)\} \\ &= \{x \in D \mid h(x) \in h(a){\uparrow}\} \\ &= \overline{\overrightarrow{h(a){\uparrow}} \circ h} \end{aligned}$$

Since h preserves finite elements, $h(a)$ is finite, hence

$$\frac{h(a){\uparrow} \in \sigma}{\overline{\overrightarrow{h(a){\uparrow}} \circ h} \in h^*\sigma}$$

□

The following easy lemmas show that topologies closed under finitary intersections correspond to weakly algebraic preorders:

Lemma 6.31 If a topology θ on D is closed under finitary intersections then the specialization preorder $\leq_\theta$ is weakly algebraic.

Proof. Assume θ closed under finitary intersections. In particular, for any finite $a \in D$, the set $a{\uparrow}^{\leq_\theta}$ is Scott-open. Weak algebraicity of $\leq_\theta$ is now straightforward. □

Lemma 6.32 If a preorder R on D is weakly algebraic, then for every $a \in K(D)$, $x \in D$,

$$a \leq_{\theta_R} x \iff aRx$$

Proof. Assume R weakly algebraic.

$\Longleftarrow$: see proof of Lemma 6.21.

$\Longrightarrow$: Assume $a \leq_{\theta_R} x$. It means that for any $V \in \theta_R$, if $a \in V$ then $x \in V$. Consider the set $a{\uparrow}^R$. It is obviously R-upper, and by weak algebraicity of R, also Scott-open, so it belongs to θ_R, hence $x \in a{\uparrow}^R$ and aRx. □

Lemma 6.33 If a preorder R on D is weakly algebraic then the topology θ_R is closed under finitary intersections.

Proof. We need to show that for any finite $a \in D$, the set $a{\uparrow}^{\leq_{\theta_R}}$ is Scott-open and R-upper. Since R is weakly algebraic, by Lemma 6.28 there is

$$a{\uparrow}^{\leq_{\theta_R}} = a{\uparrow}^R$$

The set on the right hand side is clearly R-upper, and its Scott-openness is straightforward by weak algebraicity of R. □

6.6 CPO-Bisimulations

Now we proceed to give a relational characterization of those topologies θ on D which lift a given coalgebra $h : D \to (\mathcal{P}^0 D)_\perp$ to a $\mathcal{P}^{\mathsf{BS}}$-coalgebras on $\mathbb{O}$-**TS**. To simplify matters, we restrict attention only to topologies closed under finitary intersections. This restriction, by Lemma 6.26, does not prevent the topology $k^*\sigma$ from being characterized if h is a compact coalgebra.

Definition 6.34 A preorder R on D is a *cpo-bisimulation* on a very strict coalgebra $h : D \to (\mathcal{P}^0 D)_\perp$ if $\leq \subseteq R$ and if for all $x, y \in D$ if xRy then

- $\forall x' \in hx. \forall a \leq x'. \exists y' \in hy. aRy'$
- if $\perp \notin hx$ then
  - $\perp \notin hy$
  - $\forall y' \in hy. \exists x' \in hx. x'Ry'$

Lemma 6.35 If R is a cpo-bisimulation on $h : D \to (\mathcal{P}^0 D)_\perp$ then the topology θ_R lifts h to a $\mathcal{P}^{\mathsf{BS}}$-coalgebra.

Proof. Assume R a cpo-bisimulation and take any $\mathbb{O}$-test $V \in \theta_R$. By definition of $\mathcal{P}^{\mathsf{BS}}$, it is enough to check that $w_\Box \circ (\mathcal{P}^0 V)_\perp \circ h \in \theta_R$ and $w_\Diamond \circ (\mathcal{P}^0 V)_\perp \circ h \in \theta_R$.

First consider the test $W_\Box = w_\Box \circ (\mathcal{P}^0 V)_\perp \circ h$. By definition it is strict and continuous, hence $\overline{W_\Box}$ is Scott-open on $D \setminus \{\perp\}$ and it is enough to check that it is R-upper. Take $x, y \in D$ such that $W_\Box x = \mathtt{tt}$ and xRy. By definition of $W_\Box$, for all $x' \in hx$ there is $Vx = \mathtt{tt}$. In particular, $\perp \notin hx$. Since R is a cpo-bisimulation, this means that $\perp \notin hy$ and for every $y' \in hy$ there exists $x' \in hx$ such that $x'Ry'$. Now $\overline{V}$ is R-upper and $Vx' = \mathtt{tt}$ for any $x' \in hx$, so also $Vy' = \mathtt{tt}$ for any $y' \in hy$, hence $W_\Box y = \mathtt{tt}$.

Now consider the test $W_\Diamond = w_\Diamond \circ (\mathcal{P}^0 V)_\perp \circ h$. Again, it is enough to show that $\overline{W_\Diamond}$ is R-upper. Take $x, y \in D$ such that $W_\Diamond x = \mathtt{tt}$ and xRy. By definition of $W_\Diamond$, there exists an $x' \in hx$ such that $Vx' = \mathtt{tt}$. Since $\overline{V}$ is Scott-open, there is an $a \leq x'$ such that $Va = \mathtt{tt}$. Now R is a cpo-bisimulation, so there is a $y' \in hy$ such that aRy'. But $\overline{V}$ is R-upper, hence $Vy' = \mathtt{tt}$ and $W_\Diamond y = \mathtt{tt}$. □

Lemma 6.36 Consider a very strict coalgebra $h : D \to (\mathcal{P}^0 D)_\perp$. For any topology θ on $D \setminus \{\perp\}$, if θ is closed under finitary intersections and if θ lifts h to a $\mathcal{P}^{\mathsf{BS}}$-coalgebra, then $\leq_\theta$ is a cpo-bisimulation.

Proof. Assume θ lifts h to a $\mathcal{P}^{\mathsf{BS}}$-coalgebra and that θ is closed under finitary intersections. To prove that $\leq_\theta$ is a cpo-bisimulation, assume any $x \leq_\theta y$ consider three cases:

1. Assume that there is $x' \in hx$ and a finite $a \leq x'$ such that for all $y' \in hy$ there is $a \not\leq_\theta y'$. Since θ is closed under finitary intersections, one has $a{\uparrow^{\leq_\theta}} \in \theta$. Consider the test

$$W_\Diamond = w_\Diamond \circ (\mathcal{P}^0(\overrightarrow{a{\uparrow^{\leq_\theta}}}))_\perp \circ h$$

The coalgebra h lifts to θ, hence $W_\Diamond \in \theta$. Now for any $y' \in hy$ one has $y' \notin a{\uparrow}^{\leq_\theta}$, hence $W_\Diamond y = \mathtt{ff}$. On the other hand, obviously $a \in a{\uparrow}^{\leq_\theta}$, hence $x' \in a{\uparrow}^{\leq_\theta}$ and $W_\Diamond = \mathtt{tt}$. As a result, $x \not\leq_\theta y$.

2. Assume $\bot \notin hx$ and $\bot \in hy$. Consider the test

$$W = w_\Box \circ (\mathcal{P}^0(\overrightarrow{D \setminus \{\bot\}}))_\bot \circ h$$

Since θ is a topology on $D \setminus \{\bot\}$ and lifts h to a $\mathcal{P}^{\mathsf{BS}}$-coalgebra, one has $W \in \theta$. Moreover, $Wx = \mathtt{tt}$ and $Wy = \mathtt{ff}$, hence $x \not\leq_\theta y$.

3. Assume $\bot \notin hx$ and that there is $y' \in hy$ such that for all $x' \in hx$ there is $x' \not\leq_\theta y'$. This means that for every $x' \in hx$ there exists a test $V_{x'} \in \theta$ such that $V_{x'}x' = \mathtt{tt}$ and $V_{x'}y' = \mathtt{ff}$. Consider the test

$$W_\Box = w_\Box \circ (\mathcal{P}^0(\bigvee_{x' \in \gamma x} V_{x'}))_\bot \circ h$$

The coalgebra h lifts to θ, hence $W_\Box \in \theta$ (in particular, note that θ is closed under arbitrary unions). Obviously $W_\Box x = \mathtt{tt}$ but $W_\Box y = \mathtt{ff}$, hence $x \not\leq_\theta y$.

□

We are now ready to give a characterization of $\mathcal{P}^{\mathsf{BS}}$-coalgebras:

Theorem 6.37 For a given very strict coalgebra $h : D \to (\mathcal{P}^0 D)_\bot$, specialization preorders of those topologies closed under finitary intersections that lift h to a $\mathcal{P}^{\mathsf{BS}}$-coalgebra are exactly the preorders R^F for weakly algebraic cpo-bisimulations R on h.

Proof.

$\Longrightarrow$: Let θ be closed under finitary intersections and assume that it lifts h to a $\mathcal{P}^{\mathsf{BS}}$-coalgebra. Then

- $\leq_\theta$ is a cpo-bisimulation, by Lemma 6.34,
- $\leq_\theta$ is weakly algebraic, by Fact Lemma 6.31,
- $\leq_\theta$ is finitary, by Lemmas 6.21 and 6.23, hence (as is easily checked) $\leq_\theta^F = \leq_\theta$.

$\Longleftarrow$: Take R a weakly algebraic cpo-bisimulation on h. Then

- θ_R is closed under finitary intersections, by Lemma 6.33,
- h lifts to θ_R, by Lemma 6.36,
- $\leq_{\theta_R} = R^F$, by Theorems 6.26 and 6.27.

□

Theorem 6.37 gives a characterization of those $\mathcal{P}^{\mathsf{BS}}$-coalgebras

$$h : \langle D, \theta \rangle \to \left\langle (P^0 D)_\perp D, \mathcal{P}^{\mathsf{BS}}_D \theta \right\rangle$$

for which θ is a topology closed under finitary intersections. This leaves some $\mathcal{P}^{\mathsf{BS}}$-coalgebras not characterized, but if $h : D \to (\mathcal{P}^0 D)_\perp$ is compact, then the least θ lifting h to a $\mathcal{P}^{\mathsf{BS}}$-coalgebra is indeed characterized and the full abstraction result can be stated in terms of this characterization:

Corollary 6.38 Let $h : D \to (\mathcal{P}^0 D)_\perp$ be compact, and let $k : D \to S$ be the inductive extension of h. For any $x, y \in D$,

$$k(x) \leq k(y) \iff x R^F y$$

where R is the largest weakly algebraic cpo-bisimulation on h.

Proof. By Theorems 6.37, 6.26 and Lemma 6.27. □

Remark 6.39 When coalgebras with flat carriers are considered, the statement of Theorem 6.37 can be simplified. Indeed, then $K(D) = D$, all preorders are weakly algebraic and finitary, and the definition of cpo-bisimulation (Definition 6.34) simplifies to the definition of partial bisimulation (Definition 6.10) on the transition system with divergence corresponding to the coalgebra in question.

Two problems remain open. Firstly, the author is not able to find any cpo-bisimulation on a compact coalgebra that would not be weakly algebraic. If all such cpo-bisimulations are weakly algebraic, Theorem 6.31 and Corollary 6.32 can be simplified, by replacing weakly algebraic cpo-bisimulations by simply cpo-bisimulations. Secondly, a relational characterization of *all* $\mathcal{P}^{\mathsf{BS}}$-coalgebras is presently missing. The results of Abramsky [2], who obtained a relational characterization of the full abstraction result for all transition systems with divergence, suggest that a satisfactory characterization may indeed exist.

Chapter 7

Adding Recursive Constructs to Bialgebraic Semantics

In Chapter 6, the interpretation of processes and transition systems in a category of cpos was briefly motivated by issues related to recursive operators in process languages. It was also mentioned that one can conveniently describe the behaviour of such operators treating them as syntactic abbreviations of their infinite expansions.

In this chapter, we elaborate on this issue, considering problems related to the interpretation of recursive operators in the framework of bialgebraic semantics. More specifically, we show how to merge systems of (possibly unguarded) recursive equations, modelled as certain natural transformations, with structural operational rules modelled as distributive laws, as in the bialgebraic framework of abstract GSOS.

This chapter is not directly related to the test suite framework developed in Chapters 3-6, and can be read largely independently, except for occasional references to Chapter 2 and to preliminaries in Chapter 6.

The elegant framework of abstract GSOS (see Section 2.7) covers many interesting examples of simple languages [85, 84, 13, 49]. However, it is not immediately clear how to extend it to cover variable binding and/or recursive constructs. The problem with variable binding is to some extent syntactical: signatures with variable binding constructs are not as easily expressed as endofunctors as are ordinary basic constructs of process languages. As was shown in [31] and [30], this can be remedied by interpreting operational rules in a suitable presheaf category. At the expense of making the framework considerably more complex, the authors were able to formalize the operational semantics for name-passing and value-passing constructs.

The problem with recursive constructs is of somewhat different nature. Even when variable binding is not around, it is not clear how to interpret such constructs in the bialgebraic framework. As an example, consider an unary language construct `loop` designed to repeat some computation infinitely. Traditionally, there are two ways of writing operational rules for this construct. One possibility is to write (in the presence of a sequential composition operator '`;`' in the language)

$$\frac{}{\texttt{loop t} \rightarrow \texttt{t;loop t}}$$

This rule is structural and accurately describes the intended meaning of the looping construct. However in certain situations, e.g., where the intended operational model of the language should be a labelled transition system, and not an unlabelled one, this rule is a source of problems. To merge this rule with other rules that generate a labelled transition system, one has to change the behaviour functor of the intended operational model. More importantly, the coinductive extension of the derived model does not ignore the 'silent' transitions associated with the above rule. As a result, programs like `loop a` and `a;loop a` are mapped differently by the final coalgebra semantics, which contradicts the intended meaning of the looping construct.

To remedy this problem on an abstract level, a general coalgebraic theory of weak bisimulation is needed. Some work in this direction has been done [74, 75], but no satisfactorily abstract results have been obtained yet. This means that the above rule is hard to fit into the bialgebraic framework so far.

Another option to describe the looping construct is to give the rule

$$\frac{\mathtt{t};\mathtt{loop\ t} \xrightarrow{a} \mathtt{t}'}{\mathtt{loop\ t} \xrightarrow{a} \mathtt{t}'}$$

This rule does not introduce any unwelcome silent transitions. However, it is not structural: to compute the intended operational model for a language with this rule, one uses a fixpoint construction rather than induction. This makes it impossible to fit this rule directly into the bialgebraic framework.

