On generalised coinduction and probabilistic specification formats

Bartels, F.

2004

document version
Publisher's PDF, also known as Version of record

VRIJE UNIVERSITEIT

On Generalised Coinduction and Probabilistic Specification Formats

Distributive laws in coalgebraic modelling

ACADEMISCH PROEFSCHRIFT

ter verkrijging van de graad van doctor aan
de Vrije Universiteit Amsterdam,
op gezag van de rector magnificus
prof.dr. T. Sminia,
in het openbaar te verdedigen
ten overstaan van de promotiecommissie
van de faculteit der Exacte Wetenschappen
op donderdag 3 juni 2004 om 13.45 uur
in de aula van de universiteit,
De Boelelaan 1105

door

Falk Bartels

geboren te Wilhelmshaven, Duitsland

promotoren: prof.dr. J.J.M.M. Rutten
prof.dr. J.C.M. Baeten

On Generalised Coinduction and Probabilistic Specification Formats

Distributive laws in coalgebraic modelling

Falk Bartels

April 2004

IPA Dissertation Series 2004 – 06

The work reported in this thesis was funded by NWO (Dutch Organisation for Scientific Research, ProMACS project) and has been carried out at the CWI (Center for Mathematics and Computer Science, Amsterdam) under the auspices of the research school IPA (Institute for Programming research and Algorithmics).

Contents

Preface

The present thesis reports most of the scientific results I obtained during a four year Ph.D. project funded by the Dutch Science Foundation NWO in the context of the *ProMACS* project. I have been working in the department of Software Engineering (SEN) at the CWI, the National Research Institute for Mathematics and Computer Science in the Netherlands. The CWI provides an inspiring and international working environment and it is a recommendable place for Ph.D. students. With only very few side obligations imposed on me, I could spend almost all of my time on actual research, and our frequent meetings and seminars gave me plenty of opportunity to present and discuss the ideas that I was working on. Collaboration with other research groups was particularly encouraged by the institute, and we organised regular meetings with other scientific institutions. Most important for me was Jos Baeten's formal methods group at the Technical University of Eindhoven (TU/e) – our partner in the ProMACS project – and Bart Jacobs' group at the Catholic University of Nijmegen (KUN). I also met many other Ph.D. students working in the Netherlands during the excellent spring and autumn schools organised by IPA, the research school which I was a member of. Next to the IPA courses, I benefited from the summer school on Formal Methods and Performance Analysis (FMPA 2000) and the Estonian Winter School in Computer Science (EWSCS 2002).

Jan Rutten became a professor at the Free University of Amsterdam (VU) while I was working as his student. This made it possible for him as my daily supervisor to join Jos Baeten by taking the formal role of a promoter as well. This moreover gave me the opportunity to have my promotion in Amsterdam, the town that I was working and living in. I am very grateful to Jos Baeten, who generously agreed with this change. I want to thank both of my promoters for their continuing support of my work.

Over the years, I got a chance to discuss scientific issues and work together with many people. I want to express my thanks to all of them: There are the members of my group, the Coordination Languages theme (SEN3) of Jan Rutten, and particularly those working on coalgebra, namely Jan himself, Alexandru Baltag (now Oxford University), Alexander Kurz (now University of Leicester), and Clemens Kupke. And I keep in my memory the inspiring time with Matteo Coccia, who stayed with us for an internship. Special thanks go to my colleague

Farhad Arbab for his continued willingness to answer my questions on the English grammar (despite his influence, you will still find several strange sentences, I am afraid). Further language related help I got from Juan Guillen-Scholten and Peter Zoeteweij, who turned my Dutch summary into a readable text. Next to Jan and Alexandru, I worked with Suzana Andova (now Twente University), Jos Baeten, and Erik de Vink from the TU/e in the ProMACS project. With Ana Sokolova from TU/e and Erik we later continued the study of probabilistic systems with coalgebraic means. I acknowledge fruitful discussions with Tarmo Uustalu and Varmo Vene, who I first meat in Tallinn at the above mentioned winter school, H.Peter Gumm – also for not purely scientific meetings around various schools and workshops – and Dirk Pattinson. I also would like to thank the various anonymous referees, who evaluated the papers my coauthors and I submitted mainly to the CMCS (Coalgebraic Methods in Computer Science) workshop series. As another important person for my scientific career, I want to mention Harald Rueß (University of Ulm, now SRI). As the daily supervisor for my diploma project, he brought coalgebras to my attention, which later caused me to apply for a Ph.D. position in Jan's group.

For valuable comments on drafts of this thesis I am greatly indebted to my promoters, Jos Baeten (TU/e) and Jan Rutten (CWI, VU), as well as the members of the reading committee, namely Wan Fokkink (CWI, VU), Bart Jacobs (KUN), Joost-Pieter Katoen (University of Twente), Larry Moss (University of Indiana), and Erik de Vink (TU/e). Also from Mariëlle Stoelinga (KUN, now University of California, Santa Cruz) I received helpful remarks.

The biggest part of my scientific achievements I owe to my boss and daily supervisor Jan Rutten. His experience, judgement, patience, support, and his willingness to explain and to listen made my working life in Amsterdam a success and a pleasure. Throughout, but in particular during the last months, he sacrificed much of his time reading my drafts and discussing all issues, from the overall setup down to individual sentences. I specially appreciate his conviction that one should attempt to write articles for a rather broad audience, trying to express complicated matters in the simplest possible form. As a result, many young researchers – like myself a few years ago – entered the field of coalgebra through his articles. Responding to his advice, I often rewrote my drafts in order to make them better readable, and if there is some part in any of my papers that is decently explained, then it is certainly because of his influence.

Chapter 1

Introduction

This thesis has two main subjects: generalised coinduction schemata and specification formats for probabilistic systems. We shall start this introduction with a quick overview of the thesis in order to explain the common background and relation of both parts. In the following Sections 1.1 – 1.5 we then introduce the work in more detail.

The two subjects are both based on the categorical modelling of dynamic systems and potentially infinite data structures as *coalgebras* of a functor [JR96, Rut00b]. With different choices for this functor, coalgebras describe systems with different types of behaviour, such as, for instance, labelled transition systems or infinite data streams.[1] Therefore we often use the term *behaviour functor* in this context.

The coalgebraic approach allows us to develop a general theory of dynamic systems, largely independent of their concrete behaviour type. It offers, for instance, an abstract notion of behavioural equivalence expressed in terms of *coalgebraic bisimulations* [AM89]. Moreover, in many interesting cases, we can uniformly characterise an abstract domain for behaviours of the type under consideration as a *final coalgebra* of the corresponding behaviour functor. A coalgebra is called final if for any coalgebra of the same functor there exists a unique homomorphism, i.e. a behaviour preserving map, from that coalgebra to the final one. This implies that for any state in any concrete system there is precisely one state of a final coalgebra showing the same behaviour.

The defining property of a final coalgebra gives rise to a *coinduction principle*, which can be used both to define abstract behaviours and to prove them equal. Unfortunately, these elementary principles are rather rigid: with them alone many specifications cannot be expressed directly and many equivalences cannot be proved elegantly. To remedy this situation, various generalisations of the

[1] The term behaviour is more appropriate for a process than for a data structure, but we shall use it in both cases, identifying a data structure with the process that decomposes it.

basic coinduction definition and proof principles have been considered [Mil89, Geu92, VU98, San98, UV99]. We focus on these extended principles in the first main part of our work. We shall show that several of the known generalised principles can be viewed uniformly as different instances of the same abstract schema. This provides a deeper understanding of the underlying ideas and facilitates validity proofs. Moreover, the approach allows the derivation of new principles, as we shall demonstrate as well. This abstract schema is expressed in terms of the categorical notion of a *distributive law* between two functors.

As our second main subject, we introduce well-behaved operator specification formats for probabilistic transition systems. One needs these operators to construct systems and reason about them in a compositional manner. For the latter, the operators should be well behaved, for instance, in the sense that the behaviour of a composed system should not change if one component is replaced by an equivalent one. Our formats facilitate the definition of composition operators with such properties: we prove that every specification respecting the format is valid, i.e. it defines the operators uniquely; moreover, these operators satisfy well-behavedness properties such as the one above. In the literature, formats for well-behaved specifications have mainly been studied for nondeterministic systems, for which various transition rule formats have been proposed and successfully applied (see [AFV01] for an overview). For probabilistic systems, however, formats of that kind have not been proposed.

In general, working with probabilities is more complicated than working with pure nondeterminism: when probabilities are given, transitions to the possible successor states cannot be treated independently anymore, since the probabilities associated to the individual transitions need to sum up to one. It turned out that, in order to cope with this complication, a categorical theory of well-behaved operator specifications [TP97] is rather helpful. In this abstract framework, an operator specification is a distributive law between two functors, the first of which is derived from the signature of the operators and the second from the behaviour type under consideration. We already encountered such distributive laws in the study of generalised coinduction principles.

To model probabilistic operator specifications categorically, it is important to observe that various kinds of probabilistic systems are coalgebras for suitable behaviour functors [VR99, Mos99, BSV03]. It moreover turned out that, as it was the case for LTSs, the notion of bisimilarity arising from this coalgebraic presentation yields the same process equivalence than the original notion of probabilistic bisimilarity. In order to obtain concrete specification formats, we need to be able to give concrete characterisations of the corresponding distributive laws, which are natural transformations of a certain type. To this end we first decompose the natural transformations under consideration using a collection of representation lemmata. This method allows us to derive well-behaved operator specification formats not only for probabilistic systems, but also for other types of systems and data structures that can be modelled as coalgebras, as we demonstrate with deterministic automata [Rut98] and infinite

data streams [Rut01].

Both parts of our work are thus applications of distributive laws in the coalgebraic study of dynamic systems. And when we instantiate our results on generalised coinduction with the distributive laws resulting from the setting of operator specification formats we obtain new well-behavedness statements for the operator specifications covered: guarded recursive equations with the defined operators have solutions, which are determined up to bisimilarity, and for those operators a bisimulation up-to-context technique [San98] is valid.

Next to the main subjects mentioned above, we also devote one chapter of this thesis to a detailed explanation of the categorical theory of well-behaved operator specification formats. We do so to provide the background for our work but also because we need to extend the theory presented in the literature for our purposes.

Although we work with categorical notions throughout this thesis, we limit ourselves to the use of basic concepts and do not assume a strong familiarity with the field. In particular, our formats for probabilistic systems and their properties are expressed as is common practice in process theory, i.e. without using categorical definitions. We expect that they will be of interest also for readers who work with these systems without using coalgebraic techniques.

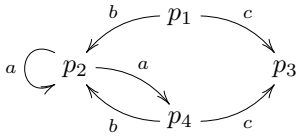
The remainder of this introduction is organised as follows. We start by motivating the coalgebraic modelling of dynamic systems (Section 1.1). Then we explain our work on generalised coinduction principles (Section 1.2) and specification formats for probabilistic systems (Section 1.3). Finally, we summarise related work (Section 1.4) and our contributions (Section 1.5).

1.1 Coalgebraic modelling

To illustrate the coalgebraic modelling of dynamic systems we consider *labelled transition systems (LTS)* as they are studied, for instance, in *process algebra* [BW90, Fok00]. Given a set of labels L, an LTS is a pair

$$\langle P, (\xrightarrow{a} \subseteq P \times P)_{a \in L} \rangle$$

of a set of states P and a family of transition relations $\xrightarrow{a}$ for each label $a \in L$. We draw a picture of such a system below, where we use the notation $p \xrightarrow{a} q$ for $\langle p, q \rangle \in \xrightarrow{a}$.



(1.1)

If $p \xrightarrow{a} q$ then we call q an *a-successor* of p, and if p has a-successors then we say that a is *enabled* in p. Note that since a state of an LTS may have more than

one a-successor for a label a – such as p_2 in the above picture – the systems are potentially *nondeterministic*.

We assume that an observer of the system cannot see directly which state it is in. He can obtain information about the current state only by checking which labels are enabled and by successively inspecting all successor states for the different labels in the same manner. We call two states *behavioural equivalent* if they cannot be distinguished this way, such as the states p_1 and p_4 in the above system. More precisely, with the type of observation explained above, the notion of behavioural equivalence is that of *(strong) bisimilarity* [Mil80, Mil89, Par81]. This is the most widely accepted process equivalence for LTSs. We restrict ourselves to strong bisimilarity here, but note that other notions are considered in the literature as well, such as weak bisimilarity or trace equivalence (see e.g. [Gla90, Gla93, Gla01]).

To motivate the coalgebraic presentation of labelled transition systems, note that the symmetric presentation of the transition structure as a family of binary relations does actually not reflect our idea of the system behaviour properly: the latter is determined by the outgoing transitions of a state, the ingoing ones are ignored. This directed interpretation is expressed better by the presentation of an LTS as a pair

$$\langle P, \alpha : P \to (\mathcal{P}P)^L \rangle, \tag{1.2}$$

where $\mathcal{P}P = \{P' \mid P' \subseteq P\}$ denotes the *powerset* of P. Here P is the same set of states as before, but instead of the transition relations we now consider a *transition function* α, which assigns to each state a description of the outgoing transitions of that state. The relation of the two presentations is described by the following equivalence:

$$p \xrightarrow{a} q \iff q \in \alpha(p)(a)$$

The presentation of an LTS as in (1.2) generalises to an abstract description of dynamic systems as *coalgebras*, which are pairs

$$\langle P, \alpha : P \to \mathrm{B}P \rangle, \tag{1.3}$$

where the set $\mathrm{B}P$ is constructed in a systematic way from P. The set-theoretic construction B is described by the notion of a *functor* from category theory [ML97], and we call the above pair a *coalgebra* of the functor B, the set P its *carrier*, and α its *structure map*.

This view led to a study of *universal coalgebra* [Rut00b] as an abstract theory of dynamic systems. It also provides, for every functor, a general notion of behaviour preserving maps between two systems. These so-called *homomorphisms* are functions from one carrier set to the other which respect the structure maps. Similarly, there is a coalgebraic notion of a bisimulation [AM89] as a binary relation on the carrier sets of two coalgebras interacting nicely with the structure maps. For the interpretation of the abstract results in the concrete case of

LTSs it is essential to observe that coalgebraic bisimilarity for the corresponding behaviour functor coincides with the (strong) bisimilarity mentioned above [RT94].

The so-called *final coalgebras*, which are defined by the property that there exists precisely one homomorphism from any given coalgebra to the final one, can be shown to provide an abstract domain for behaviours of the type under consideration. The finality condition implies that any possible system behaviour is uniquely represented in the final coalgebra in the sense that any state in any coalgebra is bisimilar to precisely one state in the final coalgebra. Finality is the basis for so-called coinductive definitions and proofs, as we shall explain in the coming section. The theory of coalgebras provides results about the existence of final coalgebras for rich classes of functors.

1.2 Generalised coinduction

In this section we explain the coinduction definition and proof principle in its basic form and discuss extensions. Here we shall use final coalgebras to model infinite data structures. As an example, we consider finite and infinite lists over elements of some set A, namely

$$A^\infty := \{\langle a_1, \ldots, a_n\rangle \mid n \in \mathbb{N}, a_i \in A\} \cup \{\langle a_1, a_2, \ldots\rangle \mid a_i \in A\}.$$

These are often referred to as *sequences* or, in functional programming, *lazy lists*. We write $\varepsilon \in A^\infty$ for the empty sequence and $(a : l) \in A^\infty$ for a sequence with a first element $a \in A$ followed by a lazy list $l \in A^\infty$. In the latter case we call a the *head* and l the *tail* of the sequence.

1.2.1 Coinductive definition principles

Suppose that, for a given function $g : A \to A$, we want to define the function $\mathtt{map}_g : A^\infty \to A^\infty$ that applies g to all elements of a given sequence. The function $\mathtt{map}_g$ should satisfy the clauses

$$\mathtt{map}_g(\varepsilon) \;:=\; \varepsilon \quad \text{and} \quad \mathtt{map}_g(a : l) \;:=\; g(a) : \mathtt{map}_g(l). \tag{1.4}$$

The problem with this "definition" is that it is *circular*: the second clause specifies the result of $\mathtt{map}_g$ in terms of $\mathtt{map}_g$ itself. Therefore it is not obvious that there exists a function satisfying these identities, and, if so, whether it is uniquely determined.

Before we discuss the problem in the given form, we recall the usual approach in the case of only finite lists, i.e. for

$$A^* := \{\langle a_1, \ldots, a_n\rangle \mid n \in \mathbb{N}, a_i \in A\}.$$

Here we would reason *inductively* to show the validity of the specification of $\mathtt{map}_g$: the reference to $\mathtt{map}_g(l)$ in the term defining $\mathtt{map}_g(a : l)$ is not troublesome because the list l is shorter than the original list $a : l$. This guarantees that after unfolding the definition of $\mathtt{map}_g$ a finite number of times we will eventually encounter an application of $\mathtt{map}_g$ to the empty sequence, so the unfolding process will terminate, which means that the result is uniquely determined.

If we allow infinite sequences, however, this reasoning based on the termination of the unfolding process fails, because it may actually not terminate. To show that the definition is valid still, we shift our attention from the reduction of the argument list to the construction of the resulting sequence: unfolding the definition of $\mathtt{map}_g$ applied to a sequence of the type $a : l$, we obtain a part of the result, namely the first element $g(a)$, and a description of the remaining sequence, namely $\mathtt{map}_g(l)$. Unfolding the latter, we obtain the second element of the result, and so on. An infinite unfolding process will thus reveal longer and longer prefixes of the result, and so every element of it is uniquely determined eventually. This way of reasoning is called *coinductive*.

Note the change of focus from inductive to coinductive reasoning: if the function to be specified occurs again in the term defining it, the induction principle requires that by repeatedly unfolding the definition the arguments become smaller and smaller, whereas the coinduction principle requires that more and more information about the result is revealed; induction constrains the argument l of the occurrence of $\mathtt{map}_g$ in the defining term $g(a) : \mathtt{map}_g(l)$, whereas coinduction looks at the context in which $\mathtt{map}_g$ occurs, which is the prefix operation $g(a) : \ldots$ in our case.

To provide a clear mathematical foundation for the above coinductive reasoning, we model lazy lists over A coalgebraically [HJ98]. Writing $1 = \{*\}$ for an arbitrary singleton set and $X + Y$ for the disjoint union of the two sets X and Y, sequences arise as the behaviour of dynamic systems that are given by a pair

$$\langle P, \alpha : P \to 1 + (A \times P) \rangle \tag{1.5}$$

of a set of states P and a transition function α (cf. equation (1.3)), i.e. a coalgebra of the functor

$$\mathrm{B}X := 1 + (A \times X). \tag{1.6}$$

For any state $p \in P$ of such a system we either have $\alpha(p) = *$, which means that the system has terminated in the state p, or $\alpha(p) = \langle a, p' \rangle$ for some $a \in A$ and $p' \in P$, which means that in p the system produces the value a and assumes a next state p'. An observer of the system notices the sequence of produced values. Depending on whether or not the system will eventually stop, this sequence is finite or infinite.

The sequences A^∞ are turned into a coalgebra $\langle A^\infty, \omega : A^\infty \to 1 + (A \times A^\infty) \rangle$

of the functor B from (1.6) by defining

$$\begin{aligned} \omega(\langle a_1, \ldots, a_n \rangle) &= \begin{cases} * & \text{if } n = 0 \\ \langle a_1, \langle a_2, \ldots, a_n \rangle \rangle & \text{otherwise} \end{cases} \\ \omega(\langle a_1, a_2, a_3, \ldots \rangle) &= \langle a_1, \langle a_2, a_3, \ldots \rangle \rangle \end{aligned}$$

As an instance of the general definition, a function $h : P \to A^\infty$ is a homomorphism from any coalgebra $\langle P, \alpha \rangle$ as in (1.5) to $\langle A^\infty, \omega \rangle$ if for all $p \in P$ we have $h(p) = \varepsilon$ if $\alpha(p) = *$ and $h(p) = a : h(p')$ if $\alpha(p) = \langle a, p' \rangle$. It is easy to see that for any $\langle P, \alpha \rangle$ there is precisely one function h with this property, which is the one that maps $p \in P$ to the sequence $h(p)$ of values the system would produce when started in p. So the coalgebra $\langle A^\infty, \omega \rangle$ is final.

The finality yields a definition principle for sequences: in order to specify a function $h : P \to A^\infty$ it suffices to provide the function α from (1.5). The homomorphism property translates into a specification format: we uniquely determine the function h by giving for each $p \in P$ an equation

$$h(p) = \varepsilon \quad \text{or} \quad h(p) = a : h(p') \quad \text{for some } a \in A \text{ and } p' \in P. \tag{1.7}$$

For coalgebras of arbitrary functors, we call the specification format obtained from such a direct translation of the finality property the *coiteration schema*, and a specification in that format is called *coiterative*. As we can see, the specification of $\mathtt{map}_g$ in (1.4) is in the format in (1.7), so it is coiterative. The coalgebra $\langle P, \alpha \rangle$ induced by the specification is

$$\langle A^\infty, (\mathtt{id}_1 + \langle g, \mathtt{id}_{A^\infty} \rangle) \circ \omega : A^\infty \to 1 + (A \times A^\infty) \rangle.$$

Many specifications of interesting functions, however, are not coiterative. As a rather simple example, we consider the function $\mathtt{zip} : A^\infty \times A^\infty \to A^\infty$ that interleaves two sequences:

$$\begin{aligned} \mathtt{zip}(\varepsilon, l') &= l' \\ \mathtt{zip}(a : l, l') &= a : \mathtt{zip}(l', l) \end{aligned}$$

The specification is not coiterative since the sequence l' given as the result in the first identity is not necessarily empty, as required by the format in (1.7). So the coiteration schema is not expressive enough to capture the given specification of `zip` directly.

We thus try to find definition schemata which are more liberal than coiteration. A schema that covers the example of `zip` has already been proposed: it is the *primitive corecursion schema* (see e.g. [Geu92, VU98]), which arises as the categorical dual of primitive recursion. The dual of course-of-value iteration yields a coinductive definition schema as well [UV99], which covers other non-coiterative specifications.

We mention both examples here to illustrate that different specifications may require different extended coiteration schemata, and each of them needs to be proved valid (see e.g. [UV99]). As a step towards a more uniform description, Lenisa [Len99a] observed that various non-coiterative functions can be written as a coiterative function precomposed with the unit η of a *pointed functor* $\langle \mathrm{T}, \eta \rangle$, which is a functor T together with a natural transformation $\eta : \mathrm{Id} \Rightarrow \mathrm{T}$. She exploits this fact to establish properties of functions defined by different schemata.

Lenisa presented functions obtained from several extended coiteration schemata in a uniform way. We carry the idea further and show that those schemata themselves, together with the validity proofs, arise as instances of a novel generalised coinduction schema. With this observation we provide a better understanding and simpler justifications of existing extended schemata, as we shall show for the duals of primitive recursion and course-of-value iteration above and the definition of a language by a nondeterministic automaton. But moreover we view our approach as a tool to develop and analyse further principles. As an example, we derive a new principle in which certain auxiliary operators may be used in a coiterative definition. The functions defined by this principle can be interpreted as solutions of guarded recursive equations. The means to characterise the class of auxiliary operators that may be used in this context shall be provided by our work on operator specification formats, which we are going to sketch in the following section.

Our generalised coinduction schema is parameterised by a *distributive law* λ of some functor S over the behaviour functor B under consideration, and we therefore call it the *λ-coiteration schema*. As it were, the functor S describes an additional syntax for the resulting schema, the semantics of which is defined by the distributive law λ.

1.2.2 Coinductive proof principles

For a large class of functors we can prove that on the carrier of a final coalgebra bisimilarity and equality coincide. This yields the following coinduction proof principle: to show that two states of a final coalgebra are equal, it suffices to exhibit a *bisimulation* relating them.

We can apply this principle to sequences, since they form a final coalgebra of the functor B from (1.6), as we mentioned above. Before we give an example, we first spell out the coalgebraic notion of a bisimulation for this case: a bisimulation between sequences is a relation $R \subseteq A^\infty \times A^\infty$ such that for all $\langle l, l' \rangle \in R$ we have that either (a) l and l' are both empty, or (b) they have the same first element and the tails are related by R again. As an example, we prove

$$\mathtt{map}_g(\mathtt{map}_{g'}(l)) = \mathtt{map}_{(g \circ g')}(l)$$

for all functions $g, g' : A \to A$ and sequences $l \in A^\infty$. To establish the identity,

we show that, for two functions $g, g' : A \to A$, the relation

$$R := \{\langle \mathtt{map}_g(\mathtt{map}_{g'}(l)), \mathtt{map}_{(g \circ g')}(l)\rangle \mid l \in A^\infty\}$$

is a bisimulation: for $l = \varepsilon$ we calculate

$$\mathtt{map}_g(\mathtt{map}_{g'}(\varepsilon)) = \varepsilon = \mathtt{map}_{(g \circ g')}(\varepsilon),$$

so the corresponding pair satisfies case (a) above; for $l = a : l'$ we find

$$\begin{array}{rcl} \mathtt{map}_g(\mathtt{map}_{g'}(a : l')) & = & \mathtt{map}_g(g'(a) : \mathtt{map}_{g'}(l')) \\ & = & g(g'(a)) : \mathtt{map}_g(\mathtt{map}_{g'}(l')) \\ \mathtt{map}_{(g \circ g')}(a : l') & = & g(g'(a)) : \mathtt{map}_{(g \circ g')}(l') \end{array}$$

which means that the pair satisfies case (b), since both sequences have $g(g'(a))$ as the first element and the two tails $\mathtt{map}_g(\mathtt{map}_{g'}(l'))$ and $\mathtt{map}_{(g \circ g')}(l')$ are related by R again. Thus we have established that for all $l \in A^\infty$ the sequences $\mathtt{map}_g(\mathtt{map}_{g'}(l))$ and $\mathtt{map}_{(g \circ g')}(l)$ are bisimilar, which implies that they are equal.

In many cases, however, the construction of a bisimulation is less straightforward: it is not always sufficient to include the pairs arising from the equations one wants to prove. In particular, we are often forced to consider pairs which appear superfluous like in the following simple example. Given a function $g : A \to A$, we want to prove

$$\mathtt{map}_g(\mathtt{zip}(l, l')) = \mathtt{zip}(\mathtt{map}_g(l), \mathtt{map}_g(l'))$$

for all $l, l' \in A^\infty$. Proceeding as above example above, we try to show that the relation

$$R := \{\langle \mathtt{map}_g(\mathtt{zip}(l, l')), \mathtt{zip}(\mathtt{map}_g(l), \mathtt{map}_g(l'))\rangle \mid l, l' \in A^\infty\}$$

is a bisimulation. For $l = \varepsilon$ and $l' \in A^\infty$ both expressions simplify to $\mathtt{map}_g(l')$, so these streams related by R are obviously identical. Still we formally need to check the assumption on a bisimulation for them. This can be avoided by using a generalised proof principle based on what we call *bisimulations up-to-equality*. Note that the problem in this simple example is not so severe in that the above relation can still be shown to be a bisimulation. In other examples we need to consider a much larger relation in order to establish this property.

For LTSs there exist various proof principles based on weaker notions than that of a bisimulation, often called bisimulations up-to-... [Mil89]. An important example is a *bisimulations up-to-context* technique [San98].

We show that the idea behind our generic λ-coiteration definition principle also yields generalised coinduction proof principles: for a given distributive law λ, we generalise bisimulations to λ-*bisimulations* and prove that, under mild assumptions, λ-bisimulations are sufficient to prove bisimilarity. The advantage

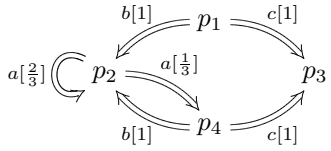
of this approach again is that once we have established the validity of the general schema, several different extended principles become available. As different instances we obtain the above mentioned bisimulation up-to-equality and bisimulation up-to-context principles. Our result about bisimulation up-to-context shows that the principle is valid for contexts built with a larger class of operators on LTSs than it was known before, and it makes the technique available for other types of systems.

1.3 Formats for probabilistic systems

As the second main contribution of this thesis, we introduce the first specification formats to define well-behaved composition operators for *probabilistic systems*. The transition structure of such a system defines a probability for each of the transitions leaving a system state. Different from nondeterministic systems, which give us only qualitative information about the possible transitions of the system, probabilistic systems provide quantitative information about how the choice between the different transitions will be made. They serve, for instance, as semantic models for probabilistic algorithms. Probabilities are also needed for performance analysis [BHK01], where one studies aspects such as the average throughput of a system, its resource utilisation, or its reliability in the presence of faulty system components.

Another reason to add quantitative information to transition systems is the need to describe the timing of the transitions. And sometimes this timing is probabilistic (see e.g. [Ber99]). In this thesis we use probabilities only for the choice of the actual transition, no timing is considered. But the categorical machinery we employ has independently been applied to timed systems by Kick [Kic02a, Kic02b, Kic03].

Adding probabilistic information to the example LTS in (1.1) we may obtain the following *probabilistic transition system (PTS)* [LS91], where the transition probabilities are written in square brackets. We draw the probabilistic transitions as double arrows in order to distinguish them from the nondeterministic ones.



We have added probability distributions over all transitions leaving one state with the same label. Alternatively, we could for each state specify one probability distribution over all its outgoing transitions, so that one distribution would range over the two transitions from state p_1 for instance. Then we would obtain a *generative system*, which chooses the transition label probabilistically

as well, instead of the above *reactive system*, which reacts probabilistically to labels provided by the environment (both names are from [GSS95]). As this discussion already suggests, the variety of different probabilistic system types that we can find in the literature is much larger than that of nondeterministic systems. We consider two kinds from this variety: the above PTS and the more complex probabilistic automata of Segala [SL94, Seg95b], which combine nondeterministic and probabilistic choice and which we refer to as *Segala systems*.

Probabilistic systems of various types, including PTS and Segala systems, can be modelled as coalgebras of suitable behaviour functors [VR99, Mos99, BSV03, BSV04]. Moreover, these coalgebraic definitions yield notions of bisimilarity which coincide with probabilistic bisimilarity as it has been defined in concrete terms for several system types.

In order to construct probabilistic systems and reason about them compositionally, we need well-behaved composition operators. As an example, we consider the specification of an asynchronous parallel composition for PTS, which is parameterised by the probability $r \in [0,1]$ that the left component reacts on an input label which is enabled for both components (so the right one reacts with probability $\bar{r} := 1 - r$ in that case). We specify the operator by the following rules for all $a \in L$ and $u \in (0,1]$:

$$\frac{x \overset{a[u]}{\Longrightarrow} x' \quad y \overset{a}{\not\longrightarrow}}{x \parallel_r y \overset{a[u]}{\Longrightarrow} x' \parallel_r y} \qquad \frac{x \overset{a}{\not\longrightarrow} \quad y \overset{a[v]}{\Longrightarrow} y'}{x \parallel_r y \overset{a[v]}{\Longrightarrow} x \parallel_r y'}$$

$$\frac{x \overset{a[u]}{\Longrightarrow} x' \quad y \overset{a}{\longrightarrow}}{x \parallel_r y \overset{a[r \cdot u]}{\Longrightarrow} x' \parallel_r y} \qquad \frac{x \overset{a}{\longrightarrow} \quad y \overset{a[v]}{\Longrightarrow} y'}{x \parallel_r y \overset{a[\bar{r} \cdot v]}{\Longrightarrow} x \parallel_r y'}$$

The premises of the type $x \overset{a}{\longrightarrow}$ and $x \overset{a}{\not\longrightarrow}$ as they appear in the rules express that the state x does or does not have a-successors.

In order to use this specification, we need to check that the rules define the parallel composition uniquely. Since we want to distinguish system states up to behavioural equivalence only, which is probabilistic bisimilarity in this setting, we moreover need to prove that the resulting operator respects this equivalence: the behaviour of the parallel composition of two processes should not change if we replace any of the two components by a probabilistically bisimilar one.

For nondeterministic systems, several syntactic transition rule formats have been proposed, which guarantee that all specifications using only rules of that type define well-behaved operators. Examples are the De Simone format [Sim85], the GSOS format [BIM95], or the tyft/tyxt rules [GV89, GV92] (see also [AFV01] for an overview).

Several authors have defined composition operators for probabilistic systems and proved that they are well-behaved (see e.g. [LS92, GSS95, And99]). Although specification formats of the type above would simplify this task considerably, none have been proposed for probabilistic systems yet. One reason is surely that,

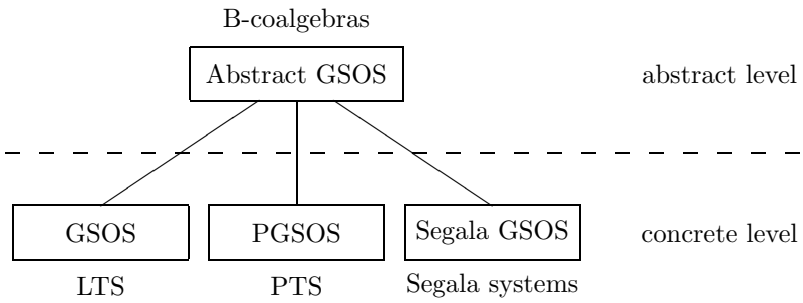


Figure 1.1: Abstract and concrete formats.

in this context, it is more complicated to work with probabilistic systems than to work with nondeterministic ones – mainly because the probabilities disallow that we treat the outgoing transitions of a state in isolation: when we construct a probabilistic system, we have to make sure that the probabilities of certain transitions together yield probability distributions, i.e. they sum up to 1; and to establish probabilistic bisimilarity, we need to relate groups of transitions leaving the bisimilar states, instead of individual transitions.

We introduce two expressive specification formats for well-behaved probabilistic composition operators: the *PGSOS format* for PTS and the *Segala GSOS format* for Segala systems. Specifications in both formats uniquely define probabilistic operators, which are well behaved with respect to probabilistic bisimilarity. Moreover, all defined operators may be used in a bisimulation up-to-context principle [San98], and guarded recursive equations involving them have solutions, determined up to probabilistic bisimilarity.

We develop PGSOS and Segala GSOS with the help of a categorical framework for specification formats for well-behaved composition operators [TP97]. Instead of a concrete system type, this approach treats arbitrary coalgebras of a behaviour functor. Operator specifications are given as natural transformations in the shape of *distributive laws*, a notion we already encountered in the study of generalised coinduction. On this level of abstraction, the models of the specifications can be described and well-behavedness properties can be established elegantly. One for instance obtains rather easily that all the specified operators respect the coalgebraic definition of bisimilarity [AM89].

To obtain the probabilistic formats, we use a subformat of the categorical framework called *abstract GSOS* [TP97], which was designed to cover the GSOS format [BIM95] for LTSs. We picture the general idea in Figure 1.1. Since the format is defined for an arbitrary behaviour functor B, it can be instantiated for many different system types including LTSs, PTSs, and Segala systems. What remains to be shown for each instance is which concrete specifications are covered by the categorical format, i.e. we have to identify a rule format so that

all its specifications give rise to natural transformations of the corresponding type. The well-behavedness of such a concrete rule format then follows from the results about the abstract one.

To establish a correspondence of abstract and concrete formats, we apply a decompositional method to analyse the natural transformations carefully. Using a collection of structural representation lemmata, we first explain transformations of a complex type in terms of simpler ones, which can then be characterised directly. With this technique we are able to give a first detailed proof of the above mentioned correspondence of abstract GSOS and the known GSOS format for LTSs. In a similar way, but based on more difficult representation results, we derive the format for PTS that corresponds to abstract GSOS. We call this new format PGSOS. For the more complex Segala systems, we define the Segala GSOS format and argue that its specifications define natural transformations of the type of abstract GSOS. We thus do not establish a one-to-one correspondence here, but our argument still allows us to conclude that the new format inherits the well-behavedness properties of abstract GSOS.

Although we use a categorical approach to derive the specifications formats and their well-behavedness properties, we express the obtained results without using coalgebraic notation. We do so to make the PGSOS and Segala GSOS format more widely applicable in process theory, in which coalgebraic terminology is not yet common.

1.4 Related work

In this section we briefly describe the work of other authors that has been most influential for this thesis. We also mention previous publications of our results.

The presentation of the bialgebraic modelling of SOS specifications is to a large extent based on work by Turi and Plotkin [TP97, Tur96], who express SOS-style specifications of composition operators as distributive laws of the free monad generated by the signature of the operators (i.e. the term monad) over the cofree comonad generated by the behaviour functor. The models of the specification are the bialgebras of the distributive law. They show that such models are well-behaved, in the sense that unique initial and final models exist, and that bisimulation is a congruence on every model. For the latter statement they assume that the behaviour functor, which needs to have cofree coalgebras for the approach in general, moreover preserves weak pullbacks. As a subformat, Turi and Plotkin introduce the abstract GSOS format and sketch a proof of its correspondence with specifications in the GSOS format [BIM95] when instantiated with (finitely branching) labelled transition systems. Since we found that the details of this proof, which establishes the correspondence in one step, are not easy to provide, we do not follow their suggestion in our proof and apply a decompositional method instead.

Distributive laws between functors with less structure than that of a monad

and comonad are studied by Lenisa, Power, and Watanabe [LPW00], who observe that the specifications in Turi and Plotkin's abstract GSOS format are distributive laws of a free monad over a cofree copointed functor.

A survey of well-behaved SOS specification formats for labelled transition systems is given by Aceto, Fokkink, and Verhoef [AFV01] in the handbook of process algebra.