Here we consider another approach. Following general guidelines from [69], recursive constructs are treated separately from the recursion-free ones, and described not with operational rules, but with recursive equations. The equations are then merged with the bialgebraic models obtained for the recursion-free fragment of the language.

This general approach was first used in [84] for the case of guarded recursion. However, unguarded equations like

$$\mathtt{loop\ t} = \mathtt{t};\mathtt{loop\ t}$$

have not been treated so far.

In this chapter, the approach introduced in [84] is modified and extended to deal with a wide range of recursive equations, including many unguarded ones. However, to keep things simple only constructs without variable binding are considered, to avoid moving to a sophisticated presheaf category.

Since unguarded recursive equations may be a source of partiality (divergence), it is convenient to interpret the operational rules and the recursive equations in a suitable category of domains rather than in the category of sets. The examples aimed to explain the constructions introduced in the paper use the category $\mathbf{Cppo}_\perp$ of pointed cpos and continuous, strict functions. However, all the constructions used can be interpreted in any $\mathbf{Cppo}_\perp$-enriched category equipped with the usual structure needed to interpret the bialgebraic framework of [85].

The structure of the chapter is as follows. Section 7.1 introduces some categorical preliminaries together with certain fixpoint constructions used in the

following. In Section 7.2, two motivating examples are shown, and the developments of the remaining sections is introduced on informal level. The examples shown use $\mathbf{Cppo}_\perp$ as the underlying category. One of the examples deals with a looping construct `loop`, and another one with a more sophisticated construct `unfolding`, which corresponds roughly to the general recursive construct μ restricted to a single recursive variable.

In Section 7.3 recursive equations are formalized abstractly as unfolding rules, and in Sections 7.4–7.6 it is shown how to merge unfolding rules with the bialgebraic framework.

In the developments presented in Sections 7.4–7.6 it repeatedly appears that the recursive equation for the construct `unfolding` considered in one of the examples is far less structured than the one for the construct `loop`. This motivates the definition of a regular unfolding rule in Section 7.7, which is satisfied by the latter but not by the former. Regular unfolding rules have some useful properties: in particular, they allow to construct a distributive law (abstract operational rules) for the full language from a distributive law for the recursion-free fragment.

7.1 Preliminaries

This section contains standard definitions and results used in this chapter. The reader is also advised to refer to Chapter 2 for definitions related to the bialgebraic framework, and to Section 6.1 for basic definitions concerning cpos.

A *pointed endofunctor* $\langle T, \eta\rangle$ on a category $\mathbb{C}$ is an endofunctor T on C together with a natural transformation $\eta : Id \to T$. For every *monad* $\langle T, \eta, \mu\rangle$ (see Definition 2.18), the pair $\langle T, \eta\rangle$ is obviously a pointed endofunctor. We often omit the names of the above natural transformations and speak of a pointed endofunctor T, or of a monad T.

An algebra for a pointed endofunctor T is an algebra $h : TX \to X$ for the endofunctor T such that $h \circ \eta_X = id_X$.

If T is a (part of a) monad (a pointed endofunctor) then by a T-algebra we will mean an algebra for the monad (the pointed endofunctor), unless otherwise stated.

A *monad morphism* between monads $\langle T, \eta, \mu\rangle$ and $\langle T', \eta', \mu'\rangle$ is a natural transformation $t : T \to T'$ such that

$$\eta' = t \circ \eta$$
$$t \circ \mu = \mu' \circ tT' \circ Tt$$

A *copointed endofunctor* (H, π) is an endofunctor H together with a natural transformation $\pi : H \to Id$.

A coalgebra for a copointed endofunctor (H, π) is a coalgebra $k : X \to HX$ for the endofunctor H such that $\pi_X \circ k = id_X$.

Any endofunctor B on a category with binary products cofreely generates a copointed endofunctor $H = Id \times B$ with $\pi = \pi_1$. Obviously then H-coalgebras are in one-to-one correspondence with B-coalgebras.

A *distributive law* of a pointed endofunctor T over a copointed endofunctor H is a natural transformation $\lambda : TH \to HT$ such that

$$\lambda \circ \eta H = H\eta$$
$$T\pi = \pi T \circ \lambda$$

If, moreover, T is a monad and

$$\lambda \circ \mu H = H\mu \circ \lambda T \circ T\lambda$$

then λ is called a distributive law of the monad T over the copointed endofunctor H.

If $H = Id \times B$ is the copointed endofunctor cofreely generated by an endofunctor B, then distributive laws of the pointed endofunctor (monad) T over H are equivalent to natural transformations $\lambda : TH \to BT$ such that $\lambda \circ \eta H = B\eta \circ \pi_2$ (and respectively, $\lambda \circ \mu H = B\mu \circ \lambda T \circ T(T\pi_1, \lambda)$). With a slight abuse of language, such natural transformations will also be called distributive laws of the copointed endofunctor (monad) T over H.

For a distributive law $\lambda : TH \to BT$ of the pointed endofunctor T over the copointed endofunctor H, a λ-*model* with carrier X is a pair $TX \xrightarrow{h} X \xrightarrow{k} BX$ of a T-algebra and B-coalgebra, satisfying the pentagonal law:

$$k \circ h = Bh \circ \lambda_X \circ T(id, k)$$

If, additionally, T is a monad and λ is a distributive law of the monad over H, then the λ-model is called a λ-*bialgebra* (see Definition 2.27).

A category is called $\mathbf{Cppo}_\perp$-enriched, if its homsets are pointed cpos (see Definition 6.1) and if composition is continuous and strict in both arguments. Since composition is strict, the collection of bottom elements of homsets may be viewed as a natural transformation $\perp$ between any two given endofunctors.

In the following sections many morphisms in a $\mathbf{Cppo}_\perp$-enriched category will be defined as fixpoints of certain continuous functions on homsets. In many applications the least fixpoints are the ones of the most interest. However, for our purposes it will be more useful to consider only fixpoints that satisfy certain conditions (for example, that factorize through some given morphism) and then take the least fixpoint from this class. This section presents some techniques aimed at defining and reasoning about such morphisms.

First, a straightforward generalization of Tarski's Fixpoint Theorem:

Theorem 7.1 (Tarski) Consider a cppo V and a continuous function $\Phi : V \to V$. For every $f \in V$ such that $\Phi f \geq f$, there exists the least fixpoint of Φ greater or equal to f, denoted by $\Phi^*(f)$. Moreover,

$$\Phi^*(f) = \bigsqcup_{n \in N} \Phi^n f$$

Note that for any continuous $\Phi : V \to V$, the set V_Φ of those elements $f \in V$ for which $\Phi f \geq f$, is a sub-cppo of V, i.e., it contains the bottom element $\perp$ and it is closed under lubs. It is easy to observe that $\Phi^* : V_\Phi \to V$ is a continuous operation (hence the notation).

In the following sections we shall prove many properties of morphisms defined as fixpoints. Most of these properties have a form of the commuting diagram involving two such morphisms. In particular, this pattern appears when proving naturality of families of morphisms defined as fixpoints. To prove such properties the following two lemmas will be used.

Lemma 7.2 Consider the following shape in a $\mathbf{Cppo}_\perp$-enriched category C:

$$\begin{array}{ccccc} X & \xrightarrow{k_1} & Y_1 & \xrightarrow{f_1} & Y_1' \\ {\scriptstyle k_2}\downarrow & & & & \downarrow{\scriptstyle l_1} \\ Y_2 & \xrightarrow[f_2]{} & Y_2' & \xrightarrow[l_2]{} & Z \end{array}$$

and two continuous functions

$$\Phi_1 : C(Y_1, Y_1') \to C(Y_1, Y_1') \qquad \Phi_2 : C(Y_2, Y_2') \to C(Y_2, Y_2')$$

such that $\Phi_1(f_1) \geq f_1$ and $\Phi_2(f_2) \geq f_2$. If

- $l_1 \circ f_1 \circ k_1 = l_2 \circ f_2 \circ k_2$
- $l_1 \circ x \circ k_1 = l_2 \circ y \circ k_2$ implies $l_1 \circ \Phi_1(x) \circ k_1 = l_2 \circ \Phi_2(y) \circ k_2$

then $l_1 \circ \Phi_1^*(f_1) \circ k_1 = l_2 \circ \Phi_2^*(f_2) \circ k_2$.

Proof. First show by induction that for any n, $l_1 \circ \Phi_1^n(f_1) \circ k_1 = l_2 \circ \Phi_2^n(f_2) \circ k_2$. Then use continuity of composition. □

In other cases, another proof principle will be more useful:

Lemma 7.3 In the setting like in Lemma 7.2, if

- $l_1 \circ f_1 \circ k_1 \leq l_2 \circ \Phi_2^*(f_2) \circ k_2$
- $l_1 \circ x \circ k_1 \leq l_2 \circ y \circ k_2$ implies $l_1 \circ \Phi_1(x) \circ k_1 \leq l_2 \circ \Phi_2(y) \circ k_2$
- $l_1 \circ \Phi_1^*(f_1) \circ k_1 \geq l_2 \circ f_2 \circ k_2$
- $l_1 \circ x \circ k_1 \geq l_2 \circ y \circ k_2$ implies $l_1 \circ \Phi_1(x) \circ k_1 \geq l_2 \circ \Phi_2(y) \circ k_2$

then $l_1 \circ \Phi_1^*(f_1) \circ k_1 = l_2 \circ \Phi_2^*(f_2) \circ k_2$.

Proof. To prove $l_1 \circ \Phi_1^*(f_1) \circ k_1 \leq l_2 \circ \Phi_2^*(f_2) \circ k_2$, show by induction from the first two assumptions that for any n, $l_1 \circ \Phi_1^n(f_1) \circ k_1 \leq l_2 \circ \Phi_2^*(f_2) \circ k_2$. Then use continuity of composition.

The proof of $l_1 \circ \Phi_1^*(f_1) \circ k_1 \geq l_2 \circ \Phi_2^*(f_2) \circ k_2$ proceeds analogously using the last two assumptions. □

The above lemmas can be used to prove naturality of families of morphisms defined as fixpoints, by taking both k_1 and l_2 to be identities. This method of proving naturality will be called *square commutation by fixpoint induction*.

Finally, a special class of monomorphisms will be considered in any $\mathbf{Cppo}_\perp$-enriched category:

Definition 7.4 A morphism $f : A \to B$ in a $\mathbf{Cppo}_\perp$-enriched category is an *upper section*, if there exists a morphism $g : B \to A$ (called *left inverse* to f) such that

$$g \circ f = id_A \qquad \text{and} \qquad f \circ g \leq id_B$$

To prove various properties involving upper sections, the following easy lemma will be used:

Lemma 7.5 Assume $f : A \to B$ is an upper section with left inverse g. For any $h : B \to C$, $k : A \to C$, if $k \circ g \circ f \leq h \circ f$ (or, equivalently, $k \leq h \circ f$), then $k \circ g \leq h$.

Proof. $k \circ g = k \circ (g \circ f) \circ g \leq h \circ f \circ g \leq h$ □

For our purposes, especially interesting examples of upper sections will be units of some monads:

Definition 7.6 A variable classifier for a monad (T, η, μ) on a $\mathbf{Cppo}_\perp$-enriched category C is a natural transformation $\upsilon : T \to Id$ such that

- all η_X are upper sections with υ_X left inverses,
- all υ_X are algebras for the monad T.

In particular, if T is freely generated by some endofunctor Σ (see Section 2.6.1), then T can be equipped with a variable classifier:

$$\upsilon = [id, \perp] \circ \iota$$

where $\iota : T \to Id + \Sigma T$ is the isomorphism given by the structure of freely generated monad, and $\perp : \Sigma T \to T$ is the natural transformation composed of the least elements of the respective homsets.

7.2 Motivating Examples

Consider a language with the following simple syntax:

$$t ::= \mathtt{0} \mid \mathtt{a} \mid t;t \mid t+t$$

(where a ranges over some fixed set of actions A), equipped with the following standard small-step operational semantics:

$$\frac{}{\mathtt{a} \xrightarrow{a} \mathtt{0}} \qquad \frac{\mathtt{x} \xrightarrow{a} \mathtt{x}'}{\mathtt{x};\mathtt{y} \xrightarrow{a} \mathtt{x}';\mathtt{y}} \qquad \frac{\mathtt{x} \not\longrightarrow \quad \mathtt{y} \xrightarrow{a} \mathtt{y}'}{\mathtt{x};\mathtt{y} \xrightarrow{a} \mathtt{y}'}$$

$$\frac{\mathtt{x} \xrightarrow{a} \mathtt{x}'}{\mathtt{x}+\mathtt{y} \xrightarrow{a} \mathtt{x}'} \qquad \frac{\mathtt{y} \xrightarrow{a} \mathtt{y}'}{\mathtt{x}+\mathtt{y} \xrightarrow{a} \mathtt{y}'}$$

In [84] it was shown how such simple languages can be interpreted in the bialgebraic framework in an arbitrary category with enough structure. For the

purposes of the example here, the category $\mathbf{Cppo}_\perp$ of pointed cpos and strict continuous functions will be used.

The syntax of the simple language mentioned above corresponds to an endofunctor Σ on $\mathbf{Cppo}_\perp$:

$$\Sigma X = 1_\perp \oplus A_\perp \oplus X_\perp \otimes X_\perp \oplus X_\perp \otimes X_\perp = (1 + A + X \times X + X \times X)_\perp$$

where $\oplus$ and $\otimes$ are the coalesced sum and the smash product of pointed cpos (see [67]), $+$ and $\times$ are the disjoint sum and the Cartesian product of cpos, and $(\_)_\perp$ is the lifting operation on cpos. This functor freely generates a monad T on $\mathbf{Cppo}_\perp$. Elements of TX are (possibly infinite) terms built over (a pointed cpo of) variables X, with some sub-terms replaced by the bottom element $\perp$, and with the ordering induced from the ordering on X, with the remark that $\perp$ is smaller than any other term.

One possible behaviour endofunctor B for interpreting the operational rules shown above is

$$BX = \mathcal{P}^0(A_\perp \otimes X_\perp) = \mathcal{P}^0((A \times X)_\perp)$$

where $\mathcal{P}^0$ is the Plotkin powerdomain with the empty set adjoined (see Chapter 6). As it turns out, the above operational semantics corresponds to a natural transformation

$$\rho : \Sigma(Id \times B) \to BT$$

defined by cases as follows:

$$\begin{aligned}
\rho_X(\iota_1(\mathtt{0})) &= \emptyset \\
\rho_X(\iota_2(\mathtt{a})) &= \{(a, \mathtt{0})\} \\
\rho_X(\iota_3(x_1, \beta_1, x_2, \beta_2)) &= \begin{cases} \{(a, \iota_3(x_1', x_2)) \mid (a, x_1') \in \beta_1\} \text{ if } \beta_1 \neq \emptyset \\ \beta_2 \text{ otherwise} \end{cases} \\
\rho_X(\iota_4(x_1, \beta_1, x_2, \beta_2)) &= \beta_1 \cup \beta_2
\end{aligned}$$

where X is a pointed cpo (of variables), x and b range over X and BX respectively, and the ι_i are the coproduct injections to ΣX. It is easy to see that ρ_X is continuous and, when suitably extended to act on bottom elements, it is also strict. Obviously, ρ is also natural in X.

In [56] it was shown that a natural transformation ρ of the type as above is equivalent to a distributive law $\lambda : TH \to BT$ of the monad T freely generated by Σ over the copointed endofunctor H cofreely generated by B. As shown in [83, 85], ρ also induces a lifting T_λ of the monad T to the category of B-coalgebras, i.e., an endofunctor on the category of B-coalgebras such that for every B-coalgebra $k : X \to BX$ the following diagram commutes:

$$\begin{array}{ccccc}
X & \xrightarrow{\eta_X} & TX & \xleftarrow{\mu_X} & TTX \\
{\scriptstyle k}\downarrow & & \downarrow{\scriptstyle T_\lambda k} & & \downarrow{\scriptstyle T_\lambda T_\lambda k} \\
BX & \xrightarrow[B\eta_X]{} & BTX & \xleftarrow[B\mu_X]{} & BTTX
\end{array}$$

Moreover, for any B-coalgebra $k : X \to BX$, there is

$$T_\lambda k = \lambda_X \circ T(id, k)$$

When used in diagrams as above, in certain situations the notation $T_\lambda k$ might lead to some confusion. One way to read it is to silently convert a morphism $k \in C(X, BX)$ to an object in the category of B-coalgebras, apply the object part of the functor T_λ to it and silently convert the result back to a morphism in C. Another way might be to treat k as a morphism between some B-coalgebras and apply the morphism part of the functor T_λ to it. In this chapter, only the former interpretation is used.

Let us extend the example shown above in two independent ways, adding some simple recursive constructs to it.

Example A: The unary looping construct `loop`. This is done by extending the syntax as follows:

$$t ::= \ldots \mid \texttt{loop}\ t$$

with the intended meaning captured by the recursive equation

$$\texttt{loop}\ t = t; \texttt{loop}\ t$$

Example B: The unary construct `unfolding`, inspired by a similar construct from Mosses's action semantics [55]. This is a version of the general recursive construct μ, restricted to only one fixed recursive variable, which is hence treated as a constant (we use the constant `0` for this purpose). The appropriate syntax extension is

$$t ::= \ldots \mid \texttt{unfolding}\ t$$

with the intended meaning captured by the recursive equation

$$\texttt{unfolding}\ t = t[\texttt{0} \mapsto \texttt{unfolding}\ t]$$

where $t[r \mapsto s]$ is the standard notation for substitution on terms.

The reuse of the constant `0` for the semantics of `unfolding` makes this recursive construct rather useless in practice. However, this does not formally change its semantic features, and makes further developments a bit simpler to present, since now the syntactic extensions in both examples correspond to the same extension of the endofunctor Σ to a new signature endofunctor Σ':

$$\Sigma' X = \Sigma X \oplus X_\perp$$

The syntactic monad freely generated by Σ' will be called T'.

Note that neither of the recursive equations mentioned here are guarded, and any of the above constructs is a potential source of divergence. Indeed, both terms `loop 0` and `unfolding 0` are immediately diverging. This explains the decision to interpret the language in a category of cpos, rather than in the category of sets.

The remainder of this section is devoted to an informal presentation of the technical developments of the following sections. This is to provide the reader with some intuition about the results presented in this chapter.

In both examples above, the recursive equation given might be seen as a natural transformation

$$r : T' \to TT'$$

which, given a term in $T'X$, performs one step of syntactic unfolding so that some recursion-free constructs appear on top of it, and splits the resulting term in two 'layers'. For instance, in example A,

$$\texttt{loop a} \rightsquigarrow (\_;\_)[\texttt{a}, \texttt{loop a}]$$

where $\rightsquigarrow$ describes the action of r on terms. The notation $(t)[t_1, \ldots, t_n]$ (where t is recursion-free and may contain some place-holders) is to represent a term split in two 'layers', i.e., an element of $TT'X$.

In example B, one can have for instance

$$\begin{aligned}
&\texttt{unfolding(a; 0)} \rightsquigarrow (_;_)[\texttt{a}, \texttt{unfolding(a; 0)}]\\
&\texttt{unfolding a} \rightsquigarrow (\texttt{a})[]\\
&\texttt{unfolding 0} \rightsquigarrow (_)[\texttt{unfolding 0}]
\end{aligned}$$

In a sense, natural transformations $r : T' \to TT'$ may be viewed as recursive specifications in the sense of, e.g., [14], that is, as sets of recursive equations with terms from $T'X$ playing the rôle of recursion variables.

A question arises how r should act on constructs from the recursion-free fragment of the language. Several options are available here. Until the end of Section 7.6, it will not be specified if, for example

$$\begin{aligned}
&\texttt{a;(b; c)} \rightsquigarrow (_;_)[\texttt{a}, \texttt{b; c}], \text{ or}\\
&\texttt{a;(b; c)} \rightsquigarrow (\texttt{a;(b;}_\texttt{))[c]}, \text{ or even}\\
&\texttt{a;(b; c)} \rightsquigarrow (\texttt{a;(b; c))}[]
\end{aligned}$$

It will only be required (in the definition of a decomposition structure in Section 7.3) that after repeating many steps of such 'unfolding' of a recursion-free term, eventually the entire term will appear in the top 'layer'. In other words, the option $\texttt{a;(b; c)} \rightsquigarrow (\_)[\texttt{a;(b; c)}]$ will be excluded.

In Section 7.7, when considering *regular* unfolding rules, the first of the above options (as small a portion of syntax is unfolded as possible) will be specifically required.

However, all the above options have something in common: all recursion-free constructs are left unchanged when unfolded. This will be properly formalized in the definition of an unfolding rule in Section 7.3.

It is reasonable to expect that if a term is built of some variables, then 'unfolding' a variable leaves it intact.

Given an 'unfolding rule' r, one can repeat its action many times. In the limit, all recursive constructs are wiped out from a given term. This construction is formalized as a natural transformation $\bar{r} : T' \to T$ in Section 7.4. This transformation replaces recursive constructs with their infinite expansions, and leaves the remaining constructs unchanged.

Given operational rules for the recursion-free fragment of the language (formalized as a distributive law $\lambda : TH \to BT$), and an unfolding rule $r : T' \to TT'$, one can define operational rules for the full language (formalized as a distributive law $\lambda^r : T'H \to BT'$ in Section 7.5). Somewhat informally

speaking, the new operational rules are defined by the following fixpoint construction: if $t \rightsquigarrow (s)[s_1, \ldots, s_n]$ and if

$$\frac{\frac{\Gamma_1}{s_1 \overset{a_1}{\to} s_1'} \quad \cdots \quad \frac{\Gamma_n}{s_n \overset{a_n}{\to} s_n'} \qquad \Gamma}{s[s_1, \ldots, s_n] \overset{a}{\to} t'}$$

(where Γ, Γ_i are some sets of premises) is a correct derivation (where the vertical dots represent a derivation using only the rules for the recursion-free fragment), then

$$\frac{\Gamma \cup \Gamma_1 \cup \cdots \cup \Gamma_n}{t \overset{a}{\to} t'}$$

is a valid rule.

Note the difference between this approach (formalized in Section 7.5) and the second traditional approach shown in the introduction to this chapter. Here the burden of a fixpoint construction is shifted to the definition of operational rules, but the rules themselves are structural, and the intended operational model can be derived from them inductively. This will allow to merge recursive equations with the bialgebraic framework.

The intended operational model is derived from T'_{λ^r}, the lifting of the pointed endofunctor T' to B-algebras which is associated with the distributive law λ^r. Intuitively, the operational model unfolds a given term according to the rule r, until the resulting term exhibits some behaviour according to the rules λ. For instance, in the example A above, the intended operational model $T'_{\lambda^r}(0)$ maps a term $\mathtt{loop}(\mathtt{a}; \mathtt{b})$ to a term $\mathtt{b}; \mathtt{loop}(\mathtt{a}; \mathtt{b})$ together with the action a.

In the remainder of this chapter the constructions hinted above are defined formally.

7.3 Unfolding Rules

The ideas presented in the preceding section can be formalized in an arbitrary $\mathbf{Cppo}_\perp$-enriched category C with products and freely generated monads of syntactic endofunctors.

From now on assume that all endofunctors on C introduced here are locally continuous (i.e., continuous as functions on homsets).

Assume we are given bialgebraic operational semantics for (the recursion-free fragment of) a language, in the sense of [85], i.e.,

- A monad $\langle T, \eta, \mu \rangle$ on C,
- A behaviour endofunctor $B : C \to C$, with the copointed endofunctor $H = Id \times B$ cofreely generated by it,
- A set of abstract GSOS rules, i.e., a distributive law of the monad T over the copointed endofunctor H:

$$\lambda : TH \to BT$$

To introduce recursive equations, consider additionally

- A monad $\langle T', \eta', \mu' \rangle$ (the full language),
- A monad morphism $t : T \to T'$.

Assume moreover, that monads T and T' come with variable classifiers υ and υ' respectively, and that $\upsilon' \circ t = \upsilon$.

For any morphism $\gamma_X : TX \to TTX$, one can define a function between cppos $\tilde{\Phi}_X : C(TX, TX) \to C(TX, TX)$ as follows:

$$\tilde{\Phi}_X \frac{TX \xrightarrow{f} TX}{TX \xrightarrow[\gamma_X]{} TTX \xrightarrow[Tf]{} TTX \xrightarrow[\mu_X]{} TX}$$

It is easy to see that $\tilde{\Phi}_X$ is continuous on the cppo $C(TX, TX)$, since T is locally continuous and C is **Cppo**-enriched.

Assume moreover that $\gamma_X \circ \eta_X = T\eta_X \circ \eta_X$, and define $\tilde{e}_X = \eta_X \circ \upsilon_X$. It is easily checked that $\tilde{e}_X \circ \eta_X = \tilde{\Phi}_X(\tilde{e}_X) \circ \eta_X$, hence, by Lemma 7.5, $\tilde{e}_X \leq \tilde{\Phi}_X(\tilde{e}_X)$ and an 'infinite decomposition' map $\bar{\gamma}_X : TX \to TX$ can be defined as follows:

$$\bar{\gamma}_X = \tilde{\Phi}^*_X(\tilde{e}_X)$$

Definition 7.7 A decomposition structure on T is a natural transformation $\gamma : T \to TT$ such that

- $\gamma \circ \eta = T\eta \circ \eta$
- $\mu \circ \gamma = id$
- $\bar{\gamma}_X = id_{TX}$ for any object X

Examples ctd. A decomposition structure $\gamma : T \to TT$ for the simple syntax T shown in Section 7.2 can be derived from a natural transformation $\gamma^0 : \Sigma \to TT$, defined by cases as follows:

$$\begin{aligned} \gamma^0_X(\mathtt{0}) &= (\mathtt{0})[] \\ \gamma^0_X(\mathtt{a}) &= (\mathtt{a})[] \\ \gamma^0_X(x_1;x_2) &= (\_;\_)[x_1, x_2] \\ \gamma^0_X(x_1+x_2) &= (\_+\_)[x_1, x_2] \end{aligned}$$

Then $\gamma : T \to TT$ defined as

$$\gamma_X = [T\eta_X \circ \eta_X, T\mu_X \circ \gamma^0_{TX}] \circ \psi_X$$

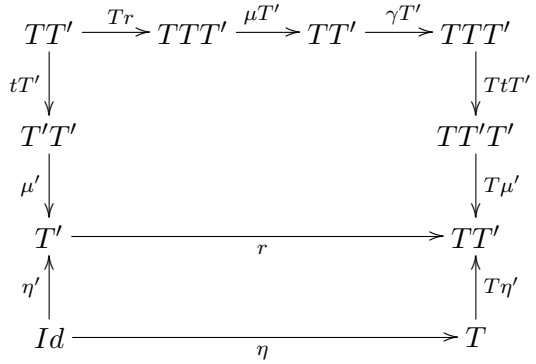
(where $\psi_X : TX \to X + \Sigma TX$ is the isomorphism arising from the free monad structure of T) is a decomposition structure for T.

As the informal discussion in Section 7.2 suggests, an 'unfolding rule' from the 'full' language T' to its 'recursion-free fragment' T might be viewed as a natural transformation $r : T' \to TT'$. This is formalized as follows.

Definition 7.8 An unfolding rule from T' to T (based on a decomposition structure γ on T) is a natural transformation

$$r : T' \to TT'$$

such that the following diagram commutes:

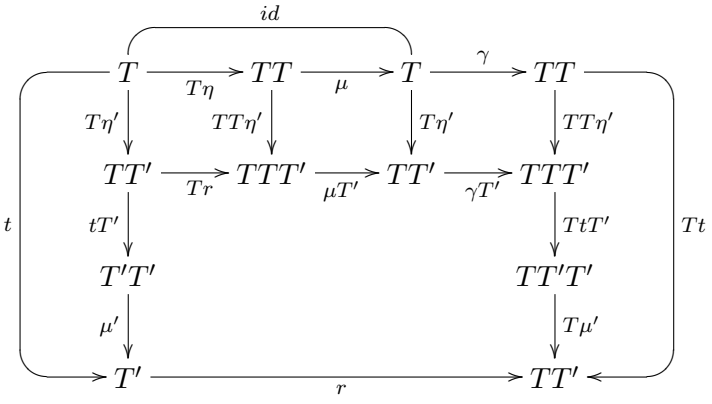


Intuitively, the bottom part of the above diagram means that variables are unfolded trivially. The top part means that the action of r on parts of terms built from syntax T is defined by γ. In particular, the following result holds:

Lemma 7.9 For any unfolding rule $r : T' \to TT'$ based on $\gamma : T \to TT$,

$$r \circ t = Tt \circ \gamma$$

Proof. Everything in the following diagram commutes:



□

Example A ctd. To define an unfolding rule $r : T' \to TT'$, one only needs to define its action on terms where the top construct is not present in the recursion-free syntax T. The action on other terms can be defined canonically, using the decomposition structure γ on T. Having defined γ from γ^0 as above, this is particularly easy: take $r_X(\iota_2(t)) = T\mu'_X \circ \gamma^0_{T'X}(t)$, where $\iota_2 : \Sigma T'X \to T'X$. For other terms, one can take e.g.,

$$r_X(\texttt{loop}\ t) = (\_;\_)[t, \texttt{loop}\ t]$$

Example B ctd. As before, one only has to define r on terms with new constructs at the top. Here take e.g.,

$$r_X(\texttt{unfolding}\ t) = (\_)[t[\texttt{0} \mapsto \texttt{unfolding}\ t]]$$

7.4 Infinite Unfolding

Given an unfolding rule $r : T' \to TT'$ one can translate any term built over the monad T' to a term over the monad T using a certain fixpoint construction, which intuitively corresponds to performing infinitely many unfolding steps.