An important reference for our work on generalised coinduction schemata are the articles of Lenisa [Len99a, Len99b], who was the first to state common properties of functions defined by various extended coiteration schemata. From her work we took the idea to use distributive laws in that context. For concrete generalised coinduction principles we mention that the categorical duals of primitive recursion and course-of-value iteration are advocated by Uustalu and Vene [VU98, UV99] and that a framework for generalised proof principles for (strong) bisimilarity in the concrete case of labelled transition systems is developed by Sangiorgi [San98]. Several examples of guarded recursive equations for infinite data streams and deterministic automata are studied by Rutten [Rut01, Rut98].

For probabilistic systems we refer to the work of Larsen and Skou [LS91], who define probabilistic transition systems and probabilistic bisimilarity, and Segala [SL94, Seg95b], who studied the second type of probabilistic automata we consider here together with a corresponding notion of bisimilarity. De Vink and Rutten [VR99] and Moss [Mos99] present probabilistic systems coalgebraically and show that the involved distribution functor preserves weak pullbacks. Concrete composition operators for different probabilistic systems are specified and studied for instance by Andova [And99], Larsen and Skou [LS92], and Van Glabbeek, Smolka, and Steffen [GSS95]. Various topics in performance analysis were treated at an EEF summer school, and are now published as its proceedings [BHK01]. Moreover, this presentation benefited from joint work with Sokolova and De Vink [BSV03, SV04] on a survey of probabilistic systems using coalgebraic techniques, which is not reported in this thesis.

Parts of the work on generalised coinduction was published first as a technical report [Bar00] and then at CMCS 2001 [Bar01, Bar03]. The PGSOS format for probabilistic transition systems has been presented in a preliminary version at CMCS 2002 [Bar02b], and then with more details as a technical report [Bar02a]. The Segala GSOS format was not published before.

1.5 Organisation of the Thesis and summary of its contributions

In Chapter 2 we settle the categorical notation used and provide the necessary background on algebra and coalgebra. The presented material is standard.

In Chapter 3 we present the categorical explanation of well-behaved SOS specification formats by Turi and Plotkin [TP97], who model specifications as dis-

tributive laws of a free monad over a cofree comonad. Our contribution is the treatment of distributive laws between functors with less structure than monads and comonads. This paves the way for the interpretation of our results on generalised coinduction schemata as new well-behavedness properties of the abstract specification formats, namely the validity of a bisimulation up-to-context principle and the solvability of guarded recursive equations. Moreover, we rigidly analyse the natural transformations arising from the categorical formats for labelled transition systems, infinite data streams, and deterministic automata, resulting in one-to-one correspondences between abstract formats and sets of transition rules in certain shapes. This analysis proceeds by first decomposing the complex natural transformations under consideration into simpler ones until a direct representation result can be applied. The advantage of this method is that it can be used to characterise the categorical format for other system types as well, as we shall exploit when we study probabilistic systems in Chapter 5.

In Chapter 4 we treat generalised coinduction schemata. We introduce the generic λ-coinduction definition and proof principle, which yields concrete schemata for any additional functor and distributive law λ of that functor over the behaviour functor under consideration. We give different proofs for the validity of the principles depending on whether the additional functor is a plain functor or carries the structure of a pointed functor or monad. As a trivial instance of the framework one obtains the standard coinduction principles; the other examples yield different extensions thereof. As known principles, we recover those that arise as the categorical duals of primitive recursion and course-of-value iteration [UV99], and the definition of a language by a nondeterministic automaton. As new principles, we derive from the λ-coiteration schema that a bisimulation up-to-context proof principle is valid for contexts built from GSOS-definable operators and that guarded recursive equations involving these operators have solutions determined up to bisimilarity.

Capitalising on our modular approach to analyse natural transformations from Chapter 3, in Chapter 5 we derive well-behaved and expressive specification formats for probabilistic systems. More precisely, we introduce PGSOS for the probabilistic transition systems of Larsen and Skou [LS91] (also known as reactive systems [GSS95]) and Segala GSOS for the probabilistic automata of Segala [SL94]. Each specification in any of the formats has two uniquely determined canonical models, bisimilarity is a congruence on all its models, the bisimulation up-to-context proof principle is valid for contexts built with the operators it defines, and guarded recursive specifications involving those operators have solutions determined up to the corresponding notion of probabilistic bisimilarity. Yet the formats are rather expressive, as we demonstrate with a list of examples for each.

In Chapter 6 we conclude by mentioning open problems and directions for future work.

In Appendix A we state and prove a collection of technical lemmata to characterise natural transformations. They form the core of our decompositional

method to analyse the concrete instances of the categorical operator specification formats. These are structural lemmata to describe complex natural transformations in terms of simpler ones as well as direct representation results for transformations of simple types. The main results are representation statements for natural transformations between functors involving the (finite) powerset and a probability distribution functor, which arise from our study of labelled transition systems and probabilistic transition systems. Although these statements are presented in the appendix, they are an important part of the contribution of the thesis.

Chapter 2

Algebras and coalgebras

In this chapter we summarise some basic concepts from universal algebra and coalgebra. We use them in this thesis mainly to model the syntax of process description languages and system behaviour respectively. The material is standard. As an introduction to our use of algebras and coalgebras we recommend an article of Jacobs and Rutten [JR96]. Since the algebras that we shall need for our purposes are of a rather simple type, we focus more on the coalgebra part. We start by fixing some notation.

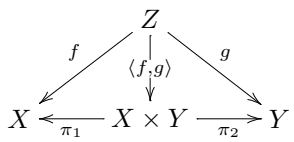
2.1 Categorical notation

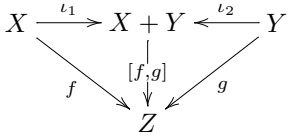
The treatment of categorical algebra and coalgebra requires some basic notation from category theory, which we introduce in this section. We do not assume much more than some familiarity with the notions of a category, a functor, a natural transformation, and a commutative diagram. Sometimes we also use the concept of a limit and colimit, mainly in the shape of products and coproducts. As a reference the reader may consult the textbook by Mac Lane [ML97], but shorter introductions are sufficient as well, such as the lecture notes by Turi [Tur01] or Van Oosten [Oos95].

We will mainly work in the category Set of sets and total functions as well as the category of endofunctors on Set. We draw double arrows for the morphisms in the latter category, i.e. for natural transformations. Moreover, the identity morphisms $\mathtt{id}_X : X \to X$ in any category are drawn as double arrows in diagrams.

For the binary product and coproduct of the objects X and Y we use the notation as pictured below, i.e. $\langle f, g \rangle$ denotes the *pairing* and $[f, g]$ denotes the

case analysis of f and g.





Note that the categorical products and coproducts in Set are simply cartesian products and disjoint unions respectively. We also use arbitrary set-indexed products and coproducts denoted by $\prod_{i\in I} X_i$ and $\coprod_{i\in I} X_i$ for families of objects $(X_i)_{i\in I}$. We write $\pi_j : \prod_{i\in I} X_i \to X_j$ and $\iota_j : X_j \to \coprod_{i\in I} X_i$ for the projections and injections. A case analysis for the arrows $(f_i : X_i \to Z)_{i\in I}$ is denoted by $[f_i] : \coprod_{i\in I} X_i \to Z$.

We use the following functors:

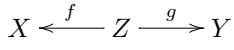
$$\mathrm{Id},\ \underline{A},\ \mathrm{F}\times\mathrm{G},\ \mathrm{F}+\mathrm{G},\ \mathrm{F}^A,\ \mathcal{P},\ \mathcal{P}_\omega,\ \mathcal{P}^+_\omega,\ \text{and}\ \mathcal{D}_\omega,$$

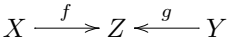
where Id is the *identity functor*; $\underline{A}$ for any object A is the *constant functor*; $\mathrm{F}\times\mathrm{G}$ and $\mathrm{F}+\mathrm{G}$ are the *product* and *coproduct* of the functors F and G; F^A is the *exponent* of the functor F by the object A; $\mathcal{P} : \mathsf{Set} \to \mathsf{Set}$ denotes the *powerset functor* defined for a set X and function $f : X \to Y$ as

$$\mathcal{P}X := \{X' \subseteq X\} \quad \text{and} \quad \mathcal{P}f := \big[X' \in \mathcal{P}X \mapsto f[X'] \in \mathcal{P}Y\big],$$

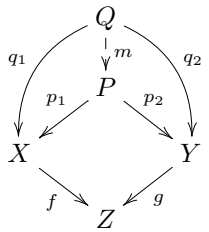
where $f[X'] = \{f(x) \mid x \in X'\} \subseteq Y$ for $X' \subseteq X$ denotes the function image; moreover, we use the symbols $\mathcal{P}_\omega$ and $\mathcal{P}^+_\omega$ for the *finite powerset functor* and the *nonempty finite powerset functor*, i.e. the restriction of the powerset functor to finite subsets and finite, nonempty subsets respectively. The *distribution functor* $\mathcal{D}_\omega$ will be given in Def. 5.1.1.

A *span* and a *cospan* between two objects X and Y are triples $\langle Z, f, g\rangle$ of an object Z and two arrows as pictured respectively below.





A *weak pullback* of a cospan $\langle Z, f, g\rangle$ between X and Y is a span $\langle P, p_1, p_2\rangle$ between X and Y satisfying $f \circ p_1 = g \circ p_2$ and such that for every span $\langle Q, q_1, q_2\rangle$ between X and Y with $f \circ q_1 = g \circ q_2$ there exists a mediating arrow $m : Q \to P$ satisfying $q_1 = p_1 \circ m$ and $q_2 = p_2 \circ m$.



Note that this definition is weaker than that of an ordinary pullback as the mediating arrow m need not be unique. A functor F is said to *preserve weak pullbacks* if it maps a weak pullback square on a weak pullback square, i.e. for the above weak pullback we have that $\langle \mathrm{F}P, \mathrm{F}p_1, \mathrm{F}p_2 \rangle$ is a weak pullback of $\langle \mathrm{F}Z, \mathrm{F}f, \mathrm{F}g \rangle$.

An object 0 of a category is called *initial* if for every object X there exists precisely one arrow $! : 0 \to X$. Dually, an object 1 of a category is called *final* if for every object X there exists precisely one arrow $! : X \to 1$. In Set the empty set is the only initial object, and every singleton set is final. When we talk about an arbitrary final set, we denote its only element by a star, i.e. $1 = \{*\}$.

2.2 Operator interpretations as algebras

In this section we recall the categorical means to model the syntax of a programming or process description language. Such a language is usually described by a grammar as in the simple example below, where $a \in L$ for some set of labels L.

$$\mathcal{E} ::= 0 \mid a \mid \mathcal{E}_1 \cdot \mathcal{E}_2 \mid \mathcal{E}_1 + \mathcal{E}_2 \tag{2.1}$$

As we shall explain, this specification gives rise to a *signature functor* S such that the interpretations of the grammar are *algebras* for S, and the sets of possibly open terms of this language over given sets of variables are captured by the *free monad* over S.

2.2.1 Algebras and congruences

A grammar as in (2.1) describes a *signature*, which is to say, a set of operator symbols Σ, where with each symbol $\sigma \in \Sigma$ a finite arity $|\sigma| \in \mathbb{N}$ is associated. In our example the signature is $\Sigma = \{0\} \cup L \cup \{\cdot, +\}$, where 0 and a for $a \in L$ are constants and the other two operators are binary, i.e. $|0| = |a| = 0$ and $|\cdot| = |+| = 2$. Here we will limit ourselves to one-sorted signatures as this one, but we remark that a generalisation of our studies to many-sorted signatures is possible.

An *interpretation* of this signature on a set P is a collection of operators

$$\big([\![\sigma(.)]\!] : P^{|\sigma|} \to P\big)_{\sigma \in \Sigma}.$$

These interpretations can be modelled elegantly as *algebras* of a functor.

Definition 2.2.1 *Let* S *be a* Set*-functor. An* **algebra of the functor** S*, or* **S-algebra** *for short, is a pair* $\langle P, \beta : \mathrm{S}P \to P \rangle$ *of a* carrier set P *and an* operation β*.*

A function $h : P \to Q$ is a **homomorphism** *between the two* S*-algebras $\langle P, \beta_P \rangle$ and $\langle Q, \beta_Q \rangle$ if it makes the following diagram commute.*

$$\begin{array}{ccc} \mathrm{S}P & \xrightarrow{\mathrm{S}h} & \mathrm{S}Q \\ {\scriptstyle \beta_P}\downarrow & & \downarrow{\scriptstyle \beta_Q} \\ P & \xrightarrow[h]{} & Q \end{array}$$

The S*-algebras together with their homomorphisms (with identities and composition as in* Set*) form a category, which we denote by* $\mathsf{Alg}^{\mathrm{S}}$*.*

Any signature Σ gives rise to a Set-functor S, defined for any set X and function $f : X \to Y$ as

$$\begin{array}{lcl} \mathrm{S}X & := & \displaystyle\coprod_{\sigma \in \Sigma} X^{|\sigma|} = \big\{\sigma(x_1, \ldots, x_{|\sigma|}) \mid \sigma \in \Sigma;\ x_1, \ldots, x_{|\sigma|} \in X\big\}, \\ \mathrm{S}f & := & \Big[\sigma(x_1, \ldots, x_n) \in \mathrm{S}X \mapsto \sigma(f(x_1), \ldots, f(x_n)) \in \mathrm{S}Y\Big]. \end{array} \tag{2.2}$$

For better readability, we have written $\sigma(x_1, \ldots, x_n)$ for $\iota_\sigma(x_1, \ldots, x_n) \in \mathrm{S}X$.

The algebras of this functor correspond to the interpretations of the operators in the signature. The algebra operation, as it were, glues together the functions $[\![\sigma(.)]\!]$ from above. Therefore we usually write such an algebra as $\langle P, [\![.]\!] : \mathrm{S}P \to P \rangle$. For a functor S arising from a signature as above, a function $h : P \to Q$ is a homomorphism from one S-algebra $\langle P, [\![.]\!]_P \rangle$ to another S-algebra $\langle Q, [\![.]\!]_Q \rangle$ if for all $\sigma(p_1, \ldots, p_n) \in \mathrm{S}P$ we have

$$h([\![\sigma(p_1, \ldots, p_n)]\!]_P) = [\![\sigma(h(p_1), \ldots, h(p_n))]\!]_Q.$$

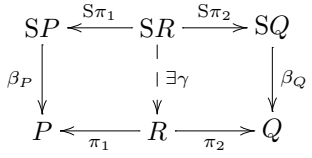
Given an interpretation $\langle P, [\![.]\!] \rangle$ of the signature Σ, one is sometimes interested in equivalence relations $R \subseteq P \times P$ that respect the operator interpretations. By this we mean that for any operator symbol $\sigma \in \Sigma$ with arity n we find

$$\langle [\![\sigma(p_1, \ldots, p_n)]\!], [\![\sigma(q_1, \ldots, q_n)]\!] \rangle \in R$$

if $\langle p_i, q_i \rangle \in R$ for $1 \leq i \leq n$. Such a relation R is called a *congruence*. We will use the following categorical generalisation of this notion. It defines a congruence as a relation between two possibly different algebras for an arbitrary functor.

Definition 2.2.2 *Let* S *be a* Set*-functor. A relation $R \subseteq P \times Q$ is a* **congruence** *between two* S*-algebras $\langle P, \beta_P \rangle$ and $\langle Q, \beta_Q \rangle$ if there is an* S*-algebra structure γ on R such that the two projections $\pi_1 : R \to P$ and $\pi_2 : R \to Q$ become homomorphisms between the respective algebras, i.e. γ makes the two squares in the following diagram commute (note that this condition determines*

γ *uniquely).*



2.2.2 Initial algebras and free monads

The syntax of a programming language is given by the set of *terms* that we can build with the operator symbols in the corresponding signature Σ. Formally this is the smallest set T that is closed under the term construction, i.e. for all operator symbols $\sigma \in \Sigma$ with arity n and elements $t_1, \ldots, t_n \in T$ we have $\sigma(t_1, \ldots, t_n) \in T$. We use the following notion to describe this set categorically.

Definition 2.2.3 *Let* S *be an endofunctor. An* **initial** S**-algebra** *is an initial object in* $\mathsf{Alg}^{\mathrm{S}}$*, i.e. an* S*-algebra* $\langle \mathcal{I}, \chi \rangle$ *such that for any* S*-algebra* $\langle P, \beta \rangle$ *there exists precisely one homomorphism from* $\langle \mathcal{I}, \chi \rangle$ *to* $\langle P, \beta \rangle$*.*

It can be shown that the set of closed terms of a signature Σ yields an initial S-algebra, where S is the functor associated to Σ. In the following we will sometimes refer to it as "the" initial algebra instead of "an" initial algebra, since initial objects are always isomorphic.

When we are interested in *open terms* over a set of variables X, we arrive at the notion of a *free algebra*.

Definition 2.2.4 *Let* S *be an endofunctor. A* **free** S**-algebra** *over the object* X *is an initial algebra*

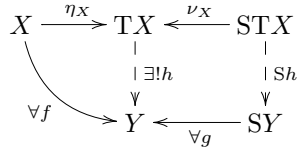
$$\langle \mathrm{T}X, [\eta_X, \nu_X] : X + \mathrm{ST}X \to \mathrm{T}X \rangle$$

of the functor $\mathrm{S}_{\mathrm{X}}Y := X + \mathrm{S}Y$*.*

In case the functor S arises from a signature Σ as in equation (2.2), which is the case that we will consider here most of the time, the elements of $\mathrm{T}X$ are the open terms of that signature over the set of variables X. An initial S-algebra is the free S-algebra over the initial set $\emptyset$.

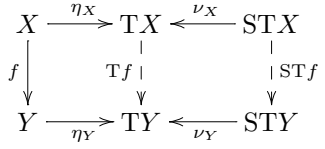
We can slightly rewrite the initiality condition of a free S-algebra $\langle \mathrm{T}X, [\eta_X, \nu_X] \rangle$ over X into the following *free induction definition principle*: for any set Y, every two arrows $f : X \to Y$ and $g : \mathrm{S}Y \to Y$ uniquely define an arrow $h : \mathrm{T}X \to Y$

making the following diagram commute:

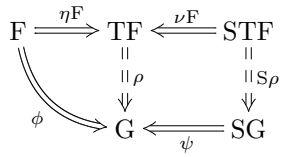


(2.3)

Definition 2.2.5 *Let* S *be a functor such that for all objects* X *there exists a free* S*-algebra* $\langle \mathrm{T}X, [\eta_X, \nu_X] \rangle$ *over* X*. The mapping* $X \mapsto \mathrm{T}X$ *(where we assume that for every* X *one particular free algebra has been chosen) extends to the* **free algebra functor** *by defining* $\mathrm{T}f : \mathrm{T}X \to \mathrm{T}Y$ *for a function* $f : X \to Y$ *to be the unique arrow making the diagram below commute (cf. the free induction principle of (2.3)).*



Using the uniqueness aspect of the free induction principle it is easy to verify that the above definition of T is functorial indeed. Note also that, with the definition of T, $\eta : \mathrm{Id} \Rightarrow \mathrm{T}$ and $\nu : \mathrm{ST} \Rightarrow \mathrm{T}$ are natural transformations. Moreover, the free induction definition principle in (2.3) extends to a definition principle for natural transformations: for two functors F and G, any two natural transformations $\phi : \mathrm{F} \Rightarrow \mathrm{G}$ and $\psi : \mathrm{SG} \Rightarrow \mathrm{G}$ uniquely define a natural transformation $\rho : \mathrm{TF} \Rightarrow \mathrm{G}$, such that the diagram below commutes.



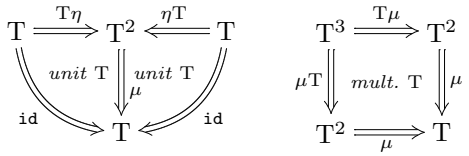
(2.4)

Each component of ρ is uniquely determined by free induction. Naturality can be established again with the uniqueness aspect of the principle.

In the following it will be convenient to treat the free algebra functor as a particular example of a *monad*.

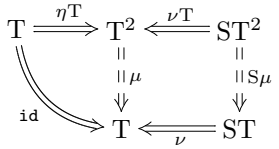
Definition 2.2.6 *A* **monad** *in a category* C *is a triple* $\langle \mathrm{T}, \eta, \mu \rangle$ *of a functor* $\mathrm{T} : \mathsf{C} \to \mathsf{C}$ *and two natural transformations* $\eta : \mathrm{Id} \Rightarrow \mathrm{T}$ *and* $\mu : \mathrm{T}^2 \Rightarrow \mathrm{T}$*, called the* unit *and* multiplication *respectively, such that the three parts of the*

two diagrams below commute.



The first two we call the unit laws *and the third the* multiplication law *of the monad.*

Definition 2.2.7 *Let* T*,* η*, and* ν *be the functor and the two natural transformations from Def. 2.2.5 of a free algebra functor over* S*. Let* $\mu : \mathrm{T}^2 \Rightarrow \mathrm{T}$ *be the unique natural transformation making the following diagram commute (cf. the definition principle in (2.4)).*

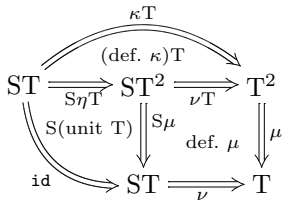


The triple $\langle \mathrm{T}, \eta, \mu \rangle$ *is a monad, which we call the* **free monad** *generated by* S*. In cases where we need the natural transformation* $\nu : \mathrm{ST} \Rightarrow \mathrm{T}$ *as well, we write the free monad as* $\langle \langle \mathrm{T}, \eta, \mu \rangle, \nu \rangle$ *or, equivalently (see Lemma 2.2.8 below),* $\langle \langle \mathrm{T}, \eta, \mu \rangle, \kappa \rangle$ *for the natural transformation* $\kappa := \nu \circ \mathrm{S}\eta : \mathrm{S} \Rightarrow \mathrm{T}$*.*

To show that the above definition yields a monad indeed, we have to check that the two unit laws and the multiplication law from Def. 2.2.6 hold. One of the unit laws is immediate from the definition of μ. The other two identities require easy diagram chases exploiting the uniqueness aspect of the definition principle in (2.4).

Lemma 2.2.8 *For the natural transformation* $\kappa : \mathrm{S} \Rightarrow \mathrm{T}$ *from Def. 2.2.7 we have* $\nu = \mu \circ \kappa\mathrm{T}$*.*

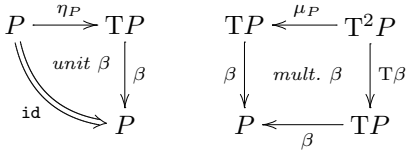
Proof: The claim is proved with the following diagram:



□

One of the special properties of the free monad $\langle \mathrm{T}, \eta, \mu \rangle$ generated by S is that it has the same algebras as S.

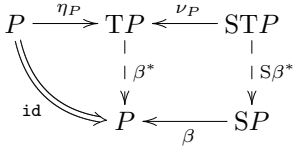
Definition 2.2.9 *An* **algebra of the monad** $\langle \mathrm{T}, \eta, \mu \rangle$ *is a* T*-algebra* $\langle P, \beta \rangle$ *such that the two diagrams below commute, which we call the* unit *and* multiplication law *of the algebra respectively.*



By $\mathsf{Alg}^{\langle \mathrm{T}, \eta, \mu \rangle}$ *we denote the category of all algebras for the monad* $\langle \mathrm{T}, \eta, \mu \rangle$.

Lemma 2.2.10 *Let* $\langle \langle \mathrm{T}, \eta, \mu \rangle, \kappa : \mathrm{S} \Rightarrow \mathrm{T} \rangle$ *be the free monad generated by a functor* S *from Def. 2.2.7. The categories* $\mathsf{Alg}^{\mathrm{S}}$ *and* $\mathsf{Alg}^{\langle \mathrm{T}, \eta, \mu \rangle}$ *are isomorphic.*

Proof: [sketch] One direction is given by $\langle P, \tilde{\beta} \rangle \mapsto \langle P, \tilde{\beta} \circ \kappa_P \rangle$ for an algebra $\langle P, \tilde{\beta} \rangle$ of the monad, and the other direction by $\langle P, \beta \rangle \mapsto \langle P, \beta^* \rangle$ for an S-algebra $\langle P, \beta \rangle$, where β^* is the unique arrow fitting into the diagram below (cf. the free induction principle in (2.3)).



For the two constructions one needs to check the following: β^* should satisfy the multiplication law (the unit law is satisfied by definition), the two constructions need to be inverses of each other, and they should preserve homomorphisms. The proofs of all items are straightforward, some use the uniqueness aspect of the free induction principle.

□

In the case of an algebra $[\![.]\!] : \mathrm{S}P \to P$ of a functor S arising from a signature Σ as in (2.2), the above inductive extension $[\![.]\!]^* : \mathrm{T}P \to P$ is given for $p \in P$, $\sigma \in \Sigma$ with arity n, and $t_1, \ldots, t_n \in \mathrm{T}P$ as

$$[\![p]\!]^* := p \quad \text{and} \quad [\![\sigma(t_1, \ldots, t_n)]\!]^* := [\![\sigma([\![t_1]\!]^*, \ldots, [\![t_n]\!]^*)]\!].$$

2.3 Transition systems as coalgebras

In this section we recall some notions and results from the theory of coalgebras. We use them to model dynamic systems and (possibly) infinite data structures. As examples we consider labelled transition systems, infinite data streams, deterministic automata, and, in Chapter 5, different kinds of probabilistic systems.

Rutten [Rut96, Rut00b] gave the first systematic treatment of universal coalgebra. For several statements we quote here we point to his paper as a convenient reference. The interested reader can find more details including pointers to the origin of the results there. As further introductions into the theory of coalgebras we recommend articles of Jacobs [Jac02] and Gumm [Gum99, Gum03b].

We illustrate the definitions again with the example of *labelled transition systems (LTSs)* from the introduction (cf. Section 1.1), which, for a fixed set of labels L, were pairs

$$\langle P, (\xrightarrow{a} \subseteq P \times P)_{a \in L} \rangle. \tag{2.5}$$

We view an LTS as a dynamic system. At any moment it is in some state $p \in P$. From p it can make transitions, to which a label from L is associated. The actual transition is chosen nondeterministically. Note that the above presentation leaves open whether the labels are a part of the choice of the system (internal nondeterminism) or whether they are prescribed by the environment (external nondeterminism). One often restricts the nondeterminism to be finite: for *finitely branching* systems one assumes that in any state $p \in P$ the system can choose from only finitely many transitions; for *image finite* systems one assumes that any state p has only finitely many outgoing transitions for each label $a \in L$.

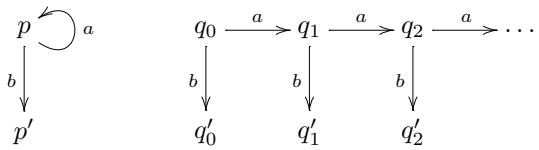
The states of an LTS are often viewed to be internal. All one is interested in is the behaviour that the system in a given state shows to an external observer. This behaviour is defined by specifying which system states look the same to this observer, i.e. by declaring a notion of behavioural equivalence of states. Various equivalences are studied in the literature, and they are all somehow based on the sets of enabled and disabled labels of the state itself and of other states reachable by sequences of labelled transitions. The most important notion is that of *(strong) bisimilarity* [Mil80, Mil89, Par81]. Here two states are indistinguishable if they are related by some *bisimulation*. A bisimulation on an LTS $\langle P, (\xrightarrow{a}) \rangle$ in turn is a relation $R \subseteq P \times P$ such that for all related states $\langle p, q \rangle \in R$ and labels $a \in L$ we have that

$$\begin{array}{l} p \xrightarrow{a} p' \text{ implies } q \xrightarrow{a} q' \text{ for some } q' \in P \text{ with } \langle p', q' \rangle \in R, \text{ and} \\ q \xrightarrow{a} q' \text{ implies } p \xrightarrow{a} p' \text{ for some } p' \in P \text{ with } \langle p', q' \rangle \in R. \end{array} \tag{2.6}$$

As an example, the states p and q_0 in the labelled transition system pictured below with set of labels $L = \{a, b\}$ are bisimilar, since the relation

$$R := \{\langle p, q_i \rangle, \langle p', q_i' \rangle \mid i \in \mathbb{N}\}$$

is a bisimulation, as one easily verifies.



2.3.1 Coalgebras and bisimilarity

We now show how to model labelled transition systems as *coalgebras of a functor*, a notion dual to that of an algebra from Definition 2.2.1. The advantages of this abstract approach to dynamic systems is that once we have found that the systems we are interested in are coalgebras of some functor, several meaningful notions and results immediately become available. For instance, there are results for an abstract notion of a bisimulation which, for the functor describing labelled transition systems, coincides with the notion of a strong bisimulation above, as we shall demonstrate.

Definition 2.3.1 *Let* B *be a* Set*-functor. A* **coalgebra for the functor** B*, or* B**-coalgebra** *for short, is a pair* $\langle P, \alpha : P \to \mathrm{B}P \rangle$ *of a* carrier set P *and a* structure map α*.*

A function $h : P \to Q$ *is a* **homomorphism** *between the two* B*-coalgebras* $\langle P, \alpha_P \rangle$ *and* $\langle Q, \alpha_Q \rangle$ *if it makes the following diagram commute.*

$$\begin{array}{ccc} P & \xrightarrow{h} & Q \\ {\scriptstyle \alpha_P}\downarrow & & \downarrow{\scriptstyle \alpha_Q} \\ \mathrm{B}P & \xrightarrow[\mathrm{B}h]{} & \mathrm{B}Q \end{array}$$

The B*-coalgebras together with their homomorphisms (with identities and composition as in* Set*) form a category, which we denote by* $\mathsf{Coalg}_{\mathrm{B}}$*. The* forgetful functor $\mathrm{U} : \mathsf{Coalg}_{\mathrm{B}} \to \mathsf{Set}$ *maps each coalgebra to its carrier and each homomorphism to itself.*

We often call the functor B used to define a class of coalgebras a *behaviour functor*. We do so only to stress the role of B, not to restrict the type of functors under consideration.

In order to show that labelled transition systems are coalgebras, we give an alternative but equivalent definition below.

Definition 2.3.2 *A* **labelled transition system (LTS)** *for the (nonempty) input alphabet* L *is a pair*

$$\langle P, \alpha : P \to (\mathcal{P}P)^L \rangle$$

of a set of states P *and a transition structure* α*, which is to say a coalgebra of the functor* $\mathrm{B} = \mathcal{P}^L$*. If no confusion about the LTS* $\langle P, \alpha \rangle$ *under consideration is likely to arise, we use the following arrow notation for* $p \in P$ *and* $a \in L$*:*

$$\begin{array}{rll} p \overset{a}{\not\longrightarrow} & \text{for} & \alpha(p)(a) = \emptyset, \\ p \overset{a}{\longrightarrow} & \text{for} & \alpha(p)(a) \neq \emptyset, \\ p \overset{a}{\longrightarrow} p' & \text{for} & p' \in \alpha(p)(a). \end{array}$$

We call $a \in L$ enabled *in state* $p \in P$ *if* $p \xrightarrow{a}$ *, otherwise it is* disabled.

From the coalgebraic definition we also obtain a notion of a map between two LTSs $\langle P, \alpha_P \rangle$ and $\langle Q, \alpha_Q \rangle$. Spelling out the definition, we get that a homomorphism from $\langle P, \alpha_P \rangle$ to $\langle Q, \alpha_Q \rangle$ is a function $h : P \to Q$ such that $p \xrightarrow{a} p'$ in $\langle P, \alpha_P \rangle$ implies $h(p) \xrightarrow{a} h(p')$ in $\langle Q, \alpha_Q \rangle$ and, moreover, all transitions of $h(p)$ arise in this way. As a consequence, p and $h(p)$ have the same set of enabled actions.

We are now going to argue that the notion of a bisimulation on an LTS generalises to arbitrary coalgebras. To this end, we first rephrase the condition in (2.6) in terms of sets of successor states. For a relation $R \subseteq P \times P$ let $\equiv_R \subseteq \mathcal{P}P \times \mathcal{P}P$ denote the relation defined to have $U \equiv_R V$ if and only if

$$\begin{array}{l} p \in U \text{ implies } \langle p, q \rangle \in R \text{ for some } q \in V, \text{ and} \\ q \in V \text{ implies } \langle p, q \rangle \in R \text{ for some } p \in U. \end{array}$$

With this definition, a relation $R \subseteq P \times P$ is a bisimulation on the LTS $\langle P, \alpha \rangle$ if for all $\langle p, q \rangle \in R$ and $a \in L$ we have that $\alpha(p)(a) \equiv_R \alpha(q)(a)$. The crucial observation now is that the relation $\equiv_R$ arises as the image of the span

$$\mathcal{P}P \xleftarrow{\mathcal{P}\pi_1} \mathcal{P}R \xrightarrow{\mathcal{P}\pi_2} \mathcal{P}P$$

which is to say that $U \equiv_R V$ if and only if there exists $W \in \mathcal{P}R$ such that $U = (\mathcal{P}\pi_1)(W)$ and $V = (\mathcal{P}\pi_2)(W)$. From this it immediately follows that the relation R is a bisimulation if and only if there exists a coalgebra structure $\gamma : R \to (\mathcal{P}R)^L$ making both parts of the diagram below commute.

$$\begin{array}{ccccc} P & \xleftarrow{\pi_1} & R & \xrightarrow{\pi_2} & P \\ {\scriptstyle\alpha}\downarrow & & {\scriptstyle\gamma}\downarrow & & \downarrow{\scriptstyle\alpha} \\ (\mathcal{P}P)^L & \xleftarrow[(\mathcal{P}\pi_1)^L]{} & (\mathcal{P}R)^L & \xrightarrow[(\mathcal{P}\pi_2)^L]{} & (\mathcal{P}P)^L \end{array}$$

The latter formulation generalises to coalgebras of an arbitrary functor B and to relations between two possibly different coalgebras. The definition is due to Aczel and Mendler [AM89].

Definition 2.3.3 *Let* B *be a* Set*-functor. A relation* $R \subseteq P \times Q$ *is a* **bisimulation** *between two* B*-coalgebras* $\langle P, \alpha_P \rangle$ *and* $\langle Q, \alpha_Q \rangle$ *if there is a* B*-coalgebra structure* γ *on* R *such that the two projections* $\pi_1 : R \to P$ *and* $\pi_2 : R \to Q$ *become homomorphisms between the respective coalgebras, i.e.* γ *makes the two squares in the following diagram commute.*

$$\begin{array}{ccccc} P & \xleftarrow{\pi_1} & R & \xrightarrow{\pi_2} & Q \\ {\scriptstyle\alpha_P}\downarrow & & {\scriptstyle\exists\gamma}\downarrow & & \downarrow{\scriptstyle\alpha_Q} \\ \mathrm{B}P & \xleftarrow[\mathrm{B}\pi_1]{} & \mathrm{B}R & \xrightarrow[\mathrm{B}\pi_2]{} & \mathrm{B}Q \end{array}$$

Two states $p \in P$ and $q \in Q$ of the above coalgebras are said to be **bisimilar**, *written as $p \sim q$, if they are related by some bisimulation.*

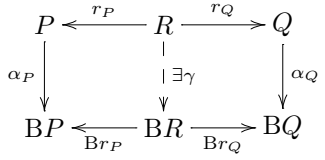
This notion is similar, but not dual to that of a congruence (cf. Def. 2.2.2). Note for instance that the mediating coalgebra structure γ is not necessarily uniquely determined here.

In the categorical setting it is often convenient not to insist on working with relations and projections but with arbitrary sets R and functions from R to P and Q. This is justified by the following lemma.

Lemma 2.3.4 (cf. [Rut00b, Lemma 5.3]) *Let $\langle P, \alpha_P \rangle$ and $\langle Q, \alpha_Q \rangle$ be two coalgebras of a functor* B*, and let $\langle R, r_P, r_Q \rangle$ be a span between P and Q. The image*

$$\langle r_P, r_Q \rangle[R] = \{\langle r_P(z), r_Q(z) \rangle \mid z \in R\} \subseteq P \times Q.$$

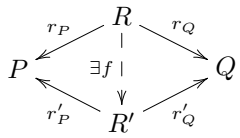
of this span is a bisimulation if and only if there is a coalgebra structure $\gamma : R \to \mathrm{B}R$ making both parts of the diagram below commute.



A corresponding statement holds for congruences (cf. Def. 2.2.2), so we also work with arbitrary spans in that context.

We usually leave the conversion of a span into the represented relation implicit. In order to do so, the following simple fact is convenient occasionally.

Lemma 2.3.5 *Let $\langle R, r_P, r_Q \rangle$ and $\langle R', r'_P, r'_Q \rangle$ be two spans between the sets P and Q. We have $\langle r_P, r_Q \rangle[R] \subseteq \langle r'_P, r'_Q \rangle[R']$ if and only if there exists a function $f : R \to R'$ making the two triangles in the diagram below commute.*



It is easy to see that the union of bisimulations is a bisimulation again. So the bisimilarity relation $\sim$ defined above is a bisimulation as well, which makes it the greatest one:

Lemma 2.3.6 *For any* Set *functor* B *there exists a greatest bisimulation $\sim$ between any two* B*-coalgebras.*

2.3.2 Final coalgebras

Among the coalgebras of a functor B we are particularly interested in finding one coalgebra $\langle \mathcal{F}, \omega \rangle$ which is *fully abstract* with respect to bisimilarity. With this we mean that for any given state p in any B-coalgebra $\langle P, \alpha \rangle$ the sought coalgebra $\langle \mathcal{F}, \omega \rangle$ should have precisely one state which is bisimilar to p. This property is equivalent to finality:

Definition 2.3.7 *A* **final** B**-coalgebra** *is a final object in* $\mathsf{Coalg}_{\mathrm{B}}$*, i.e. a* B*-coalgebra* $\langle \mathcal{F}, \omega \rangle$ *such that for every* B*-coalgebra* $\langle P, \alpha \rangle$ *there exists precisely one homomorphism from* $\langle P, \alpha \rangle$ *to* $\langle \mathcal{F}, \omega \rangle$*.*

Since any two final objects are isomorphic, we often refer to a final coalgebra as *the* final coalgebra.