Formally, given r, for any fixed object X in C, one can define a function $\Phi_X : C(T'X, TX) \to C(T'X, TX)$ acting as follows:

$$\Phi_X \frac{T'X \xrightarrow{f} TX}{T'X \xrightarrow[r_X]{} TT'X \xrightarrow[Tf]{} TTX \xrightarrow[\mu_X]{} TX}$$

It is easy to see that Φ_X is continuous on the cppo $C(T'X, TX)$, since T is locally continuous and C is **Cppo**-enriched.

Given an object X, the 'infinite unfolding' map $\bar{r}_X : T'X \to TX$ is defined as follows:

$$\bar{r}_X = \Phi_X^*(e_X)$$

where $e_X = \eta_X \circ \upsilon'_X$.

Intuitively speaking, this map unfolds a term from $T'X$ performing infinitely many steps of unfolding as defined by r, and leaving the variables from X unchanged.

Note that $\bar{r}_X$ is properly defined, since $\Phi_X(e_X) \geq e_X$. This follows from Lemma 7.5, since it is easily checked that $\Phi_X(e_X) \circ \eta'_X = e_X \circ \eta'_X$.

Example A ctd. With definition of r as shown at the end of Section 7.3, one has for example

$$\begin{aligned} \bar{r}_0(\texttt{loop a}) &= \texttt{a;a;a;a;a;}\ldots \\ \bar{r}_0(\texttt{loop 0}) &= \bot \end{aligned}$$

Example B ctd. With definition of r as shown at the end of Section 7.3, here one has

$$\begin{aligned} \bar{r}_0(\texttt{unfolding a}) &= \texttt{a} \\ \bar{r}_0(\texttt{unfolding(a;0)}) &= \texttt{a;a;a;a;a;}\ldots \\ \bar{r}_0(\texttt{unfolding 0}) &= \bot \end{aligned}$$

The maps $\bar{r}_X$ are in fact natural in X:

Lemma 7.10 The family $(\bar{r}_X : T'X \to TX)_{X \in C}$ derived from $r : T' \to TT'$ forms a natural transformation $\bar{r} : T' \to T$.

Proof. Using square commutation by fixpoint induction.

Base case: The family $(e_X : T'X \to TX)_{X \in C}$ is a natural transformation.

Induction step: Assume that the family $(f_X : T'X \to TX)_{X \in C}$ is a natural transformation. Then everything in the definition of $\Phi_X(f_X)$ is natural in X, so the family $(\Phi_X(f_X) : T'X \to TX)_{X \in C}$ is also a natural transformation.

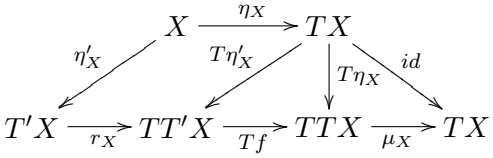
The lemma follows by square commutation by fixpoint induction (Lemma 7.2). □

Lemma 7.11 For any natural transformation $r : T' \to TT'$, $\bar{r} \circ \eta' = \eta$.

Proof. Componentwise by fixpoint induction on Φ_X.

Base case: Obvious by definition of e_X.

Induction step: Assume that for some $f : T'X \to TX$, $f \circ \eta'_X = \eta_X$. Then also $\Phi_X(f) \circ \eta'_X = \eta_X$, since everything in the following diagram commutes.



□

Remark 7.12 The latter result shows that $\bar{r}$ satisfies one of the axioms of a monad morphism from T' to T. However, in general $\bar{r}$ is not a monad morphism: it is not the case that

$$\bar{r} \circ \mu' = \mu \circ T\bar{r} \circ \bar{r}T'$$

Indeed, recall the example B presented in Section 7.2 (with definition of r given in Section 7.3) and consider a term

$$s = (\texttt{unfolding}\_)[\texttt{a}; \texttt{0}] \in T'T'0$$

Then $\mu_0(T\bar{r}_0(\bar{r}_{T'0}(s))) = \texttt{a}; \texttt{0}$, but $\bar{r}_0(\mu'_0(s))$ is the infinite term $\texttt{a}; \texttt{a}; \texttt{a}; \ldots$.

This problem will be addressed in Section 7.7.

For any unfolding rule r, the unfolding transformation $\bar{r} : T' \to T$ is a left inverse to $t : T \to T'$:

Lemma 7.13 For any unfolding rule r, $\bar{r} \circ t = id$.

Proof. (componentwise) Recall that r is based on some decomposition structure γ on T, and that $id = \bar{\gamma}_X = \tilde{\Phi}^*_X(\tilde{e}_X)$.

To prove that $\Phi^*_X(e_X) \circ t_X = \tilde{\Phi}^*_X(\tilde{e}_X)$, proceed by simultaneous fixpoint induction on Φ_X and $\tilde{\Phi}_X$.

Base case: $e_X \circ t_X = \eta_X \circ \upsilon'_X \circ t_X = \eta_X \circ \upsilon_X = \tilde{e}_X$

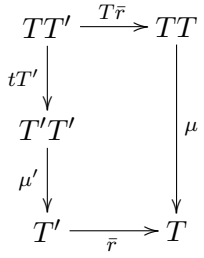
Induction step: Assume $f \circ t_X = g$ for some $f : T'X \to TX$, $g : TX \to TX$. Then

$$\begin{aligned} \Phi_X(f) \circ t_X = \mu_X \circ Tf \circ r_X \circ t_X &= \text{(Lemma 7.9)} \\ \mu_X \circ Tf \circ Tt_X \circ \gamma &= \\ \mu_X \circ Tg \circ \gamma_X &= \tilde{\Phi}_X(g) \end{aligned}$$

The lemma follows from Lemma 7.2. □

In the following sections, the following technical lemma will be often used:

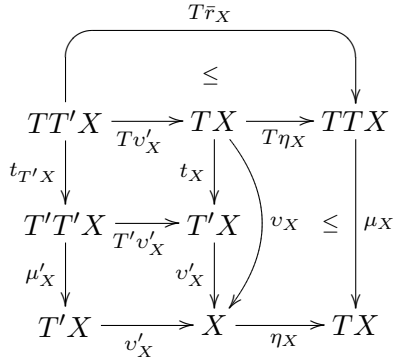
Lemma 7.14 For any unfolding rule r, the diagram



commutes.

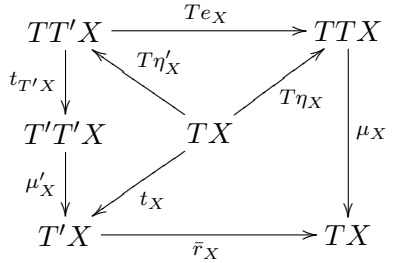
Proof. Proceed componentwise using Lemma 7.3.

Base case 1: To see that $e_X \circ \mu'_X \circ t_{T'X} \leq \mu_X \circ T\bar{r}_X$, chase the following diagram:



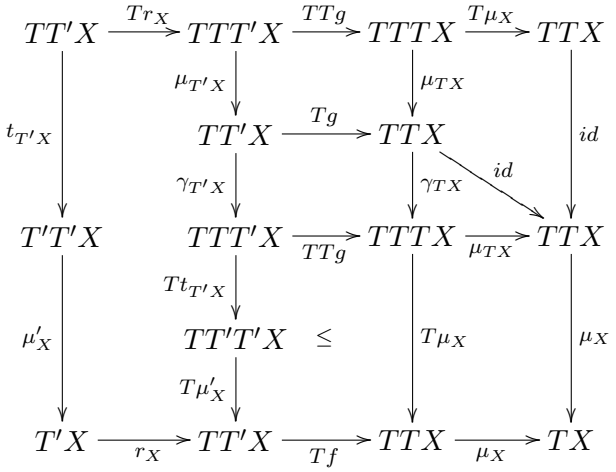
In particular, the lower left square commutes since υ'_X is a T'-algebra, and the upper region by definition of $\bar{r}$.

Base case 2: Note that locally continuous functors preserve upper sections and their left inverses. This means that $T\eta'_X$ is an upper section, and to check that $\bar{r}_X \circ \mu'_X \circ t_{T'X} \geq \mu_X \circ Te_X$, by Lemma 7.5 it is enough to check that $\bar{r}_X \circ \mu'_X \circ t_{T'X} \circ T\eta'_X = \mu_X \circ Te_X \circ T\eta'_X$. To see this, chase the diagram



In particular, the lower right square commutes by Lemma 7.13.

Induction step 1: Consider two morphisms $f : T'X \to TX$, $g : T'X \to TX$ such that $f \circ \mu'_X \circ t_{T'X} \leq \mu_X \circ Tg$. Then the following diagram commutes:



(the upper right square commutes only when postcomposed with μ_X).

Induction step 2: Proceed like in Induction step 1, changing all $\leq$ to $\geq$.

The lemma follows from Lemma 7.3. □

7.5 Merging Unfolding Rules with Distributive Laws

Operational semantics formalized as a distributive law $\lambda : TH \to BT$ of a monad T over a copointed endofunctor H gives rise to an operational interpretation of the language T. Given an unfolding rule $r : T' \to TT'$, one can extend λ to a distributive law $\lambda^r : T'H \to BT'$, thus providing an operational interpretation of terms built from syntax T'.

More formally, for a given distributive law $\lambda : TH \to BT$ and an unfolding rule $r : T' \to TT'$, one defines the function $\Psi_X : C(T'HX, BT'X) \to C(T'HX, BT'X)$ as follows:

$$\Psi_X \frac{T'HX \xrightarrow{f} BT'X}{T'HX \xrightarrow[r_{HX}]{} TT'HX \xrightarrow[T(T'\pi_1, f)]{} THT'X \xrightarrow[\lambda_{T'X}]{} BTT'X \xrightarrow[Bt_{T'X}]{} BT'T'X \xrightarrow[B\mu'_X]{} BT'X}$$

It is easy to see that Ψ_X is continuous, since T and B are locally continuous, and C is **Cppo**-enriched.

Based on this function, the morphism $\lambda^r_X : T'HX \to BT'X$ is defined as follows:

$$\lambda^r_X = \Psi^*_X(d_X)$$

where $d_X = B\eta'_X \circ \pi_2 \circ \upsilon'_{HX}$

Note that this map is well defined, because

$$\Psi_X(d_X) \geq d_X$$

This follows from Lemma 7.5, since (as is easily checked) $\Psi_X(d_X) \circ \eta'_{HX} = d_X \circ \eta'_{HX}$.

Example A ctd. The map λ^r_X, given a term (together with the behaviour of all variables from X), unfolds it using rule r until the resulting term can show some behaviour according to the distributive law λ. With the definition of r as shown at the end of Section 7.3, one has for example

$$\begin{aligned} \lambda^r_0(\mathtt{loop\ a}) &= \{(a, \mathtt{loop\ a})\} \\ \lambda^r_0(\mathtt{loop(a + b)}) &= \{(a, \mathtt{loop(a + b)}), (b, \mathtt{loop(a + b)})\} \\ \lambda^r_0(\mathtt{loop\ 0}) &= \bot \end{aligned}$$

(note that no variables appear here).

Example B ctd. With definition of r as shown at the end of Section 7.3, here one has

$$\begin{aligned} \lambda^r_0(\mathtt{unfolding(a; a)}) &= \{(a, \mathtt{a})\} \\ \lambda^r_0(\mathtt{unfolding(a; 0)}) &= \{(a, \mathtt{unfolding(a; 0)})\} \\ \lambda^r_0(\mathtt{unfolding\ 0}) &= \bot \end{aligned}$$

Lemma 7.15 The family $(\lambda^r_X : T'HX \to BT'X)_{X \in C}$ is a natural transformation from $T'H$ to BT'. Moreover, it is a distributive law of the pointed endofunctor T' over the copointed endofunctor H.

Proof. Naturality is shown using square commutation by fixpoint induction (Lemma 7.2). For the base case, d is a natural transformation. For the induction step, assume a family $(f_X : T'HX \to BT'X)_{X \in C}$ to be natural in X. Then everything in the definition of $\Psi_X(f)$ is natural in X, so the family $(\Psi_X(f) : T'HX \to BT'X)_{X \in C}$ is also natural in X.

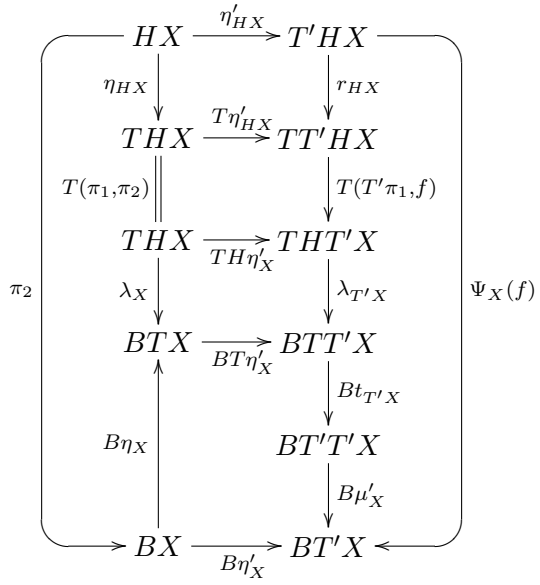
The distributive law axiom

$$\lambda^r_X \circ \eta'_{HX} = B\eta'_X \circ \pi_2$$

is shown by ordinary fixpoint induction on Ψ_X. The base case follows directly from definition of d_X. For the induction step, take a morphism $f : T'HX \to BT'X$, such that

$$f \circ \eta'_{HX} = B\eta'_X \circ \pi_2$$

Then everything in the diagram



commutes. □

Remark 7.16 In general λ^r is not a distributive law of the monad T over the copointed endofunctor H (for a counterexample, see Remark 7.12).

The natural transformation λ^r induces a mapping T'_{λ^r} on B-coalgebras. For a B-coalgebra $k : X \to BX$, define

$$T'_{\lambda^r}(k) = \lambda^r_X \circ T'(id, k)$$

Lemma 7.17 T'_{λ^r} is an endofunctor lifting the pointed endofunctor T' to B-coalgebras.

Proof. Functoriality follows easily from naturality of λ^r. The structure of pointed endofunctor follows from Lemma 7.15. □

The functor T'_{λ^r} may be also defined independently. For any object X, define the function $\Upsilon_X : C(T'X, BT'X) \to C(T'X, BT'X)$ as follows:

$$\Upsilon_X \frac{T'X \xrightarrow{f} BT'X}{T'X \xrightarrow[r_x]{} TT'X \xrightarrow[T_\lambda f]{} BTT'X \xrightarrow[Bt_{T'X}]{} BT'T'X \xrightarrow[B\mu'_X]{} BT'X}$$

It is easy to see that Υ_X is continuous on the cppo $C(T'X, BT'X)$, since T_λ acts as a continuous function on $C(T'X, BT'X)$ and C is **Cppo**-enriched.

For a given H-coalgebra $k : X \to BX$, define $k^\downarrow : T'X \to BT'X$

$$k^\downarrow = d_X \circ T'(id, k)$$

The following result gives an independent characterization of T'_{λ^r}:

Theorem 7.18 For any B-coalgebra $k : X \to BX$,

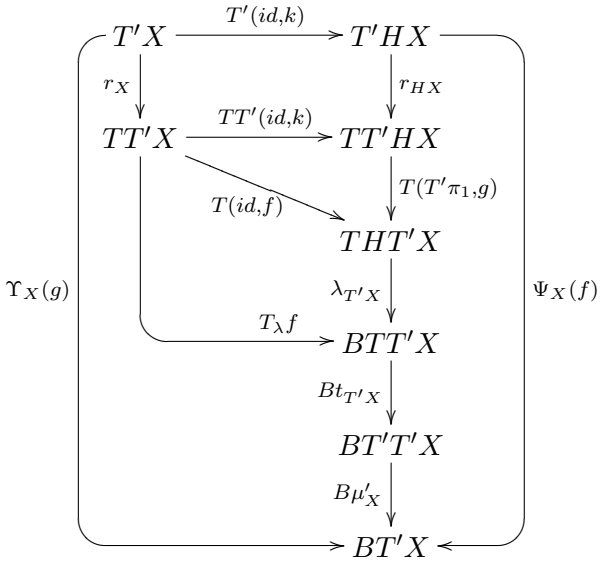
$$T'_{\lambda^r}(k) = \Upsilon^*_X(k^\downarrow)$$

Proof. By simultaneous fixpoint induction on Ψ_X and Υ_X. For the base case, one has to show that

$$d_X \circ T'(id, k) = k^\downarrow$$

which is just the definition of $k^\downarrow$.

For the induction step, consider two maps $f : TH'X \to BT'X$, $g : T'X \to BT'X$ such that $f \circ T'(id, k) = g$. Then the following diagram commutes:



□

Intuitively, for a given B-coalgebra $k : X \to BX$, the coalgebra $T'_{\lambda^r}k : T'X \to BT'X$ unfolds a given term from $T'X$ according to the unfolding map r, until the resulting term exhibits some behaviour according to the distributive law λ, assuming that variables in the terms show behaviour as defined by k.

Remark 7.19 From Remark 7.16 it follows that in general T'_{λ^r} does not lift the monad T' to the category of B-coalgebras: it is not always the case that the diagram

$$\begin{array}{ccc} T'T'X & \xrightarrow{\mu'_X} & T'X \\ {\scriptstyle T'_{\lambda^r}T'_{\lambda^r}k}\downarrow & & \downarrow{\scriptstyle T'_{\lambda^r}k} \\ BT'T'X & \xrightarrow[B\mu'_X]{} & BT'X \end{array}$$

commutes. This problem will be addressed in Section 7.7.

7.6 From Unfolding Rules to Models

In the preceding sections it was shown how, given a distributive law λ of a monad T over a copointed endofunctor H, a translation $t : T \to T'$, and an unfolding rule $r : T' \to TT'$, one can derive the infinite unfolding $\bar{r} : T' \to T$

and the distributive law λ^r of the pointed endofunctor T' over the copointed endofunctor H. This section shows how the two latter notions are related, and how they allow to construct λ^r-models from λ-bialgebras and vice versa.

First, two theorems showing how λ, λ^r, t and $\bar{r}$ relate:

Theorem 7.20 The diagram

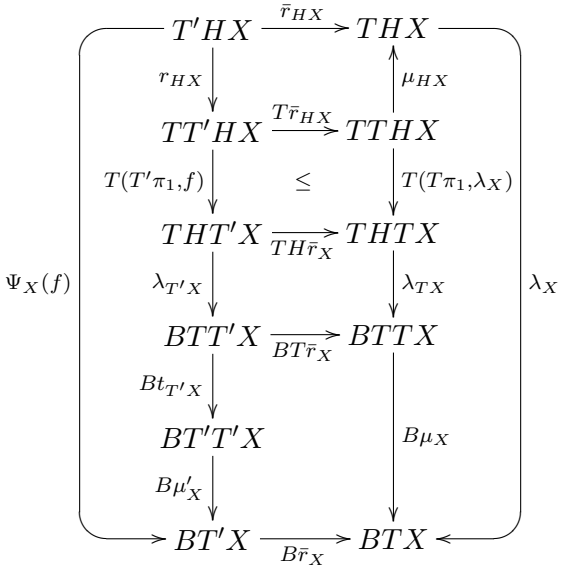
$$\begin{array}{ccc} T'H & \xrightarrow{\lambda^r} & BT' \\ {\scriptstyle \bar{r}H}\downarrow & & \downarrow{\scriptstyle B\bar{r}} \\ TH & \xrightarrow[\lambda]{} & BT \end{array}$$

commutes.

Proof. First, prove that $B\bar{r}_X \circ \lambda^r_X \leq \lambda_X \circ \bar{r}_{HX}$ by fixpoint induction on Ψ_X.

Base case: $B\bar{r}_X \circ d_X \leq \lambda_X \circ \bar{r}_{HX}$ follows from definition of d_X and from Lemma 7.5, since (as is easily checked) $B\bar{r}_X \circ d_X \circ \eta'_{HX} = \lambda_X \circ \bar{r}_{HX} \circ \eta'_{HX}$.

Induction step: Assume $B\bar{r}_X \circ f \leq \lambda_X \circ \bar{r}_{HX}$ for some $f : T'HX \to BT'X$. Then the same inequality holds with f replaced by $\Psi_X(f)$, since the following diagram commutes:

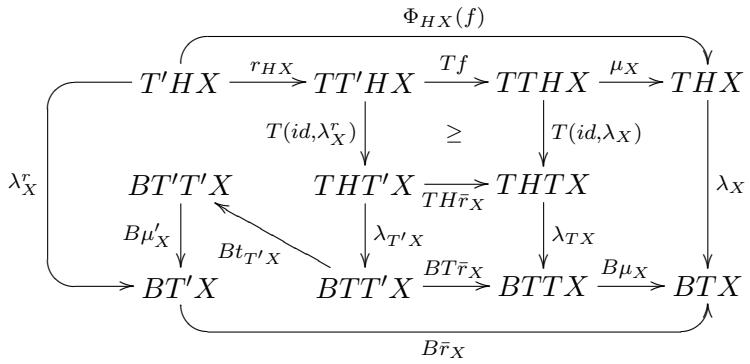


(The four regions in the middle of the diagram commute, counting from the top: by definition of $\bar{r}_X$, by the inductive assumption, by naturality of λ and by Lemma 7.14. The region to the right commutes since λ is a distributive law of the monad T over H.)

Then proceed to prove that $B\bar{r}_X \circ \lambda^r_X \geq \lambda_X \circ \bar{r}_{HX}$ by fixpoint induction on Φ_{HX}.

Base case: $B\bar{r}_X \circ \lambda^r_X \geq \lambda_X \circ e_{HX}$ follows from definition of e_X and from Lemma 7.5, since (as is easily checked) $B\bar{r}_X \circ \lambda^r_X \circ \eta'_{HX} \geq \lambda_X \circ e_{HX} \circ \eta'_{HX}$

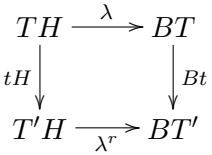
Induction step: Assume $B\bar{r}_X \circ \lambda^r_X \geq \lambda_X \circ f$ for some $f : T'X \to TX$. Then the same inequality holds with f replaced by $\Phi_{HX}(f)$. To prove this, consider the diagram



Here everything commutes (in particular, the large region to the left commutes by definition of λ^r, and the bottom region commutes by Lemma 7.14).

This completes the proof of Theorem 7.20. □

Theorem 7.21 The diagram



commutes.

Proof. To prove this, it is convenient to represent $\lambda_X : HTX \to BTX$ as a fixpoint of a certain function, resembling the function Ψ_X used to define λ^r_X in Section 7.5. The appropriate function $\tilde{\Psi}_X : C(THX, BTX) \to C(THX, BTX)$ acts as follows:

$$\tilde{\Psi}_X \frac{THX \xrightarrow{f} BTX}{THX \xrightarrow[\gamma_{HX}]{} TTHX \xrightarrow[T(T\pi_1,f)]{} THTX \xrightarrow[\lambda_{TX}]{} BTTX \xrightarrow[B\mu_X]{} BTX}$$

It is easy to see that $\tilde{\Psi}_X$ is continuous, since T is locally continuous, and C is **Cppo**-enriched.

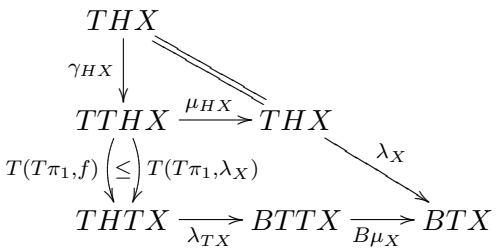
Having defined $\tilde{d}_X = B\eta_X \circ \pi_2 \circ \upsilon_{HX}$, the following lemma holds:

Lemma 7.22 $\tilde{\Psi}^*_X(\tilde{d}_X) = \lambda_X$.

Proof. First, prove that $\tilde{\Psi}^*_X(\tilde{d}_X) \leq \lambda_X$ by fixpoint induction on $\tilde{\Psi}_X$.

Base case: $\tilde{d}_X \leq \lambda_X$ follows from definition of $\tilde{d}_X$ and from Lemma 7.5, since (as is easily checked) $\tilde{d}_X \circ \eta_{HX} = \lambda_X \circ \eta_{HX}$.

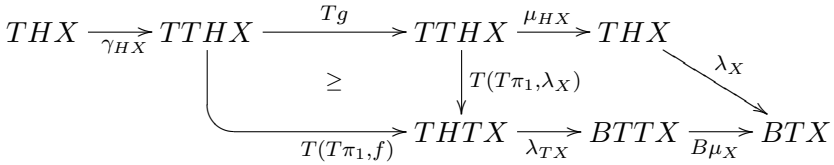
Induction step: Assume $f \leq \lambda_X$ for some $f : THX \to BTX$. Then also $\tilde{\Psi}_X(f) \leq \lambda_X$. To prove this, chase the diagram



To prove that $\tilde{\Psi}^*_X(\tilde{d}_X) \geq \lambda_X$, recall from Section 7.3 that $\tilde{\Phi}^*_X(\tilde{e}_X) = id_{TX}$ for any object X and prove $\tilde{\Psi}^*_X(\tilde{d}_X) \geq \lambda_X \circ \tilde{\Phi}^*_{HX}(\tilde{e}_{HX})$ by simultaneous induction on $\tilde{\Psi}_X$ and $\tilde{\Phi}_{HX}$.

Base case: Immediate by definition of $\tilde{e}_{HX}$ and $\tilde{d}_X$.

Induction step: Take two morphisms $f : THX \to BTX$, $g : THX \to THX$ such that $f \geq \lambda_X \circ g$. Without any loss of generality, assume that $g \leq \tilde{\Phi}^*_{HX}(\tilde{e}_{HX}) = id_{THX}$. Then chase the diagram

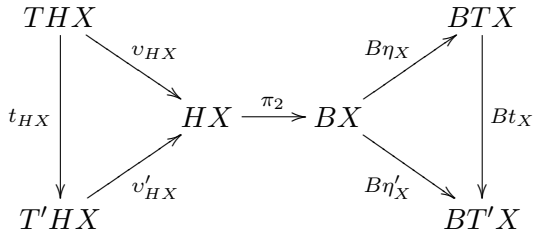


The show the middle triangle commutes use the fact that $g \leq id_{THX}$, hence $T\pi_1 \circ g \leq T\pi_1$.

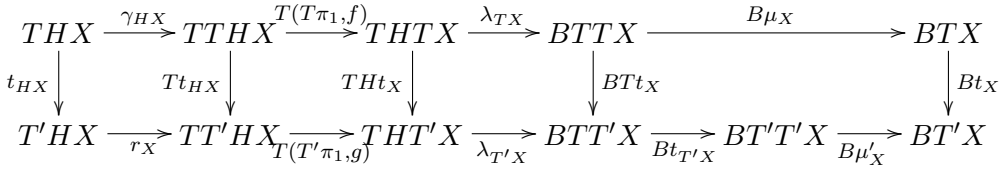
The lemma follows from Lemma 7.2. □

The proof of Theorem 7.21 proceeds componentwise by induction on $\tilde{\Psi}_X$ and Ψ_X.

Base case: See definitions of d_X and $\tilde{d}_X$, and chase the diagram



Induction step: Take two morphisms $f : THX \to BTX$, $g : T'HX \to BT'X$ such that $g \circ t_{HX} = Bt_X \circ f$. Then the following diagram commutes:

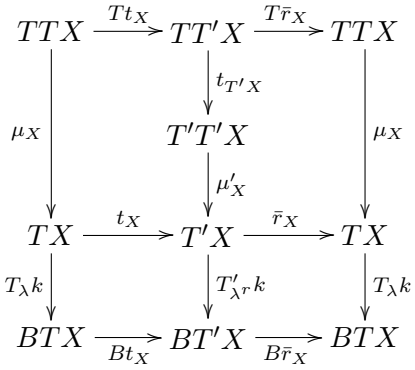


In particular, the leftmost square commutes by Lemma 7.9.