Theorem 2.3.8 *(cf. [Rut00b, Theorems 9.2 and 9.3]) Let* B *be a* Set*-functor with a final coalgebra* $\langle \mathcal{F}, \omega \rangle$*. For two* B*-coalgebras* $\langle P, \alpha_P \rangle$ *and* $\langle Q, \alpha_Q \rangle$ *and two states* $p \in P$ *and* $q \in Q$ *we have that* $p \sim q$ *implies* $h_P(p) = h_Q(q)$*, where* $h_P : \langle P, \alpha_P \rangle \to \langle \mathcal{F}, \omega \rangle$ *and* $h_Q : \langle Q, \alpha_Q \rangle \to \langle \mathcal{F}, \omega \rangle$ *are the unique homomorphisms given by finality. If* B *preserves weak pullbacks (cf. Section 2.1) then the converse holds as well.*

In particular, we obtain the following *coinduction proof principle* to prove states of the final coalgebra equal:

Corollary 2.3.9 *Let* $\langle \mathcal{F}, \omega \rangle$ *be the final coalgebra of the* Set*-functor* B*, and let* $p, q \in \mathcal{F}$*. We have* $p \sim q$ *if and only if* $p = q$*.*

So in order to show that two states p and q of a final coalgebra are equal it suffices to exhibit a bisimulation $R \subseteq \mathcal{F} \times \mathcal{F}$ with $\langle p, q \rangle \in R$. We shall give an example for this principle in Section 2.3.3.

Not every functor has a final coalgebra. To find counterexamples we can use a property every final coalgebra has according to *Lambek's Lemma*:

Lemma 2.3.10 *(cf. [Rut00b, Theorem 9.1]) Let* B *be a functor with a final coalgebra* $\langle \mathcal{F}, \omega \rangle$*. The structure map* ω *is an isomorphism.*

With this property we can for instance show that the (unrestricted) powerset functor $\mathcal{P}$ does not have a final coalgebra: if $\langle \mathcal{F}, \omega \rangle$ was a final $\mathcal{P}$-coalgebra, we had $\mathcal{F} \simeq \mathcal{P}\mathcal{F}$; this is impossible, since with Cantor's Theorem no set is isomorphic to its own powerset. From this consideration we immediately get that no LTS as in Def. 2.3.2 can be final.

Still, it has been shown for a large class of functors that they have final coalgebras. We quote a rather strong result that uses the notion of a *bounded functor*

(cf. [Rut00b, Def. 6.7]): a Set-functor B is called bounded, if there exists a global bound to the size of the carrier set of any one-generated B-coalgebra. A one-generated coalgebra in turn is a coalgebra with a state from which all other states can be reached via the coalgebra structure.

Theorem 2.3.11 *(cf. [Rut00b, Theorem 10.4]) Every bounded functor has a final coalgebra.*

With this argument one can show that all functors built from the identity functor Id, the constant functor $\underline{A}$ for any set A, product and coproduct of such functors, the exponent $(-)^A$ of such a functor with a constant set A, and the finite powerset functor $\mathcal{P}_\omega$ have final coalgebras (cf. [Rut00b, Theorem 10.6]). For the interested reader we mention that a construction of the final coalgebra for the finite powerset functor $\mathcal{P}_\omega$ is explained in some detail by Worrell [Wor00, Section 3.6].

It follows from the above theorem that there exists a final *image finite* LTS. So from now on we implicitly assume image finiteness when we work with labelled transition systems, i.e. we adopt the following modification of Definition 2.3.2:

Definition 2.3.12 *An* **image finite labelled transition system** *for a (non-empty) input alphabet L is a pair*

$$\langle P, \alpha : P \to (\mathcal{P}_\omega P)^L \rangle,$$

i.e. a coalgebra of the functor $\mathrm{B} = (\mathcal{P}_\omega)^L$. So an LTS $\langle P, \alpha \rangle$ from Def. 2.3.2 is image finite if for all $p \in P$ and $a \in L$ we have that $\alpha(p)(a)$ is a finite set.

Once we know that the behaviour functor B under consideration has a final coalgebra, we often work on this coalgebra directly. Below we shall demonstrate this for the examples of infinite streams and languages. We can use finality directly as a definition principle to access the inhabitants of such a datatype:

Definition 2.3.13 *Let $\langle \mathcal{F}, \omega \rangle$ be a final coalgebra of the functor* B*. Every arrow $\alpha : X \to \mathrm{B}X$ uniquely determines an arrow $h : X \to \mathcal{F}$ making the diagram below commute.*

$$\begin{array}{ccc} X & \overset{\exists! h}{- - \to} & \mathcal{F} \\ {\scriptstyle \forall\alpha}\downarrow & \textit{coiteration} & \downarrow{\scriptstyle \omega} \\ \mathrm{B}X & \underset{\mathrm{B}h}{- - \to} & \mathrm{B}\mathcal{F} \end{array}$$

We say that h is defined by α through the **coiteration schema**.

We will give an example for this principle when we treat streams coalgebraically in the next section.

2.3.3 Examples of coalgebras

We have seen that unrestricted as well as image finite labelled transition systems are coalgebras for a suitable behaviour functor. In this section we give a few examples on how to model other kinds of systems and datatypes as coalgebras. We spell out the corresponding conditions for homomorphisms and bisimulations and present final systems for some of them.

LTSs with state predicates

One advantage of the coalgebraic approach is that we can often easily also treat minor variations of the behaviour type under consideration. To illustrate this point, we mention a variant of (image finite) labelled transition systems, which we modelled as coalgebras of the functor $(\mathcal{P}_\omega)^L$ in Def. 2.3.12. Suppose we want to extend this model so that to each state an attribute from a set of attributes A is associated. Such a system can be modelled as a pair $\langle P, \langle a, \alpha\rangle : P \to A \times (\mathcal{P}_\omega P)^L\rangle$, i.e. a coalgebra of the functor $A \times (\mathcal{P}_\omega)^L$. As one example, transition systems with a termination predicate $\surd$ can be modelled by taking $A := \mathcal{P}\{\surd\} \simeq 2$.

We can apply Theorem 2.3.11 to obtain that this functor possesses a final coalgebra as well. For the characterisation of a homomorphism

$$h : \langle P, \langle a_P, \alpha_P\rangle\rangle \to \langle Q, \langle a_Q, \alpha_Q\rangle\rangle$$

we get an extra clause demanding that h should preserve the attributes, i.e. that for all $p \in P$ we have $a_P(p) = a_Q(h(p))$. Similarly, in the definition of a bisimulation we further require that related states have the same attribute.

Streams

We now want to consider systems where to each state we assign a real number as an attribute and a successor state. These systems are coalgebras of the functor $\mathrm{B} := \underline{\mathbb{R}} \times \mathrm{Id}$. They are studied e.g. by Rutten [Rut01].

Let $\langle P, \langle h_P, t_P\rangle : P \to \mathbb{R} \times P\rangle$ and $\langle Q, \langle h_Q, t_Q\rangle : Q \to \mathbb{R} \times Q\rangle$ be two B-coalgebras. A function $f : P \to Q$ is a homomorphism between those coalgebras just in case for all $p \in P$ we have $h_P(p) = h_Q(f(p))$ and $f(t_P(p)) = t_Q(f(p))$, and a relation $R \subseteq P \times Q$ is a bisimulation between the two coalgebras if for all $\langle p, q\rangle \in R$ we have $h_P(p) = h_Q(q)$ and $\langle t_P(p), t_Q(q)\rangle \in R$.

Let

$$\mathbb{R}^\omega := \{\langle s_0, s_1, s_2, \ldots\rangle \mid s_i \in \mathbb{R}\ (i \in \mathbb{N})\}$$

be the set of infinite sequences of real numbers, which we will call *streams* over $\mathbb{R}$. The set $\mathbb{R}^\omega$ can be turned into a final B-coalgebra $\langle \mathbb{R}^\omega, \langle h, t\rangle : \mathbb{R}^\omega \to \mathbb{R} \times \mathbb{R}^\omega\rangle$

by defining h and t to return the *head* and *tail* of the given stream, i.e.

$$h(\langle s_0, s_1, s_2, \ldots\rangle) := s_0 \quad \text{and} \quad t(\langle s_0, s_1, s_2, \ldots\rangle) := \langle s_1, s_2, \ldots\rangle. \tag{2.7}$$

We will sometimes use bold letters like $\mathbf{s} \in \mathbb{R}^\omega$ for a stream and write $\mathbf{s} = s_0 : \mathbf{s}'$ to express that $\mathbf{s}$ has head s_0 and tail $\mathbf{s}'$.

To illustrate the coiteration schema from Def 2.3.13, we define the function $\mathtt{map}_g : \mathbb{R}^\omega \to \mathbb{R}^\omega$ that applies a given function $g : \mathbb{R} \to \mathbb{R}$ to all elements of a stream. For any $\mathbf{s} = (s_0 : \mathbf{s}') \in \mathbb{R}^\omega$ the function satisfies

$$\mathtt{map}_g(\mathbf{s}) \;=\; g(s_0) : \mathtt{map}_g(\mathbf{s}'). \tag{2.8}$$

To see that this equation uniquely defines $\mathtt{map}_g$, notice that it can equivalently be expressed by requiring that the function should make the following diagram commute, which is an instantiation of the coiteration schema.

$$\begin{array}{ccc} \mathbb{R}^\omega & \overset{\mathtt{map}_g}{\dashrightarrow} & \mathbb{R}^\omega \\ {\scriptstyle\langle g\circ h, t\rangle}\downarrow & & \downarrow{\scriptstyle\langle h, t\rangle} \\ \mathbb{R}\times\mathbb{R}^\omega & \underset{\mathtt{id}\times\mathtt{map}_g}{\dashrightarrow} & \mathbb{R}\times\mathbb{R}^\omega \end{array}$$

As an example of a coinduction proof we show

$$\mathtt{map}_{(g'\circ g)} = \mathtt{map}_{g'} \circ \mathtt{map}_g$$

for any two functions $g, g' : \mathbb{R} \to \mathbb{R}$. With the coinduction proof principle from Corollary 2.3.9 it suffices to establish

$$\mathtt{map}_{(g'\circ g)}(\mathbf{s}) \sim \mathtt{map}_{g'}(\mathtt{map}_g(\mathbf{s})) \quad \text{for all } \mathbf{s} \in \mathbb{R}^\omega.$$

To this end one easily derives from the definitions that the relation

$$R = \{\langle \mathtt{map}_{(g'\circ g)}(\mathbf{s}), \mathtt{map}_{g'}(\mathtt{map}_g(\mathbf{s}))\rangle \mid \mathbf{s} \in \mathbb{R}^\omega\}$$

is a bisimulation.

In some examples we will more generally work with streams A^ω over some set A, which can be turned into the final coalgebra of the functor $\mathrm{B} = \underline{A} \times \mathrm{Id}$.

Streams over real numbers modelling power series are also studied e.g. by McIlroy [McI99]. As a variation, the lazy lists over a set A from the introduction form a final coalgebra of the functor $\underline{1} + \underline{A} \times \mathrm{Id}$ (see e.g. the article of Hensel and Jacobs [HJ98]).

Deterministic automata

As the last example for now we consider deterministic automata for an alphabet L. Following Rutten [Rut98], we model such an automaton as a pair

$$\langle P, \langle o, t\rangle : P \to 2 \times P^L\rangle,$$

where $2 = \{\bot, \top\}$ is the set of truth values. In other words, a deterministic automaton is a coalgebra of the functor $\mathrm{B} := \underline{2} \times \mathrm{Id}^L$. The function $o : P \to 2$ models the acceptance predicate and $t : P \to P^L$ encodes the transition function. For $p \in P$ we write $p{\downarrow}$ if $o(p) = \top$ and $p{\uparrow}$ otherwise. For $a \in L$ we moreover define $p_a := t(p)(a)$ and write $p \xrightarrow{a} p_a$.

Let $\langle P, \langle o_P, t_P\rangle : P \to 2 \times P^L\rangle$ and $\langle Q, \langle h_Q, t_Q\rangle : Q \to 2 \times Q^L\rangle$ be two deterministic automata. A function $f : P \to Q$ is a homomorphism between those coalgebras just in case for all $p \in P$ we have $p{\downarrow} \iff (f(p)){\downarrow}$ and $(f(p))_a = f(p_a)$ for all $a \in L$. A relation $R \subseteq P \times Q$ is a bisimulation between the two automata if for all $\langle p, q\rangle \in R$ we have $p{\downarrow} \iff q{\downarrow}$ and for all $a \in L$ we find $\langle p_a, q_a\rangle \in R$.

Let L^* denote the set of finite words over L, including the empty word ε, and let

$$\mathcal{L} := \{M \subseteq L^*\} = \mathcal{P}L^*$$

be the set of *languages* over L. It can be turned into a deterministic automaton (i.e. a B-coalgebra $\langle \mathcal{L}, \langle o, t\rangle : \mathcal{L} \to 2 \times \mathcal{L}^L\rangle$) by defining for any language $M \in \mathcal{L}$

$$M{\downarrow} \iff \varepsilon \in M \quad \text{and} \quad M_a := \{w \in L^* \mid aw \in M\}. \tag{2.9}$$

Rutten [Rut98] shows that this coalgebra is final. Given any deterministic automaton $\langle P, \langle o_P, t_P\rangle\rangle$, the final homomorphism $h : P \to \mathcal{L}$ from $\langle P, \langle o_P, t_P\rangle\rangle$ to $\langle \mathcal{L}, \langle o, t\rangle\rangle$ maps every state $p \in P$ to the language $h(p)$ it accepts. In Section 4.4.3 we will use a generalised coinduction principle to define the functions that map the states of *nondeterministic* automata to the languages they accept.

2.3.4 Comonads

We shall need the concept dual to that of a monad from Def. 2.2.6 as well, though it will be less important for our work. Since all definitions and statements are dual to those in Section 2.2.2, the description is very brief here and is mainly meant to introduce the notation.

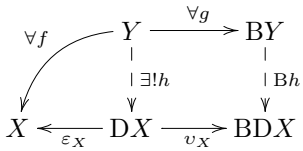
Definition 2.3.14 *Let* B *be an endofunctor. A* **cofree** B**-coalgebra** *over the object X is a final coalgebra*

$$\langle \mathrm{D}X, \langle \varepsilon_X, \upsilon_X\rangle : \mathrm{D}X \to X \times \mathrm{BD}X\rangle$$

of the functor $\mathrm{B}_X Y := X \times \mathrm{B}Y$.

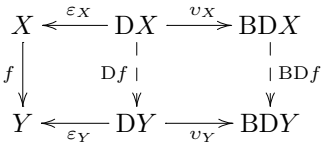
The definition of a cofree B-coalgebra $\langle \mathrm{D}X, \langle \varepsilon_X, \upsilon_X\rangle\rangle$ gives rise to the *cofree coinduction definition principle*: for any set Y, every two arrows $f : Y \to X$

and $g : Y \to \mathrm{B}Y$ uniquely define an arrow $h : Y \to \mathrm{D}X$ making the following diagram commute:

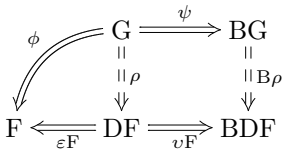


(2.10)

Definition 2.3.15 *Let* B *be a functor such that for all objects X there exists a cofree* B*-coalgebra $\langle \mathrm{D}X, \langle \varepsilon_X, \upsilon_X \rangle\rangle$ over X. The mapping $X \mapsto \mathrm{D}X$ (where we assume that for every X one particular cofree coalgebra has been chosen) extends to the* **cofree coalgebra functor** *by defining $\mathrm{D}f : \mathrm{D}X \to \mathrm{D}Y$ for a function $f : X \to Y$ to be the unique arrow making the diagram below commute (cf. the cofree coinduction principle of (2.10)).*

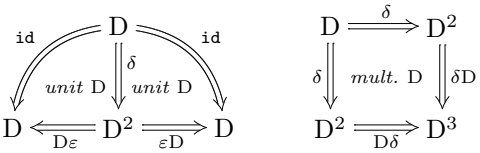


Here again $\varepsilon : \mathrm{D} \Rightarrow \mathrm{Id}$ and $\upsilon : \mathrm{D} \Rightarrow \mathrm{BD}$ become natural transformations and the cofree coinduction definition principle in (2.10) extends to a definition principle for natural transformations: for two functors F and G, any two natural transformations $\phi : \mathrm{G} \Rightarrow \mathrm{F}$ and $\psi : \mathrm{G} \Rightarrow \mathrm{BG}$ uniquely define a natural transformation $\rho : \mathrm{G} \Rightarrow \mathrm{DF}$, such that the diagram below commutes.



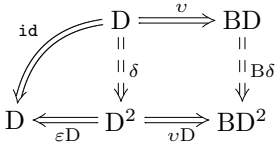
(2.11)

Definition 2.3.16 *A* **comonad** *in a category* C *is a triple $\langle \mathrm{D}, \varepsilon, \delta \rangle$ of a functor $\mathrm{D} : \mathsf{C} \to \mathsf{C}$ and two natural transformations $\varepsilon : \mathrm{D} \Rightarrow \mathrm{Id}$ and $\delta : \mathrm{D} \Rightarrow \mathrm{D}^2$, called the* counit *and* comultiplication *respectively, such that the three parts of the two diagrams below commute.*



The first two we call the counit laws *and the third the* comultiplication law *of the comonad.*

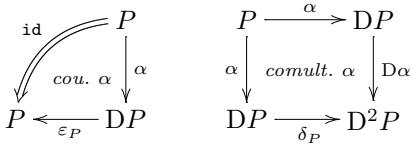
Definition 2.3.17 *Let* D, $\varepsilon : \mathrm{D} \Rightarrow \mathrm{Id}$, *and* $\upsilon : \mathrm{D} \Rightarrow \mathrm{BD}$ *be the functor and the two natural transformations from Def. 2.3.15 of a cofree coalgebra functor over* B. *Let* $\delta : \mathrm{D} \Rightarrow \mathrm{D}^2$ *be the unique natural transformation making the following diagram commute (cf. the definition principle in (2.11)).*



The triple $\langle \mathrm{D}, \varepsilon, \delta \rangle$ *is a comonad, which we call the* **cofree comonad** *generated by* B. *In cases where we need the natural transformation* $\upsilon : \mathrm{D} \Rightarrow \mathrm{BD}$ *as well, we write the cofree comonad as* $\langle \langle \mathrm{D}, \varepsilon, \delta \rangle, \upsilon \rangle$ *or, equivalently (see Lemma 2.3.18 below),* $\langle \langle \mathrm{D}, \varepsilon, \delta \rangle, \vartheta \rangle$ *for the natural transformation* $\vartheta := \mathrm{B}\varepsilon \circ \upsilon : \mathrm{D} \Rightarrow \mathrm{B}$.

Lemma 2.3.18 *For the natural transformation* $\vartheta : \mathrm{D} \Rightarrow \mathrm{B}$ *from Def. 2.3.17 we have* $\upsilon = \vartheta\mathrm{D} \circ \delta$.

Definition 2.3.19 *A* **coalgebra of the comonad** $\langle \mathrm{D}, \varepsilon, \delta \rangle$ *is a* D*-coalgebra* $\langle P, \alpha \rangle$ *such that the two diagrams below commute, which we call the* counit *and* comultiplication law *of the coalgebra respectively.*



By $\mathsf{Coalg}_{\langle \mathrm{D}, \varepsilon, \delta \rangle}$ *we denote the category of all coalgebras for the comonad* $\langle \mathrm{D}, \varepsilon, \delta \rangle$.

Lemma 2.3.20 *Let* $\langle \langle \mathrm{D}, \varepsilon, \delta \rangle, \vartheta : \mathrm{D} \Rightarrow \mathrm{B} \rangle$ *be the cofree comonad generated by a functor* B *from Def. 2.3.17. The categories* $\mathsf{Coalg}_{\mathrm{B}}$ *and* $\mathsf{Coalg}_{\langle \mathrm{D}, \varepsilon, \delta \rangle}$ *are isomorphic, i.e. there exists a one-to-one correspondence between* B*-coalgebras and coalgebras for the comonad* $\langle \mathrm{D}, \varepsilon, \delta \rangle$ *which respects homomorphisms.*

Chapter 3

A bialgebraic approach to structural operational semantics

Plotkin's *structural operational semantics (SOS)* [Plo81] is a popular tool for giving semantics to programming languages. An SOS specification consists of a collection of derivation rules. With those rules, the transitions of a labelled transition system can be specified following the syntactic structure of the terms of the programming language. It has been observed that important properties of the specification, such as invariance under bisimilarity, can be established simply by inspection of the syntactic format of its rules. Examples of proposed rule formats are the De Simone format [Sim85], the GSOS format [BIM95], or the tyft/tyxt rules [GV89, GV92]. A survey of the most important formats and the properties they guarantee is given by Aceto et al. [AFV01].

In this chapter, we study a bialgebraic approach to well-behaved SOS specification formats as introduced by Turi and Plotkin [TP97], which has received considerable attention since its introduction (see Section 3.7 on related work). Here specifications are modelled as certain natural transformations called *distributive laws* of a monad modelling the syntax of the programming language over a comonad modelling computations or, as we say here, system behaviour. The main advantages of this abstract categorical formulation are that it allows for more transparent proofs and that it can be instantiated with different kinds of systems behaviours, as we shall demonstrate. At the same time it captures rather expressive concrete specification formats like the GSOS rules mentioned above.

We give a step by step introduction to the subject, which does not assume a strong background in category theory and should make the theory more easily accessible. To this end, we focus on the category of sets and total functions,

Set, and motivate the necessary categorical notions by concrete examples.

We present a new proof showing that bisimilarity on every bialgebra for a distributive law is a congruence. With our argument we can dispense with assumptions on the coalgebra functor.

We extend Turi and Plotkin's work in that we also consider distributive laws between functors with less structure than that of a monad and comonad as specifications. Although such specifications are less expressive than the original ones, we have two reasons for studying them: first, as we shall prove in Chapter 4, they have additional properties; second, this way we generalise the approach to system types that cannot be described by comonads, like for instance labelled transition systems with unbounded nondeterminism.

The success of the methodology of Turi and Plotkin depends crucially on our ability to relate the specifications in the abstract formats, i.e. natural transformations of a certain type, to practically usable concrete formats. By the latter we mean for instance sets of transition rules of some shape. To establish an equivalence between a type of natural transformation and certain sets of transition rules, one needs to analyse the naturality condition carefully. This has not been done satisfactorily for any of the interesting cases yet. As our main contribution to Turi and Plotkin's work, we make such an analysis in the case of labelled transition systems and sets of rules in the GSOS format. Thereby we proceed in a decompositional manner, which allows us to adapt our argument to variations of the LTS behaviour easily. Moreover, it helps us to spell out the categorical format for streams, deterministic automata, and, later in Chapter 5, for probabilistic systems.

This chapter is organised as follows: first, we briefly recall Plotkin's SOS (Section 3.1); next, we introduce a simple but not very expressive categorical specification format and prove it well-behaved in several respects (Section 3.2); the abstract format is then related to concrete rule style format for LTS, stream systems, and deterministic automata (Section 3.3); next, we show how to improve the expressiveness of the format while keeping its well-behavedness (Section 3.4); as an example of such a more expressive format we study an abstract format related to GSOS (Section 3.5); then we formally compare the different formats treated (Section 3.6); we conclude by reviewing some related work (Section 3.7).

3.1 Plotkin's SOS and congruence formats

The operational semantics of programming languages is often given as a labelled transition system on the set of program terms and the transition structure is specified using Plotkin's structural operational semantics, or SOS for short [Plo81]. With Σ being the signature of the operators of the programming language, such a specification is given by a set of transition rules of the following

shape

$$\frac{\{u_i \xrightarrow{b_i} u_i' \mid i \in I\} \quad \{v_j \xrightarrow{c_j}\!\!\!\!\!\!\rightarrow \mid j \in J\}}{t \xrightarrow{a} t'}$$

where t, t', u_i, u_i', v_j for $i \in I$ and $j \in J$ are (possibly open) terms for the signature Σ, and $a, b_i, c_j \in L$ are labels. For $i \in I$ and $j \in J$ we call $u_i \xrightarrow{b_i} u_i'$ and $v_j \xrightarrow{c_j}\!\!\!\!\!\!\rightarrow$ the (*positive* and *negative*) *premises* of this rule, $t \xrightarrow{a} t'$ is its *conclusion*, the terms t and t' are its *source* and *target* respectively, and a is the *label* of the rule.

Traditionally, the semantics of an SOS specification is given by one particular LTS: its states are the closed terms of the signature under consideration and the transitions are precisely those that can be derived from the rules – at least in the simple case that this set of transitions is uniquely determined. Here we take a slightly different perspective: for an arbitrary labelled transition system $\langle P, (\xrightarrow{a})_{a \in L} \rangle$ (cf. equation (2.5) on page 25) we want to find an interpretation of the operators in Σ. So to each symbol $\sigma \in \Sigma$ we want to associate a function $[\![\sigma(.)]\!] : P^n \to P$, where n is the arity of σ. The set of rules is meant to determine the behaviour of the states $[\![\sigma(p_1, \ldots, p_n)]\!]$ for arbitrary given states $p_1, \ldots, p_n \in P$ of the LTS. Accordingly, we are interested only in rules with a source of the shape $\sigma(x_1, \ldots, x_n)$, where $\sigma \in \Sigma$ is some operator symbol with arity n and $x_1, \ldots, x_n$ are state variables.

An LTS with operator interpretations as above is a model of such a specification if, for all $\sigma \in \Sigma$ with arity n and $p_1, \ldots, p_n \in P$, the transition structure of the LTS assigns to the state $[\![\sigma(p_1, \ldots, p_n)]\!] \in P$ precisely those outgoing transitions that can be inferred from the rules.[1]

Usually one wants (the models of) the specification to satisfy certain basic properties. Most importantly, in order to be able to take bisimilarity as a process equivalence, one wants bisimilarity to be a congruence for the operator interpretations of a model. It has been found that properties of this type are guaranteed for specifications that involve only transition rules of a certain shape. Such a shape may or may not allow the negative premises, i.e. the premises of the type $v_j \xrightarrow{c_j}\!\!\!\!\!\!\rightarrow$. Moreover, it may restrict some of the terms u_i, u_i', v_j, t, and t' appearing in the different places of the rule to be a plain variable or the application of just one operator to variables. And it may prescribe that certain variables are allowed to appear only in some of the terms. An important example of such a rule shape is the GSOS format [BIM95] that we will recall in Section 3.5. For an overview of such formats see the article by Aceto et al. [AFV01]. In the following we shall present a categorical approach to derive these well-behaved specification formats.

[1] To be precise, generalising from the transition system on the set of terms to arbitrary ones actually requires us to apply the term evaluation $[\![.]\!]^* : \mathrm{T}P \to P$ (cf. Lemma 2.2.10), i.e. the inductive extension of the operator interpretation of the model, to all terms mentioned in the rules, but for simplicity of presentation we usually omit it in our notation.

3.2 A simple categorical specification format

In this section we will introduce a simple categorical operator specification format. Its expressiveness is rather limited but it will be the basis for more powerful formats to be introduced later.

We shall develop the format out of the following simple example specification of the synchronous parallel composition of two processes. For this example, we take the signature Σ to contain just one binary operator symbol, which we write in infix notation as $x|y$. Its outgoing transitions are specified by the following rules.

$$\frac{x \xrightarrow{a} x' \quad y \xrightarrow{a} y'}{x|y \xrightarrow{a} x'|y'} \qquad (\forall a \in L) \tag{3.1}$$

An LTS $\langle P, (\xrightarrow{a})_{a\in L}\rangle$ with an associated operation $[\![.|.]\!] : P \times P \to P$ is a model of the specification if for all $p, q \in P$ and $a \in L$ we have that $[\![p|q]\!] \xrightarrow{a} z$ precisely if $z = [\![p'|q']\!]$ for two states $p', q' \in P$ with $p \xrightarrow{a} p'$ and $q \xrightarrow{a} q'$.

In order to express the example in categorical terms, we first note that the structures under consideration, i.e. labelled transition systems with interpretations of operators, are actually *bialgebras* as defined below. Then we shall show that the above definition of a model can be expressed elegantly with this formulation.

Definition 3.2.1 *Given two* Set*-functors* S *and* B*, an* $\langle \mathrm{S}, \mathrm{B}\rangle$**-bialgebra** *is a triple* $\langle P, \beta, \alpha\rangle$ *consisting of a set* P *and two functions* $\beta : \mathrm{S}P \to P$ *and* $\alpha : P \to \mathrm{B}P$*, i.e. an* S*-algebra and a* B*-coalgebra structure on a common carrier set. A* **homomorphism** *from one* $\langle \mathrm{S}, \mathrm{B}\rangle$*-bialgebra* $\langle P, \beta_P, \alpha_P\rangle$ *to another* $\langle \mathrm{S}, \mathrm{B}\rangle$*-bialgebra* $\langle Q, \beta_Q, \alpha_Q\rangle$ *is a function* $h : P \to Q$ *satisfying*

$$h \circ \beta_P = \beta_Q \circ \mathrm{S}h \quad \textit{and} \quad \alpha_Q \circ h = \mathrm{B}h \circ \alpha_P,$$

i.e. it is both an S*-algebra and a* B*-coalgebra homomorphism for the involved algebra and coalgebra structures. By* $\mathsf{Bialg}^{\mathrm{S}}_{\mathrm{B}}$ *we denote the category of* $\langle \mathrm{S}, \mathrm{B}\rangle$*-bialgebras and their homomorphisms. A bisimulation (congruence) between two bialgebras is a bisimulation (congruence) between the contained coalgebras (algebras).*

Let us again regard the interpretation of the operators in the signature Σ as an algebra of a functor S as in (2.2) on page 20 and model (image finite) labelled transition systems coalgebraically as in Def. 2.3.12, i.e. as B-coalgebras for the functor $\mathrm{B} = {\mathcal{P}_\omega}^L$. Then an LTS with associated interpretations of the operators in Σ can be modelled as an $\langle \mathrm{S}, \mathrm{B}\rangle$-bialgebra $\langle P, [\![.]\!], \alpha\rangle$.

In the example of the synchronous parallel composition we have

$$\mathrm{S}P = \{\ p|q\ \mid p, q \in P\} \simeq P^2.$$

An $\langle \mathrm{S}, \mathrm{B}\rangle$-bialgebra $\langle P, [\![.]\!], \alpha\rangle$ is a model of the rules in (3.1) if for all $a \in L$ and $p, q \in P$ it satisfies

$$\alpha([\![p|q]\!])(a) = \{[\![p'|q']\!] \mid p' \in \alpha(p)(a),\ q' \in \alpha(q)(a)\}.$$

Alternatively, we can write this condition as

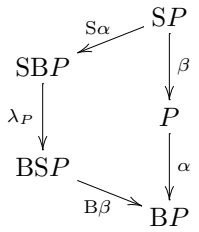
$$\alpha \circ [\![.]\!] = (\mathcal{P}_\omega[\![.]\!])^L \circ \lambda_P \circ \mathrm{S}\alpha, \tag{3.2}$$

where $\lambda_P : \mathrm{S}(\mathcal{P}_\omega P)^L \to (\mathcal{P}_\omega \mathrm{S}P)^L$ is given, for $\phi, \psi \in (\mathcal{P}_\omega P)^L$ and $a \in L$, by

$$\lambda_P(\phi|\psi)(a) = \{p'|q' \in \mathrm{S}P \mid p' \in \phi(a),\ q' \in \psi(a)\}.$$

Note that λ_P is independent of $[\![.]\!]$ and α and that it is natural in P. So we have translated our transition rules from (3.1) into a natural transformation $\lambda : \mathrm{S}(\mathcal{P}_\omega)^L \Rightarrow (\mathcal{P}_\omega \mathrm{S})^L$. This natural transformation $\lambda : \mathrm{SB} \Rightarrow \mathrm{BS}$ *distributes* the signature functor over the behaviour functor $\mathrm{B} = (\mathcal{P}_\omega)^L$, which motivates the following name.

Definition 3.2.2 *Let* S *and* B *be functors. A* **distributive law** *of* S *over* B *is a natural transformation* $\lambda : \mathrm{SB} \Rightarrow \mathrm{BS}$. *A* **bialgebra for a distributive law** λ*, or* **λ-bialgebra** *for short, is an* $\langle \mathrm{S}, \mathrm{B}\rangle$*-bialgebra* $\langle P, \beta, \alpha\rangle$ *for which the following diagram commutes.*



The full subcategory of $\mathsf{Bialg}^{\mathrm{S}}_{\mathrm{B}}$ *containing all* λ*-bialgebras is denoted by* λ-Bialg.

We have shown that the specification of the synchronous parallel composition gives rise to a distributive law λ of the signature functor over the behaviour functor for labelled transition systems in such a way that the models of the specification are precisely the λ-bialgebras (compare the above definition with equation (3.2)). In the next section we show that the category of λ-bialgebras is well-behaved in several respects. For instance, bisimilarity is a congruence on every λ-bialgebra. This leads us to consider distributive laws λ and λ-bialgebras as specifications and their models in the first place. But then the question arises which concrete specifications are subsumed by the abstract approach. For the case of labelled transition systems it turns out that we can characterise those specifications by restricting the format of the transition rules that may be used. So the categorical setting yields a congruence format similar to the ones known in the literature. But different from the conventional study of congruence

formats, the categorical approach is parametric in the behaviour functor under consideration. This enables us to instantiate the results also with other types of systems, as we show for streams and deterministic automata in Sections 3.3.2 and 3.3.3.

In Turi and Plotkin's work distributive laws are defined between two functors taken from a monad and comonad respectively. In that case the distributive law is assumed to respect the monad and comonad structure in a sense we shall define later. For the moment we just consider the simple case where no further structure is involved, but later we shall find that the more complex settings can be used to obtain more expressive formats.

3.2.1 Properties

In this section we prove basic properties of the category of λ-bialgebras, λ-Bialg, for a distributive law λ.

Theorem 3.2.3 *Let λ be a distributive law of a functor* S *over a functor* B.

(i) *Let $\langle \mathcal{I}, \chi \rangle$ be an initial* S*-algebra. There is a unique* B*-coalgebra structure α on $\mathcal{I}$ such that $\langle \mathcal{I}, \chi, \alpha \rangle$ is a λ-bialgebra. Moreover, this is an initial λ-bialgebra.*

(ii) *Dually, let $\langle \mathcal{F}, \omega \rangle$ be a final* B*-coalgebra. There is a unique* S*-algebra structure β on $\mathcal{F}$ such that $\langle \mathcal{F}, \beta, \omega \rangle$ is a λ-bialgebra. Moreover, this is a final λ-bialgebra.*

This statement is an immediate consequence of the following two lemmata.

Lemma 3.2.4 *Let* $\mathrm{F} : \mathsf{C} \to \mathsf{C}$ *be a functor.*

(i) *There is a unique* F*-coalgebra structure on an initial object* 0 *of* C *and it yields an initial* F*-coalgebra.*

(ii) *Dually, there is a unique* F*-algebra structure on a final object* 1 *of* C *and it yields a final* F*-algebra.*

Proof: We prove item (ii), the other one is dual. The only F-algebra structure on a final object 1 is the unique final morphism $!_{\mathrm{F}1} : \mathrm{F}1 \to 1$. It yields a final F-algebra, since for any F-algebra $\langle S, \beta \rangle$ there is precisely one morphism from S to 1, namely the unique final morphism $!_S : S \to 1$, and it is indeed a homomorphism, again by finality.

$$\begin{array}{ccc} \mathrm{F}S & \xrightarrow{\mathrm{F}!_S} & \mathrm{F}1 \\ {\scriptstyle\beta}\downarrow & \text{finality} & \downarrow{\scriptstyle !_{\mathrm{F}1}} \\ S & \xrightarrow[!_S]{} & 1 \end{array}$$

□

Theorem 3.2.3 follows from the above simple observation if we instantiate it with the functors defined below.

Lemma 3.2.5 *Let λ be a distributive law of a functor* S *over a functor* B.

(i) *The functor* $\mathrm{B} : \mathsf{Set} \to \mathsf{Set}$ *lifts to a functor* $\mathrm{B}_\lambda : \mathsf{Alg}^{\mathrm{S}} \to \mathsf{Alg}^{\mathrm{S}}$ *defined for an* S*-algebra* $\langle P, \beta\rangle$ *and an* S*-algebra homomorphism* h *as*

$$\begin{aligned} \mathrm{B}_\lambda\langle P,\beta\rangle &:= \langle \mathrm{B}P, \mathrm{B}\beta \circ \lambda_P\rangle \\ \mathrm{B}_\lambda h &:= \mathrm{B}h \end{aligned}$$

An $\langle \mathrm{S}, \mathrm{B}\rangle$*-bialgebra* $\langle P, \beta, \alpha\rangle$ *is a* λ*-bialgebra if and only if* $\langle\langle P, \beta\rangle, \alpha\rangle$ *is a* B_λ*-coalgebra, i.e.* α *is an* S*-algebra homomorphism from* $\langle P, \beta\rangle$ *to* $\mathrm{B}_\lambda\langle P, \beta\rangle$.