This, by Lemma 7.2, completes the proof of Theorem 7.21. □

From Theorems 7.20 and 7.21, a few important corollaries can be drawn:

Corollary 7.23 For any B-coalgebra $k : X \to BX$, the following diagram commutes:



Proof. The bottom left square commutes by Theorem 7.21, the bottom right square by Theorem 7.20, the top right region by Lemma 7.14, and the top left region since t is a monad morphism. □

The coalgebraic part of Corollary 7.23 means that interpreting a term from $T'X$ by unfolding it infinitely with $\bar{r}_X$, and then by any λ-bialgebra morphism from TX, is adequate with respect to the operational model $T'_{\lambda^r}k$, which unfolds a given term only until it exhibits some behaviour. The algebraic part of the corollary means that the map $\bar{r}_X$ is compositional with respect to syntax T (it is a T-algebra morphism).

Remark 7.24 Note that $\bar{r}_X$ is not fully compositional in general (for a counterexample, see Remark 7.12). This problem will be addressed in Section 7.7.

Remark 7.25 The structure

$$TT'X \xrightarrow{t_{T'X}} T'T'X \xrightarrow{\mu'_X} T'X \xrightarrow{T'_{\lambda^r}k} BT'X$$

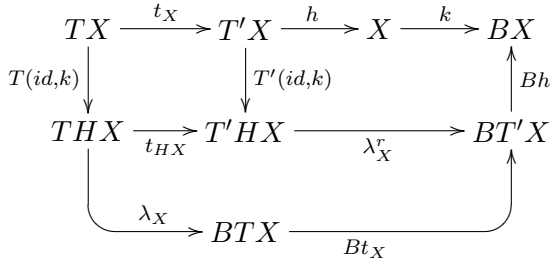
is not a λ-bialgebra in general. Therefore Corollary 7.23 cannot be read as showing that t_X and $\bar{r}_X$ are λ-bialgebra morphisms.

Corollary 7.26 For any λ^r-model $T'X \xrightarrow{h} X \xrightarrow{k} BX$,

$$TX \xrightarrow{t_X} T'X \xrightarrow{h} X \xrightarrow{k} BX$$

is a λ-bialgebra.

Proof. The diagram



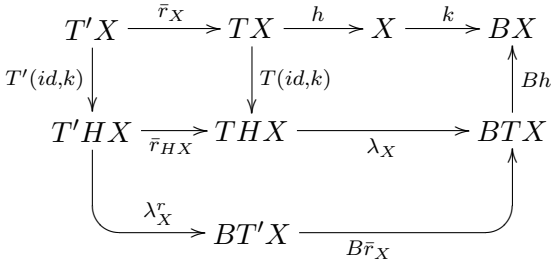
commutes by Theorem 7.21, and by naturality of t. □

Corollary 7.27 For any λ-bialgebra $TX \xrightarrow{h} X \xrightarrow{k} BX$,

$$T'X \xrightarrow{\bar{r}_X} TX \xrightarrow{h} X \xrightarrow{k} BX$$

is a λ^r-model.

Proof. The diagram



commutes by Theorem 7.20 and by naturality of $\bar{r}$. □

Corollaries 7.26 and 7.27 mean that any bialgebraic model of the recursion-free fragment of a language can be used to interpret the full language, and vice versa, along syntactic translations of $\bar{r}$ and t respectively.

The fact that λ^r is not a distributive law of the monad T' over H is the only reason that prevents t and $\bar{r}$ from being morphisms of distributive laws in the sense of [88].

7.7 Regular Unfolding Rules

In the preceding sections, recursive constructs were added to languages by means of unfolding rules $r : T' \to TT'$. This notion is quite general, covering rather strange examples, e.g., the `unfolding` construct. This generality also led to some technical problems:

- $\lambda^r : T'H \to BT'$ is not a distributive law of the monad T' over the copointed endofunctor H,
- $\bar{r} : T' \to T$ is not a monad morphism,
- the maps $\bar{r}_X$ are not compositional.

In this section these problems will be remedied by considering a special class of *regular unfolding rules*. These exclude the example B shown throughout this paper, but still cover many examples of recursive equations, including many unguarded ones (in particular, the example A).

For this purpose, the general framework described so far is specialized with the following additional assumptions:

- T is the monad freely generated by an endofunctor Σ,
- T' is the monad freely generated by an endofunctor Σ',
- $t : T \to T'$ is the monad morphism generated by a natural transformation $t : \Sigma \to \Sigma'$,
- $r : T' \to TT'$ is induced from a natural transformation

$$r^0 : \Sigma' \to TT'$$

such that

$$r^0 \circ t = T\eta' \circ \phi$$

where $\phi : \Sigma \to T$ is the inclusion arising from the free monad structure of T.

Given such r^0, the transformation r is derived according to the formula

$$r = [(\eta T' \circ \eta'), (T\mu' \circ r^0 T')] \circ \iota'$$

where $\iota' : T' \to Id + \Sigma' T'$ is the isomorphism arising from the free monad structure of T'.

It is easy to check that r defined this way satisfies the definition of an unfolding rule, presented in Section 7.3, and is based on the decomposition structure $\gamma : T \to TT$ defined as

$$\gamma = [\eta T \circ \eta, \phi T] \circ \iota$$

Unfolding rules r derived as above will be called *regular*.

Note that since T' is freely generated by Σ', its unit η' is an upper section and it can be canonically equipped with a left inverse $\upsilon' : T' \to Id$:

$$\upsilon' = [id, \bot] \circ \iota'$$

The same holds for T.

Example A ctd.: Definition of an appropriate r^0 is immediately derived from definitions of γ^0 and r given in Example A in Section 7.3.

Example B ctd.: Note that the intended recursive equation for the construct `unfolding` cannot be captured as a regular unfolding rule. Indeed, consider two terms t_1, t_2. If an unfolding rule r is derived from a natural transformation $r^0 : \Sigma' \to TT'$ as above, then the term $r(\texttt{unfolding}\ t_2)$ should be obtained from $r(\texttt{unfolding}\ t_1)$ just by replacing every copy of t_1 with t_2. This is, however, not the intended behaviour of `unfolding`: as a counterexample, consider $t_1 = \texttt{a};\texttt{0}$, $t_2 = \texttt{a};\texttt{a}$, where $r(\texttt{unfolding}\ t_1) = \texttt{a};\texttt{unfolding}\ t_1$ and $r(\texttt{unfolding}\ t_2) = t_2$.

To remedy the technical problems encountered in previous sections, some technical lemmas are needed:

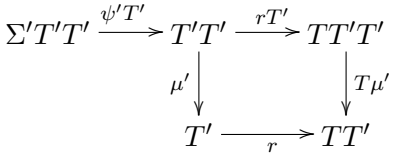
Lemma 7.28 The following diagram commutes:

$$\begin{array}{ccc} \Sigma' T' & \xrightarrow{r^0 T'} & TT'T' \\ {\scriptstyle \psi'}\downarrow & & \downarrow{\scriptstyle T\mu'} \\ T' & \xrightarrow[r]{} & TT' \end{array}$$

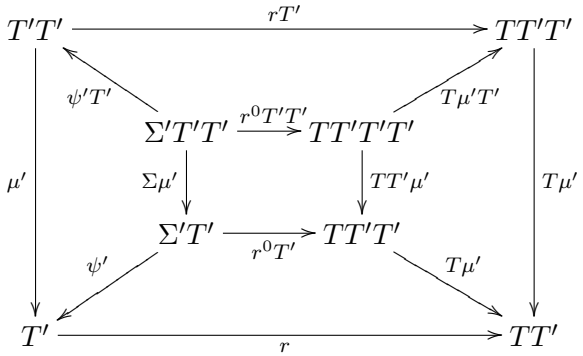
where $\psi' : \Sigma' T' \to T'$ is the inclusion arising from the free monad structure of T'.

Proof. Immediate by definition of r from r^0. □

Lemma 7.29 For a regular r, the square as in the following diagram is made commute by the map $\psi' T'$:



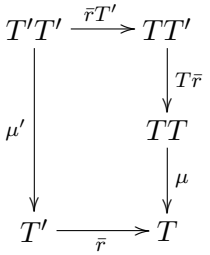
Proof. Everything in the following diagram commutes.



The top and the bottom squares commute by Lemma 7.28, the middle square by naturality of r^0, and the left square by the inductive definition of μ' in the freely generated monad T'. □

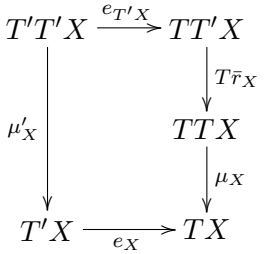
Theorem 7.30 For a regular r, the transformation $\bar{r}$ is a monad morphism from T' to T.

Proof. One of the laws for the monad morphism for $\bar{r}$ was proved to hold in Lemma 7.11. The remaining law is



To prove this, proceed componentwise using square commutation by fixpoint induction (Lemma 7.2).

For the base case, it must be shown that that the diagram



commutes. Given that T' is freely generated by Σ', it is enough to show that the diagram commutes when precomposed with $\eta'_{T'X} : T'X \to T'T'X$

or with $\psi'_{T'X} : \Sigma'T'T'X \to T'T'X$. Both cases are checked easily (in the latter case, both sides of the diagram are equal to the bottom element of the homset $C(\Sigma'T'T'X, TX)$).

For the induction step, take two morphisms $f : T'T'X \to TT'X$, $g : T'X \to TX$ and assume that the diagram

$$\begin{array}{ccc} T'T'X & \xrightarrow{f} & TT'X \\ \downarrow{\scriptstyle \mu'_X} & & \downarrow{\scriptstyle T\bar{r}_X} \\ T'X & \xrightarrow[g]{} & TTX \\ & & \downarrow{\scriptstyle \mu_X} \\ & & TX \end{array}$$

commutes. Then the diagram

$$\begin{array}{ccccccc} T'T'X & \xrightarrow{r_{T'X}} & TT'T'X & \xrightarrow{Tf} & TTT'X & \xrightarrow{\mu_{T'X}} & TT'X \\ \downarrow{\scriptstyle \mu'_X} & & \downarrow{\scriptstyle T\mu'_X} & & \downarrow{\scriptstyle TT\bar{r}_X} & & \downarrow{\scriptstyle T\bar{r}_X} \\ T'X & \xrightarrow[r_X]{} & TT'X & \xrightarrow[Tg]{} & TTX & \xrightarrow[\mu_X]{} & TX \\ & & & & \downarrow{\scriptstyle T\mu_X} & & \downarrow{\scriptstyle \mu_X} \\ & & & & TTTX & \xrightarrow[\mu_{TX}]{} & TTX \end{array}$$

commutes when precomposed with $\eta'_{T'X} : T'X \to T'T'X$ or with $\psi'_{T'X} : \Sigma'T'T'X \to T'T'X$. In particular, the left square commutes when precomposed with $\psi'_{T'X} : \Sigma'T'T'X \to T'T'X$ by Lemma 7.29.

This completes the proof of Theorem 7.30. □

Theorem 7.31 For a regular r, λ^r is a distributive law of the monad T' over the copointed endofunctor H.

Proof. One of the laws for the distributive law for λ^r was proved to hold in Lemma 7.15. The remaining law is

$$\begin{array}{ccccc} T'T'H & \xrightarrow{T'(T'\pi_1, \lambda^r)} & T'HT' & \xrightarrow{\lambda^r T'} & BT'T' \\ \downarrow{\scriptstyle \mu' H} & & & & \downarrow{\scriptstyle B\mu'} \\ T'H & & \xrightarrow[\lambda^r]{} & & BT' \end{array}$$

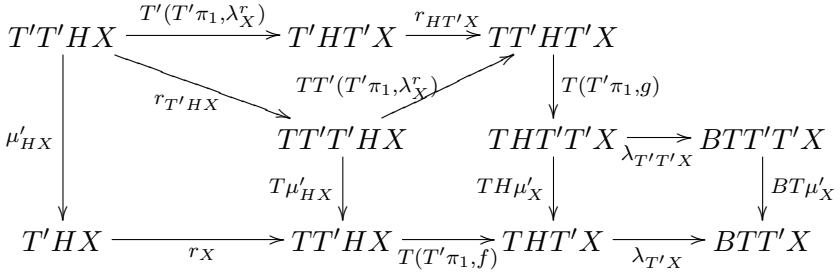
To prove this, proceed componentwise by induction on $\Psi_{T'X}$ and Ψ_X using Lemma 7.2.

Base case: It is enough to show that the following diagram commutes:

$$\begin{array}{ccccc} T'T'HX & \xrightarrow{T'(T'\pi_1, \lambda^r_X)} & T'HT'X & \xrightarrow{d_{T'X}} & BT'T'X \\ \downarrow{\scriptstyle \mu'_{HX}} & & & & \downarrow{\scriptstyle B\mu'_X} \\ T'HX & & \xrightarrow[d_X]{} & & BT'X \end{array}$$

As in Theorem 7.30, it is enough to show that this diagram commutes when precomposed with $\eta'_{T'HX}$ or with $\psi'_{T'HX}$. Both cases are checked easily, by definition of d.

Induction step: Take two morphisms $f : T'HX \to BTX$ and $g : T'HT'X \to BT'T'X$ such that $f \circ \mu'_{HX} = B\mu'_X \circ g \circ T'(T'\pi_1, \lambda^r_X)$. Then chase the diagram



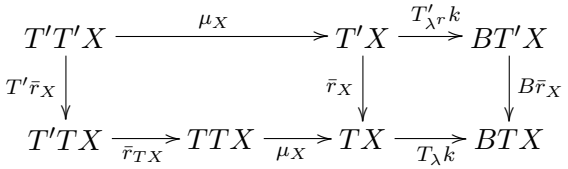
precomposing it with $\eta'_{T'HX} : T'HX \to T'T'HX$ and with $\psi'_{T'HX} : \Sigma'T'T'HX \to T'T'HX$. In particular, the bottom left square precomposed with $\psi'_{T'HX}$ commutes by Lemma 7.29. The middle pentagon commutes by the inductive assumption, and the upper square — by naturality of r.