(ii) *Dually, the functor* $\mathrm{S} : \mathsf{Set} \to \mathsf{Set}$ *lifts to a functor* $\mathrm{S}_\lambda : \mathsf{Coalg}_{\mathrm{B}} \to \mathsf{Coalg}_{\mathrm{B}}$ *defined for a* B*-coalgebra* $\langle P, \alpha\rangle$ *and a* B*-coalgebra homomorphism* h *as*

$$\begin{aligned} \mathrm{S}_\lambda\langle P,\alpha\rangle &:= \langle \mathrm{S}P, \lambda_P \circ \mathrm{S}\alpha\rangle \\ \mathrm{S}_\lambda h &:= \mathrm{S}h \end{aligned}$$

An $\langle \mathrm{S}, \mathrm{B}\rangle$*-bialgebra* $\langle P, \beta, \alpha\rangle$ *is a* λ*-bialgebra if and only if* $\langle\langle P, \alpha\rangle, \beta\rangle$ *is a* S_λ*-algebra, i.e.* β *is a* B*-coalgebra homomorphism from* $\mathrm{S}_\lambda\langle P, \alpha\rangle$ *to* $\langle P, \alpha\rangle$.

Proof: We treat item (ii). To see that S_λ is a functor indeed, we have to check that for a B-coalgebra homomorphism $h : \langle P, \alpha_P\rangle \to \langle Q, \alpha_Q\rangle$ we get that $\mathrm{S}_\lambda h := \mathrm{S}h$ is a B-coalgebra homomorphism from $\mathrm{S}_\lambda\langle P, \alpha_P\rangle$ to $\mathrm{S}_\lambda\langle Q, \alpha_Q\rangle$. This easily follows from the naturality of λ, as the right diagram below shows.

$$\begin{array}{ccc} P & \xrightarrow{h} & Q \\ {\scriptstyle\alpha_P}\downarrow & (*) & \downarrow{\scriptstyle\alpha_Q} \\ \mathrm{B}P & \xrightarrow[\mathrm{B}h]{} & \mathrm{B}Q \end{array} \qquad \Rightarrow \qquad \begin{array}{ccc} \mathrm{S}P & \xrightarrow{\mathrm{S}h} & \mathrm{S}Q \\ {\scriptstyle\mathrm{S}\alpha_P}\downarrow & \mathrm{S}(*) & \downarrow{\scriptstyle\mathrm{S}\alpha_Q} \\ \mathrm{SB}P & \xrightarrow{\mathrm{SB}h} & \mathrm{SB}Q \\ {\scriptstyle\lambda_P}\downarrow & \text{nat. } \lambda & \downarrow{\scriptstyle\lambda_Q} \\ \mathrm{BS}P & \xrightarrow[\mathrm{BS}h]{} & \mathrm{BS}Q \end{array}$$

The second claim is immediate by definition.

□

Theorem 3.2.3 provides two canonical models of the specification λ, in case the initial S-algebra and final B-coalgebra exist.

Let us again assume that S arises from a signature Σ as in equation (2.2) on page 20, that B describes labelled transition systems as in Definition 2.3.12, and that λ was induced by a set of SOS rules, as in the above example of a synchronous parallel composition. Then the carrier of the initial S-algebra is the set of closed terms built from the operators in Σ. With the first item of the above theorem, the rules can be viewed as a specification of a unique transition system structure

on them. This is often the intended model of an SOS specification. The states of the final B-coalgebra uniquely represent all possible LTS behaviours up to bisimilarity. So the model given by the second item essentially provides a unique collection of operators on these behaviours satisfying the transition rules.

Theorem 3.2.6 *Let λ be a distributive law of the functor* S *over the functor* B. *The greatest bisimulation $R \subseteq P \times Q$ between any two λ-bialgebras $\langle P, \beta_P, \alpha_P \rangle$ and $\langle Q, \beta_Q, \alpha_Q \rangle$ is a congruence.*

Proof: Let $\gamma : R \to \mathrm{B}R$ be a mediating coalgebra structure. This means that the two projections $\pi_1 : R \to P$ and $\pi_2 : R \to Q$ are homomorphisms between the coalgebras below, i.e. they exist in $\mathsf{Coalg}_{\mathrm{B}}$.

$$\begin{array}{ccccc} \mathrm{S}_\lambda\langle P, \alpha_P\rangle & \xleftarrow{\mathrm{S}_\lambda\pi_1} & \mathrm{S}_\lambda\langle R, \gamma\rangle & \xrightarrow{\mathrm{S}_\lambda\pi_2} & \mathrm{S}_\lambda\langle Q, \alpha_Q\rangle \\ \downarrow{\scriptstyle \beta_P} & & & & \downarrow{\scriptstyle \beta_Q} \\ \langle P, \alpha_P\rangle & \xleftarrow[\pi_1]{} & \langle R, \gamma\rangle & \xrightarrow[\pi_2]{} & \langle Q, \alpha_Q\rangle \end{array}$$

Since S_λ is a functor on $\mathsf{Coalg}_{\mathrm{B}}$, the two arrows $\mathrm{S}_\lambda\pi_1$ and $\mathrm{S}_\lambda\pi_2$ are homomorphisms as well. The algebra operations β_P and β_Q are homomorphisms by the assumption on $\langle P, \beta_P, \alpha_P\rangle$ and $\langle Q, \beta_Q, \alpha_Q\rangle$ being λ-bialgebras (cf. Lemma 3.2.5 (ii)). So we get a span

$$\langle \mathrm{S}_\lambda\langle R, \gamma\rangle, \beta_P \circ \mathrm{S}_\lambda\pi_1, \beta_Q \circ \mathrm{S}_\lambda\pi_2\rangle$$

between $\langle P, \alpha_P\rangle$ and $\langle Q, \alpha_Q\rangle$ in $\mathsf{Coalg}_{\mathrm{B}}$. This means that the span

$$\langle \mathrm{S}R, \beta_P \circ \mathrm{S}\pi_1, \beta_Q \circ \mathrm{S}\pi_2\rangle$$

resulting after the application of the forgetful functor $\mathrm{U} : \mathsf{Coalg}_{\mathrm{B}} \to \mathsf{Set}$ is a bisimulation between $\langle P, \alpha_P\rangle$ and $\langle Q, \alpha_Q\rangle$, witnessed by the the coalgebra structure of $\mathrm{S}_\lambda\langle R, \gamma\rangle$ (i.e. $\lambda_R \circ \mathrm{S}\gamma : \mathrm{S}R \to \mathrm{BS}R$). The image of this span is contained in R, since the latter was assumed to be a greatest bisimulation between the same coalgebras. So by Lemma 2.3.5 there exists a morphism $m : \mathrm{S}R \to R$ making the two squares in the diagram below commute.

$$\begin{array}{ccccc} \mathrm{S}P & \xleftarrow{\mathrm{S}\pi_1} & \mathrm{S}R & \xrightarrow{\mathrm{S}\pi_2} & \mathrm{S}Q \\ \downarrow{\scriptstyle \beta_P} & & \downarrow{\scriptstyle m} & & \downarrow{\scriptstyle \beta_Q} \\ P & \xleftarrow[\pi_1]{} & R & \xrightarrow[\pi_2]{} & Q \end{array}$$

This proves that R is a congruence as wanted.

□

Turi and Plotkin base their proof of the corresponding statement [TP97, Corollary 7.5] on the construction of a greatest bisimulation as a pullback of the

final coalgebra homomorphisms. Therefore they need to assume that a final B-coalgebra exists and that B preserves weak pullbacks. These assumptions are not necessary for our proof.

Let S be again given by a signature Σ, let B describe labelled transition systems, and let λ be induced by a set of SOS rules. Recall that the initial λ-bialgebra from Theorem 3.2.3 (i) has the set of terms of the signature as its carrier and a transition structure which is uniquely defined by the transition rules. The above theorem states that the notion of strong bisimilarity associated to this transition structure is a congruence for the terms of the language. This is a central property of a set of transition rules for instance in the area of process algebra (see e.g. [GV92]). But note that the above result also allows models of the specification other than the one on the set of terms.

3.3 Rule formats derived from distributive laws

We have shown that the specification of the synchronous parallel composition for labelled transition systems in terms of transition rules gives rise to a distributive law of the signature functor over the corresponding behaviour functor. The models of the specification could alternatively be characterised as the bialgebras for this distributive law. The category of all such bialgebras in turn is well-behaved in several respects. Taken together we concluded that the original specification was well-behaved.

In this section we want to make precise for which specifications this chain of reasoning applies, i.e. we characterise the sets of transition rules that correspond to distributive laws as above. The resulting format will be less powerful than other known well-behaved specification formats, but in Section 3.4 we shall show how to generalise the approach.

The task boils down to characterising natural transformations of the respective type in concrete terms. To this end, we employ a collection of simple lemmata that allow us to decompose complex natural transformations into simpler ones or give direct descriptions of natural transformations of a simple structure. In order not to disturb the flow of reading with too many technicalities here, we state and prove them in Appendix A.1.

This collection of lemmata helps us to derive concrete representations of the distributive laws of S over B in the case that S is the functor associated to a signature Σ and B is a behaviour functor modelling any of the types of systems considered in Section 2.3.3. First we give the argument for labelled transition systems in detail. For it we moreover need a characterisation result for simple natural transformations involving the powerset functor, which we prove in Appendix A.2. For now this is the most important and at the same time the most complicated case. Later in this section we briefly mention streams and automata. In Chapter 5 we shall use the techniques developed here to derive novel specification formats for probabilistic systems.

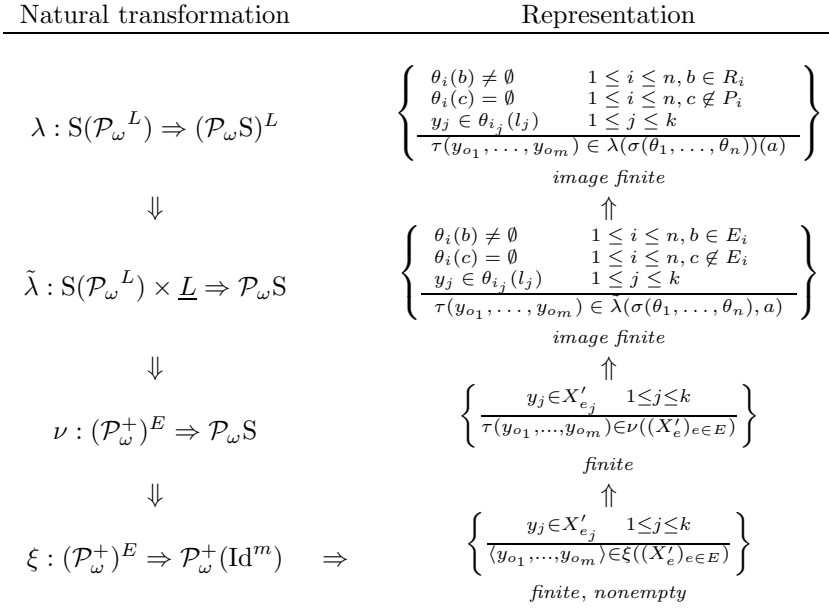


Figure 3.1: The outline of our approach.

3.3.1 LTS

In this part we analyse which specifications give rise to distributive laws of a functor S associated to a signature Σ (cf. equation (2.2) on page 20) over the functor $\mathrm{B} = {\mathcal{P}_\omega}^L$ modelling labelled transition systems (cf. Definition 2.3.12). Recall that these are natural transformations of the type

$$\lambda : \mathrm{S}({\mathcal{P}_\omega}^L) \Rightarrow (\mathcal{P}_\omega \mathrm{S})^L. \tag{3.3}$$

Using the lemmata from Appendix A.1, we first decompose those transformations into simpler ones and then apply a direct representation theorem developed in Appendix A.2. Putting together the pieces, we shall obtain a characterisation in terms of transition rules. The approach is outlined in Figure 3.1. In the following we explain the development in detail.

Decomposing the natural transformations

First of all, by Lemma A.1.1 and the adjunction $\mathrm{Id} \times \underline{L} \dashv (\mathrm{Id})^L$, natural transformations (3.3) are in one-to-one correspondence with those of the shape

$$\tilde{\lambda} : \mathrm{S}({\mathcal{P}_\omega}^L) \times \underline{L} \Rightarrow \mathcal{P}_\omega \mathrm{S}. \tag{3.4}$$

Let us write $\mathrm{F} := \mathrm{S}({\mathcal{P}_\omega}^L) \times \underline{L}$ for the functor in the domain of $\tilde{\lambda}$. For the next decomposition step, we express this functor equivalently as a coproduct, which

can be done according to Lemma A.1.4 as $\mathrm{F} \simeq \coprod_{z \in \mathrm{F}1} \mathrm{F}|_z$, where 1 is a final set and

$$\mathrm{F}|_z X := \{\phi \in \mathrm{F}X \mid (\mathrm{F}!)(\phi) = z\}$$

for the unique arrow $! : X \to 1$ and $z \in \mathrm{F}1$.

With this presentation of F we can apply Lemma A.1.2, which states that a natural transformation with a coproduct in the domain can be split into natural transformations from each summand. So $\tilde{\lambda}$ can be described by a family of natural transformations

$$(\nu^z : \mathrm{F}|_z \Rightarrow \mathcal{P}_\omega \mathrm{S})_{z \in \mathrm{F}1}. \tag{3.5}$$

We shall now characterise the individual natural transformations ν^z for $z \in \mathrm{F}1$. To this end we first derive a concrete description of the functors $\mathrm{F}|_z$. With $\mathcal{P}_\omega 1 = \{\emptyset, 1\} \simeq 2$ and $2^L \simeq \mathcal{P}L$ we get that the functor F from the domain of our natural transformations in (3.4) maps the singleton set 1 to

$$\begin{array}{rcl} \mathrm{F}1 & = & \mathrm{S}(\mathcal{P}_\omega 1^L) \times L \\ & \simeq & \mathrm{S}(2^L) \times L \\ & \simeq & \mathrm{S}\mathcal{P}L \times L \\ & = & \{\langle \sigma(E_1, \ldots, E_n), a\rangle \mid \sigma \in \Sigma \text{ with arity } n; E_1, \ldots, E_n \subseteq L; a \in L\}. \end{array}$$

The isomorphism is given by

$$\langle \sigma(\theta_1, \ldots, \theta_n), a\rangle \in \mathrm{F}1 \mapsto \langle \sigma(E_1, \ldots, E_n), a\rangle \in \mathrm{S}\mathcal{P}L \times L$$

where $E_i := \{l \in L \mid \theta_i(l) \neq \emptyset\}$. For simplicity we will use elements from the latter set to describe those of F1 without making the isomorphism explicit. For every set X and $z = \langle \sigma(E_1, \ldots, E_n), a\rangle \in \mathrm{F}1$ we calculate

$$\begin{array}{rcl} \mathrm{F}|_z X & := & \{\phi \in \mathrm{F}X \mid (\mathrm{F}!)(\phi) = z\} \\ & = & \{\langle \sigma(\theta_1, \ldots, \theta_n), a\rangle \mid \theta_1, \ldots, \theta_n \in (\mathcal{P}_\omega X)^L \text{ such that} \\ & & \qquad \forall l \in L, 1 \leq i \leq n : \ \theta_i(l) \neq \emptyset \iff l \in E_i\} \\ & \simeq & \{\langle \sigma(\tilde{\theta}_1, \ldots, \tilde{\theta}_n), a\rangle \mid \tilde{\theta}_i \in (\mathcal{P}^+_\omega X)^{E_i}, 1 \leq i \leq n\} \\ & \simeq & \displaystyle\prod_{i=1}^{n} (\mathcal{P}^+_\omega X)^{E_i}. \\ & \simeq & (\mathcal{P}^+_\omega X)^E, \end{array} \tag{3.6}$$

where $E := E_1 + \cdots + E_n$. So we will study the natural transformations ν^z from (3.5) as examples of natural transformations of the following type for a set E:

$$\nu : (\mathcal{P}^+_\omega)^E \Rightarrow \mathcal{P}_\omega \mathrm{S} \tag{3.7}$$

Now we focus on the codomain of these natural transformations. Writing $|\tau|$ for the arity of $\tau \in \Sigma$ and using Lemma A.1.5 to distribute the nonempty powerset functor over a coproduct in the step $(*)$ below we calculate

$$\begin{aligned}
\mathcal{P}_\omega \mathrm{S} &\simeq 1 + \mathcal{P}_\omega^+ \mathrm{S} \\
&\overset{(2.2)}{=} 1 + \mathcal{P}_\omega^+ \Big(\coprod_{\tau \in \Sigma} \mathrm{Id}^{|\tau|} \Big) \\
&\overset{(*)}{\simeq} 1 + \coprod_{M \in \mathcal{P}_\omega^+ \Sigma} \Big(\prod_{\tau \in M} \mathcal{P}_\omega^+ (\mathrm{Id}^{|\tau|}) \Big) \\
&\simeq \coprod_{M \in \mathcal{P}_\omega \Sigma} \Big(\prod_{\tau \in M} \mathcal{P}_\omega^+ (\mathrm{Id}^{|\tau|}) \Big)
\end{aligned}$$

Lemma A.1.3 states that a natural transformation from a functor preserving finality into a coproduct hits just one of the summands, so that ν arises from a natural transformation of the type $(\mathcal{P}_\omega^+)^E \Rightarrow \prod_{\tau \in M} \mathcal{P}_\omega^+(\mathrm{Id}^{|\tau|})$ for some $M \in \mathcal{P}_\omega \Sigma$. This in turn can by Lemma A.1.2 (b) be split into natural transformations into each factor $\mathcal{P}_\omega^+(\mathrm{Id}^{|\tau|})$ for $\tau \in M$. So ν in (3.7) is given by

$$M \in \mathcal{P}_\omega \Sigma \quad \text{and} \quad \big(\xi^\tau : (\mathcal{P}_\omega^+)^E \Rightarrow \mathcal{P}_\omega^+ (\mathrm{Id}^{|\tau|}) \big)_{\tau \in M} \tag{3.8}$$

At this point we stop the decomposition, since for natural transformations

$$\xi : (\mathcal{P}_\omega^+)^E \Rightarrow \mathcal{P}_\omega^+ (\mathrm{Id}^m) \tag{3.9}$$

as they appear in the above representation we can use a direct characterisation result stated and proved in Corollary A.2.8 in the Appendix.

Constructing the rule format

Each natural transformation ξ as in (3.9) can be characterised according to Corollary A.2.8 by a nonempty, finite set of derivation rules as in the expression below.

$$\xi \simeq \left\{ \frac{y_j \in X'_{e_j} \quad 1 \le j \le k}{\langle y_{o_1}, \ldots, y_{o_m} \rangle \in \xi\big((X'_e)_{e \in E}\big)} \;\Big|\; k, e_j, o_i \right\}_{\mathit{finite, nonempty}} \tag{3.10}$$

Any rule mentions distinct variables $y_1, \ldots, y_k$ for some $k \in \mathbb{N}$, where y_j ranges over the elements of X_{e_j} for some $e_j \in E$. For $1 \le i \le m$ the variable y_{o_i} is placed in the i-th position of the resulting tuple with $1 \le o_i \le k$.

So with (3.8) a natural transformation $\nu : (\mathcal{P}_\omega^+)^E \Rightarrow \mathcal{P}_\omega \mathrm{S}$ can be represented by a set $M \in \mathcal{P}_\omega \Sigma$ and for each $\tau \in M$ a set of rules as in (3.10) with $m = |\tau|$. To simplify this characterisation, we union the sets of rules for all $\tau \in M$. We tag each individual rule with the corresponding τ, so that we do not lose

information. Thus we write a rule as in the notation below. The resulting set will be finite still, but not necessarily nonempty, because M can be empty.

$$\nu \simeq \left\{ \frac{y_j \in X'_{e_j} \quad 1 \leq j \leq k}{\tau(y_{o_1}, \ldots, y_{o_{|\tau|}}) \in \nu\big((X'_e)_{e \in E}\big)} \;\middle|\; \tau, k, e_j, o_i \right\}_{\mathit{finite}} \tag{3.11}$$

For $z = \langle \sigma(E_1, \ldots, E_n), a \rangle \in \mathrm{F1}$ we treated ν^z appearing in (3.5) equivalently as $\nu : (\mathcal{P}_\omega^+)^E \Rightarrow \mathcal{P}_\omega \mathrm{S}$ for $E = E_1 + \cdots + E_n$. To adapt our rule notation to the original type of ν^z, we rewrite the rules as below, where $1 \leq i_j \leq n$ and $l_j \in E_{i_j}$.

$$\nu^z \simeq \left\{ \frac{y_j \in \theta_{i_j}(l_j) \quad 1 \leq j \leq k}{\tau(y_{o_1}, \ldots, y_{o_{|\tau|}}) \in \nu^z(\sigma(\theta_1, \ldots, \theta_n), a)} \;\middle|\; \tau, k, i_j, l_j, o_i \right\}_{\mathit{finite}} \tag{3.12}$$

To obtain a characterisation for $\tilde{\lambda}$ in (3.4), we again union the sets of rules for all $z \in \mathrm{F1}$. And again we need to rewrite each rule such that the corresponding $z = \langle \sigma(E_1, \ldots, E_n), a \rangle$ is encoded. We already included σ and a, but we need to account for the E_i still, which are implicit in the notation above through the typing of ν^z (remember that its parameters $\theta_i : L \to \mathcal{P}_\omega X$ need to be such that $\theta_i(b) \neq \emptyset$ if and only if $b \in E_i$). This will result in new premises, as written below. The resulting set will not necessarily be finite, because there may be infinitely many labels or operator symbols. But there will be finitely many rules for the same σ, E_i, and a only, a property that we will call *image finiteness*.

$$\tilde{\lambda} \simeq \left\{ \frac{\begin{array}{ll} \theta_i(b) \neq \emptyset & 1 \leq i \leq n, b \in E_i \\ \theta_i(c) = \emptyset & 1 \leq i \leq n, c \notin E_i \\ y_j \in \theta_{i_j}(l_j) & 1 \leq j \leq k \end{array}}{\tau(y_{o_1}, \ldots, y_{o_m}) \in \tilde{\lambda}(\sigma(\theta_1, \ldots, \theta_n), a)} \;\middle|\; \begin{array}{l} \sigma, a, E_i, \tau, \\ k, i_j, l_j, o_i \end{array} \right\}_{\substack{\mathit{image} \\ \mathit{finite}}} \tag{3.13}$$

To turn this set of rules into a representation for λ from (3.3) we just need to replace $\tilde{\lambda}(\sigma(\theta_1, \ldots, \theta_n), a)$ in the conclusion by $\lambda(\sigma(\theta_1, \ldots, \theta_n))(a)$. But moreover for ease of notation we will replace the set E_i of enabled labels for the i-th argument by two disjoint sets $R_i, P_i \subseteq L$ of *required* and *prohibited* labels. The resulting rule is applicable when for the i-th argument the set of enabled labels contains all required ones and none of the prohibited ones, i.e. $R_i \subseteq E_i$ and $E_i \cap P_i = \emptyset$. This way we can write several rules which differ in the sets of enabled labels only as one.

The following theorem summarises the result of our development.

Theorem 3.3.1 *Every natural transformation λ from (3.3) can be characterised*

by an image finite set of rules of the shape

$$\frac{\begin{array}{ll}\theta_i(b) \neq \emptyset & 1 \leq i \leq n, b \in R_i \\ \theta_i(c) = \emptyset & 1 \leq i \leq n, c \in P_i \\ y_j \in \theta_{i_j}(l_j) & 1 \leq j \leq k\end{array}}{\tau(y_{o_1}, \ldots, y_{o_m}) \in \lambda(\sigma(\theta_1, \ldots, \theta_n))(a)}$$

where $a \in L$; $\sigma, \tau \in \Sigma$ (with arity n and m respectively); $R_i, P_i \subseteq L$ such that $R_i \cap P_i = \emptyset$ $(1 \leq i \leq n)$; $k \in \mathbb{N}$; $1 \leq i_j \leq n$, $l_j \in R_{i_j}$ $(1 \leq j \leq k)$; $1 \leq o_i \leq k$ $(1 \leq i \leq m)$.

The image finiteness assumption means that for all $\sigma \in \Sigma$ (with arity n), $E_1, \ldots, E_n \subseteq L$, and $a \in L$ the set contains only finitely many rules (in the notation above) for this a and σ, such that $R_i \subseteq E_i$ and $E_i \cap P_i = \emptyset$ for $1 \leq i \leq n$.

To match this rule notation with the usual transition rules, we have to remember the pentagonal diagram an $\langle \mathrm{S}, \mathrm{B} \rangle$-bialgebra $\langle P, [\![.]\!], \alpha \rangle$ should satisfy in order to be a λ-bialgebra (Def. 3.2.2): the (P-component of the) natural transformation λ is applied to arguments of the form $\sigma(\alpha(p_1), \ldots, \alpha(p_n))$ and the result is evaluated using $[\![.]\!]$. Instantiated with the arguments above, we can use the arrow notation introduced in Definition 2.3.12 to rewrite the rules characterising λ.

Corollary 3.3.2 *The classes of λ-bialgebras for any λ as in (3.3) are precisely those that can be characterised by an image finite set of rules as below.*

$$\frac{\begin{array}{ll}x_i \xrightarrow{b} & 1 \leq i \leq n, b \in R_i \\ x_i \xrightarrow{c} \!\!\!\!\!\!\rightarrow & 1 \leq i \leq n, c \in P_i \\ x_{i_j} \xrightarrow{l_j} y_j & 1 \leq j \leq k\end{array}}{[\![\sigma(x_1, \ldots, x_n)]\!] \xrightarrow{a} [\![\tau(y_{o_1}, \ldots, y_{o_m})]\!]}$$

It determines the $\langle \mathrm{S}, \mathrm{B} \rangle$-bialgebras $\langle P, [\![.]\!], \alpha \rangle$ for which the states $[\![\sigma(p_1, \ldots, p_n)]\!]$ for all $\sigma \in \Sigma$ and $p_1, \ldots, p_n \in P$ allow precisely the transitions derivable from the rules.

Compared to other congruence formats, this one is rather weak. It does allow negative premises, but transitions from $[\![\sigma(p_1, \ldots, p_n)]\!]$ can lead only to states expressible as $[\![\tau(q_1, \ldots, q_m)]\!]$ for some operator symbol τ (with arity m), where each q_j is an immediate successor of some p_i. In Section 3.4 we will discuss extensions of the approach.

3.3.2 Streams

We will now briefly spell out which specifications give rise to distributive laws of a functor S associated to the signature Σ over the functor $\mathrm{B} = \underline{\mathbb{R}} \times \mathrm{Id}$ for

stream systems, which were introduced in Section 2.3.3. So we consider natural transformations of the shape

$$\lambda : \mathrm{S}(\underline{\mathbb{R}} \times \mathrm{Id}) \Rightarrow \underline{\mathbb{R}} \times \mathrm{S} \tag{3.14}$$

Using $\mathrm{S} = \coprod_{\sigma\in\Sigma} X^{|\sigma|}$ (cf. equation (2.2) on page 20) and Lemma A.1.2 about natural transformations from a coproduct, we find that each such λ is equivalent to a family of transformations

$$\big(\nu^\sigma : (\underline{\mathbb{R}} \times \mathrm{Id})^{|\sigma|} \Rightarrow \underline{\mathbb{R}} \times \mathrm{S}\big)_{\sigma\in\Sigma} \tag{3.15}$$

Lemma A.1.7 allows us to remove certain occurrences of the identity functor from the domain of a natural transformation. Applied here it yields that, for each $\sigma \in \Sigma$ (with arity n), each ν^σ is equivalent to a natural transformation

$$\tilde{\nu}^\sigma : \underline{\mathbb{R}}^n \Rightarrow \underline{\mathbb{R}} \times \mathrm{S}(\underline{N} + \mathrm{Id}), \tag{3.16}$$

where $N = \{1, \ldots, n\}$. The domain of this natural transformation is a constant functor and is thus by Lemma A.1.6 characterised by a plain function

$$h^\sigma : \mathbb{R}^n \to \mathbb{R} \times \mathrm{S}N \tag{3.17}$$

As a result, a specification gives rise to a distributed law as in (3.14) if and only if for all operators $\sigma \in \Sigma$ and $u_1, \ldots, u_n \in \mathbb{R}$, where n is the arity of σ, it specifies a derivation

$$\frac{x_i \xrightarrow{u_i} y_i \quad 1 \le i \le n}{\sigma(x_1, \ldots, x_n) \xrightarrow{v} \tau(y_{p_1}, \ldots, y_{p_m})}$$

for some value $v \in \mathbb{R}$, operator symbol $\tau \in \Sigma$ (with arity say m), and indices $1 \le p_i \le n$ for $1 \le i \le m$.

The specification of $\mathtt{map}_g$ from equation (2.8) on page 32 for instance can be expressed this way. Further examples are the specifications of (unary) prefix operations $u.x$ for $u \in \mathbb{R}$ and of a (binary) pointwise summation $x + y$ of streams:

$$\frac{x \xrightarrow{v} x'}{u.x \xrightarrow{u} v.x'}(\forall u, v \in \mathbb{R}) \qquad \frac{x \xrightarrow{u} x' \quad y \xrightarrow{v} y'}{x + y \xrightarrow{u+v} x' + y'}(\forall u, v \in \mathbb{R})$$

Note that in the first case we did not give the more intuitive rules $u.x \xrightarrow{u} x$ without premises for all $u \in \mathbb{R}$, because they do not fit into the simple format. But they will fit into extensions of the format to be discussed in the coming sections.

3.3.3 Deterministic automata

As the last example we consider deterministic automata modelled as coalgebras of the functor $\mathrm{B} = \underline{2} \times \mathrm{Id}^L$. We analyse distributive laws of the functor

S associated to a signature Σ over this behaviour functor, which are natural transformations

$$\lambda : \mathrm{S}(\underline{2} \times \mathrm{Id}^L) \Rightarrow \underline{2} \times \mathrm{S}^L. \tag{3.18}$$

An argument similar to the one in the case of streams above shows that those are equivalent to families of functions

$$\left(h^\sigma : 2^{|\sigma|} \to 2 \times (\mathrm{S}(\{1, \ldots, |\sigma|\} \times L))^L\right)_{\sigma \in \Sigma} \tag{3.19}$$

Translated into a transition rule format, such a specification corresponds to a set of rules of the following two types:

$$\frac{x_i{\downarrow}\ (i \in P) \quad x_i{\uparrow}\ (i \in N)}{\sigma(x_1, \ldots, x_n){\downarrow}}$$

and

$$\frac{x_i \xrightarrow{l} y_i^l\ (1 \le i \le n, l \in L) \quad x_i{\downarrow}\ (i \in P) \quad x_i{\uparrow}\ (i \in N)}{\sigma(x_1, \ldots, x_n) \xrightarrow{a} \tau(y_{i_1}^{l_1}, \ldots, y_{i_m}^{l_m})}$$

where $\sigma, \tau \in \Sigma$ (with arity n and m); $P, N \subseteq \{1, \ldots, n\}$ with $P \cap N = \emptyset$; $a \in L$; $1 \le i_j \le n$; and $l_j \in L$ for $1 \le j \le m$ (we omit a premise $x_i \xrightarrow{b} y_i^l$ if y_i^l does not appear in the target). For each $\sigma \in \Sigma$, $a \in L$, and $A \subseteq \{1, \ldots, n\}$ there should be exactly one rule of the second type with this σ and a such that $P \subseteq A$ and $A \cap N = \emptyset$.

For instance, the specification of the binary union operation $\cup$ below fits into this format.

$$\frac{x{\downarrow}}{(x \cup y){\downarrow}} \qquad \frac{y{\downarrow}}{(x \cup y){\downarrow}} \qquad \frac{x \xrightarrow{a} x_a \quad y \xrightarrow{a} y_a}{x \cup y \xrightarrow{a} x_a \cup y_a}(\forall a \in L) \tag{3.20}$$

3.4 Extensions of the simple format

In Section 3.3.1 we illustrated the limitation of our categorical format in its current form with the example of labelled transition systems: when $\sigma(x_1, \ldots, x_n)$ is the source of a rule, then the target can only be of the shape $\tau(y_1, \ldots, y_m)$ for some operator symbol τ (with arity m), where the y_j are successors of the x_i.

In this section we first develop four extensions of this approach in separation. Later we show how to combine some of them. The known GSOS format [BIM95] for transition system specifications for instance emerges from such a combination, as we shall explain in Section 3.5.

The four extensions improve the format to allow the target of a rule with source $\sigma(x_1, \ldots, x_n)$ to be, respectively,

(1a) $\tau(\tilde{y}_1, \ldots, \tilde{y}_m)$, where $\tau \in \Sigma$ with arity m and $\tilde{y}_j$ is either one of the x_i or ranges over their immediate successors, like in:

$$\frac{x \xrightarrow{a} y}{\sigma(x) \xrightarrow{a} \tau(x, y)}$$

(1b) either $\tau(y_1, \ldots, y_m)$ or y, where $\tau \in \Sigma$ is an operator symbol with arity m and y and y_j range over successors of the x_i, like in:

$$\frac{x \xrightarrow{b} y}{\sigma(x) \xrightarrow{a} y}$$

(2a) $t \in \mathrm{T}Y$, i.e. an arbitrary term with variables ranging over the immediate successors of the x_i (this extension subsumes the one in (1b)), like in:

$$\frac{x \xrightarrow{a} y}{\sigma(x) \xrightarrow{a} \tau(\sigma(y), y)}$$

(2b) $\tau(\tilde{y}_1, \ldots, \tilde{y}_m)$, where $\tau \in \Sigma$ with arity m and each $\tilde{y}_j$ is either one of the x_i or ranges over their successors after an arbitrary number of steps (this extension subsumes the one in (1a)), like in:

$$\frac{x \xrightarrow{a} y \quad y \xrightarrow{b} z}{\sigma(x) \xrightarrow{a} \tau(x, z)}$$

To obtain the extensions (1a) and (1b) we equip the behaviour functor B and the signature functor S with the structure of a *copointed* and *pointed functor* respectively. For (2a) and (2b) we employ *monads* and *comonads*.

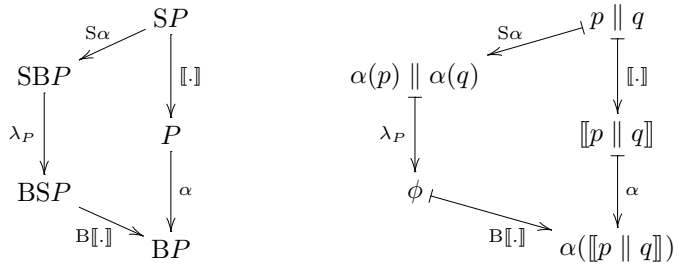
3.4.1 Using (co)pointed functors

As an example of a specification that does not give rise to a distributive law of the functor S associated to a signature Σ over the behaviour functor B under consideration we study the *asynchronous parallel composition* of labelled transition systems. It is given by the rules below for all $a \in L$.

$$\frac{x \xrightarrow{a} x'}{x \parallel y \xrightarrow{a} x' \parallel y} \qquad \frac{y \xrightarrow{a} y'}{x \parallel y \xrightarrow{a} x \parallel y'} \tag{3.21}$$

Neither of the two rules fits in the format from Corollary 3.3.2, because x or y, which are the arguments of $x \parallel y$ in the source, appear in the target. But as of yet, our format allows only immediate successors in the target.

To locate the weak spot in the current approach, imagine that there was a distributive law λ capturing the above definition and that $\langle P, [\![.]\!], \alpha \rangle$ was a λ-bialgebra. So the left diagram below commutes.



Let $p, q \in P$.The transitions rules (3.21) say that

$$\alpha([\![p \parallel q]\!]) = \{[\![p' \parallel q]\!] \mid p' \in \alpha(p)(a)\} \cup \{[\![p \parallel q']\!] \mid q' \in \alpha(q)(a)\},$$

so we can read off the right diagram above that λ_P ought to map $\alpha(p) \parallel \alpha(q) \in \mathrm{SB}P$ to $\phi \in \mathrm{BS}P$, so that for all $a \in L$

$$\phi(a) = \{\ p' \parallel q\ \mid p' \in \alpha(p)(a)\} \cup \{\ p \parallel q'\ \mid q' \in \alpha(q)(a)\}.$$

But the problem is that p and q appear in this expression although they are not in the input of λ_P, since, by applying $\mathrm{S}\alpha$, the arguments p and q are first replaced by their outgoing transitions $\alpha(p)$ and $\alpha(q)$.

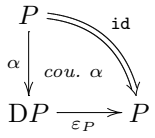
One could now modify the pentagonal diagram from the definition of a λ-bialgebra in some suitable but maybe ad-hoc way so that the information about p and q is conveyed, and then try to prove similar properties as in Section 3.2.1 for this new definition. Such an approach has actually been taken, leading to the notion of a "generalised distributive law" (see [CHL03]). Here we will adopt a different approach allowing us to – at least for a moment – still work with distributive laws and their bialgebras as before, but now for modified functors and with additional coherence axioms.

Copointed functors

Our distributive law is given the information about $\alpha(p)$ and $\alpha(q)$, where p and q are the arguments of the operator under consideration, but we need to access p and q themselves. This was not a problem if our coalgebras were such that the transition structure, as it were, remembered the current state, i.e. p was encoded in $\alpha(p)$. To state this property formally, we recall the notion of a *copointed (endo)functor*.

Definition 3.4.1 *A* **copointed (endo)functor** *on a category* C *is a pair* $\langle \mathrm{D}, \varepsilon \rangle$ *of a functor* $\mathrm{D} : \mathsf{C} \to \mathsf{C}$ *and a natural transformation* $\varepsilon : \mathrm{D} \Rightarrow \mathrm{Id}$*, called the* counit.

A **coalgebra for a copointed functor** $\langle \mathrm{D}, \varepsilon \rangle$ *is a* D*-coalgebra* $\langle P, \alpha \rangle$ *making the diagram below commute, which we call the* counit law for α*.*



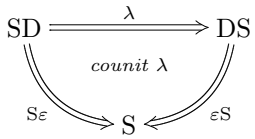
The category of all coalgebras for the copointed functor $\langle \mathrm{D}, \varepsilon \rangle$ *is denoted by* $\mathsf{Coalg}_{\langle \mathrm{D}, \varepsilon \rangle}$*. Given an algebra functor* S*, the full subcategory of* $\mathsf{Bialg}^{\mathrm{S}}_{\mathrm{D}}$ *consisting of those* $\langle \mathrm{S}, \mathrm{D} \rangle$*-bialgebras* $\langle P, \beta, \alpha \rangle$ *such that* α *satisfies the counit law is denoted by* $\mathsf{Bialg}^{\mathrm{S}}_{\langle \mathrm{D}, \varepsilon \rangle}$*.*

We proceed in two steps: first, we adapt the setting of distributive laws and their bialgebras to copointed functors; second, given a (plain) behaviour functor B, we shall define in a canonical way a (cofree) copointed functor $\langle \mathrm{D}, \varepsilon \rangle$, to which the framework from the first step is applied. The results obtained this way can be translated back into the original setting, because the copointed functor $\langle \mathrm{D}, \varepsilon \rangle$ is such that its coalgebras are in one-to-one correspondence with B-coalgebras.

Distributive laws and copointed functors

For a given functor S and a copointed functor $\langle \mathrm{D}, \varepsilon \rangle$ we can straightforwardly adapt the framework from Section 3.2: we again use distributive laws λ of S over D as specifications and λ-bialgebras as models, where this time we require the coalgebra operations of the bialgebras to be coalgebras for the copointed functor. It turns out that the properties we stated in Section 3.2.1 still hold if we make one assumption on the distributive laws under consideration.

Definition 3.4.2 *A* **distributive law** *of a functor* S *over a copointed functor* $\langle \mathrm{D}, \varepsilon \rangle$ *is a natural transformation* $\lambda : \mathrm{SD} \Rightarrow \mathrm{DS}$ *making the following diagram commute, which we call the* counit law for λ*.*



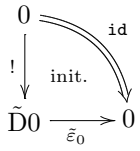
In this context we further require for a λ*-bialgebra* $\langle P, \beta, \alpha \rangle$ *(cf. Definition 3.2.2) that* $\langle P, \alpha \rangle$ *is a coalgebra of the copointed functor (cf. Def. 3.4.1).*

Corollary 3.4.3 *The statement of Theorem 3.2.3 still holds if we replace the functor* B *by the copointed functor* $\langle \mathrm{D}, \varepsilon \rangle$*, and correspondingly assume* λ *to satisfy the counit law and all coalgebra structures under consideration to be coalgebras for the copointed functor.*

Similar to the simple setting, this statement follows from two Lemmata below which are adaptations of Lemmata 3.2.4 and 3.2.5.

Lemma 3.4.4 *Let $\langle \tilde{\mathrm{D}}, \tilde{\varepsilon} \rangle$ be a copointed functor on a category C. There is a unique $\tilde{\mathrm{D}}$-coalgebra structure on an initial object 0 of C and it yields an initial coalgebra for the copointed functor $\langle \tilde{\mathrm{D}}, \tilde{\varepsilon} \rangle$.*

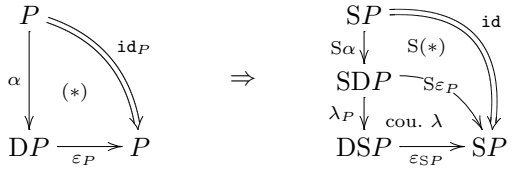
Proof: In addition to the proof of Lemma 3.2.4 (i) we need to check that the unique initial arrow $! : 0 \to \tilde{\mathrm{D}}0$ is a coalgebra structure for the copointed functor, i.e. the diagram below should commute. This easily follows from initiality.



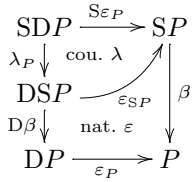
□

Lemma 3.4.5 *Let λ be a distributive law of the functor S over the copointed functor $\langle \mathrm{D}, \varepsilon \rangle$. (i) The functor S lifts to a functor S_λ on $\mathsf{Coalg}_{\langle \mathrm{D}, \varepsilon \rangle}$, and (ii) the copointed functor $\langle \mathrm{D}, \varepsilon \rangle$ lifts to a copointed functor $\langle \mathrm{D}_\lambda, \tilde{\varepsilon} \rangle$ on $\mathsf{Alg}^{\mathrm{S}}$.*

Proof: In addition to the proof of Lemma 3.2.5 we have one extra proof obligation for each statement. For (i) we check that for any coalgebra $\langle P, \alpha \rangle$ for the copointed functor $\langle \mathrm{D}, \varepsilon \rangle$ we indeed get that $\mathrm{S}_\lambda \langle P, \alpha \rangle = \langle \mathrm{S}P, \lambda_P \circ \mathrm{S}\alpha \rangle$ is a coalgebra for the copointed functor as well. This easily follows with the counit law of λ, as the right diagram below demonstrates.



For (ii) we need to prove that the component of the natural transformation $\tilde{\varepsilon}$ at each S-algebra $\langle P, \beta \rangle$, i.e. $\tilde{\varepsilon}_{\langle P, \beta \rangle} := \varepsilon_P : \mathrm{D}P \to P$ is indeed an arrow in $\mathsf{Alg}^{\mathrm{S}}$, i.e. an S-algebra homomorphism from $\mathrm{D}_\lambda \langle P, \beta \rangle$ to $\langle P, \beta \rangle$. This follows from the naturality of ε and again the counit law for λ, as shown below.



□

With the above auxiliary statements we can straightforwardly adapt the proof of Theorem 3.2.6 to this setting as well, which stated that bisimilarity is a congruence on every λ-bialgebra.

Corollary 3.4.6 *The statement of Theorem 3.2.6 still holds if we replace* B *by a copointed functor* $\langle \mathrm{D}, \varepsilon \rangle$ *and correspondingly assume that* λ *and all coalgebra structures under consideration satisfy the appropriate counit law.*

This variant essentially states that Theorem 3.2.6 still holds under the additional assumption that the mediating coalgebra structure in the definition of a bisimulation is a coalgebra for the copointed functor.

Cofree copointed functors

The specification of the asynchronous parallel composition led us to adapt our categorical format to the setting of coalgebras for copointed functors. Because for such a coalgebra $\langle P, \alpha \rangle$ and any $p \in P$ the information about p is kept in $\alpha(p)$, which was needed in the example. But the coalgebras there (i.e. labelled transition systems) are not coalgebras of a copointed functor. So the idea is to try and rephrase the example in the setting of coalgebras for a suitable copointed functor. Such a functor is given by the *cofree copointed functor* generated by the original behaviour functor B.

Definition 3.4.7 *The* **cofree copointed functor** $\langle \mathrm{D}, \varepsilon \rangle$ *generated by a* Set*-functor*[2] B *is defined as*

$$\mathrm{D} := \mathrm{Id} \times \mathrm{B} \quad \textit{and} \quad \varepsilon := \pi_1 : \mathrm{Id} \times \mathrm{B} \Rightarrow \mathrm{Id}.$$

Moreover, we use the name $\vartheta := \pi_2 : \mathrm{D} \Rightarrow \mathrm{B}$ *for the natural transformation given by the second projection. To mention* ϑ *explicitly, we sometimes write the cofree copointed functor as* $\langle \langle \mathrm{D}, \varepsilon \rangle, \vartheta \rangle$.

More abstractly and more generally, we can define a cofree copointed functor as an object in a category of copointed functors satisfying a universal property (cf. e.g. [LPW00, LPW04]). Here we limit ourselves to this more concrete definition, since we shall not need the categorical characterisation.

When dealing with cofree copointed functors, we mainly use the following two immediate consequences of the above definition:

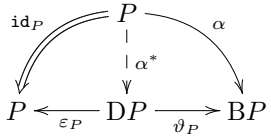
Corollary 3.4.8 *Let* $\langle \langle \mathrm{D}, \varepsilon \rangle, \vartheta \rangle$ *be the cofree copointed functor generated by* B.

[2] More generally, the definition makes sense in any category with binary coproducts.

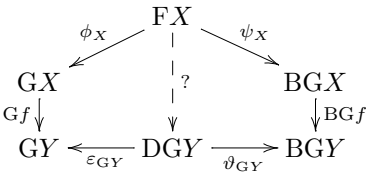
(i) *Any two arrows* $f : X \to Y$ *and* $g : X \to \mathrm{B}Y$ *uniquely determine an arrow* $h : X \to \mathrm{D}Y$ *making the two triangles in the left diagram below commute.*

This property extends to natural transformations: two natural transformations $\phi : \mathrm{F} \Rightarrow \mathrm{G}$ *and* $\psi : \mathrm{F} \Rightarrow \mathrm{BG}$ *uniquely determine a natural transformation* $\rho : \mathrm{F} \Rightarrow \mathrm{DG}$ *making both triangles in the right diagram above commute.*

(ii) *The categories* $\mathsf{Coalg}_{\mathrm{B}}$ *and* $\mathsf{Coalg}_{\langle \mathrm{D}, \varepsilon \rangle}$ *are isomorphic. The isomorphism is given by* $\langle P, \tilde{\alpha} \rangle \mapsto \langle P, \vartheta_P \circ \tilde{\alpha} \rangle$ *for a coalgebra* $\langle P, \tilde{\alpha} \rangle$ *for the cofree copointed functor* $\langle \mathrm{D}, \varepsilon \rangle$ *and* $\langle P, \alpha \rangle \mapsto \langle P, \alpha^* \rangle$ *for a* B*-coalgebra* $\langle P, \alpha \rangle$*, where* α^* *is the (by the first item) unique arrow fitting in the diagram below.*



Proof: The definition principle for arrows in the first item immediately follows from the universal property of the binary product. The extension to natural transformations is componentwise. Naturality follows from the uniqueness aspect of the principle: for any function $f : X \to Y$ we have that both $\mathrm{DG}f \circ \rho_X$ and $\rho_Y \circ \mathrm{F}f$ fit as the unique arrow into the diagram below, as is easily verified using the naturality of all transformations involved.



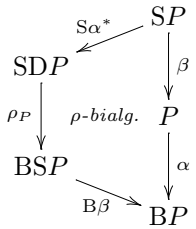
For the second item one easily verifies that the two constructions are inverses of each other and that they preserve homomorphisms.

□

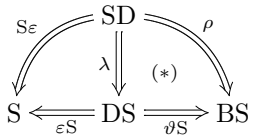
Remark 3.4.9 *We will apply the first item above mainly as we used it in the second: for a fixed arrow* f *(or natural transformation* ϕ*) there is a one-to-one correspondence between plain arrows* g *and arrows* h *satisfying* $\varepsilon_Y \circ h = f$ *(or plain natural transformations* ψ *and natural transformations* ρ *satisfying* $\varepsilon\mathrm{G} \circ \rho = \phi$*). We will then say that* g and h correspond under f *(*ψ and ρ correspond under ϕ*).*

Starting with a signature functor S and a behaviour functor B we take a distributive law λ of S over the cofree copointed functor $\langle \mathrm{D}, \varepsilon \rangle$ generated by B as a specification. The models are the $\langle \mathrm{S}, \mathrm{B} \rangle$-bialgebras $\langle P, \beta, \alpha \rangle$ such that $\langle P, \beta, \alpha^* \rangle$ is a λ-bialgebra, where α^* is the coalgebra structure for the copointed functor corresponding to α according to Corollary 3.4.8 (ii). Using the following lemma we can express the above approach directly.

Lemma 3.4.10 *Let* $\mathrm{S}, \mathrm{B} : \mathsf{Set} \to \mathsf{Set}$ *be two functors and let* $\langle \mathrm{D}, \varepsilon \rangle$ *be the cofree copointed functor generated by* B. *There is a one-to-one correspondence between distributive laws* λ *of* S *over the copointed functor* $\langle \mathrm{D}, \varepsilon \rangle$ *and (plain) natural transformations* $\rho : \mathrm{SD} \Rightarrow \mathrm{BS}$. *Moreover,* $\langle P, \beta, \alpha^* \rangle$ *is a* λ*-bialgebra just in case the* $\langle \mathrm{S}, \mathrm{B} \rangle$*-bialgebra* $\langle P, \beta, \alpha \rangle$ *(where* α *corresponds to* α^* *according to Corollary 3.4.8 (ii)) makes the diagram below for the corresponding natural transformation* ρ *commute. In this case we call* $\langle P, \beta, \alpha \rangle$ *a* **ρ-bialgebra**.

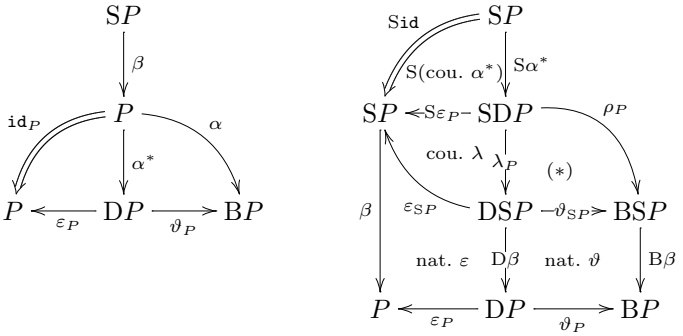


Proof: The correspondence of natural transformations is an immediate consequence of Corollary 3.4.8 (i):



For the second statement, observe that for any $\langle \mathrm{S}, \mathrm{B} \rangle$-bialgebra $\langle P, \beta, \alpha \rangle$ the two paths in the definition of a λ-bialgebra and a ρ-bialgebra pairwisely correspond to each other under β (i.e. $\alpha \circ \beta$ corresponds to $\alpha^* \circ \beta$ and $\mathrm{B}\beta \circ \rho_P \circ \mathrm{S}\alpha^*$ corresponds to $\mathrm{D}\beta \circ \lambda_P \circ \mathrm{S}\alpha^*$), as the two diagrams below show. In the case of the left one this immediately follows from the correspondence of α and α^* (see Corollary 3.4.8 (ii)). In the case of the right one we use counit laws, naturality,

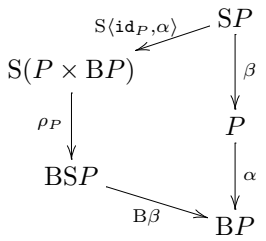
and the correspondence of λ and ρ from above in the part marked by $(*)$.



So the diagram for $\langle P, \beta, \alpha^* \rangle$ being a λ-bialgebra commutes if and only if the one for $\langle P, \beta, \alpha \rangle$ being a ρ-bialgebra does.

□

Remembering $\mathrm{D} := \mathrm{Id} \times \mathrm{B}$ from the definition of a cofree copointed functor, the above development is summarised as follows: given a signature functor S and a behaviour functor B, we can take natural transformations $\rho : \mathrm{S}(\mathrm{Id} \times \mathrm{B}) \Rightarrow \mathrm{BS}$ as specifications, the models of which are the ρ-bialgebras, i.e. the $\langle \mathrm{S}, \mathrm{B} \rangle$-bialgebras $\langle P, \beta, \alpha \rangle$ making the following diagram commute.



From Corollary 3.4.3 we derive that an initial S-algebra or a final B-coalgebra extends to an initial or final ρ-bialgebra respectively, from Corollary 3.4.6 we derive that the greatest bisimulation between any two ρ-bialgebras is a congruence.

The specification of the parallel composition in (3.21) gives rise to the natural transformation

$$\rho : \mathrm{S}(\mathrm{Id} \times (\mathcal{P}_\omega)^L) \Rightarrow (\mathcal{P}_\omega \mathrm{S})^L$$

defined for any set P; $p, q \in P$; and $\phi, \psi \in (\mathcal{P}_\omega P)^L$ by

$$\rho_P(\langle p, \phi \rangle \parallel \langle q, \psi \rangle) := [\, a \mapsto \{p' \parallel q \mid p' \in \phi(a)\} \cup \{p \parallel q' \mid q' \in \psi(a)\} \,] \,.$$

The corresponding ρ-bialgebras are precisely the models of the transition rules.

We could now again derive the rule format to which natural transformations ρ as above correspond. But we will skip this effort here and rather do it for a more interesting format after we have introduced further extensions.

The dual case: pointed functors

In this section we dualise the approach based on copointed functors from above. This will result in a reasonable extension of the simple format as well, as the following example specification is supposed to motivate. It is about a variant of the synchronous parallel composition from Section 3.2 where one component drops out at the moment that it cannot do a transition. It satisfies the following transition rules, each for all $a \in L$.

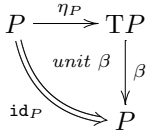
$$\frac{x \xrightarrow{a} x' \quad y \xrightarrow{a} y'}{x \otimes y \xrightarrow{a} x' \otimes y'} \qquad \frac{x \xrightarrow{a} x' \quad y \xrightarrow{a}\!\!\!\!\!\rightarrow}{x \otimes y \xrightarrow{a} x'} \qquad \frac{x \xrightarrow{a}\!\!\!\!\!\rightarrow \quad y \xrightarrow{a} y'}{x \otimes y \xrightarrow{a} y'} \tag{3.22}$$

This specification does not fit into the simple format from Corollary 3.3.2 because the rules of the second and third type declare transitions to plain states; no operator is applied.

An approach corresponding to the one in the previous section based on copointed functors leads us to the introduction of the dual concept of a *pointed functor*.

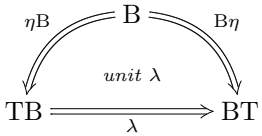
Definition 3.4.11 *A* **pointed (endo)functor** *on a category* C *is a pair* $\langle \mathrm{T}, \eta \rangle$ *of a functor* $\mathrm{T} : \mathsf{C} \to \mathsf{C}$ *and a natural transformation* $\eta : \mathrm{Id} \Rightarrow \mathrm{T}$*, called the* unit*.*

An **algebra for a pointed functor** $\langle \mathrm{T}, \eta \rangle$ *is a* T*-algebra* $\langle P, \beta \rangle$ *making the diagram below commute, which we call the* unit law for $\langle P, \beta \rangle$*.*



The category of all algebras for the pointed functor $\langle \mathrm{T}, \eta \rangle$ *is denoted by* $\mathsf{Alg}^{\langle \mathrm{T}, \eta \rangle}$*. For another functor* B*, a* $\langle \langle \mathrm{T}, \eta \rangle, \mathrm{B} \rangle$*-bialgebra is a* $\langle \mathrm{T}, \mathrm{B} \rangle$*-bialgebra* $\langle P, \beta, \alpha \rangle$ *such that* β *satisfies the unit law. The full subcategory of* $\mathsf{Bialg}^{\mathrm{T}}_{\mathrm{B}}$ *containing all* $\langle \langle \mathrm{T}, \eta \rangle, \mathrm{B} \rangle$*-bialgebras is denoted by* $\mathsf{Bialg}^{\langle \mathrm{T}, \eta \rangle}_{\mathrm{B}}$*.*

A **distributive law** *of a pointed functor* $\langle \mathrm{T}, \eta \rangle$ *over a functor* B *is a natural transformation* $\lambda : \mathrm{TB} \Rightarrow \mathrm{BT}$ *making the following diagram commute, which we call the* unit law for λ*.*



In this setting, a λ*-bialgebra is an object of* $\mathsf{Bialg}^{\langle \mathrm{T}, \eta \rangle}_{\mathrm{B}}$ *satisfying the pentagonal law from Def. 3.2.2.*

Dualising the proof of Corollary 3.4.3 we obtain the following statement.

Corollary 3.4.12 *The statement of Theorem 3.2.3 still holds if we replace the functor* S *by the pointed functor* $\langle \mathrm{T}, \eta \rangle$*, and correspondingly assume* λ *to satisfy the unit law and all algebra structures under consideration to be algebras for the pointed functor.*

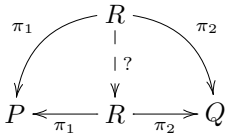
The corresponding variant of Theorem 3.2.6 does not simply follow by dualisation. It can still be shown to hold however.

Corollary 3.4.13 *The statement of Theorem 3.2.6 still holds if we replace the functor* S *by the pointed functor* $\langle \mathrm{T}, \eta \rangle$*, and correspondingly assume* λ *to satisfy the unit law and all algebra structures under consideration to be algebras for the pointed functor.*

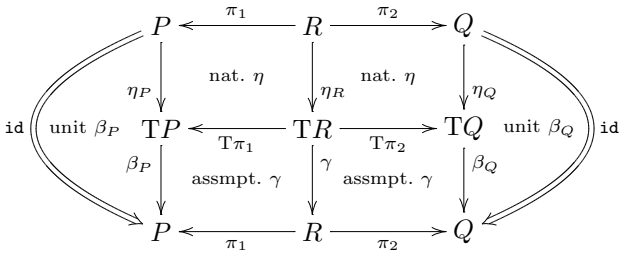
In addition to the proof of Theorem 3.2.6 we need to show that the algebra operation $m : \mathrm{T}R \to R$ obtained there to witness the congruence property is an algebra for the pointed functor $\langle \mathrm{T}, \eta \rangle$. This follows from the lemma below.

Lemma 3.4.14 *Let* $\langle P, \beta_P \rangle$ *and* $\langle Q, \beta_Q \rangle$ *be algebras for the pointed functor* $\langle \mathrm{T}, \eta \rangle$*. If* $R \subseteq P \times Q$ *is a congruence between* $\langle P, \beta_P \rangle$ *and* $\langle Q, \beta_Q \rangle$ *viewed as plain* T*-algebras, then it is also a congruence between them viewed as algebras for the pointed functor.*

Proof: We need to show that the mediating algebra structure $\gamma : \mathrm{T}R \to R$ witnessing the congruence property of R satisfies the unit law, i.e. $\mathtt{id}_R = \gamma \circ \eta_R$. We shall show that both arrows $\mathtt{id}_R$ and $\gamma \circ \eta_R$ make the two triangles in the diagram below commute, which implies that they are equal since the span $\langle R, \pi_1, \pi_2 \rangle$ is jointly monic.



We can obviously plug the identity function into the above diagram. That $\gamma \circ \eta_R$ makes both triangles commute as well follows with the unit laws of β_P and β_Q and the naturality of η:



□

Free pointed functors

Dual to the case of copointed functors, we define a *free pointed functor*.

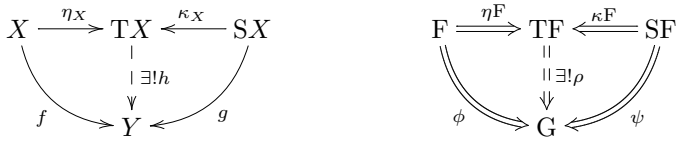
Definition 3.4.15 *The* **free pointed functor** $\langle \mathrm{T}, \eta \rangle$ *generated by a* Set*-functor* S *is defined as*

$$\mathrm{T} := \mathrm{Id} + \mathrm{S} \quad \textit{and} \quad \eta := \iota_1 : \mathrm{Id} \Rightarrow \mathrm{Id} + \mathrm{S}.$$

Moreover, we use the name $\kappa := \iota_1 : \mathrm{S} \Rightarrow \mathrm{T}$ *for the natural transformation given by the second injection. In order to make this transformation explicit, we sometimes write* $\langle \langle \mathrm{T}, \eta \rangle, \kappa \rangle$.

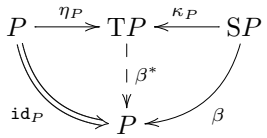
Corollary 3.4.16 *Let* $\langle \langle \mathrm{T}, \eta \rangle, \kappa \rangle$ *be the free pointed functor generated by* S.

(i) *Any two arrows* $f : X \to Y$ *and* $g : \mathrm{S}X \to Y$ *uniquely determine an arrow* $h : \mathrm{T}X \to Y$ *making the two triangles in the left diagram below commute.*



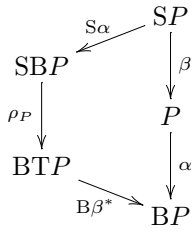
This property extends to natural transformations: two natural transformations $\phi : \mathrm{F} \Rightarrow \mathrm{G}$ *and* $\psi : \mathrm{SF} \Rightarrow \mathrm{G}$ *uniquely determine a natural transformation* $\rho : \mathrm{TF} \Rightarrow \mathrm{G}$ *making both triangles in the right diagram above commute.*

(ii) *The categories* $\mathsf{Alg}^{\mathrm{S}}$ *and* $\mathsf{Alg}^{\langle \mathrm{T}, \eta \rangle}$ *are isomorphic. An* S*-algebra* $\langle P, \beta \rangle$ *corresponds to the algebra* $\langle P, \beta^* \rangle$ *of the free copointed functor* $\langle \mathrm{T}, \eta \rangle$*, where* β^* *is the (by the first item) unique arrow fitting in the diagram below.*

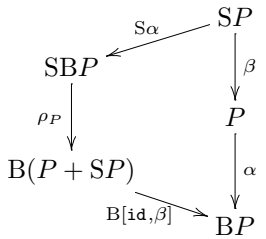


Lemma 3.4.17 *Let* $\mathrm{S}, \mathrm{B} : \mathsf{Set} \to \mathsf{Set}$ *be two functors and let* $\langle \mathrm{T}, \eta \rangle$ *be the free pointed functor generated by* S*. There is a one-to-one correspondence between distributive laws* λ *of the pointed functor* $\langle \mathrm{T}, \eta \rangle$ *over* B *and (plain) natural transformations* $\rho : \mathrm{SB} \Rightarrow \mathrm{BT}$*. For an* $\langle \mathrm{S}, \mathrm{B} \rangle$*-bialgebra* $\langle P, \beta, \alpha \rangle$ *we moreover have that* $\langle P, \beta^*, \alpha \rangle$ *(where* β^* *extends* β *according to Corollary 3.4.16 (ii)) is a* λ*-bialgebra if and only if* $\langle P, \beta, \alpha \rangle$ *makes the diagram below for the corresponding*

natural transformation ρ commute. In this case we call $\langle P, \beta, \alpha\rangle$ a **ρ-bialgebra**.



Spelling out the definition of a free pointed functor, we conclude from this statement that we can alternatively accept natural transformations $\rho : \mathrm{SB} \Rightarrow \mathrm{B}(\mathrm{S} + \mathrm{Id})$ as well-behaved specifications. Their models are the $\langle \mathrm{S}, \mathrm{B}\rangle$-bialgebras $\langle P, \beta, \alpha\rangle$ making the diagram below commute.



Our example specification (3.22) fits into this framework: the rules give rise to the natural transformation

$$\rho : \mathrm{S}({\mathcal{P}_\omega}^L) \Rightarrow (\mathcal{P}_\omega(\mathrm{Id} + \mathrm{S}))^L$$

defined for any set X and $\phi, \psi \in (\mathcal{P}_\omega X)^L$ as

$$\rho_X(\phi \otimes \psi) := a \mapsto \begin{cases} \iota_1[\phi(a)] & \text{if } \psi(a) = \emptyset, \\ \iota_1[\psi(a)] & \text{if } \phi(a) = \emptyset, \\ \iota_2[\{x' \otimes y' \mid x' \in \phi(a), y' \in \psi(a)\}] & \text{otherwise.} \end{cases}$$

3.4.2 Using (co)monads

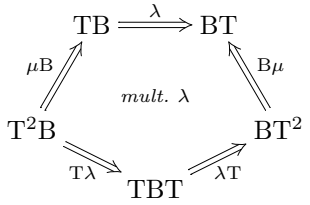
Employing free pointed functors enabled us to also allow rules with a single variable ranging over successor states as a target. In this section we extend the framework to rules with a target described by arbitrary terms in the signature over successor variables, i.e. we allow more than one operator application.

To motivate the development, we present an example of a specification in the setting of deterministic automata. Consider the sequential composition of such systems. It is defined by the following rules for all $a \in L$, which use the union operator specified in (3.20) on page 52.

$$\frac{x{\downarrow} \quad y{\downarrow}}{(x.y){\downarrow}} \qquad \frac{x{\uparrow} \quad x \xrightarrow{a} x_a}{x.y \xrightarrow{a} x_a.y} \qquad \frac{x{\downarrow} \quad x \xrightarrow{a} x_a \quad y \xrightarrow{a} y_a}{x.y \xrightarrow{a} (x_a.y) \cup y_a} \tag{3.23}$$

The last rule does not fit into any of the formats considered so far, because it declares a transition to a state described by the application of two operators: the sequential composition again and the union. As before, we shall try to extend our framework so that we can still capture this specification by a distributive law λ of an algebra functor over a coalgebra functor. In the previous section a similar situation led us to represent the algebras alternatively as algebras for a pointed functor. Here it turns out that we need to consider *algebras of a monad* from Section 2.2.2.

Definition 3.4.18 *A* **distributive law** *of a monad* $\langle \mathrm{T}, \eta, \mu \rangle$ *over a functor* B *is a natural transformation* $\lambda : \mathrm{TB} \Rightarrow \mathrm{BT}$ *satisfying the unit law from Def. 3.4.11 as well as the* multiplication law*, stating that the diagram below commutes.*



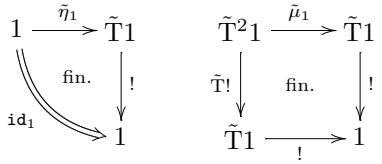
In this setting, we in addition require that for a λ*-bialgebra* $\langle P, \beta, \alpha \rangle$ *(cf. Definition 3.2.2) the algebra* $\langle P, \beta \rangle$ *is an algebra for the monad* $\langle \mathrm{T}, \eta, \mu \rangle$ *(cf. Definition 2.2.9).*

Corollary 3.4.19 *Theorem 3.2.3 still holds if we take* λ *to be a distributive law of the monad* $\langle \mathrm{T}, \eta, \mu \rangle$ *over the functor* B *and all algebras under consideration are assumed to be algebras for the monad* $\langle \mathrm{T}, \eta, \mu \rangle$*.*

The proof of the corollary is again based on extentions of Lemmata 3.2.4 and 3.2.5.

Lemma 3.4.20 *Let* $\langle \tilde{\mathrm{T}}, \tilde{\eta}, \tilde{\mu} \rangle$ *be a monad on a category* C*. There is a unique* $\tilde{\mathrm{T}}$*-algebra structure on a final object* 1 *of* C *and it yields a final algebra for the monad* $\langle \tilde{\mathrm{T}}, \tilde{\eta}, \tilde{\mu} \rangle$*.*

Proof: In addition to the proof of Lemma 3.2.4 we need to show that the final arrow $! : \tilde{\mathrm{T}}1 \to 1$ satisfies the unit and multiplication law (the dual of the former was actually shown already for Lemma 3.4.4). Both laws hold by finality.



□

Lemma 3.4.21 *Let λ be a distributive law of the monad $\langle \mathrm{T}, \eta, \mu \rangle$ over the functor* B. (i) *The monad $\langle \mathrm{T}, \eta, \mu \rangle$ lifts to a monad $\langle \mathrm{T}_\lambda, \tilde{\eta}, \tilde{\mu} \rangle$ on $\mathsf{Coalg}_{\mathrm{B}}$, and* (ii) *the functor* B *lifts to a functor B_λ on $\mathsf{Alg}^{\langle \mathrm{T}, \eta, \mu \rangle}$.*

Proof: We define T_λ as in Lemma 3.2.5. Dualising the proof of Lemma 3.4.5 we know already for (i) that η lifts to a natural transformation $\tilde{\eta}$ on $\mathsf{Coalg}_{\mathrm{B}}$ and for (ii) that $\mathrm{B}_\lambda\langle P, \beta \rangle$ satisfies the unit law in case the T-algebra $\langle P, \beta \rangle$ does. With the multiplication law for λ and the naturality of λ and μ we moreover get

- for (i) that also $\mu : \mathrm{T}^2 \Rightarrow \mathrm{T}$ lifts to a natural transformation $\tilde{\mu} : {\mathrm{T}_\lambda}^2 \Rightarrow \mathrm{T}_\lambda$ on $\mathsf{Coalg}_{\mathrm{B}}$, i.e. that $\tilde{\mu}_{\langle P, \alpha \rangle} := \mu_P$ for a B-coalgebra $\langle P, \alpha \rangle$ is a homomorphism from

  $$ {\mathrm{T}_\lambda}^2\langle P, \alpha \rangle = \langle \mathrm{T}^2 P, \lambda_{\mathrm{T}P} \circ \mathrm{T}\lambda_P \circ \mathrm{T}^2\alpha \rangle $$

  to

  $$ \mathrm{T}_\lambda\langle P, \alpha \rangle = \langle \mathrm{T}P, \lambda_P \circ \mathrm{T}\alpha \rangle, $$

  $$\begin{array}{ccc}
  \mathrm{T}P & \xleftarrow{\mu_P} & \mathrm{T}^2P \\
  {\scriptstyle \mathrm{T}\alpha}\downarrow & \text{nat. } \mu & \downarrow{\scriptstyle \mathrm{T}^2\alpha} \\
  \mathrm{TB}P & \xleftarrow{\mu_{\mathrm{B}P}} & \mathrm{T}^2\mathrm{B}P \\
  & & \downarrow{\scriptstyle \mathrm{T}\lambda_P} \\
  {\scriptstyle \lambda_P}\downarrow & \text{mult. } \lambda & \mathrm{TBT}P \\
  & & \downarrow{\scriptstyle \lambda_{\mathrm{T}P}} \\
  \mathrm{BT}P & \xleftarrow[\mathrm{B}\mu_P]{} & \mathrm{BT}^2P
  \end{array}$$

- and for (ii) that $\mathrm{B}_\lambda\langle P, \beta \rangle = \langle \mathrm{B}P, \mathrm{B}\beta \circ \lambda_P \rangle$ satisfies the multiplication law if $\langle P, \beta \rangle$ does.

$$\begin{array}{ccc}
\mathrm{T}^2P & \xrightarrow{\mu_P} & \mathrm{T}P \\
{\scriptstyle \mathrm{T}\beta}\downarrow & (*) & \downarrow{\scriptstyle \beta} \\
\mathrm{T}P & \xrightarrow[\beta]{} & P
\end{array}
\quad \Longrightarrow \quad
\begin{array}{ccccc}
\mathrm{T}^2\mathrm{B}P & & \xrightarrow{\mu_{\mathrm{B}P}} & & \mathrm{TB}P \\
{\scriptstyle \mathrm{T}\lambda_P}\downarrow & & \text{mult. } \lambda & & \downarrow{\scriptstyle \lambda_P} \\
\mathrm{TBT}P & \xrightarrow{\lambda_{\mathrm{T}P}} & \mathrm{BT}^2P & \xrightarrow{\mathrm{B}\mu_P} & \mathrm{BT}P \\
{\scriptstyle \mathrm{TB}\beta}\downarrow & \text{nat. } \lambda & {\scriptstyle \mathrm{BT}\beta}\downarrow & \mathrm{B}(*) & \downarrow{\scriptstyle \mathrm{B}\beta} \\
\mathrm{TB}P & \xrightarrow[\lambda_P]{} & \mathrm{BT}P & \xrightarrow[\mathrm{B}\beta]{} & \mathrm{B}P
\end{array}$$

□

Actually the converse of statement (ii) of the above lemma holds as well. More precisely, a distributive law of the monad $\langle \mathrm{T}, \eta, \mu \rangle$ over the functor B is *equivalent* to a lifting of the functor B to the category $\mathsf{Alg}^{\langle \mathrm{T}, \eta, \mu \rangle}$ of algebras for the

monad.[3] We can exploit this correspondence e.g. in order to prove Lemma 3.4.24 (i) below, but since it is available only in the cases that involve a monad (or dually a comonad), we will give a more elementary proof there which is similar to the arguments for the simpler cases above.

The result about bisimilarity as a congruence still holds in the setting of monads:

Corollary 3.4.22 *The statement of Theorem 3.2.6 still holds if we replace the functor* S *by the monad* $\langle \mathrm{T}, \eta, \mu \rangle$*, and correspondingly assume* λ *to satisfy the unit and multiplication law and all algebra structures under consideration to be algebras for the monad.*

In addition to the proof of Theorem 3.2.6 we need to show that the the T-algebra operation on the greatest bisimulation is an algebra for the monad. This follows from the lemma below.