This, by Lemma 7.2, completes the proof. □

Theorems 7.30 and 7.31 allow to rephrase many results from previous sections in a more structured fashion, provided the unfolding rule r is regular. Theorems 7.21 and 7.20 say now that t and $\bar{r}$ are morphisms of distributive laws in the sense of [88]. In Corollaries 7.26 and 7.27 one can replace λ^r-models with λ^r-bialgebras. Also the functor T'_{λ^r} lifts the monad structure of T' to the category of B-coalgebras, as an easy corollary from Theorem 7.31.

Moreover, for regular r the map $\bar{r}$ is fully compositional:

Corollary 7.32 If r is regular, then for any B-coalgebra $k : X \to BX$, the map $\bar{r}_X$ is a λ^r-bialgebra morphism as shown in the diagram:



Proof. The algebraic part follows from Theorem 7.30, and the coalgebraic part from Corollary 7.23. □

7.8 Concluding Remarks

We have seen how to fit a rather general class of recursive constructs into the bialgebraic semantic framework developed by Turi and Plotkin. The behaviour of recursive constructs is not modelled with operational rules, but with separate recursive equations, formalised as certain natural transformations called unfolding rules. These equations are then merged, using certain fixpoint constructions, with natural transformations corresponding to the operational semantics of the recursion-free fragment of the language in question. The result is particularly well-structured if the original unfolding rules satisfy an additional property, called regularity. Regular unfolding rules, when merged with distributive laws

emerging from operational rules, yield new distributive laws, giving bialgebraic semantics to the language extended with recursive constructs.

The framework presented here is quite general and covers many examples of recursive equations, including many unguarded ones. As such equations can be a source of partiality (divergence), it is convenient to interpret them in a suitable $\mathbf{Cppo}_\perp$-enriched category. This allows to perform the fixpoint constructions needed to merge the equations with recursion-free operational rules.

Working in enriched categories makes dealing with recursive constructs simpler, but there is a valid question whether and to what extent a similar framework can be realized in the category of sets and functions, without imposing any additional (and always unwelcome) structure on the programs considered. A particularly interesting option is to interpret recursive equations using completely iterative monads generated by syntactic functors [7], rather than usual freely generated monads. In this framework, one can find solutions to recursive equations, under the mild assumption that the equations are ideal. This makes it rather straightforward, given an ideal recursive equation formalized as an unfolding rule $r : T' \to TT'$, to construct the infinite unfolding map $\bar{r}$ using universal properties of final coalgebras rather than existence of fixpoints. Therefore it seems that the framework presented in Section 7.4 can be realized in the category of sets, using completely iterative monads.

However, it is much less clear how to apply the same idea to the developments presented in Sections 7.5 and 7.6. This is due to divergence that may arise from unguarded recursive equations. Indeed, consider a rather unusual looping construct `loop2` with the intended behaviour captured by the (ideal) recursive equation

$$\texttt{loop2}\ t = (\texttt{loop2}\ t);\ t$$

formalised as an appropriate unfolding rule r. Using properties of iterative monads, one can then define the infinite unfolding map $\bar{r}$ and, e.g.,

$$\bar{r}_0(\texttt{loop2 a}) = (((\ldots;\texttt{a});\texttt{a});\texttt{a});\texttt{a})$$

However, it is much harder to define the action of the distributive law λ_0^r on the term `loop2 a`, since this term can never show any real behaviour. It is an open problem whether this drawback can be avoided and whether all developments presented in this paper can be realized without resorting to categories of cpos.

Bibliography

[1] M. Abadi and A. D. Gordon. A calculus for cryptographic protocols: The spi calculus. In *4th ACM Conference on Computer and Communications Security*, pages 36–47. ACM Press, 1997.

[2] S. Abramsky. Domain equation for bisimulation. *Information and Computation*, 92:161–218, 1991.

[3] S. Abramsky and A. Jung. Domain theory. In S. Abramsky, D. Gabbay, and T. S. E. Maibaum, editors, *Handbook of Logic in Computer Science*, volume 3. Oxford University Press, 1995.

[4] S. Abramsky and S. Vickers. Quantales, observational logic and process semantics. *Mathematical Structures in Computer Science*, 3:161–227, 1993.

[5] L. Aceto, W. J. Fokkink, and C. Verhoef. Structural operational semantics. In J. A. Bergstra, A. Ponse, and S. Smolka, editors, *Handbook of Process Algebra*. Elsevier, 2002.

[6] L. Aceto and A. Ingólfsdóttir. CPO models for compact GSOS languages. *Information and Computation*, 129:107–141, 1996.

[7] P. Aczel, J. Adamek, and J. Velebil. A coalgebraic view of infinite trees and iteration. In *Proc. CMCS'01*, volume 44 of *Electronic Notes in Theoretical Computer Science*, 2001.

[8] P. Aczel and N. Mendler. A final coalgebra theorem. In *Proc. CTCS'89*, volume 389 of *Lecture Notes in Computer Science*, pages 357–365, 1989.

[9] H. H. Andersen and M. Mendler. An asynchronous process algebra with multiple clocks. In *Proc. ESOP '94*, volume 788 of *Lecture Notes in Computer Science*. Springer-Verlag, 1994.

[10] J. C. M. Baeten and W. P. Weijland. *Process Algebra*. Cambridge University Press, 1990.

[11] M. Barr. Terminal coalgebras for endofunctors on sets. *Theoretical Computer Science*, 114:299–315, 1993.

[12] F. Bartels. GSOS for probabilistic transition systems. In L. Moss, editor, *Proc. CMCS'02*, volume 65 of *Electronic Notes in Theoretical Computer Science*. Elsevier, 2002.

[13] F. Bartels. *On Generalised Coinduction and Probabilistic Specification Formats*. PhD dissertation, CWI, Amsterdam, 2004.

[14] J. A. Bergstra, W. J. Fokkink, and A. Ponse. Process algebra with recursive operators. In J. A. Bergstra, A. Ponse, and S. Smolka, editors, *Handbook of Process Algebra*. Elsevier, 2002.

[15] J. A. Bergstra and J. W. Klop. Process algebra for synchronous communication. *Information and Control*, 60:109–137, 1984.

[16] J. A. Bergstra, A. Ponse, and S. Smolka. *Handbook of Process Algebra*. Elsevier, 2002.

[17] B. Bloom, W. J. Fokkink, and R. J. van Glabbeek. Precongruence formats for decorated trace semantics. *ACM Transactions on Computational Logic*, 5:26–78, 2004.

[18] B. Bloom, S. Istrail, and A. Meyer. Bisimulation can't be traced. *Journal of the ACM*, 42:232–268, 1995.

[19] M. Boreale and F. Gadducci. Denotational testing semantics in coinductive form. In *Proc. MFCS'03*, volume 2747 of *Lecture Notes in Computer Science*. Springer-Verlag, 2003.

[20] S. D. Brookes, C. A. R. Hoare, and A. W. Roscoe. A theory of communicating sequential processes. *Journal of the ACM*, 31:560–599, 1995.

[21] L. Cardelli and A. D. Gordon. Mobile ambients. In *Proc. FOSSACS '98*, volume 1378 of *Lecture Notes in Computer Science*. Springer-Verlag, 1998.

[22] A. Cheng and M. Nielsen. Open maps at work. In *Proc. FSTTCS'95*, volume 1026 of *Lecture Notes in Computer Science*, 1995.

[23] A. Corradini, R. Heckel, and U. Montanari. Compositional SOS and beyond: a coalgebraic view of open systems. *Theoretical Computer Science*, 280:163–192, 2002.

[24] V. Danos and J. Krivine. Formal molecular biology done in CCS. In *Proc. BioCONCUR'03*, 2003.

[25] R. De Nicola and M. Hennessy. Testing equivalences for processes. *Theoretical Computer Science*, 34:83–133, 1984.

[26] J. Desharnais, A. Edalat, and P. Panangaden. Bisimulation for labelled Markov processes. *Information and Computation*, 179:163–193, 2002.

[27] U. Engberg and M. Nielsen. A calculus of communicating systems with name-passing. Technical Report DAIMI PB-208, Computer Science Department, Aarhus University, 1986.

[28] M. P. Fiore. *Axiomatic Domain Theory in Categories of Partial Maps*. Distinguished Dissertations in Computer Science. Cambridge University Press, 1996.

[29] M. P. Fiore. A coinduction principle for recursive data types based on bisimulation. *Information and Computation*, 127:186–198, 1996.

[30] M. P. Fiore, G. D. Plotkin, and D. Turi. Abstract syntax with variable binding. In *Proc. LICS'99*, pages 193–202. IEEE Computer Society Press, 1999.

[31] M. P. Fiore and D. Turi. Semantics of name and value passing. In *Proc. LICS'01*, pages 93–104. IEEE Computer Society Press, 2001.

[32] W. J. Fokkink, R. J. van Glabbeek, and P. de Wind. Compositionality of Hennessy-Milner logic through structural operational semantics. In *Proc. FCT'03*, volume 2751 of *Lecture Notes in Computer Science*, pages 412–422. Springer Verlag, 2003.

[33] M. Fowler, K. Beck, et al. *Refactoring: Improving the Design of Existing Code*. Addison-Wesley, 1999.

[34] R. J. van Glabbeek. The linear time – branching time spectrum II. In E. Best, editor, *Proc. CONCUR'93*, volume 715 of *Lecture Notes in Computer Science*, pages 66–81, 1993.

[35] R. J. van Glabbeek. The linear time – branching time spectrum I. In J. A. Bergstra, A. Ponse, and S. Smolka, editors, *Handbook of Process Algebra*. Elsevier, 1999.

[36] J. A. Goguen, J. W. Thatcher, et al. Initial algebra semantics and continuous algebras. *Journal of the ACM*, 24:68–95, 1977.

[37] H. P. Gumm. State based systems are coalgebras. *Cubo – Matemática Educacional*, 5:239–262, 2003.

[38] M. Hennessy. *Algebraic Theory of Processes*. MIT Press, 1988.

[39] M. Hennessy. *The semantics of programming languages: An elementary introduction using structural operational semantics*. Wiley, 1990.

[40] M. Hennessy and R. Milner. Algebraic laws for nondeterminism and concurrency. *Journal of the ACM*, 32:137–161, 1985.

[41] C. Hermida and B. Jacobs. Structural induction and coinduction in a fibrational setting. *Information and Computation*, 145(2):107–152, 1998.

[42] C. A. R. Hoare. *Communicating Sequential Processes*. Prentice Hall, 1985.

[43] B. Jacobs. *Categorical Logic and Type Theory*. Number 141 in Studies in Logic and the Foundations of Mathematics. North Holland, 1999.

[44] B. Jacobs. Trace semantics for coalgebras. In *Proc. CMCS 2004*, Electronic Notes in Theoretical Computer Science, 2004. To appear.

[45] B. Jacobs and J. Hughes. Simulations in coalgebra. *Electronic Notes in Theoretical Computer Science*, 82, 2003.

[46] B. Jacobs and J. J. M. M. Rutten. A tutorial on (co)algebras and (co)induction. *Bulletin of the European Association for Theoretical Computer Science*, 62, 1996.

[47] A. Joyal, M. Nielsen, and G. Winskel. Bisimulation from open maps. *Information and Computation*, 127(2):164–185, 1996.

[48] G. M. Kelly. *Basic Concepts of Enriched Category Theory*. Cambridge University Press, 1982.

[49] M. Kick. Rule formats for timed processes. In *Proc. CMCIM'02*, volume 68 of *Electronic Notes in Theoretical Computer Science*. Elsevier, 2002.

[50] B. Klin. An abstract approach to process equivalence and a coinduction principle for traces. In *Proc. CMCS 2004*, Electronic Notes in Theoretical Computer Science, 2004. To appear.

[51] B. Klin. Adding recursive constructs to bialgebraic semantics. *Journal of Logic and Algebraic Programming*, 2004. Special issue on structural operational semantics, to appear.

[52] B. Klin and P. Sobocinski. Syntactic formats for free: An abstract approach to process equivalence. In *Proc. CONCUR 2003*, volume 2671 of *Lecture Notes in Computer Science*, 2003.

[53] K. G. Larsen and A. Skou. Bisimulation through probabilistic testing. *Information and Computation*, 94:1–28, 1991.

[54] S. Lasota. Coalgebra morphisms subsume open maps. *Theoretical Computer Science*, 280:123–135, 2002.

[55] S. B. Lassen, P. D. Mosses, and D. A. Watt. An introduction to AN-2, the proposed new version of action notation. In *Proc. 3rd Workshop on Action Semantics*, BRICS NS-00-6. Aarhus University, 2000.

[56] M. Lenisa, J. Power, and H. Watanabe. Distributivity for endofunctors, pointed and co-pointed endofunctors, monads and comonads. In *Proc. CMCS'00*, volume 33 of *Electronic Notes for Theoretical Computer Science*. Elsevier, 2000.

[57] S. Mac Lane. *Categories for the Working Matematician*. Springer-Verlag, second edition, 1998.

[58] R. Milner. A calculus of communicating systems. *Journal of the ACM*, 1980.

[59] R. Milner. *Communication and Concurrency*. Prentice Hall, 1988.

[60] R. Milner. *Communicating and Mobile Systems: the π-Calculus*. Cambridge University Press, 1999.

[61] R. Milner and M. Tofte. *The definition of Standard ML*. MIT Press, revised edition, 1997.

[62] L. Moss. Coalgebraic logic. *Annals of Pure and Applied Logic*, 96:177–317, 1999.

[63] D. M. Park. Concurrency and automata on infinite sequences. *Lecture Notes in Computer Science*, 140:195–219, 1981.

[64] D. Pattinson. Semantical principles in the modal logic of coalgebras. In *Proc. STACS 2001*, volume 2010 of *Lecture Notes in Computer Science*. Springer Verlag, 2001.

[65] A. M. Pitts. A co-induction principle for recursively defined domains. *Theoretical Computer Science*, 124(2):195–219, 1994.

[66] G. D. Plotkin. A powerdomain construction. *SIAM Journal of Computing*, 5:452–487, 1976.

[67] G. D. Plotkin. Lecture notes in domain theory (the Pisa notes). 1981.

[68] G. D. Plotkin. A structural approach to operational semantics. DAIMI Report FN-19, Computer Science Department, Aarhus University, 1981.

[69] G. D. Plotkin. Bialgebraic semantics and recursion (extended abstract). In *Proc. CMCS'01*, volume 44 of *Electronic Notes on Theoretical Computer Science*, 2001.