Lemma 3.4.23 *Let* $\langle P, \beta_P \rangle$ *and* $\langle Q, \beta_Q \rangle$ *be algebras for the monad* $\langle \mathrm{T}, \eta, \mu \rangle$*. If* $R \subseteq P \times Q$ *is a congruence between* $\langle P, \beta_P \rangle$ *and* $\langle Q, \beta_Q \rangle$ *viewed as plain* T*-algebras, then it is also a congruence between them viewed as algebras for the monad.*

Proof: In addition to the proof of Lemma 3.4.14 we need to show that the algebra structure γ on R witnessing the congruence property satisfies the multiplication law. The argument is similar to the one for the unit law: using the naturality of μ and the multiplication laws for β_P and β_Q one easily shows that both $\gamma \circ \mu_R$ and $\gamma \circ \mathrm{T}\gamma$ make the two squares of the diagram diagram below commute; this implies that they are equal, since the span $\langle R, \pi_1, \pi_2 \rangle$ is jointly monic.

$$\begin{array}{ccccc}
\mathrm{T}^2P & \xleftarrow{\mathrm{T}^2\pi_1} & \mathrm{T}^2R & \xrightarrow{\mathrm{T}^2\pi_2} & \mathrm{T}^2Q \\
{\scriptstyle \mathrm{T}\beta_P}\downarrow & & \vdots & & \downarrow{\scriptstyle \mathrm{T}\beta_Q} \\
\mathrm{T}P & & \vdots\ ? & & \mathrm{T}Q \\
{\scriptstyle \beta_P}\downarrow & & \vdots & & \downarrow{\scriptstyle \beta_Q} \\
P & \xleftarrow[\pi_1]{} & R & \xrightarrow[\pi_2]{} & Q
\end{array}$$

□

[3] To obtain a distributive law $\lambda^{\bar{\mathrm{B}}}$ from a lifting $\bar{\mathrm{B}} : \mathsf{Alg}^{\langle \mathrm{T}, \eta, \mu \rangle} \to \mathsf{Alg}^{\langle \mathrm{T}, \eta, \mu \rangle}$ of the functor B, we set

$$\lambda^{\bar{\mathrm{B}}}_X := b(\mu_X) \circ \mathrm{TB}\eta_X : \mathrm{TB}X \to \mathrm{BT}X$$

where $b(\mu_X) : \mathrm{TBT}X \to \mathrm{BT}X$ is the T-algebra structure with $\bar{\mathrm{B}}\langle \mathrm{T}X, \mu_X \rangle = \langle \mathrm{BT}X, b(\mu_X) \rangle$. It is not too difficult to check that this defines a natural transformation indeed which satisfies the unit and multiplication law, and that moreover the construction of the lifting B_λ from the above proof and this one are inverses of each other.

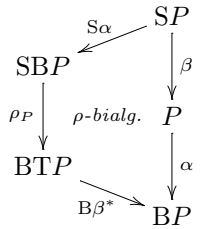
Free monads

Similar to the approaches involving (co)pointed functors, we will choose the monad $\langle \mathrm{T}, \eta, \mu \rangle$ such that the plain S-algebras correspond to algebras for the monad. As stated in Lemma 2.2.10, this correspondence holds if we take the *free monad* generated by S from Definition 2.2.7. Specifications will now be the distributive laws λ of the free monad generated by S over the behaviour functor B. As in the case of a pointed functor such a distributive law is generated by a plain natural transformation ρ of a simpler type and we can express the condition on a λ-bialgebra directly in terms of ρ.

Lemma 3.4.24 *Let* S *and* B *be functors such that* S *generates the free monad* $\langle \mathrm{T}, \eta, \mu \rangle$ *and let* $\langle P, \beta, \alpha \rangle$ *be an* $\langle \mathrm{S}, \mathrm{B} \rangle$*-bialgebra.*

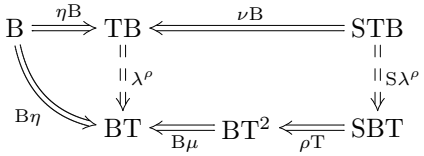
(i) *Distributive laws* λ *of the monad* $\langle \mathrm{T}, \eta, \mu \rangle$ *over the functor* B *are in one-to-one correspondence with (plain) natural transformations* $\rho : \mathrm{SB} \Rightarrow \mathrm{BT}$*, and,*

(ii) *with* β^* *being the extension of* β *to an algebra for the monad* $\langle \mathrm{T}, \eta, \mu \rangle$ *according to Lemma 2.2.10,* $\langle P, \beta^*, \alpha \rangle$ *is a* λ*-bialgebra if and only if* $\langle P, \beta, \alpha \rangle$ *is a* ρ*-bialgebra for* ρ *as above, where this is meant to say that the diagram below commutes.*



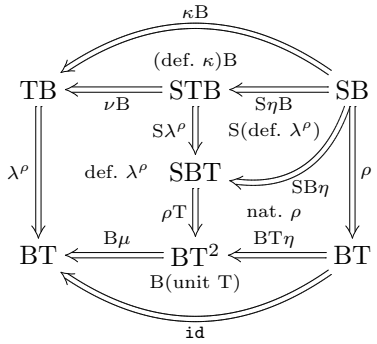
Proof: Let $\nu : \mathrm{ST} \Rightarrow \mathrm{T}$ and $\kappa : \mathrm{S} \Rightarrow \mathrm{T}$ be the two natural transformations from Definition 2.2.7.

For item (i) the correspondence is given by $\lambda \mapsto \rho^\lambda$ and $\rho \mapsto \lambda^\rho$ where $\rho^\lambda := \lambda \circ \kappa\mathrm{B}$ and λ^ρ is the unique natural transformation fitting in the diagram below (cf. the definition principle in (2.4) on page 22).

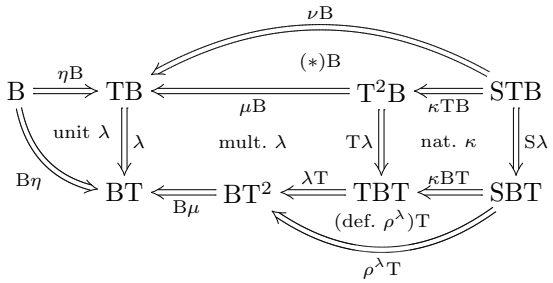


Both constructions are inverse to each other: $\rho^{\lambda^\rho} := \lambda^\rho \circ \kappa\mathrm{B} = \rho$ follows from the following diagram chase, which employs the naturality or ρ and one unit

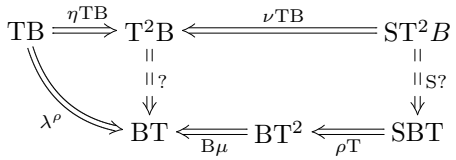
law of the monad.



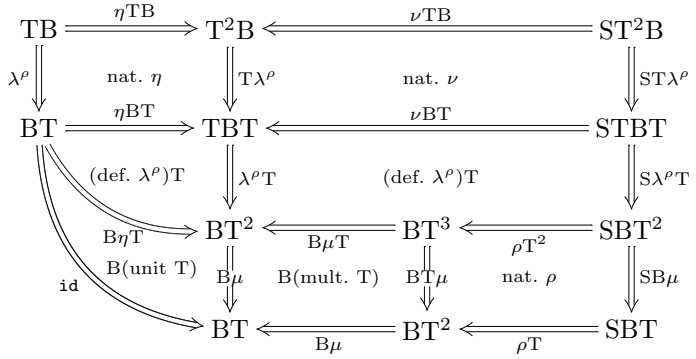
To see $\lambda^{\rho^\lambda} = \lambda$, the diagram below shows that λ itself satisfies the defining property of λ^{ρ^λ}. It uses the unit and multiplication law for λ, the naturality of κ, and Lemma 2.2.8 (referred to by $(*)$ in the diagram below).



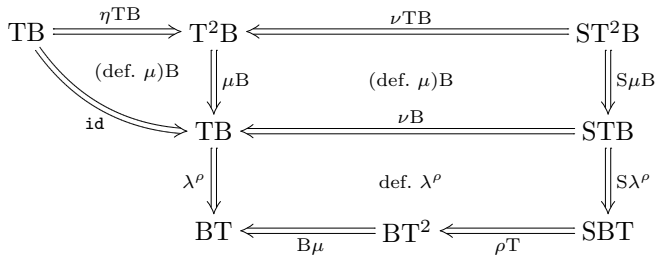
Moreover, we need to check that λ^ρ defined as above satisfies the unit and multiplication law. The unit law holds by definition. For the multiplication law we shall show that both $\lambda^\rho \circ \mu\mathrm{B}$ and $\mathrm{B}\mu \circ \lambda^\rho\mathrm{T} \circ \mathrm{T}\lambda^\rho$ fit as the unique arrow into this inductive definition:



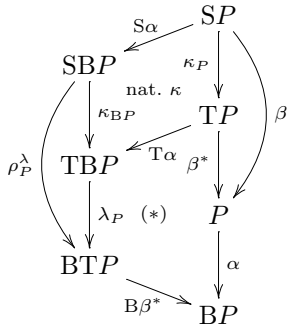
For $\mathrm{B}\mu \circ \lambda^\rho\mathrm{T} \circ \mathrm{T}\lambda^\rho$, the situation is as follows.



And for $\lambda^\rho \circ \mu\mathrm{B}$ we calculate as below.

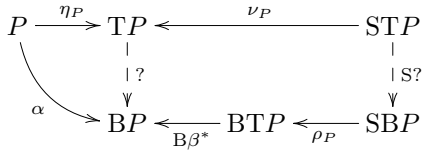


Next we prove item (ii). For one direction assume that $\langle P, \beta^*, \alpha\rangle$ is a λ-bialgebra (referred to by $(*)$ in the diagram below). Using naturality of κ it easily follows that $\langle P, \beta, \alpha\rangle$ is a ρ^λ-bialgebra:

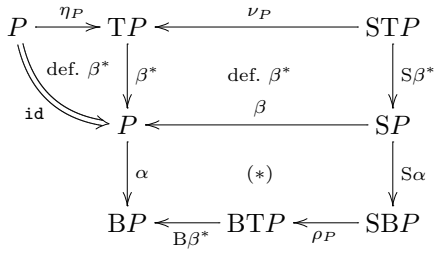


For the other direction assume that $\langle P, \beta, \alpha\rangle$ is a ρ-bialgebra. We obtain that $\langle P, \beta^*, \alpha\rangle$ is a λ^ρ-bialgebra, because, as we shall see, both $\alpha \circ \beta^*$ and $\mathrm{B}\beta^* \circ \lambda^\rho_P \circ \mathrm{T}\alpha$ are uniquely determined (cf. the free induction principle in (2.3) on page

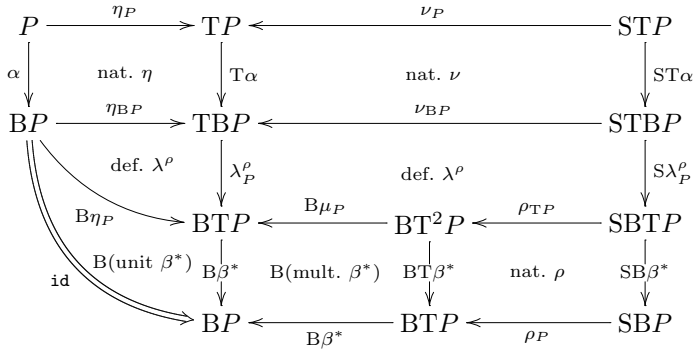
22) by the same inductive characterisation below.



The diagram below shows the situation for $\alpha \circ \beta^*$, the part $(*)$ of which commutes by our assumption on $\langle P, \beta, \alpha \rangle$ being a ρ-bialgebra.



The next diagram shows that $\mathrm{B}\beta^* \circ \lambda^\rho_P \circ \mathrm{T}\alpha$ fits as well:



□

With the above development, we know that we can take natural transformations $\rho : \mathrm{SB} \Rightarrow \mathrm{BT}$ as well-behaved specifications with ρ-bialgebras as their models, where T is the functor of the free monad generated by the algebra functor S. Such natural transformations correspond to rule formats that allow terms of the signature under consideration in the targets of the rules. But the resulting format does not yet allow the transition rules in (3.23) specifying the sequential composition, because the terms in the targets mention not only successors of the variables in the source but again those variables themselves. A similar situation led us to study copointed functors in Section 3.4.1 and we will only be able to cover the specification of the sequential composition if we combine both approaches. We shall do so in the next section, but before we mention that the use of monads can be dualised of course:

In case the behaviour functor B generates a cofree comonad $\langle \mathrm{D}, \varepsilon, \delta \rangle$ we can use natural transformations $\rho : \mathrm{SD} \Rightarrow \mathrm{BS}$ as well-behaved specifications with ρ-bialgebras as models, where the latter are defined in the apparent way. This format allows *lookahead*, i.e. we can use chains of transitions in the premises.

3.5 Abstract GSOS as a mixed format

In the previous section we have extended our use of distributive laws between plain functors as specifications in Section 3.2 to settings where the algebra functor comes as a pointed functor or monad, and dually for coalgebra functors taken from copointed functors or comonads. This involved distributive laws satisfying extra coherence axioms. For (co)free (co)pointed functors or (co)monads we then demonstrated that such distributive laws are equivalent to plain natural transformations of a different type. This led to more expressive specification formats. It is easy to see that the addition of the extra structure on the algebra side is independent from that on the coalgebra side, so we can for instance consider distributive laws of a pointed functor over a comonad. Combining the proofs of Theorem 3.2.3 and Corollaries 3.4.3 and 3.4.19 (as well as their duals) we for instance get the following more general statement.

Corollary 3.5.1 *Let λ be a distributive law of a functor* S*, pointed functor $\langle \mathrm{T}, \eta \rangle$, or monad $\langle \mathrm{T}, \eta, \mu \rangle$ over a functor* B*, copointed functor $\langle \mathrm{D}, \varepsilon \rangle$, or comonad $\langle \mathrm{D}, \varepsilon, \delta \rangle$. An initial algebra of* S*, $\langle \mathrm{T}, \eta \rangle$, or $\langle \mathrm{T}, \eta, \mu \rangle$ respectively extends uniquely to a λ-bialgebra and this λ-bialgebra is initial. Dually, a final coalgebra of* B*, $\langle \mathrm{D}, \varepsilon \rangle$, or $\langle \mathrm{D}, \varepsilon, \delta \rangle$ respectively extends uniquely to a λ-bialgebra and this λ-bialgebra is final.*

When we again use (co)free (co)pointed functors or (co)monads in such a mixed setting, however, we have to check whether we can still explain the corresponding distributive laws in terms of plain natural transformations. This turns out to be possible in all cases but for the most complex one, namely that of a distributive law of a free monad over a cofree comonad. We summarise the results in the table in Figure 3.5. In this presentation, S and B are functors, S generates the free pointed functor $\langle \tilde{\mathrm{T}}, \tilde{\eta} \rangle$ or the free monad $\langle \mathrm{T}, \eta, \mu \rangle$, and B generates the cofree copointed functor $\langle \tilde{\mathrm{D}}, \tilde{\varepsilon} \rangle$ or the cofree comonad $\langle \mathrm{D}, \varepsilon, \delta \rangle$. A natural transformation ρ of the type listed in any of the fields of the table is a representation of a distributive law λ between the respective structures.

For the upper left entry of the table in Figure 3.5, i.e. for distributive laws between plain functors, there is nothing to show. For the remaining entries in the upper row and the left column we have shown the correspondence of natural transformations ρ of the given type and distributive laws between the respective structures already in the Lemmata 3.4.10 and 3.4.24 and their duals.

The other three filled entries of the table in Figure 3.5 are not proved yet, namely those marked by (a), (b), and (c). In Lemma 3.5.2 further below we will

λ	functor S	pointed functor $\langle \tilde{\mathrm{T}}, \tilde{\eta} \rangle$	monad $\langle \mathrm{T}, \eta, \mu \rangle$
functor B	$\lambda : \mathrm{SB} \Rightarrow \mathrm{BS}$ (immediate)	$\rho : \mathrm{SB} \Rightarrow \mathrm{B}\tilde{\mathrm{T}}$ (L. 3.4.10 dual)	$\rho : \mathrm{SB} \Rightarrow \mathrm{BT}$ (Lemma 3.4.24)
copointed functor $\langle \tilde{\mathrm{D}}, \tilde{\varepsilon} \rangle$	$\rho : \mathrm{S}\tilde{\mathrm{D}} \Rightarrow \mathrm{BS}$ (Lemma 3.4.10)	$\rho : \mathrm{S}\tilde{\mathrm{D}} \overset{(a)}{\Rightarrow} \mathrm{B}\tilde{\mathrm{T}}$ (without proof)	$\rho : \mathrm{S}\tilde{\mathrm{D}} \overset{(b)}{\Rightarrow} \mathrm{BT}$ (Lemma 3.5.2)
comonad $\langle \mathrm{D}, \varepsilon, \delta \rangle$	$\rho : \mathrm{SD} \Rightarrow \mathrm{BS}$ (L. 3.4.24 dual)	$\rho : \mathrm{SD} \overset{(c)}{\Rightarrow} \mathrm{B}\tilde{\mathrm{T}}$ (L. 3.5.2 dual)	unknown

Figure 3.2: Representation of distributive laws between (co)free structures.

prove the correspondence (b) involving distributive laws of a free monad over a cofree copointed functor. A dual argument proves the correspondence marked by (c). Since the statement for (a) is easier, we leave out the proof.

We do not know how to resolve all the coherence axioms for a distributive law of the free monad $\langle \mathrm{T}, \eta, \mu \rangle$ over the cofree comonad $\langle \mathrm{D}, \varepsilon, \delta \rangle$ in general, so we cannot provide a natural transformation for the lower right field of the table. Looking at the other cases, one may try natural transformations

$$\rho : \mathrm{SD} \Rightarrow \mathrm{BT} \tag{3.24}$$

as a candidate. But they turn out to be too liberal. To see this, note that a natural transformations of the type in (3.24) can capture rules with lookahead and arbitrary terms in the target. Consider the following rule for a unary operator σ on streams of real numbers (cf. Sections 2.3.3 and 3.3.2) as an example:

$$\frac{s \xrightarrow{u} s' \quad s' \xrightarrow{v} s''}{\sigma(s) \xrightarrow{u+v} \sigma(\sigma(s''))}$$

This rule gives rise to a natural transformation ρ as in (3.24). Assume that ρ would correspond to a distributive law λ of the free monad generated by the signature of the operator symbol σ over the cofree comonad generated by the behaviour functor $\mathrm{B} = \underline{\mathbb{R}} \times \mathrm{Id}$ for stream systems. The final λ-bialgebra would provide a unique interpretation of σ on streams of real numbers satisfying the rule. But such an interpretation does not exist, as one can easily see trying to compute, for instance, the result of $\sigma(\langle 1, 1, 1, \ldots \rangle)$.

Note that with the above we are not arguing that the case of a free monad and cofree comonad is not meaningful. On the contrary, it is the most expressive and thus the most interesting one. We simply do not have an easy way to deal with it in general.

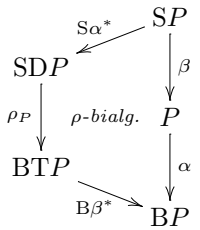
We now turn back to the field marked by (b) in the table in Figure 3.5. Natural transformations of the type (b) are the most important ones in Turi and

Plotkin's paper [TP97] as well as in the subsequent literature (see e.g. [Tur97, LPW00, Bar03, Wat02]). For labelled transition systems, they correspond to specifications in the well known GSOS format [BIM95].

Lemma 3.5.2 *Let* $\mathrm{S}, \mathrm{B} : \mathsf{Set} \to \mathsf{Set}$ *be functors, let* $\langle \mathrm{T}, \eta, \mu \rangle$ *be the free monad generated by* S*, and let* $\langle \mathrm{D}, \varepsilon \rangle$ *be the cofree copointed functor generated by* B*.*

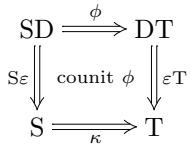
(i) *Distributive laws* λ *of the free monad* $\langle \mathrm{T}, \eta, \mu \rangle$ *over the cofree copointed functor* $\langle \mathrm{D}, \varepsilon \rangle$ *correspond to (plain) natural transformations* $\rho : \mathrm{SD} \Rightarrow \mathrm{BT}$*.*

(ii) *Let* $\langle P, \beta, \alpha \rangle$ *be an* $\langle \mathrm{S}, \mathrm{B} \rangle$*-bialgebra. For* λ *and* ρ *corresponding to each other as above, with* β^* *being the* T*-algebra corresponding to* β *(cf. Lemma 2.2.10), and* α^* *being the* D*-coalgebra corresponding to* α *(cf. Corollary 3.4.8 (ii)), we have that* $\langle P, \beta^*, \alpha^* \rangle$ *is a* λ*-bialgebra if and only if* $\langle P, \beta, \alpha \rangle$ *is a* ρ*-bialgebra. By the latter we mean that the diagram below commutes.*

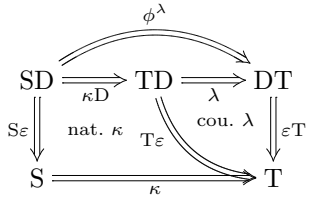


Proof: Let $\kappa : \mathrm{S} \Rightarrow \mathrm{T}$, $\nu : \mathrm{ST} \Rightarrow \mathrm{T}$ and $\vartheta : \mathrm{D} \Rightarrow \mathrm{B}$ be the natural transformations from Definitions 2.2.7 and 3.4.7.

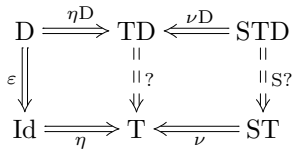
For the proof of item (i) we know from Lemma 3.4.24 that distributive laws λ of the monad $\langle \mathrm{T}, \eta, \mu \rangle$ over the *functor* D, i.e. natural transformations $\lambda : \mathrm{TD} \Rightarrow \mathrm{DT}$ satisfying the unit and multiplication law, correspond to plain natural transformations $\phi : \mathrm{SD} \Rightarrow \mathrm{DT}$ (where we write the correspondence as $\lambda \mapsto \phi^\lambda$ and $\phi \mapsto \lambda^\phi$). We show that λ satisfies the counit law if and only if the corresponding ϕ satisfies the following counit law.



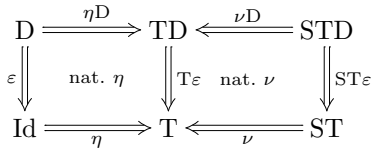
It follows from the naturality of κ that ϕ^λ satisfies this law if λ satisfies the respective counit law, as the following diagram shows.



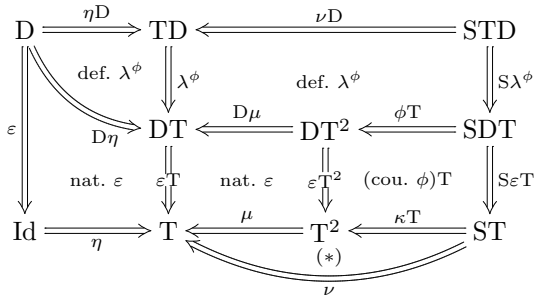
We prove that λ^ϕ satisfies the counit law if ϕ does by showing that both $\mathrm{T}\varepsilon$ and $\varepsilon\mathrm{T} \circ \lambda^\phi$ are uniquely determined (cf. the principle from (2.4) on page 22) by the same inductive characterisation below.



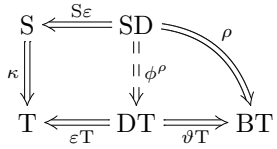
For $\mathrm{T}\varepsilon$ this follows from the naturality of η and ν, as shown below.



It is shown by the following diagram that $\varepsilon\mathrm{T} \circ \lambda^\phi$ satisfies the characterisation as well. Its part $(*)$ commutes by Lemma 2.2.8, the others by definition, the naturality of ε, and the above counit law for ϕ.



So we obtain a one-to-one correspondence between distributive laws λ of the monad $\langle \mathrm{T}, \eta, \mu \rangle$ over the copointed functor $\langle \mathrm{D}, \varepsilon \rangle$ and natural transformations $\phi : \mathrm{SD} \Rightarrow \mathrm{DT}$ satisfying the counit law above. As a second step, we show that the latter correspond to plain natural transformations ρ as in the statement. This correspondence is given by the maps $\phi \mapsto \rho^\phi$ and $\rho \mapsto \phi^\rho$ where $\rho^\phi := \vartheta\mathrm{T} \circ \phi$ and ϕ^ρ is the (by Corollary 3.4.8 (i)) unique natural transformation fitting into the diagram below.



It is easy to see that these constructions yield the desired correspondence.

The proof of item (ii) is again split into showing two correspondences, where the type of model in the middle is formulated using ϕ. Both parts correspond to the respective parts of the proofs of Lemma 3.4.24 and Lemma 3.4.10.

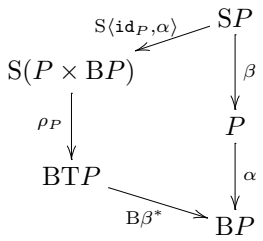
□

Spelling out $\mathrm{D} = \mathrm{Id} \times \mathrm{B}$ and $\alpha^* = \langle \mathtt{id}, \alpha \rangle$, we obtain the following abstract format:

Definition 3.5.3 *Given a signature functor* S *generating the monad* $\langle \mathrm{T}, \eta, \mu \rangle$*, and given a behaviour functor* B*, we define a specification in* **abstract GSOS** *to be a natural transformation*

$$\rho : \mathrm{S}(\mathrm{Id} \times \mathrm{B}) \Rightarrow \mathrm{BT}.$$

A **model** *of* ρ *is an* $\langle \mathrm{S}, \mathrm{B} \rangle$*-bialgebra* $\langle P, \beta, \alpha \rangle$ *making the diagram below commute, where* β^* *is the inductive extension of* β *(cf. Lemma 2.2.10).*



Natural transformations ρ are called specifications in *abstract GSOS* by Turi and Plotkin [TP97], because – as we shall see in the next section – the instantiation of the format with labelled transition systems leads to transition rules of a type corresponding to the GSOS format.

The models of specifications in abstract GSOS are well-behaved in the sense that with Lemma 3.5.2 we can apply Corollary 3.5.1 and a corresponding variant of Theorem 3.2.6 to get the following properties.

Corollary 3.5.4 *Let* ρ *be a specification in abstract GSOS. An initial* S*-algebra can be extended in a unique way to a model of* ρ*, and this model is initial. Dually, a final* B*-coalgebra, if it exists, can be extended in a unique way to a model of* ρ *and this model is final.*

Corollary 3.5.5 *Bisimilarity between two models of a specification in abstract GSOS is a congruence.*

With this format we are now able to cover the sequential composition of automata as specified in (3.23) (using the union from (3.20) from page 52). The signature Σ contains two binary operations, the sequential composition and union operator. Let S be the functor associated to this signature. The rules give rise to the following specification ρ in abstract GSOS: the natural transformation

$$\rho : \mathrm{S}(\mathrm{Id} \times (\underline{2} \times \mathrm{Id}^L)) \Rightarrow \underline{2} \times \mathrm{T}^L$$

is defined for all sets X; $x, y \in X$; $b, c \in 2$; and $\phi, \psi \in X^L$ as

$$\rho_X(\langle x, \langle b, \phi\rangle\rangle \cup \langle y, \langle c, \psi\rangle\rangle) \;:=\; \langle b \vee c, a \mapsto \phi(a) \cup \psi(a)\rangle$$

$$\rho_X(\langle x, \langle b, \phi\rangle\rangle.\langle y, \langle c, \psi\rangle\rangle) \;:=\; \langle b \wedge c, a \mapsto \begin{cases} (\phi(a).y) \cup \psi(a) & \text{if } b = \top, \\ \phi(a).y & \text{otherwise.} \end{cases}\rangle$$

3.5.1 Concrete rule formats derived from abstract GSOS

In Section 3.3 we analysed distributive laws of a functor S derived from a signature Σ over a behaviour functor B for a few important system types, i.e. for several instances of B. We are now going to do the same for specifications in abstract GSOS, which is to say natural transformations $\rho : \mathrm{S}(\mathrm{Id} \times \mathrm{B}) \Rightarrow \mathrm{BT}$, where T is the functor from the term monad over S. The derivation here is slightly more complicated but still similar to the one in Section 3.3, where we considered the natural transformations $\lambda : \mathrm{SB} \Rightarrow \mathrm{BS}$. Therefore our explanations will be rather brief this time. We just make a general remark on the difference between the two settings:

- The extra occurrence of $\mathrm{Id} \times \ldots$ in the source of the natural transformation is essentially handled by Lemma A.1.7, which states that natural transformations of the type $\mathrm{Id}^A \times \mathrm{F} \Rightarrow \mathrm{G}$ for some set A correspond to those of the type $\mathrm{F} \Rightarrow \mathrm{G}(\underline{\mathrm{A}} + \mathrm{Id})$.
- Since the functor S is associated to a *finitary* signature Σ, i.e. all operator symbols have a finite arity, the functors T and S are similar in structure: T can be viewed to arise from a signature as well, more precisely we have

$$\mathrm{T} \simeq \coprod_{t \in \mathrm{T}1} \mathrm{Id}^{|t|_*}, \tag{3.25}$$

  where $|t|_*$ denotes the number of occurrences of the one variable $*$ in t. As a result, the replacement of the functor S by T in the target of the natural transformation does not bring any extra complication.

LTS

Modifying the development in Section 3.3.1 as indicated above, we get that specifications in abstract GSOS instantiated for labelled transition systems can be characterised as follows:

Corollary 3.5.6 *Any natural transformation*

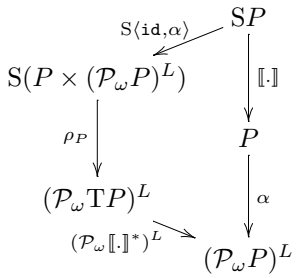
$$\rho : \mathrm{S}(\mathrm{Id} \times {\mathcal{P}_\omega}^{\mathrm{L}}) \Rightarrow (\mathcal{P}_\omega \mathrm{T})^L$$

corresponds to an image finite (where this has the same meaning as in Theorem 3.3.1) set of derivation rules of the shape

$$\frac{\begin{array}{ll}\theta_i(b) \neq \emptyset & 1 \leq i \leq n, b \in R_i \\ \theta_i(c) = \emptyset & 1 \leq i \leq n, c \in P_i \\ y_j \in \theta_{i_j}(l_j) & 1 \leq j \leq k\end{array}}{t \in \rho(\sigma(\langle x_1, \theta_1\rangle, \ldots, \langle x_n, \theta_n\rangle))(a)}$$

where $a \in L$; $\sigma \in \Sigma$ with arity n; $R_i, P_i \subseteq L$ such that $R_i \cap P_i = \emptyset$ $(1 \leq i \leq n)$; $k \in \mathbb{N}$; $1 \leq i_j \leq n$, $l_j \in R_{i_j}$ $(1 \leq j \leq k)$; $t \in \mathrm{T}(X + Y)$ with $X = \{x_1, \ldots, x_n\}$ and $Y = \{y_1, \ldots, y_k\}$.

To obtain a direct characterisation of the models of a specification ρ in abstract GSOS, first recall that those are the $\langle \mathrm{S}, \mathrm{B}\rangle$-bialgebras $\langle P, [\![.]\!], \alpha\rangle$ for $\mathrm{B} = {\mathcal{P}_\omega}^L$ which make the following diagram commute, where $[\![.]\!]^*$ is the inductive extension of $[\![.]\!]$ to terms:



In this setting, the specification ρ is applied to arguments of the type

$$\sigma(\langle p_1, \alpha(p_1)\rangle, \ldots, \langle p_n, \alpha(p_n)\rangle)$$

for $\sigma(p_1, \ldots, p_n) \in \mathrm{S}P$. Substituting $\alpha(p_i)$ for θ_i in the rules in Corollary 3.5.6, we can rewrite the premises using the arrow notation from Definition 2.3.12. As for the conclusion, note that with the diagram above

$$t \in \rho_P(\sigma(\langle p_1, \alpha(p_1)\rangle, \ldots, \langle p_n, \alpha(p_n)\rangle))(a)$$

implies $[\![t]\!]^* \in \alpha([\![\sigma(p_1, \ldots, p_n)]\!])(a)$, or, with the arrow notation,

$$[\![\sigma(p_1, \ldots, p_n)]\!] \xrightarrow{a} [\![t]\!]^*.$$

Taken together, we can characterise the models of any specification in abstract GSOS instantiated for labelled transition systems by an image finite set of rules of the following shape:

$$\frac{\begin{array}{ll}x_i \xrightarrow{b} & 1 \leq i \leq n, b \in R_i \\ x_i \xrightarrow{c} \!\!\!\!\rightarrow & 1 \leq i \leq n, c \in P_i \\ x_{q_j} \xrightarrow{a_j} y_j & 1 \leq j \leq k\end{array}}{\sigma(x_1, \ldots, x_n) \xrightarrow{a} t}$$

We leave the application of the operator interpretations $[\![.]\!]$ and $[\![.]\!]^*$ implicit.

These rules are a generalisation of the GSOS rules of Bloom, Istrail, and Meyer [BIM95] to possibly infinite sets of labels. This extension led to the introduction of positive premises without a variable for the successor state (i.e. the uppermost type of premises), which are not present in the original format. In principle, every premise of this type can be replaced by a transition leading to a successor state variable (i.e. one of the bottommost type) which is not used in the target t. But with infinite sets of labels this forces us to allow infinitely many successor variables. At this point it is actually not necessary to keep their number finite, but the situation will change when we turn to probabilistic systems in Chapter 5.

The corresponding instances of our statements about the abstract GSOS framework, i.e. Corollaries 3.5.4 and 3.5.5, yield that every GSOS specification has an initial and final model and that bisimilarity is a congruence on every model. These are known properties of the format. In Chapter 4 we shall moreover obtain new well-behavedness results for the GSOS format from the abstract framework.

Streams

A specification in abstract GSOS instantiated with the stream functor $\mathrm{B} = \underline{\mathbb{R}} \times \mathrm{Id}$ is a natural transformation

$$\rho : \mathrm{S}(\mathrm{Id} \times (\underline{\mathbb{R}} \times \mathrm{Id})) \Rightarrow \underline{\mathbb{R}} \times \mathrm{T}$$

The models of such a specification can be characterised by a set of rules determining for each $\sigma \in \Sigma$ (with arity n) and $u_1, \ldots, u_n \in \mathbb{R}$ a derivation of the shape

$$\frac{x_i \xrightarrow{u_i} y_i \quad (1 \leq i \leq n)}{\sigma(x_1, \ldots, x_n) \xrightarrow{v} t}$$

for some $v \in \mathbb{R}$ and $t \in \mathrm{T}(X + Y)$, where $X := \{x_1, \ldots, x_n\}$ and $Y := \{y_1, \ldots, y_n\}$.

As an example, notice that most of the operators Rutten considers for his stream calculus [Rut01, Rut00a] fit into this format.[4] We recall the specification of the sum $+$ and shuffle product $\otimes$, which are both binary operators on streams of

[4] The only exception is the operator Λ_c [Rut01, page 29]. Its definition is equivalent to the rule

$$\frac{x \xrightarrow{u} x' \quad x' \xrightarrow{v} x''}{\Lambda_c(x) \xrightarrow{u} \Lambda_c(x' + X \otimes x'')}$$

which uses lookahead. Therefore the rule does not fit into the abstract GSOS format. It can be shown, however, that it can be covered by a distributive law of the free monad over the cofree comonad. This means that it is well-behaved in all respects studied by Turi and Plotkin [TP97].

real numbers (the first one was already shown in Section 3.3.2). They are given by the following sets of rules:

$$\frac{x \xrightarrow{u} x' \quad y \xrightarrow{v} y'}{x + y \xrightarrow{u+v} x' + y'} (u, v \in \mathbb{R}) \tag{3.26}$$

$$\frac{x \xrightarrow{u} x' \quad y \xrightarrow{v} y'}{x \otimes y \xrightarrow{u \cdot v} (x' \otimes y) + (x \otimes y')} (u, v \in \mathbb{R}) \tag{3.27}$$

The sum operator was already specifiable in the simple format, but the shuffle product uses the greater expressiveness of abstract GSOS: the rules declare transitions to a state described by more than one application of an operator and this description moreover uses both the variables in the source (x and y) as well as those for their successors (x' and y').

Automata

Instantiated with the behaviour functor $\mathrm{B} = \underline{2} \times \mathrm{Id}^L$ for deterministic automata, specifications in abstract GSOS become natural transformations

$$\rho : \mathrm{S}(\underbrace{\mathrm{Id} \times (\underline{2} \times \mathrm{Id}^L)}_{\simeq \underline{2} \times \mathrm{Id}^{L+1}}) \Rightarrow \underline{2} \times \mathrm{T}^L,$$

which are equivalent to families of functions

$$\left(h^\sigma : 2^{|\sigma|} \to 2 \times (\mathrm{T}(\{1, \ldots, |\sigma|\} \times (L + 1)))^L\right)_{\sigma \in \Sigma}$$

Translated into a characterisation of the models, we obtain the following rule format: a specification again contains sets of rules of two types, namely

$$\frac{x_i{\downarrow}\ (i \in P) \quad x_i{\uparrow}\ (i \in N)}{\sigma(x_1, \ldots, x_n){\downarrow}}$$

and

$$\frac{x_i \xrightarrow{b} y_i^l\ (1 \le i \le n, l \in L) \quad x_i{\downarrow}\ (i \in P) \quad x_i{\uparrow}\ (i \in N)}{\sigma(x_1, \ldots, x_n) \xrightarrow{a} t}$$

where $\sigma \in \Sigma$ with arity n; $P, N \subseteq \{1, \ldots, n\}$ with $P \cap N = \emptyset$; $a \in L$; and $t \in \mathrm{T}(X + Y)$, where $X := \{x_1, \ldots, x_n\}$ and $Y := \{y_i^l \mid 1 \le i \le n,\ l \in L\}$ (we again omit a premise $x_i \xrightarrow{b} y_i^l$ if y_i^l does not occur in t). For each $\sigma \in \Sigma$, $a \in L$, and $A \subseteq \{1, \ldots, n\}$ there should be exactly one rule of the second type with this σ and a such that $P \subseteq A$ and $A \cap N = \emptyset$.

We have already argued that the specification of the sequential composition by the rules (3.23) on page 64 fits into the abstract GSOS format, and we indeed

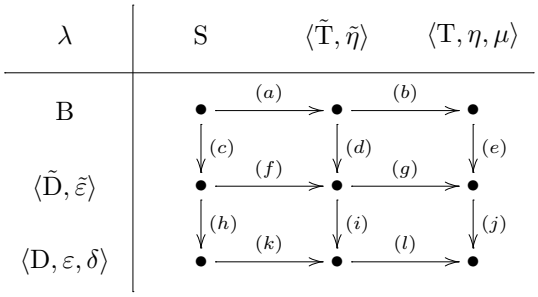


Figure 3.3: Distributive laws between all combinations of cofree structures.

find that the given rules are in the shape above. As another example, we define the Kleene star operation as follows:

$$\frac{}{(x^*)\!\downarrow} \qquad \frac{x \xrightarrow{a} x_a}{x^* \xrightarrow{a} x_a.(x^*)}$$

3.6 Comparing the different formats

Given a signature Σ with associated functor S and a behaviour functor B we have specified classes of $\langle \mathrm{S}, \mathrm{B}\rangle$-bialgebras by distributive laws λ of S over B. Driven by examples that did not fit into this simple framework, we later also considered distributive laws between functors with additional structure, namely the free pointed functor $\langle \tilde{\mathrm{T}}, \tilde{\eta}\rangle$ or monad $\langle \mathrm{T}, \eta, \mu\rangle$ over S and the cofree copointed functor $\langle \tilde{\mathrm{D}}, \tilde{\varepsilon}\rangle$ or comonad $\langle \mathrm{D}, \varepsilon, \delta\rangle$ over B, provided they existed. This way we could indeed capture the specifications that did not fit into the simple setting.

Taking all possible combinations of these structures, we obtain distributive laws of nine kinds, depicted by bullets in the table in Figure 3.3. In this section, we show formally that the addition of the (co)free constructions extends the expressiveness of the approach. To this end we argue that a distributive law λ of any of the types can be turned into a distributive law λ^* of a type positioned below or to the right of it (or both) in the table, such that λ and λ^* have the same models. We do so by establishing liftings along the arrows (a) through (l) in Figure 3.3. We will not prove all possible cases in all detail here, but we sketch the argument.

Lemma 3.6.1 *(i) A distributive law λ of a functor* S *over a functor* B *lifts to a distributive law λ^* of the functor* S *over the cofree copointed functor $\langle \tilde{\mathrm{D}}, \tilde{\varepsilon}\rangle$ generated by* B *(cf. Def. 3.4.7) such that an $\langle \mathrm{S}, \mathrm{B}\rangle$-bialgebra $\langle P, \beta, \alpha\rangle$ is a λ-bialgebra if and only if $\langle P, \beta, \tilde{\alpha}^*\rangle$ is a λ^*-bialgebra, where $\tilde{\alpha}^*$ is the extension of*

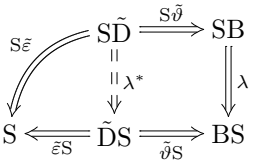
α to a coalgebra structure for the copointed functor (cf. Corollary 3.4.8 (ii)).

(ii) The statement still holds if we replace S *by any pointed functor* $\langle \tilde{\mathrm{T}}, \tilde{\eta} \rangle$ *or monad* $\langle \mathrm{T}, \eta, \mu \rangle$.

(iii) The above statements dualise.

Item (i) of the above lemma corresponds to arrow (c) in Figure 3.3, and item (ii) covers arrows (d) and (e). With item (iii) we moreover obtain the dual arrows (a), (f) and (k).

Proof: We define λ^* to be the (by Corollary 3.4.8 (i)) unique natural transformation fitting into the diagram below, where $\tilde{\vartheta} : \tilde{\mathrm{D}} \Rightarrow \mathrm{B}$ is the natural transformation from Def. 3.4.7.



The left triangle in the diagram above establishes the required counit law for λ^*. We show that, with this definition of λ^*, the liftings

$$\mathrm{S}_\lambda : \mathsf{Coalg}_{\mathrm{B}} \to \mathsf{Coalg}_{\mathrm{B}} \quad \text{and} \quad \mathrm{S}_{\lambda^*} : \mathsf{Coalg}_{\langle \tilde{\mathrm{D}}, \tilde{\varepsilon} \rangle} \to \mathsf{Coalg}_{\langle \tilde{\mathrm{D}}, \tilde{\varepsilon} \rangle}$$

from Lemmata 3.2.5 and 3.4.5 correspond to each other via the isomorphism $\mathsf{Coalg}_{\mathrm{B}} \simeq \mathsf{Coalg}_{\langle \tilde{\mathrm{D}}, \tilde{\varepsilon} \rangle}$ (cf. Lemma 3.4.8 (ii)), i.e.

$$(\mathrm{S}_\lambda \langle P, \alpha \rangle)^* = \mathrm{S}_{\lambda^*}(\langle P, \alpha \rangle^*) \tag{3.28}$$

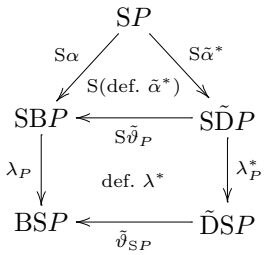
for all B-coalgebras $\langle P, \alpha \rangle$, where by $\langle P, \alpha \rangle^* := \langle P, \tilde{\alpha}^* \rangle$ we denote the coalgebra in $\mathsf{Coalg}_{\langle \tilde{\mathrm{D}}, \tilde{\varepsilon} \rangle}$ corresponding to $\langle P, \alpha \rangle$ in $\mathsf{Coalg}_{\mathrm{B}}$ under the above isomorphism.
It is easier to actually show the equality in $\mathsf{Coalg}_{\mathrm{B}}$. So we claim that

$$\mathrm{S}_\lambda \langle P, \alpha \rangle = \langle \mathrm{S}P, \lambda_P \circ \mathrm{S}\alpha \rangle$$

is the B-coalgebra to which

$$\mathrm{S}_{\lambda^*}(\langle P, \alpha \rangle^*) = \langle \mathrm{S}P, \lambda^*_P \circ \mathrm{S}\tilde{\alpha}^* \rangle$$

corresponds. Bearing in mind that any coalgebra $\langle Q, \tilde{\alpha}_Q \rangle$ for the copointed functor $\mathsf{Coalg}_{\langle \tilde{\mathrm{D}}, \tilde{\varepsilon} \rangle}$ corresponds to the B-coalgebra $\langle Q, \tilde{\vartheta}_Q \circ \tilde{\alpha}_Q \rangle$, the claim follows from the diagram below.



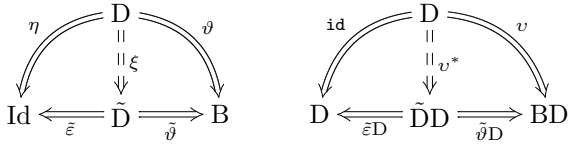
With the correspondence of S_λ and S_{λ^*} we easily obtain the correspondence of λ-bialgebras and λ^*-bialgebras: by definition, $\langle P, \beta, \alpha\rangle$ is a λ-bialgebra if and only if β is a B-coalgebra homomorphism from $\mathrm{S}_\lambda\langle P, \alpha\rangle$ to $\langle P, \alpha\rangle$. By the isomorphism $\mathsf{Coalg}_{\mathrm{B}} \simeq \mathsf{Coalg}_{\langle \tilde{\mathrm{D}}, \tilde{\varepsilon}\rangle}$ the latter is the case if and only if β is a $\langle \tilde{\mathrm{D}}, \tilde{\varepsilon}\rangle$-coalgebra homomorphism from $(\mathrm{S}_\lambda\langle P, \alpha\rangle)^*$ to $\langle P, \alpha\rangle^*$, i.e. from $\mathrm{S}_{\lambda^*}\langle P, \tilde{\alpha}^*\rangle$ to $\langle P, \tilde{\alpha}^*\rangle$ using (3.28). This in turn is equivalent to $\langle P, \beta, \tilde{\alpha}^*\rangle$ being a λ^*-bialgebra.

For item (ii) it is sufficient to show that λ^* as defined above satisfies a unit or multiplication law if λ does. Both proofs are straightforward. Item (iii) follows immediately by dualising the proofs.

□

Next we prove a lifting of distributive laws over a cofree copointed functor $\langle \tilde{\mathrm{D}}, \tilde{\varepsilon}\rangle$ to those over the cofree comonad $\langle \mathrm{D}, \varepsilon, \delta\rangle$. To this end we first study the relation of the two cofree structures.

Lemma 3.6.2 *Let $\langle\langle \tilde{\mathrm{D}}, \tilde{\varepsilon}\rangle, \tilde{\vartheta} : \tilde{\mathrm{D}} \Rightarrow \mathrm{B}\rangle$ and $\langle\langle \mathrm{D}, \varepsilon, \delta\rangle, \upsilon : \mathrm{D} \Rightarrow \mathrm{BD}\rangle$ be the cofree pointed functor and comonad over the same functor* B *(cf. Def. 3.4.7 and 2.3.17). Define $\xi : \mathrm{D} \Rightarrow \tilde{\mathrm{D}}$ and $\upsilon^* : \mathrm{D} \Rightarrow \tilde{\mathrm{D}}\mathrm{D}$ to be the natural transformations fitting into the diagrams below (cf. Corollary 3.4.8), where $\upsilon : \mathrm{D} \Rightarrow \mathrm{BD}$, $\vartheta : \mathrm{D} \Rightarrow \mathrm{B}$, and $\tilde{\vartheta} : \tilde{\mathrm{D}} \Rightarrow \mathrm{B}$ are defined as in Definitions 2.3.17 and 3.4.7.*

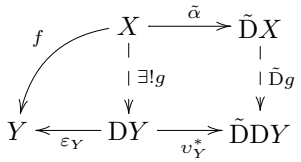


(i) The composed isomorphism

$$\mathsf{Coalg}_{\langle \mathrm{D}, \varepsilon, \delta\rangle} \simeq \mathsf{Coalg}_{\mathrm{B}} \simeq \mathsf{Coalg}_{\langle \tilde{\mathrm{D}}, \tilde{\varepsilon}\rangle}$$

(cf. Lemma 2.3.20 and Corollary 3.4.8 (ii)) relates the coalgebra $\langle P, \alpha\rangle$ for the comonad $\langle \mathrm{D}, \varepsilon, \delta\rangle$ to the coalgebra $\langle P, \xi_P \circ \alpha\rangle$ for the copointed functor $\langle \tilde{\mathrm{D}}, \tilde{\varepsilon}\rangle$.

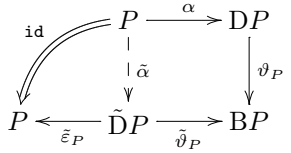
(ii) Every coalgebra $\langle X, \tilde{\alpha}\rangle$ for the copointed functor $\langle \tilde{\mathrm{D}}, \tilde{\varepsilon}\rangle$ and arrow $f : X \to Y$ determine a unique arrow $g : X \to \mathrm{D}Y$ fitting into the diagram below.



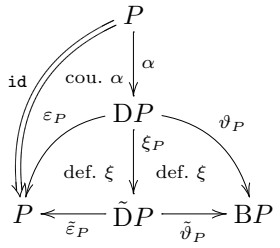
This principle lifts to natural transformations.

Proof: For item (i), spelling out both isomorphisms we find that $\langle P, \alpha\rangle$ corresponds to $\langle P, \tilde{\alpha}\rangle$, where $\tilde{\alpha}$ is the unique arrow (cf. Corollary 3.4.8 (i)) fitting

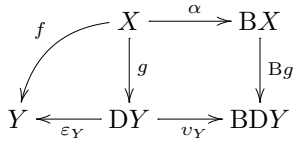
into the diagram below.



That $\xi_P \circ \alpha$ satisfies this characterisation follows from the following diagram.



For item (ii), with the correspondence of $\mathsf{Coalg}_{\mathrm{B}}$ and $\mathsf{Coalg}_{\langle \tilde{\mathrm{D}}, \tilde{\varepsilon} \rangle}$ it is easy to see that an arrow g makes the diagram in the statement commute if and only if it fits into the diagram below, where α is the B-coalgebra corresponding to $\tilde{\alpha}$. So the statement follows from the cofree coinduction principle from (2.10) on page 34.



□

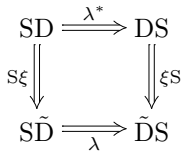
Lemma 3.6.3 *Let the copointed functor $\langle \tilde{\mathrm{D}}, \tilde{\varepsilon} \rangle$ and the comonad $\langle \mathrm{D}, \varepsilon, \delta \rangle$ be cofree over the same functor* B*. (i) A distributive law λ of a functor* S *over the copointed functor $\langle \tilde{\mathrm{D}}, \tilde{\varepsilon} \rangle$ lifts to a distributive law λ^* of the functor* S *over the comonad $\langle \mathrm{D}, \varepsilon, \delta \rangle$ such that $\langle P, \beta, \tilde{\alpha} \rangle$ is a λ-bialgebra if and only if $\langle P, \beta, \alpha \rangle$ is a λ^*-bialgebra, where $\langle P, \tilde{\alpha} \rangle$ and $\langle P, \alpha \rangle$ correspond to each other under the isomorphism $\mathsf{Coalg}_{\langle \tilde{\mathrm{D}}, \tilde{\varepsilon} \rangle} \simeq \mathsf{Coalg}_{\langle \mathrm{D}, \varepsilon, \delta \rangle}$.*

(ii) The statement still holds if we replace S *by any pointed functor $\langle \tilde{\mathrm{T}}, \tilde{\eta} \rangle$ or monad $\langle \mathrm{T}, \eta, \mu \rangle$.*

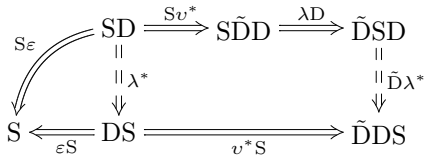
(iii) The above statements dualise.

Proof: [sketch] With Lemma 3.6.2 (i) we can reuse parts of the proof of Lemma

3.6.1 if we can find a distributive law λ^* making the following diagram commute.



It can be defined by the principle of Lemma 3.6.2 (ii) to fit into the following diagram (note that with the counit law for λ the upper arrow $\lambda\mathrm{D} \circ \mathrm{S}\upsilon^*$ satisfies the counit law as required for this principle).



Again, λ^* satisfies the counit law by definition. It can moreover be shown that it satisfies the comultiplication law as required. For item (ii) it is sufficient to show that λ^* satisfies a unit or multiplication law if λ does.

□

Item (i) of the above lemma corresponds to arrow (h) in Figure 3.3, and item (ii) covers arrows (i) and (j). With item (iii) we moreover obtain the dual arrows (b), (g) and (k). Thus, Lemmata 3.6.1 and 3.6.3 together provide all claimed liftings.

3.7 Comparison with related work

As already mentioned, the material presented in this chapter is largely based on an article by Turi and Plotkin [TP97]. Our main contribution to it is the accurate analysis of the natural transformations under consideration (cf. Sections 3.3 and 3.5.1). Turi and Plotkin state that abstract GSOS (cf. Def. 3.5.3) corresponds to the GSOS format of Bloom, Istrail, and Meyer [BIM95] in the case of labelled transition systems with a finite set of labels and sketch a proof. Since we found that it is not so easy to add the details to this sketch, we took a different approach by first decomposing the natural transformations structurally in a number of steps in our proof. This method moreover has the advantage that it is easily adaptable to variations of the behaviour under consideration. We will benefit from this property in Chapter 5, when we study specification formats for probabilistic transition systems. More as a side effect, we extended the result of Turi and Plotkin to systems with arbitrary sets of labels. This generalisation was apparently not obvious, since Turi and Plotkin asked for it in an open question in loc. cit. Moreover, we spelled out the formats for stream systems and deterministic automata.

We gave an alternative proof to show that the greatest bisimulation between two bialgebras for a distributive law is a congruence (cf. Lemma 2.3.6). Turi and Plotkin construct this greatest bisimulation as the pullback of the final coalgebra maps. For that they need to assume that the behaviour functor under consideration possesses a final coalgebra and preserves weak pullbacks; assumptions which were quoted in most of the successive literature. For our proof the greatest bisimulation need not be a pullback. All we need is that any such relation exists, which is guaranteed without assumptions on the functor in Set, our category of interest.

Another difference is that we studied distributive laws between two functors with less structure than that of a monad and comonad respectively as formats in their own right. This was inspired by an observation of Lenisa, Power, and Watanabe [LPW00], who found that the abstract GSOS format corresponds to distributive laws of a free monad over a cofree copointed functor. Although the formats assuming less structure are less expressive than the one based on monads and comonads, they have two advantages: first, they are applicable also in contexts where the monad or comonad structure is not given, which generalises parts of the bialgebraic approach, for instance, to behaviour functors not possessing a final coalgebra; second, as we shall demonstrate in Chapter 4, the less complex formats have additional well-behavedness properties.

Next we mention successive articles elaborating on Turi and Plotkin's bialgebraic approach to SOS specification formats. Among those who add to the general theory of bialgebraic semantics are the following. Watanabe [Wat02] studies different types of transformations between distributive laws. This extends the theory to topics such as conservative extensions of specifications or the implementation of one operator by several others. Power [Pow03, LPW04] proposes a categorical construction of a big-step semantics from the small-step semantics given by abstract GSOS and remarks on how to merge specifications. Klin [Kli04b, Kli04a] studies the bialgebraic framework in the context of domains and fixpoint constructions. This allows to add possibly unguarded recursive specifications of operators. Fiore and Turi [FT01] apply the theory in the setting of syntax with variable binding, which causes them to work with categories of presheaves.

We postpone the discussion of the papers concerned with the derivation of concrete specification formats from this theory [Tur97, Kic02a, Kic02b] until Chapter 5, where we derive formats for probabilistic systems.

Chapter 4

Generalised coinduction

The coinduction proof principle and the coiteration definition schema from Corollary 2.3.9 and Definition 2.3.13 are the basic coinductive definition and proof principles. But for many applications they are too rigid: with B being the behaviour functor under consideration, we often encounter functions into the carrier of a final B-coalgebra that cannot be characterised directly as coiterative arrows, and the bisimulations needed for coinduction proofs are often more complex than desirable. In Chapter 1 we have illustrated this with the specification of the function `zip` that merges two lazy lists (cf. Section 1.2.1) and the proof of an identity involving it (cf. Section 1.2.2).

Therefore, several extensions of the basic coinduction principles are discussed in the literature. Amongst them are the categorical duals of the primitive recursion and course-of-value iteration definition schemata for inductive definitions. These have been proposed and shown to be meaningful, e.g., by Uustalu and Vene [VU98, UV99]. Another example is the Flattening Lemma of Moss [Mos01]. An extended coinductive proof principle is the bisimulation up-to-context technique of Sangiorgi [San98].

Lenisa [Len99a] observed that many functions that cannot be characterised as coiterative arrows directly can be obtained by precomposing a coiterative arrow with the unit of a suitable *pointed functor* (cf. Definition 3.4.11). In order to prove properties of these functions, she assumes that there exists a *distributive law* of this pointed functor over the behaviour functor B.

Elaborating on her idea, in this chapter we introduce a new coinductive proof and definition principle based on a functor S that distributes over the behaviour functor B via a distributive law λ.

On the one hand, we obtain an abstract definition principle that we call the *λ-coiteration schema*. Under mild assumptions it is shown to uniquely characterise functions into a final B-coalgebra. Amongst those functions are examples that do not arise as (standard) coiterative arrows, i.e. functions that cannot be

defined directly by the coiteration schema. On the other hand, we introduce the notion of a λ-*bisimulation* and derive a coinduction proof principle based on it. In many cases, this principle allows us to carry out bisimulation proofs with relations that are smaller than the ordinary bisimulation relations we would need for the same task.

We also study the cases where λ is a distributive law of a pointed functor or monad over B. We show that in both cases λ-coiteration is a conservative extension of standard coiteration, in the sense that every coiterative arrow is also a λ-coiterative arrow. Moreover, in both cases the proof principle based on λ-bisimulations generalises the standard one, since every ordinary bisimulation is a λ-bisimulation.

We call λ-coinduction a *generalised coinduction principle*, because standard coinduction arises as a special instance of it: the coiteration schema (cf. Definition 2.3.13) is a special instance of the λ-coiteration schema, and a bisimulation is a special instance of a λ-bisimulation. We show that several known extended definition schemata, like the duals of primitive recursion and course of value iteration mentioned above, arise as λ-coiteration schemata for suitable distributive laws λ. Our approach provides new simple justifications for the validity of those principles as instances of our abstract statements.

Moreover, we are able to derive new schemata with little effort. Applied to the setting of Chapter 3 for instance, where we viewed distributive laws as operator specifications, the λ-coiteration definition and proof principles yield solutions of *guarded recursive specifications* and a *bisimulation up-to-context* proof principle for the operators definable in the respective specification formats. As one instance, we obtain that the bisimulation up-to-context proof principle is valid for operators defined by the GSOS format [BIM95]. With this statement we generalise a result by Sangiorgi [San98], who proved the validity of the up-to-context technique for the weaker De Simone specification format [Sim85]. In Chapter 5, where we consider probabilistic systems, we obtain corresponding properties also for the novel probabilistic formats that we shall introduce there.

We proceed as follows: in the first two sections we introduce the λ-coinduction proof and definition principles involving distributive laws between plain functors; in Section 4.3 we extend the results to settings where the algebra and coalgebra functors are equipped with the additional structure of a (co)pointed functor or monad; in Section 4.4 we spell out several instances of the abstract format, obtaining known and new concrete extensions of the basic coinduction principles; in Section 4.5 we compare our approach with related work.

4.1 The λ-coinduction proof principle

In this section we introduce the λ-coinduction proof principle in the simple setting of a distributive law λ of a given signature functor S over a given behaviour functor B. It deviates from the standard coinduction proof principle in that

it is based on λ-bisimulations (to be defined below) instead of the ordinary bisimulations from Def. 2.3.3.

Our motivating examples are about infinite streams of reals numbers $\mathbb{R}^\omega$, which can be modelled as a final coalgebra, as we mentioned in Section 2.3.3. We use the sum + and (shuffle) product $\otimes$ on $\mathbb{R}^\omega$, which are defined by the following equations for $\mathbf{s}, \mathbf{t} \in \mathbb{R}^\omega$ with $\mathbf{s} = s_0 : \mathbf{s}'$ and $\mathbf{t} = t_0 : \mathbf{t}'$ (cf. the specifications in (3.26) and (3.27) from Section 3.5.1):

$$\begin{aligned} \mathbf{s} + \mathbf{t} &= (s_0 + t_0) : (\mathbf{s}' + \mathbf{t}') \\ \mathbf{s} \otimes \mathbf{t} &= (s_0 \cdot t_0) : ((\mathbf{s}' \otimes \mathbf{t}) + (\mathbf{s} \otimes \mathbf{t}')) \end{aligned}$$

We want to prove that the shuffle product is commutative, i.e. that we have

$$\mathbf{s} \otimes \mathbf{t} = \mathbf{t} \otimes \mathbf{s} \quad \text{for all } \mathbf{s}, \mathbf{t} \in \mathbb{R}^\omega.$$

Recall that $\langle \mathbb{R}^\omega, \langle h, t \rangle \rangle$, where $h : \mathbb{R}^\omega \to \mathbb{R}$ and $t : \mathbb{R}^\omega \to \mathbb{R}^\omega$ from equation (2.7) on page 32 return the head and tail of a stream, is a final coalgebra of the functor $\mathrm{B} := \underline{\mathbb{R}} \times \mathrm{Id}$. We would thus try to establish this identity with the coinduction proof principle (cf. Corollary 2.3.9). This means that we need to come up with a bisimulation relation $R \subseteq \mathbb{R}^\omega \times \mathbb{R}^\omega$ relating the streams we aim to prove equal. The evident candidate is thus

$$R := \{\langle \mathbf{s} \otimes \mathbf{t}, \mathbf{t} \otimes \mathbf{s} \rangle \mid \mathbf{s}, \mathbf{t} \in \mathbb{R}^\omega\}. \tag{4.1}$$

The definition of a bisimulation relation requires first that the heads of any two streams related should be equal, i.e. $h(\mathbf{s} \otimes \mathbf{t}) = h(\mathbf{t} \otimes \mathbf{s})$ for all $\mathbf{s}, \mathbf{t} \in \mathbb{R}^\omega$. This easily follows from the definitions. But second, the tails of any two related streams should be related again. For the pairs in R the specification of $\otimes$ yields

$$\begin{aligned} t(\mathbf{s} \otimes \mathbf{t}) &= (\mathbf{s}' \otimes \mathbf{t}) + (\mathbf{s} \otimes \mathbf{t}') \\ &\overset{(*)}{=} (\mathbf{s} \otimes \mathbf{t}') + (\mathbf{s}' \otimes \mathbf{t}), \\ t(\mathbf{t} \otimes \mathbf{s}) &= (\mathbf{t}' \otimes \mathbf{s}) + (\mathbf{t} \otimes \mathbf{s}'), \end{aligned} \tag{4.2}$$

where we again write $\mathbf{s} = s_0 : \mathbf{s}'$ and $\mathbf{t} = t_0 : \mathbf{t}'$ and for the step $(*)$ we use the commutativity of the sum, which can be shown by a straightforward coinduction proof. Since both tails are given as sums of two streams, we do not obtain a pair such as those related by R. In order to establish the bisimulation property, we thus have to add further pairs. Observing that the respective summands in the above expressions are related by R, i.e.

$$\langle \mathbf{s} \otimes \mathbf{t}', \mathbf{t}' \otimes \mathbf{s} \rangle \in R \quad \text{and} \quad \langle \mathbf{s}' \otimes \mathbf{t}, \mathbf{t} \otimes \mathbf{s}' \rangle \in R,$$

we can take

$$\tilde{R} := \Big\{ \langle \sum_{i=1}^{n} (\mathbf{s}_i \otimes \mathbf{t}_i), \sum_{i=1}^{n} (\mathbf{t}_i \otimes \mathbf{s}_i) \rangle \;\Big|\; n \in \mathbb{N}; \mathbf{s}_i, \mathbf{t}_i \in \mathbb{R}^\omega (1 \leq i \leq n) \Big\}. \tag{4.3}$$

But in order to show that $\tilde{R}$ is a bisimulation, we have to deal with more complex terms than before, which we could try to handle, for instance, by an induction on the number of summands n in the terms under consideration. However, since the relation $\tilde{R}$ arose from R in a systematic way as a closure under the sum, we can hope that there is a proof principle to automate such an induction.

It turns out indeed that we get a valid proof principle for bisimilarity if we modify the definition of a bisimulation relation such that the successors are required to arise as sums of pairwisely related streams. This principle as such is not really universal, but it motivates the first step of the development of an abstract framework. This will be powerful enough to provide the justification of several known and new variations and generalisations of the bisimulation proof technique.

The computation in (4.2) shows that the tails of $\mathbf{s} \otimes \mathbf{t}$ and $\mathbf{t} \otimes \mathbf{s}$ can be expressed as sums of streams that are pairwise related by R, i.e. we have

$$\langle t(\mathbf{s} \otimes \mathbf{t}), t(\mathbf{t} \otimes \mathbf{s}) \rangle \in R' := \{\langle \mathbf{x}_1 + \mathbf{x}_2, \mathbf{y}_1 + \mathbf{y}_2 \rangle \mid \langle \mathbf{x}_1, \mathbf{y}_1 \rangle,\ \langle \mathbf{x}_2, \mathbf{y}_2 \rangle \in R\}$$

With this observation we find that R satisfies a condition similar to that of a bisimulation: setting, for $\mathbf{s} = s_0 : \mathbf{s}$ and $\mathbf{t} = t_0 : \mathbf{t}$,

$$\gamma'(\langle \mathbf{s} \otimes \mathbf{t}, \mathbf{t} \otimes \mathbf{s} \rangle) := \langle s_0 \cdot t_0, \langle (\mathbf{s}' \otimes \mathbf{t}) + (\mathbf{t} \otimes \mathbf{s}'), (\mathbf{s} \otimes \mathbf{t}') + (\mathbf{t}' \otimes \mathbf{s}) \rangle \rangle$$

we find a function $\gamma' : R \to \mathbb{R} \times R'$ that makes both parts of the diagram below commute (for a similar situation see Sangiorgi [San98]).

$$\begin{array}{ccccc}
\mathbb{R}^\omega & \xleftarrow{\pi_1} & R & \xrightarrow{\pi_2} & \mathbb{R}^\omega \\
{\scriptstyle\langle h,t\rangle}\downarrow & & \downarrow{\scriptstyle\gamma} & & \downarrow{\scriptstyle\langle h,t\rangle} \\
\mathbb{R} \times \mathbb{R}^\omega & \xleftarrow[\mathtt{id}\times\pi_1']{} & \mathbb{R} \times R' & \xrightarrow[\mathtt{id}\times\pi_2']{} & \mathbb{R} \times \mathbb{R}^\omega
\end{array}$$

Next, note that we can write the relation R' above as

$$R' = \langle + \circ (\pi_1 \times \pi_1), + \circ (\pi_2 \times \pi_2) \rangle[R \times R] \subseteq \mathbb{R}^\omega \times \mathbb{R}^\omega,$$

so the function γ' factors as

$$\gamma' = (\mathtt{id}_{\mathbb{R}} \times \langle + \circ (\pi_1 \times \pi_1), + \circ (\pi_2 \times \pi_2) \rangle) \circ \gamma$$

for some $\gamma : R \to \mathbb{R} \times (R \times R)$. This way we can eliminate the relation R' in the above diagram to obtain the one below.

$$\begin{array}{ccccc}
\mathbb{R}^\omega & \xleftarrow{\pi_1} & R & \xrightarrow{\pi_2} & \mathbb{R}^\omega \\
{\scriptstyle\langle h,t\rangle}\downarrow & & \downarrow{\scriptstyle\gamma} & & \downarrow{\scriptstyle\langle h,t\rangle} \\
\mathbb{R} \times \mathbb{R}^\omega & \xleftarrow[\mathtt{id}\times(+\circ(\pi_1\times\pi_1))]{} & \mathbb{R} \times (R \times R) & \xrightarrow[\mathtt{id}\times(+\circ(\pi_2\times\pi_2))]{} & \mathbb{R} \times \mathbb{R}^\omega
\end{array}$$

In the following we shall show that the existence of such a γ is sufficient here to conclude that R is contained in some bisimulation. But first we derive a more abstract description of the diagram above. To this end, note first that with $\mathrm{S} := \mathrm{Id} \times \mathrm{Id}$ we obtain $R \times R = \mathrm{S}R$ and $(\pi_i \times \pi_i) = \mathrm{S}\pi_i$. Second, with $\mathrm{B} := \underline{\mathbb{R}} \times \mathrm{Id}$, the $\langle \mathrm{S}, \mathrm{B}\rangle$-bialgebra $\langle \mathbb{R}^\omega, +, \langle h, t\rangle\rangle$ is a λ-bialgebra for the distributive law $\lambda : \mathrm{SB} \Rightarrow \mathrm{BS}$, where for any set X the component

$$\lambda_X : (\mathbb{R} \times X) \times (\mathbb{R} \times X) \to \mathbb{R} \times (X \times X)$$

is defined for all $u, v \in \mathbb{R}$ and $x, y \in X$ by

$$\lambda_X(\langle u, x\rangle, \langle v, y\rangle) := \langle u + v, \langle x, y\rangle\rangle. \tag{4.4}$$

Using this, the above situation is an instance of the following notion.

Definition 4.1.1 *Let* S *and* B *be functors, and let* $\langle P, \beta_P, \alpha_P\rangle$ *and* $\langle Q, \beta_Q, \alpha_Q\rangle$ *be two* $\langle \mathrm{S}, \mathrm{B}\rangle$*-bialgebras. We call a relation* $R \subseteq P \times Q$ *an* $\langle \mathrm{S}, \mathrm{B}\rangle$**-bisimulation** *between* $\langle P, \beta_P, \alpha_P\rangle$ *and* $\langle Q, \beta_Q, \alpha_Q\rangle$ *if there exists a* BS*-coalgebra structure* γ *on* R *such that the diagram below commutes.*

$$\begin{array}{ccccc} P & \xleftarrow{\pi_1} & R & \xrightarrow{\pi_2} & Q \\ {\scriptstyle \alpha_P}\downarrow & & \downarrow{\scriptstyle \gamma} & & \downarrow{\scriptstyle \alpha_Q} \\ \mathrm{B}P & \xleftarrow[\mathrm{B}(\beta_P \circ \mathrm{S}\pi_1)]{} & \mathrm{BS}R & \xrightarrow[\mathrm{B}(\beta_Q \circ \mathrm{S}\pi_2)]{} & \mathrm{B}Q \end{array}$$

If the two bialgebras $\langle P, \beta_P, \alpha_P\rangle$ *and* $\langle Q, \beta_Q, \alpha_Q\rangle$ *are* λ*-bialgebras for a distributive law* λ *of* S *over* B*, then we shall speak of* λ*-bisimulations instead of* $\langle \mathrm{S}, \mathrm{B}\rangle$*-bisimulations for short.*

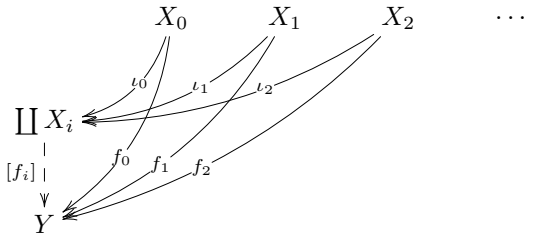
So the relation R from (4.1) is a λ-bisimulation for the distributive law λ from (4.4) specifying the sum of two streams of real numbers. In order to use λ-bisimulations for a proof principle, we shall prove that any two states related by a λ-bisimulation are bisimilar.

Theorem 4.1.2 *Let* S *and* B *be two functors on a category* C *with countable coproducts, and let* λ *be a distributive law of* S *over* B*. Any* λ*-bisimulation between two* λ*-bialgebras* $\langle P, \beta_P, \alpha_P\rangle$ *and* $\langle Q, \beta_Q, \alpha_Q\rangle$ *is contained in some (ordinary) bisimulation between* $\langle P, \alpha_P\rangle$ *and* $\langle Q, \alpha_Q\rangle$*.*

With this theorem and Corollary 2.3.9 we immediately obtain the following *λ-coinduction proof principle.*

Corollary 4.1.3 *Let* S *and* B *be two functors on a category* C *with countable coproducts, let* $\langle \mathcal{F}, \omega\rangle$ *be a final* B*-coalgebra, and let* λ *be a distributive law of* S *over* B*. If* $R \subseteq \mathcal{F} \times \mathcal{F}$ *is a* λ*-bisimulation on the final* λ*-bialgebra* $\langle \mathcal{F}, \beta_{\mathcal{F}}, \omega\rangle$ *from Theorem 3.2.3 (ii) then* $\langle p, q\rangle \in R$ *implies* $p = q$*.*

Before we prove Theorem 4.1.2, we need to introduce some notation. The *countable coproduct* of the objects X_i for $i \in \mathbb{N}$ is an object $\coprod X_i$ together with injections $\iota_j : X_j \to \coprod X_i$ for $j \in \mathbb{N}$ such that the following holds: for any object Y and arrows $f_j : X_j \to Y$ for $j \in \mathbb{N}$ there is a unique arrow $[f_i] : \coprod X_i \to Y$, the *countable case analysis*, such that $[f_i] \circ \iota_j = f_j$ for all $j \in \mathbb{N}$.