[70] G. D. Plotkin. The origins of structural operational semantics. *Journal of Logic and Algebraic Programming*, 2004. Special issue on structural operational semantics, to appear.

[71] J. Power and D. Turi. A coalgebraic foundation for linear time semantics. *Electronic Notes in Theoretical Computer Science*, 29, 1999.

[72] V. R. Pratt. Chu spaces. Course notes for the School in Category Theory and Applications, Coimbra, Portugal, July 1999.

[73] A. Regev, W. Silverman, and E. Shapiro. Representation and simulation of biochemical processes using the π-calculus process algebra. In *Proc. Pacific Symposium of Biocomputing*, 2001.

[74] J. Rothe and D. Mašulović. Towards weak bisimulations for coalgebras. In *Procs. CMCS'02*, volume 68 of *Electronic Notes in Theoretical Computer Science*, 2002.

[75] J. J. M. M. Rutten. A note on coinduction and weak bisimilarity for while programs. Report SEN-R9826, CWI, Amsterdam, 1988.

[76] J. J. M. M. Rutten. A structural co-induction theorem. In *Proc. MFPS'93*, volume 802 of *Lecture Notes in Computer Science*, 1994.

[77] J. J. M. M. Rutten. Universal coalgebra: a theory of systems. *Theoretical Computer Science*, 249:3–80, 2000.

[78] J. J. M. M. Rutten and D. Turi. Initial algebra and final coalgebra semantics for concurrency. In J. de Bakker et al., editor, *Proc. of the REX workshop A Decade of Concurrency – Reflections and Perspectives*, volume 803 of *LNCS*, pages 530–582. Springer-Verlag, 1994.

[79] D. Sannella and A. Tarlecki. Algebraic preliminaries. In E. Astesiano, H.-J. Kreowski, and B. Krieg-Brückner, editors, *Algebraic Foundations of System Specification*. Springer-Verlag, 1999.

[80] R. de Simone. Higher-level synchronising devices in Meije-SCCS. *Theoretical Computer Science*, 37:245–267, 1985.

[81] M. Smyth and G. D. Plotkin. The category-theoretic solution of recursive domain equations. *SIAM J. Comput.*, 11:761–783, 1982.

[82] S. Thatte. XLANG: web services for business process design. Microsoft document. Available at `http://www.gotdotnet.com/team/xml_wsspecs/xlang-c/default.htm`.

[83] D. Turi. *Functorial Operational Semantics and its Denotational Dual*. PhD thesis, Vrije Universiteit, Amsterdam, 1996.

[84] D. Turi. Categorical modeling of structural operational rules: case studies. In *Proc. CTCS'97*, volume 1290 of *Lecture Notes in Computer Science*, pages 127–146. Springer-Verlag, 1997.

[85] D. Turi and G. D. Plotkin. Towards a mathematical operational semantics. In *Proc. LICS'97*, pages 280–291. IEEE Computer Society Press, 1997.

[86] F. W. Vaandrager. On the relationship between process algebra and input/output automata. In *Proc. LICS'91*, pages 387–398, 1991.

[87] E. de Vink and J. J. M. M. Rutten. Bisimulation for probabilistic transition systems: a coalgebraic approach. *Theoretical Computer Science*, 221:271–293, 1999.

[88] H. Watanabe. Well-behaved translations between structural operational semantics. *Electronic Notes in Theoretical Computer Science*, 65, 2002.

Concept Index

action, 13
    of a functor, 28
    unobservable, 2
adjoint
    left, 26
    right, 26
Alexandrov topology, 92
algebra, 17
    carrier, 17
    for a monad, 20
    for a pointed endofunctor, 113
    free, 18
    initial, 18
    morphism, 17
    structure, 17
arity, 16

behaviour, 1, 20
    deterministic, 1
    endofunctor, 21
    probabilistic, 1
    timed, 1
bialgebra, 23, 114
    morphism, 23
bialgebraic semantics, 5
bifibration, 26
bisimulation, 15, 34, 50
    cpo-, 107
    equivalence, 15, 34, 51
    from coalgebra spans, 95, 97
    ordered, 97
    partial, 94
    preorder, 51
    weak, 6
BPA, 74

categorical logic, 8
category, 17
    base, 26
    enriched, 97
    total, 26
category theory, 3, 17–23
check
    basic flat, 75
    basic term, 76
    failure, 84
    positive, 78
    positive term, 79
Chu space, 8
closure, 39
coalgebra, 20
    carrier, 20
    compact, 105
    final, 21
    for a copointed endofunctor, 113
    morphism, 20
    structure, 20
coalgebraic logic, 6
coinduction principle, 51
    for traces, 62
coinductive extension, 21
complete partial order, *see* cpo
congruence, 17
    format, 17
    on an algebra, 19
    structural, 1
continuous function, 92
cpo, 92
    algebraic, 93
    pointed, 92
CTr-blocking set, 81
    satisfiable, 81
CTr-format, 81

de Simone format, 74
decomposition structure, 121
distributive law, 22, 113

Egli-Milner order, 93
endofunctor, 17
    behaviour, 21
    copointed, 22, 113
    pointed, 113
    polynomial, 18, 72
    syntactic, 18

failure
    equivalence, 14, 58, 86
    pair, 14

preorder, 14, 58, 86
failure trace format, 88
failures-aware relation, 53
fibration, 26
of relations, 32
of test suites, 35
fibre, 26
final semantics, 4
finite element, 93
Fl-blocking set, 86
satisfiable, 86
Fl-format, 87
functor, 17
fibred, 28

van Glabbeek spectrum, 14
Grothendieck construction, 26
GSOS, 16
abstract, 21, 31
image finite, 16

Hennessy-Milner logic, 14

inductive extension, 18
initial semantics, 4
input/output, 2

kernel pair, 95
ordered, 97

labelled synchronization tree, 13
labelled transition system, *see* LTS
Lambek lemma, 18, 21
language construct, 16
lifting
abstract GSOS, 31
algebras, 28
coalgebras, 30
coproducts, 27
endofunctors, 27
polynomial, 72
final objects, 26
freely generated monads, 29
initial objects, 27
products, 26
lifting functor, 93
lifting monads to coalgebras, 117
literal, 16
negative, 16
positive, 16
LTS, 13
finitely branching, 13
induced by Λ, 17

minimal blocking set, 81
modal logic, 13
λ-model, 114
monad, 19
freely generated, 19
morphism, 113

natural transformation, 17

one-by-many simulation, 52
completed, 53
failure, 53
failure trace, 53
ready, 53
ready trace, 54
open maps, 6
operational semantics, 2, 15

pentagonal law, 23
Plotkin powerdomain, 93
possible futures equivalence, 65
possible worlds equivalence, 65
precongruence, 17
on an algebra, 19
preorder, 15
algebraic, 102
finitary, 102
strongly finitary, 102
weakly algebraic, 102
process, 13
algebra, 1
equivalence, 2, 14
preorder, 2, 14

quantales, 6
quasi-preorder, 52
quasi-upper set, 52

readiness-aware relation, 53
ready
equivalence, 14, 60
pair, 14

preorder, 14, 60
reindexing, 26
rule, 16
conclusion, 16
for `f`, 16
premise, 16
source, 16
target, 16

Scott topology, 92
Sierpinski space, 99
signature, 16
simulation, 15, 48
2-nested, 66
equivalence, 65, 66
preorder, 66
equivalence, 15, 49
preorder, 15, 49
ready, 15, 49
equivalence, 15, 50
preorder, 15, 50
specialization
equivalence, 38
functor, 37
preorder, 38, 99
square commutation by fixpoint induction, 115
state, 2
strict function, 92
syntax, 1

Tarski theorem, 114
term, 16
closed, 16
test, 36
basic flat, 75
basic term, 76
failure, 84
intersection, 43
positive, 79
positive term, 79
suite, 36
union, 43
test constructor, 39
topology, 7
Alexandrov, 92
closed under finitary intersections, 105
Scott, 92
Tr-format, 74
trace, 14
completed, 14
equivalence, 14, 57
preorder, 14, 57
decorated, 52
equivalence, 14, 35, 56, 63
failure, 14
equivalence, 14, 59
preorder, 14, 59
preorder, 14, 55
ready, 14
equivalence, 14, 62
preorder, 14, 62
trace-aware relation, 52
completed, 53
failure, 53
ready, 54
transition
relation, 13
system
labelled, *see* LTS
probabilistic, 2
specification, 16
timed, 2
with divergence, 94
with unobservable steps, 2

unfolding rule, 121
upper set, 48

van Glabbeek spectrum, 14
variable binding, 1
variable classifier, 116
very strict function, 92

List of Symbols

$[a]$
modal operator, 14, 66

$\bowtie$
test suite construction, 72

$\bigsqcup_D$
least upper bound, 92

$\bot$
bottom element
in a cpo, 92
in a fibre, 27
logical constant, 14, 66

$\boxplus$
coproduct in a total category, 27, 71

$\boxtimes$
product in a total category, 26, 71

$\cong_{\mathsf{W}}$
process equivalence, 14, 66

$\Circledcirc$
5-test operator, 68
logical operator, 66

$\equiv_\theta$
specialization equivalence, 38, 43

$\wedge$
logical function, 72
logical operator, 14, 66
meet in a fibre, 26
test intersection, 43

$\leq_\theta$
specialization preorder, 38, 43, 99

$\lesssim_{\mathrm{EM}}$
Egli-Milner order, 93

$\vee$
join in a fibre, 27
logical operator, 14, 66
test union, 43

$\models_h$
satisfaction relation, 14, 66

$\rightrightarrows$
test suite notation, 36

$\not\longrightarrow$
failure check, 84

$\oplus$
coalesced sum, 117

$\otimes$
smash product, 117

$[\![-]\!]_h$
semantics of formulae, 46, 68

$\sqcap, \sqcup$
5-test operators, 68

$\sqsubseteq_{\mathsf{W}}$
process preorder, 14, 66

$\top$
logical constant, 14, 66
top element in a fibre, 26

$\langle a \rangle$
modal operator, 14, 66

$\xrightarrow{a}$
positive check, 79

$x \overset{Q}{\not\longrightarrow}$
negated transition, 13

2
test value set, 37, 43

5
test value set, 67

γ
decomposition structure, 121

η
monad unit, 19

Φ^*
fixpoint, 114

Φ_h, Ψ_h
operators on fibres, 28, 30

$\Lambda_{\mathtt{f}}, \Lambda_{\mathtt{f}Q}$
sets of rules, 81

λ-Bialg
category of bialgebras, 23

λ^r
distributive law, 126

μ
monad multiplication, 19

ΣX
set of terms, 16

Σ
functor, 18
signature, 16, 18
Σ**-Alg**
category of algebras, 18
$\bar{\Sigma}$
set of constructs, 16
σ_D
Scott topology, 92
θ_R
test suite, 48, 55
topology, 103
θ_{2S}
test suite, 69
υ
variable classifier, 121
$\upsilon(-)$
test for a check, 75, 76, 79, 84
Ξ
set of variables, 16
ζ
test suite, 38, 41

Acpo$_\perp$
category, 93
ar
arity function, 16
B^{2S}
functor, 68
B^{Tr}, B^{CTr}, B^{Fl}, B^{FlTr}, B^{Rd}, B^{RdTr}, B^{S}, B^{RdS}, B^{BS}
functors, 45
B^{W}
functor, 40
B**-Coalg**
category of coalgebras, 21
Cl^t
closure, 100
$\mathrm{Cl}^{\mathsf{2S}}$
closure, 68
$\mathrm{Cl}^\vee$
closure, 72
$\mathrm{Cl}^\top, \mathrm{Cl}^\wedge, \mathrm{Cl}^{\vee\wedge}$
closures, 44
Cppo$_\perp$
category, 117
$D_\perp$
lifted cpo, 93
$\mathsf{E}, \mathtt{T}, \mathtt{F}$
constant 5-tests, 67, 69
F_X, B_X, Σ_X
functor actions, 28, 33, 72
f^*
reindexing function, 26, 33, 36
$f_!$
left adjoint to f^*, 26
$\mathcal{F}_{\mathsf{W}}$
modal formulae, 13, 66
h_λ
intended operational model, 23
$I(x)$
set of initials, 13
$K(D)$
set of finite elements, 93
$k^*\omega$
test suite, 40
the least lifting, 30
$\mathbb{O}$
Sierpinski space, 99
$\mathcal{P}$
powerset functor, 21
$\mathcal{P}^0$
Plotkin powerdomain with empty set, 93
$\mathcal{P}_{\mathrm{f}}$
finite powerset functor, 21
$\mathcal{P}_{\mathrm{f}}(A \times -)$
behaviour functor, 21
Pos
category of posets, 26
$\check{Q}$
modal operator, 14
$\tilde{Q}$
modal operator, 14
R^F
finitary preorder, 104
Rel
category of relations, 33
$\bar{r}$
infinite unfolding, 123
r
unfolding rule, 121

Set
category of sets, 18
Sp_R
specialization functor action, 37
Spec_R
specialization functor, 37
$T_\Sigma X$
set of terms, 16
T_λ
lifting of a monad, 117
$\mathsf{Tr}, \mathsf{CTr}, \mathsf{Fl}, \mathsf{FlTr}, \mathsf{Rd}, \mathsf{RdTr}, \mathsf{BS}$
test constructor sets, 44
T**-Alg**
category of algebras, 20
$\mathtt{tt}, \mathtt{ff}$
test values, 37, 43, 99
$\mathtt{tt}, \mathtt{ff}, \mathtt{tt}_\neg, \mathtt{ff}_\neg, \mathtt{err}$
test values, 67
$\langle T, \eta, \mu \rangle$
monad, 19
t
monad morphism, 121
$\mathcal{T}_\mathcal{V}$
functor, 36
$u_{\langle a \rangle}, u_{[a]}$
5-test constructors, 67
$\mathcal{V}$-**TS**
category of test suites, 36
$\mathcal{V}, \mathcal{W}$
sets of test values, 36
$\overline{V}$
set from a test, 43, 99
W
set of test constructors, 40
symbol, 14
$w_\Box, w_\Diamond$
$\mathbb{O}$-test constructors, 100
$w_{\langle a \rangle}, w_{[a]}, \tilde{w}_Q, \tilde{w}_{aQ}, \check{w}_Q, \check{w}_{aQ}$
2-test constructors, 44
X^*
fibre, 26, 33
$X^\star$
closure in a cpo, 93
$\overrightarrow{X}$
test from a set, 43, 99