(4.5)

In Set, the countable coproduct of the sets $X_1, X_2, \ldots$ and the countable case analysis for the functions $f_i : X_i \to Y$ can be described as follows.

$$\begin{array}{rcl} \coprod X_i & = & \{\langle i, x\rangle \mid i \in \mathbb{N},\ x \in X_i\} \\ [f_i] & = & [\langle i, x\rangle \mapsto f_i(x)] \end{array}$$

Definition 4.1.4 *We define a pointed functor* $\langle \mathrm{H}, \iota_0 : \mathrm{Id} \Rightarrow \mathrm{H}\rangle$ *by*

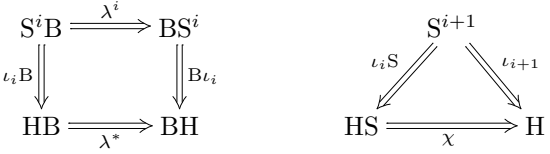
$$\mathrm{H} := \coprod_{i=0}^{\infty} \mathrm{S}^i \quad \textit{and} \quad \iota_0 : \mathrm{S}^0 \Rightarrow \mathrm{H},$$

where S^i *is the* i*-fold application of* S*, i.e.* $\mathrm{S}^0 := \mathrm{Id}$ *and* $\mathrm{S}^{i+1} := \mathrm{S} \circ \mathrm{S}^i$*.*

Any S*-algebra* $\beta : \mathrm{S}X \to X$ *lifts to an algebra* $\beta^* : \mathrm{H}X \to X$ *for the pointed functor by defining*

$$\beta^* := [\beta^i] : \mathrm{H}X \to X \quad \textit{where} \quad \beta^0 := \mathtt{id}_X \quad \textit{and} \quad \beta^{i+1} := \beta \circ \mathrm{S}\beta^i.$$

Moreover, a distributive law λ *of* S *over any functor* B *lifts to a distributive law* λ^* *of the pointed functor* $\langle \mathrm{H}, \iota_0\rangle$ *over* B*. Also, we use the natural transformation* $\chi : \mathrm{HS} \Rightarrow \mathrm{H}$*. They are defined to be the (by the universal property of the countable coproduct defining* H*) unique natural transformations fitting into the respective diagrams below for all* $i \in \mathbb{N}$*.*



Here $\lambda^i : \mathrm{S}^i\mathrm{B} \Rightarrow \mathrm{BS}^i$ *is defined inductively by*

$$\lambda^0 := \mathtt{id} \quad \textit{and} \quad \lambda^{i+1} := \lambda\mathrm{S}^i \circ \mathrm{S}\lambda^i.$$

Note that the diagram for $i = 0$ *in the definition of* λ^* *establishes the unit law for* λ^**.*

Lemma 4.1.5 *Let λ be a distributive law of the functor S over the functor B, let $\langle X, \beta, \alpha\rangle$ be an $\langle \mathrm{S}, \mathrm{B}\rangle$-bialgebra, and let λ^* and β^* be the liftings of λ and β from Definition 4.1.4. If $\langle X, \beta, \alpha\rangle$ is a λ-bialgebra then $\langle X, \beta^*, \alpha\rangle$ is a λ^*-bialgebra.*

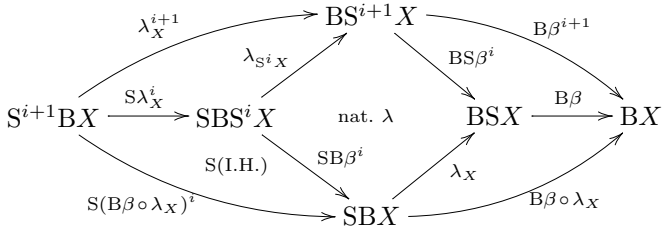
Proof: It is immediate to see that the mapping $\langle X, \beta\rangle \mapsto \langle X, \beta^*\rangle$ defines a functor from $\mathsf{Alg}^{\mathrm{S}}$ to $\mathsf{Alg}^{\mathrm{H}}$, i.e. that it preserves homomorphisms. If $\langle X, \beta, \alpha\rangle$ is a λ-bialgebra then α is an S-algebra homomorphism from $\langle X, \beta\rangle$ to $\mathrm{B}_\lambda\langle X, \beta\rangle = \langle \mathrm{B}X, \mathrm{B}\beta \circ \lambda_X\rangle$ (cf. Lemma 3.2.5 (i)). This implies that α is an H-algebra homomorphism from $\langle X, \beta^*\rangle$ to $\langle \mathrm{B}X, (\mathrm{B}\beta \circ \lambda_X)^*\rangle$. Again with Lemma 3.2.5 (i) this means that $\langle X, \beta^*, \alpha\rangle$ is a λ^*-bialgebra, because

$$\langle \mathrm{B}X, (\mathrm{B}\beta \circ \lambda_X)^*\rangle = \mathrm{B}_{\lambda^*}\langle X, \beta^*\rangle,$$

as we shall show. The above equation amounts to stating $(\mathrm{B}\beta \circ \lambda_X)^* = \mathrm{B}\beta^* \circ \lambda^*_X$. By the universal property of the countable coproduct it suffices to show for all $i \in \mathbb{N}$ that both sides of the latter equation are equal when precomposed with $\iota_i : \mathrm{S}^i\mathrm{B}X \to \mathrm{HB}X$:

$$\begin{array}{rcl}
(\mathrm{B}\beta \circ \lambda_X)^* \circ \iota_i & = & (\mathrm{B}\beta \circ \lambda_X)^i \\
& \overset{(*)}{=} & \mathrm{B}\beta^i \circ \lambda^i_X \\
& = & \mathrm{B}(\beta^* \circ \iota_i) \circ \lambda^i_X \\
& = & \mathrm{B}\beta^* \circ \mathrm{B}\iota_i \circ \lambda^i_X \\
& = & \mathrm{B}\beta^* \circ \lambda^*_X \circ \iota_i.
\end{array}$$

The equation $(\mathrm{B}\beta \circ \lambda_X)^i = \mathrm{B}\beta^i \circ \lambda^i_X$ used in the step marked with $(*)$ follows by induction on i: for $i = 0$ both sides are defined to be $\mathtt{id}_{\mathrm{B}X}$; the induction step for $i + 1$ is valid by the diagram below, which uses the induction hypothesis, naturality of λ and definitions. The lower composite is $(\mathrm{B}\beta \circ \lambda_X)^{i+1}$ by definition.



□

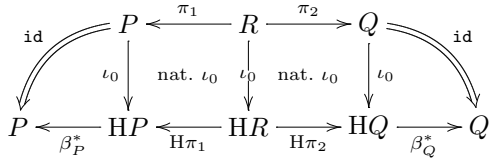
We are now prepared to prove our main theorem.

Proof: [Theorem 4.1.2] Let $R \subseteq P \times Q$ be a λ-bisimulation between $\langle P, \beta_P, \alpha_P\rangle$ and $\langle Q, \beta_Q, \alpha_Q\rangle$. We show that the image of the span

$$\langle \mathrm{H}R, \beta^*_P \circ \mathrm{H}\pi_1, \beta^*_Q \circ \mathrm{H}\pi_2\rangle \tag{4.6}$$

is a bisimulation between $\langle P, \alpha_P\rangle$ and $\langle Q, \alpha_Q\rangle$ that contains R.

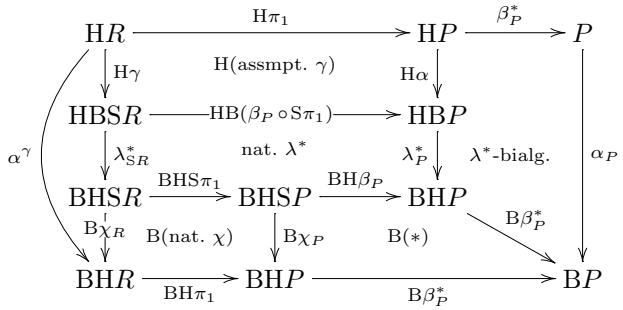
With Lemma 2.3.5, R is contained in the image of the span (4.6) because with the injection ι_0 all parts of the diagram below commute. The two triangles are the unit laws of β_P^* and β_Q^*, which hold by definition of the latter.



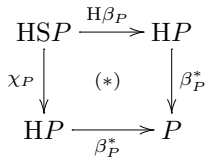
Let γ be the intermediate BS-coalgebra structure on R demanded by the definition of a λ-bisimulation. By

$$\alpha^\gamma := \mathrm{B}\chi_R \circ \lambda^*_{\mathrm{S}R} \circ \mathrm{H}\gamma \tag{4.7}$$

we define a B-coalgebra structure on $\mathrm{H}R$. We show that α^γ witnesses that the span (4.6) is a bisimulation between $\langle P, \alpha_P\rangle$ and $\langle Q, \alpha_Q\rangle$, i.e. that $\beta_P^* \circ \mathrm{H}\pi_1$ is a coalgebra homomorphism from $\langle \mathrm{H}R, \alpha^\gamma\rangle$ to $\langle P, \alpha_P\rangle$ and respectively for the side of Q. We prove the property for P with the diagram below, which uses that $\langle P, \beta_P^*, \alpha_P\rangle$ is a λ^*-bialgebra by Lemma 4.1.5. The one for Q follows in the same way.



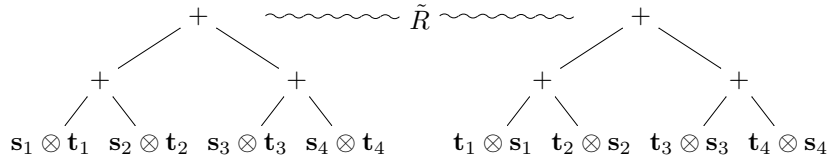
By $(*)$ in the diagram we refer to the property of β_P^* pictured below, which easily follows from the definitions. It resembles the multiplication law for an algebra for a monad.



□

With this theorem, in order to show that two states in two λ-bialgebras are bisimilar it suffices to exhibit a λ-bisimulation relating them. In the case of our example we have seen that the relation R from (4.1) is a λ-bisimulation, so we

get that all related states are bisimilar as wanted. The standard bisimulation, say $\tilde{R}$, constructed from R in the proof of Theorem 4.1.2 is slightly different from the one we gave in (4.3) though. It relates the states obtained by applications of the sum operator arranged in full binary trees of any depth k, as shown in this picture for $k = 2$:



The B-coalgebra structure α^γ assigns to such a pair a successor pair of trees with the depth increased by one.

In Section 4.3 we will give alternative constructions of the standard bisimulation and the mediating coalgebra structure α^γ exploiting the additional structure of a (co)pointed functor or monad replacing the plain functors S and B.

4.2 The λ-coiteration definition principle

We are now going to employ the same idea as in the previous section to derive variations and generalisations of the coiteration definition schema (cf. Definition 2.3.13). We start by giving an example of a specification that does not fit into the standard principle. It again involves streams of real numbers. We consider the following definition of the stream $\mathtt{pow} = \langle 1, 2, 4, 8, \ldots \rangle$ of powers of two:

$$\mathtt{pow} \;:=\; 1 : (\mathtt{pow} + \mathtt{pow}). \tag{4.8}$$

As before, we use that $\langle \mathbb{R}^\omega, \langle h, t \rangle \rangle$ (cf. equation (2.7) from page 32) is a final coalgebra of the functor $\mathrm{B} := \underline{\mathbb{R}} \times \mathrm{Id}$. So we try to define the stream $\mathtt{pow} \in \mathbb{R}^\omega$ with the coiteration schema. Therefore we need to come up with a B-coalgebra $\langle P, \langle o, s \rangle : P \to \mathbb{R} \times P \rangle$ such that $\mathtt{pow} = f(p)$ for some $p \in P$, where f is the unique homomorphism from $\langle P, \langle o, s \rangle \rangle$ to the final coalgebra $\langle \mathbb{R}^\omega, \langle h, t \rangle \rangle$. We can easily see that with the definition above the carrier P of such a coalgebra needs to be infinite: repeatedly applying $t(f(x)) = f(s(x))$ from the homomorphism property of f and using equation (4.8) we get

$$\mathtt{pow} \;=\; f(p) \;=\; 1 : (2 : \ldots (2^{n-1} : f(s^n(p)) \ldots)$$

for all $n \in \mathbb{N}$; we can read of that all streams $f(s^n(p))$ are different, so all $s^n(p) \in P$ need to be different, which is possible only for an infinite set P. The coalgebra structure $\langle o, s \rangle$ needs to be defined on all these infinitely many elements.

The specification of pow, however, defines one head and one tail only, namely the ones of pow itself. So it directly gives rise to a coalgebraic structure on a one

element set $1 = \{*\}$. Because of the twofold occurrence of `pow` in the expression for the tail, this does not define a B-coalgebra structure, but rather a function of the type $\phi : 1 \to \mathbb{R} \times (1 \times 1)$, namely

$$\phi(*) := \langle 1, \langle *, * \rangle \rangle, \tag{4.9}$$

which is a BS-coalgebra structure for $\mathrm{S} := \mathrm{Id} \times \mathrm{Id}$.

Note that functions of that type also appeared in the definition of a λ-bisimulation in the previous section. The condition on the arrows in that definition comes in here as well: for a function $f : 1 \to \mathbb{R}^\omega$, the stream $f(*) \in \mathbb{R}^\omega$ satisfies the specification in (4.8) if and only if f fits into the following diagram.

$$\begin{array}{ccccc} 1 & & \xrightarrow{\qquad f \qquad} & & \mathbb{R}^\omega \\ {\scriptstyle\phi}\downarrow & & & & \downarrow{\scriptstyle\langle h,t\rangle} \\ \mathbb{R} \times (1 \times 1) & \xrightarrow[\mathtt{id} \times (f \times f)]{} & \mathbb{R} \times (\mathbb{R}^\omega \times \mathbb{R}^\omega) & \xrightarrow[\mathtt{id} \times +]{} & \mathbb{R} \times \mathbb{R}^\omega \end{array} \tag{4.10}$$

The specification of `pow` is thus valid if and only if this diagram characterises the arrow f uniquely.

The observation that, for the distributive law λ of S over B from (4.4), the $\langle \mathrm{S}, \mathrm{B} \rangle$-bialgebra $\langle \mathbb{R}^\omega, +, \omega \rangle$ is a final λ-bialgebra leads to the following definition as a generalisation of the above diagram.

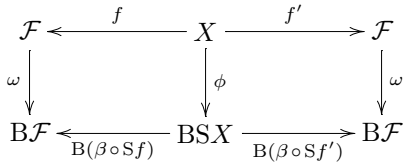
Definition 4.2.1 *Let* $\mathrm{S}, \mathrm{B} : \mathsf{C} \to \mathsf{C}$ *be functors, let* $\langle \mathcal{F}, \omega \rangle$ *be a final* B*-coalgebra and let* λ *be a distributive law of* S *over* B*. For a* BS*-coalgebra* $\langle X, \phi \rangle$ *we call an arrow* $f : X \to \mathcal{F}$ *a* **λ-coiterative arrow induced by** ϕ *if it makes the diagram below commute, where* β *is the unique arrow such that* $\langle \mathcal{F}, \beta, \omega \rangle$ *is a* λ*-bialgebra (cf. Theorem 3.2.3* (ii)*):*

$$\begin{array}{ccccc} X & & \xrightarrow{\qquad f \qquad} & & \mathcal{F} \\ {\scriptstyle\phi}\downarrow & & & & \downarrow{\scriptstyle\omega} \\ \mathrm{BS}X & \xrightarrow[\mathrm{BS}f]{} & \mathrm{BS}\mathcal{F} & \xrightarrow[\mathrm{B}\beta]{} & \mathrm{B}\mathcal{F} \end{array}$$

In a setting where a final B-coalgebra exists, the following theorem states that with the same assumption on the underlying category as in Theorem 4.1.2 every BS-coalgebra uniquely determines a λ-coiterative arrow.

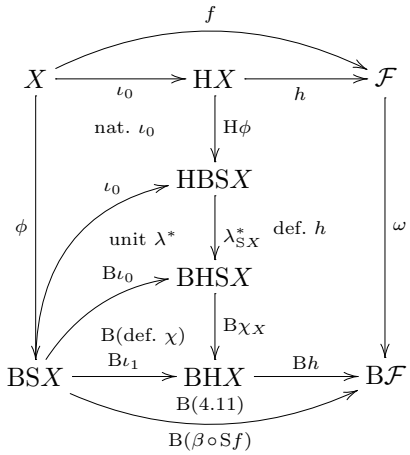
Theorem 4.2.2 *Let* $\mathrm{S}, \mathrm{B} : \mathsf{C} \to \mathsf{C}$ *be functors on a category* C *with countable coproducts, let* $\langle \mathcal{F}, \omega \rangle$ *be a final* B*-coalgebra and let* λ *be a distributive law of* S *over* B*. For any* BS*-coalgebra* $\langle X, \phi \rangle$ *there exists a unique* λ*-coiterative arrow* $f : X \to \mathcal{F}$ *induced by* ϕ*.*

Proof: Let $\langle \mathcal{F}, \beta, \omega \rangle$ be the final λ-bialgebra according to Theorem 3.2.3 (ii). The *uniqueness* of a λ-coiterative arrow easily follows from the λ-coiteration proof principle: given two λ-coiterative arrows f and f', ϕ witnesses that $\langle X, f, f' \rangle$ is a λ-bisimulation on $\langle \mathcal{F}, \beta, \omega \rangle$.



This λ-bisimulation relates the pairs $\langle f(x), f'(x) \rangle$ for all $x \in X$, so by Corollary 4.1.3 we have $f(x) = f'(x)$ for all $x \in X$ and thus $f = f'$.

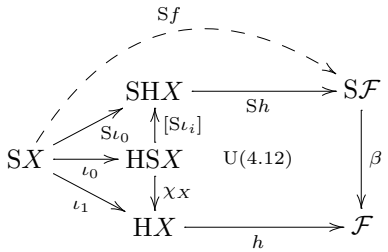
The *existence* part requires more work. We use H, β^*, λ^*, and χ as in Def. 4.1.4. We again turn the BS-coalgebra $\langle X, \phi \rangle$ into a B-coalgebra $\langle \mathrm{H}X, \alpha^\phi \rangle$, where $\alpha^\phi := \mathrm{B}\chi_X \circ \lambda^*_{\mathrm{S}X} \circ \mathrm{H}\phi$. Let $h : \mathrm{H}X \to \mathcal{F}$ be the coiterative arrow from $\langle \mathrm{H}X, \alpha^\phi \rangle$ to $\langle \mathcal{F}, \omega \rangle$. The following diagram shows that $f := h \circ \iota_0$ fits as a λ-coiterative arrow from $\langle X, \phi \rangle$ to $\langle \mathcal{F}, \beta, \omega \rangle$.



Its bottommost part commutes because it arises by applying the functor B to the equation

$$\beta \circ \mathrm{S}f = h \circ \iota_1, \tag{4.11}$$

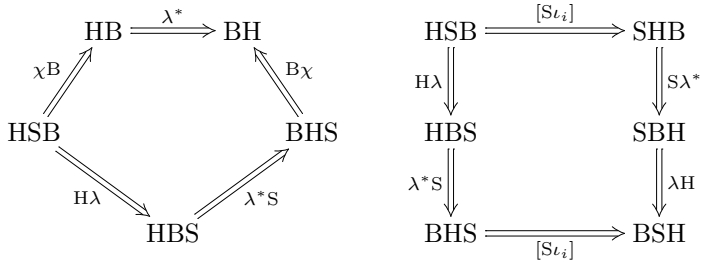
which in turn is proved by the diagram below.



Both triangles commute by definition, and the rectangle arises as the image of the diagram (4.12) below under the forgetful functor $\mathrm{U} : \mathsf{Coalg}_{\mathrm{B}} \to \mathsf{C}$. It is a diagram in the category of B-coalgebras, and it commutes by finality.

$$\begin{array}{ccc} \mathrm{S}_\lambda\langle \mathrm{H}X, \alpha^\phi\rangle & \xrightarrow{\mathrm{S}_\lambda h} & \mathrm{S}_\lambda\langle \mathcal{F}, \omega\rangle \\ {\scriptstyle [\mathrm{S}\iota_i]}\uparrow & & \\ \langle \mathrm{HS}X, \alpha^{(\lambda_{\mathrm{S}X} \circ \mathrm{S}\phi)}\rangle \quad \text{finality} & & \downarrow{\scriptstyle \beta} \\ {\scriptstyle \chi_X}\downarrow & & \\ \langle \mathrm{H}X, \alpha^\phi\rangle & \xrightarrow[h]{} & \langle \mathcal{F}, \omega\rangle \end{array} \tag{4.12}$$

It remains to be checked that all drawn arrows exist in $\mathsf{Coalg}_{\mathrm{B}}$, that is, they are homomorphisms between the respective coalgebras. The arrow h is a homomorphism by its definition. This implies that $\mathrm{S}_\lambda h$ is a homomorphism as well. The algebra operation β is a homomorphism by the assumption on $\langle \mathcal{F}, \beta, \omega\rangle$ being a λ-bialgebra (cf. Lemma 3.2.5 (ii) for the last two statements). For the remaining two arrows $[\mathrm{S}\iota_i]$ and χ_X the homomorphism property can be checked by routine calculations. Notably, the easily established commutativity of the two diagrams below can be used. The left one resembles a multiplication law for λ^*.



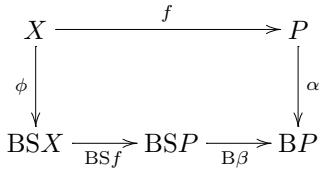
□

Applying the above theorem to our example, we find that ϕ from equation (4.9) uniquely defines an arrow $f : 1 \to \mathbb{R}^\omega$ fitting into the diagram (4.10). Which means that there is a unique stream $\mathtt{pow} := f(*) \in \mathbb{R}^\omega$ satisfying the specification in (4.8).

In order to obtain a slightly broader perspective on specifications of B-behaviours that can be captured by arrows $\phi : X \to \mathrm{BS}X$, we give the following definition, which does not rely on the existence of a final B-coalgebra anymore.

Definition 4.2.3 *Given two functors* $\mathrm{B}, \mathrm{S} : \mathsf{C} \to \mathsf{C}$*, where* B *is viewed as a behaviour functor, we call an arrow* $\phi : X \to \mathrm{BS}X$ *a* **guarded recursive definition**. *A* **solution** *of this definition in an* $\langle \mathrm{S}, \mathrm{B}\rangle$*-bialgebra* $\langle P, \beta, \alpha\rangle$ *is an*

arrow $f : X \to P$ making the diagram below commute.



In Section 4.4.4 we explain why we call ϕ a guarded recursive definition. With this notion, a λ-coiterative arrow induced by ϕ appears as a solution of ϕ in the final λ-bialgebra.

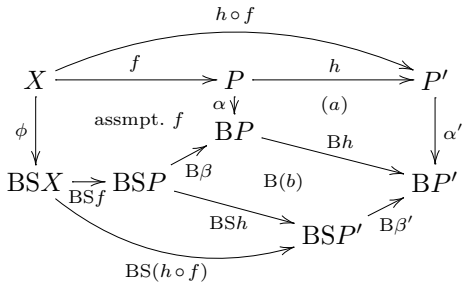
Since two solutions of a guarded recursive specification in two λ-bialgebras give rise to a λ-bisimulation, Theorem 4.1.2 immediately implies the following corollary.

Corollary 4.2.4 *Given a distributive law λ of the functor S over the functor B in Set, we get that the solutions of a guarded recursive definition $\phi : X \to \mathrm{BS}X$ in λ-bialgebras are determined up to bisimilarity. By this we mean that when f and f' are two solutions of ϕ in the λ-bialgebras $\langle P, \beta, \alpha\rangle$ and $\langle P', \beta', \alpha'\rangle$ respectively, then for any $x \in X$ we have $f(x) \sim f'(x)$ in the respective coalgebras $\langle P, \alpha\rangle$ and $\langle P', \alpha'\rangle$.*

Moreover, solutions are preserved by composition with bialgebra homomorphisms.

Lemma 4.2.5 *Let $\phi : X \to \mathrm{BS}X$ be a guarded recursive specification with a solution $f : X \to P$ in the $\langle \mathrm{S}, \mathrm{B}\rangle$-bialgebra $\langle P, \beta, \alpha\rangle$ and let $h : P \to P'$ be a bialgebra homomorphism from that bialgebra to a bialgebra $\langle P', \beta', \alpha'\rangle$. Then $h \circ f$ is a solution of ϕ in $\langle P', \beta', \alpha'\rangle$.*

Proof: By the definition of a bialgebra homomorphism, h is both (a) a coalgebra homomorphism from $\langle P, \alpha\rangle$ to $\langle P', \alpha'\rangle$ and (b) an algebra homomorphism from $\langle P, \beta\rangle$ to $\langle P', \beta'\rangle$. With this we get the following commuting diagram.



□

With this lemma, a solution in an arbitrary λ-bialgebra leads to a solution in the final one, i.e. a λ-coiterative arrow. Any alternative construction of a solution

thus leads to an alternative justification of the λ-coiteration schema, possibly with different assumptions than those in Theorem 4.2.2. As an example of one such construction, we can easily obtain a λ-bialgebra on the carrier of a final BS-coalgebra, if the latter exists, such that the final BS-homomorphisms are solutions in that bialgebra. Since we shall not need this construction in the following sections, we omit the details.

4.3 λ-coinduction and additional structure

In the previous two sections we have introduced λ-coinduction as an abstract schema to obtain variations of the coinduction proof and definition principle. It was based on a distributive law λ between plain functors. As in Section 3.4 in the previous chapter, we now also consider distributive laws between functors carrying the structure of (co)pointed functors and monads. In the context of λ-coinduction, the availability of such additional structure has two effects: on the one hand, it allows us to use more expressive specifications and guarded recursive definitions; on the other hand, we have more structure available to carry out the constructions needed for proofs. In this section we develop the theory. In Section 4.4 we give a list of examples for λ-coinduction, all of which use some of the additional structure.

4.3.1 Adding structure to the algebra functor

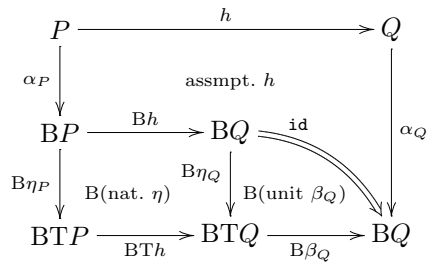
Considering a pointed functor or monad instead of the plain algebra functor S does not create new proof obligations in the context of λ-coinduction: the presence of the extra structure requires us to show that all algebra operations under consideration satisfy the unit or multiplication law. But the proofs of Theorems 4.1.2 and 4.2.2 mentioned only the algebra operation on the final B-coalgebra. We have shown already that this algebra operation satisfies the unit or multiplication law in case the distributive law λ does, namely in the proofs of Corollaries 3.4.12 and 3.4.19 in Section 3.4. So Theorems 4.1.2 and 4.2.2 hold in the setting of pointed functors or monads as well. Still, we can benefit from the additional structure: First, the unit natural transformation of a pointed functor or monad allows us to conclude that λ-coinduction and λ-coiteration are not merely variations but conservative extensions of the standard proof and definition principle, as we shall prove for Lemma 4.3.1. Second, we can use the additional structure for the constructions needed in the proofs. We show this in Lemma 4.3.2 for a pointed functor and in Lemma 4.3.3 for a monad. In the latter case the new construction allows us to dispense with our assumption on the underlying category. Moreover, for any given guarded recursive definition ϕ we can use the monad structure to construct a minimal model of λ in which ϕ has a solution. So we can guarantee solutions independent of whether or not the behaviour functor has a final coalgebra.

λ-coiteration extending coiteration

If the algebras under consideration satisfy a unit law, we can show that λ-coinduction is a conservative extension of standard coinduction.

Lemma 4.3.1 *Let λ be a distributive law of a pointed functor* $\langle \mathrm{T}, \eta \rangle$ *over a functor* B. *(i) The coiteration definition schema is generalised by the λ-coiteration schema, since every coiterative arrow h from the* B*-coalgebra* $\langle P, \alpha_P \rangle$ *to a final* B*-coalgebra* $\langle \mathcal{F}, \omega \rangle$ *is a λ-coiterative arrow from the* BT*-coalgebra* $\langle P, \mathrm{B}\eta_P \circ \alpha_P \rangle$ *to* $\langle \mathcal{F}, \omega \rangle$. *(ii) Similarly, for λ-bialgebras the λ-coinduction proof principle generalises the standard coinduction principle in that every bisimulation is a λ-bisimulation.*

Proof: Both statements easily follow from this observation: for a B-coalgebra $\langle P, \alpha_P \rangle$ and a $\langle \langle \mathrm{T}, \eta \rangle, \mathrm{B} \rangle$-bialgebra $\langle Q, \beta_Q, \alpha_Q \rangle$ (cf. Def. 3.4.11) a B-coalgebra homomorphism $h : \langle P, \alpha_P \rangle \to \langle Q, \alpha_Q \rangle$ is also a solution of the guarded recursive definition $\mathrm{B}\eta_P \circ \alpha_P$ in $\langle Q, \beta_Q, \alpha_Q \rangle$, as the diagram below shows.



Note that for this observation we need the unit law of the $\langle \mathrm{T}, \eta \rangle$-algebra $\langle Q, \beta_Q \rangle$ only. It holds without assuming that the bialgebra under consideration is a λ-bialgebra and it does not use the unit law for λ. The latter comes into play in the proof of item (i), where we take $\langle Q, \beta_Q, \alpha_Q \rangle$ to be the final λ-bialgebra $\langle \mathcal{F}, \beta, \omega \rangle$ from Theorem 3.2.3 (ii): as we have stated in Corollary 3.4.12, the algebra structure β satisfies the unit law if λ does.

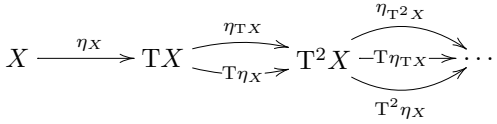
□

Pointed Functors

Next we study the impact of the unit of a pointed functor $\langle \mathrm{T}, \eta \rangle$ on the construction of a B-coalgebra from a BT-coalgebra $\langle X, \phi \rangle$. The proofs of Theorems 4.1.2 and 4.2.2 used a construction that turned a guarded recursive definition $\phi : X \to \mathrm{BT}X$ into a B-coalgebra on the carrier $\mathrm{H}X = X + \mathrm{T}X + \mathrm{T}^2X + \ldots$ (note that the algebra functor is now T instead of S). We can use the unit η to represent the set X in $\mathrm{T}X$, which enables us to work with a quotient of the carrier of the above B-coalgebra only. The unit law for λ guarantees that the previous coalgebra structure defines a coalgebra structure on the quotient.

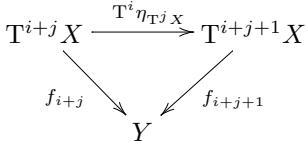
Apart from making the coalgebra smaller, this new construction is better behaved in the presence of a copointed behaviour functor, as we shall show in Section 4.3.2.

We modify the definition of the functor $\mathrm{H} := \coprod_{i\in\mathbb{N}} \mathrm{T}^i$ from Def. 4.1.4 (with S replaced by T) as follows: for any set X, let $\mathrm{H}X$ be the colimit of the diagram pictured below.[1]



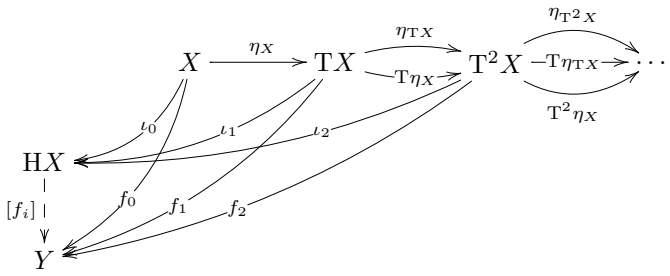
(4.13)

This of course requires that this colimit exists in the underlying category, which is the case for Set. The definition of a colimit moreover provides arrows $\iota_i : \mathrm{T}^i X \to \mathrm{H}X$ for all $i \in \mathbb{N}$ satisfying $\iota_{i+j} = \iota_{i+j+1} \circ \mathrm{T}^i \eta_{\mathrm{T}^j X}$ for all $i, j \in \mathbb{N}$.



(4.14)

And for functions $(f_i : \mathrm{T}^i X \to Y)_{i\in\mathbb{N}}$ such that the triangles above commute for all $i, j \in \mathbb{N}$ there is a unique arrow $[f_i] : \mathrm{H}X \to Y$ satisfying $f_j = [f_i] \circ \iota_j$ for all j. The situation is pictured in the diagram below.



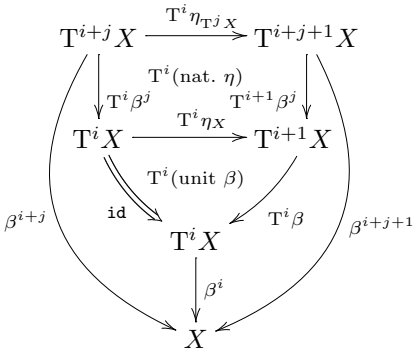
The action of H on an arrow $h : X \to Y$ is defined as $\mathrm{H}h := [\iota_i \circ \mathrm{T}^i h]$. The functions $f_i = \iota_i \circ \mathrm{T}^i h$ used in this definition make the triangles in (4.14) commute, as one easily verifies using the naturality of η.

Lemma 4.3.2 *Let $\langle \mathrm{T}, \eta\rangle$ be a pointed functor. The functions and natural transformations $\beta^* : \mathrm{H}X \to X$ (for $\beta : \mathrm{T}X \to X$ satisfying the unit law), $\lambda^* : \mathrm{HB} \Rightarrow \mathrm{BH}$ (for a distributive law λ of $\langle \mathrm{T}, \eta\rangle$ over B), and $\chi : \mathrm{HT} \Rightarrow \mathrm{H}$ can be defined as in Def. 4.1.4 still (with S replaced by T) if we modify the definition of the functor H to be a colimit of the diagram (4.13).*

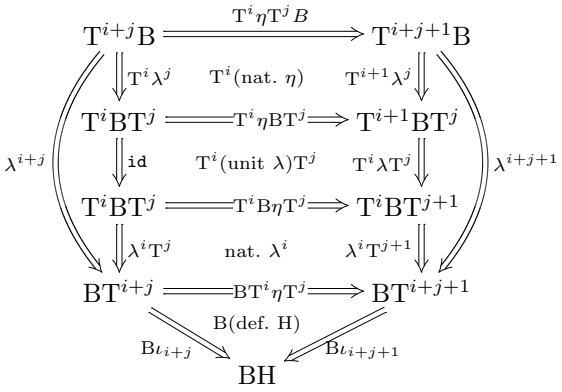
[1] Lenisa [Len99a] (see also [LPW00] for an improved presentation) describes a similar construction but with arrows $\eta_{\mathrm{T}^j X} : \mathrm{T}^j X \to \mathrm{T}^{j+1} X$ for all $j \in \mathbb{N}$ only. For now this was sufficient, but our choice has advantages when B is replaced by a copointed functor,as we shall see in the next section.

Proof: For all three cases we have to check that the arrows from HX are defined by functions $(f_i : \mathrm{T}^i X \to Y)_{i\in\mathbb{N}}$ making the triangles (4.14) commute.

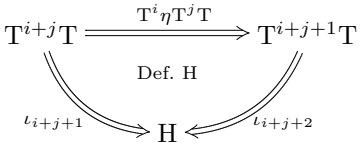
- The extension of an algebra $\langle X, \beta\rangle$ of the pointed functor $\langle \mathrm{T}, \eta\rangle$ to an algebra $\langle X, \beta^*\rangle$ for the pointed functor $\langle \mathrm{H}, \iota_0\rangle$ is justified by the unit law for β, as the diagram below shows for all $i, j \in \mathbb{N}$. Note that the definition of β^i from Def. 4.1.4 (with S exchanged by T of course) easily implies $\beta^{i+j} = \beta^i \circ \mathrm{T}^i\beta^j$ for all $i, j \in \mathbb{N}$.



- With the unit law of λ the definition of the lifting $\lambda^* : \mathrm{HB} \Rightarrow \mathrm{BH}$ does not cause problems, as the diagram below shows. Note that the definition of λ^i in Def. 4.1.4 easily implies $\lambda^{i+j} = \lambda^i\mathrm{T}^j \circ \mathrm{T}^i\lambda^j$ for all $i, j \in \mathbb{N}$.



- The definition of $\chi : \mathrm{HT} \Rightarrow \mathrm{H}$ as $\chi := [\iota_{i+1}]$ remains valid as well. The corresponding diagrams commute trivially, but note that we need the arrows $\mathrm{T}^i\eta\mathrm{T}^{j+1}$ to be present in the diagram (4.13) defining H in order to establish the commutativity of the diagram (4.14) for $\mathrm{T}^i\eta\mathrm{T}^j$:



□

With this lemma we have actually reproved Theorem 4.1.2 using a B-coalgebra $\langle \mathrm{H}X, \alpha^\phi \rangle$ with $\alpha^\phi := \mathrm{B}\chi_X \circ \lambda^*_{\mathrm{T}X} \circ \mathrm{H}\phi$ based on the modified definition of H as a colimit of the diagram (4.13).

At this point, the extra complication caused by the arrows in the diagram (4.13) does not really pay off: The carrier of our coalgebra is a bit "smaller", but we still need an assumption on the underlying category of a similar type, namely that the colimit of the diagram exists. But, when we come to study copointed functors on the coalgebra side, it will turn out that this construction has the advantage of yielding a coalgebra for the copointed functor.

Monads

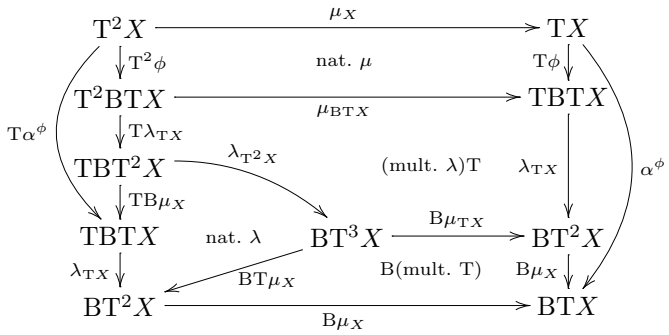
In the case of a distributive law λ of a monad $\langle \mathrm{T}, \eta, \mu \rangle$ over the functor B the construction of a B-coalgebra from a BT-coalgebra can be simplified considerably. Carrying further the above approach, we can add all arrows $\mathrm{T}^i\mu_{\mathrm{T}^j X} : \mathrm{T}^{i+j+2}X \Rightarrow \mathrm{T}^{i+j+1}X$ to the diagram (4.13) defining HX and check whether we can still define the required structures. But, as one can easily verify, the colimit of the described diagram is TX itself. So we obtain H = T, which renders the definition of the liftings β^* and λ^* unnecessary, and $\chi = \mu$. One advantage is that we do not need an assumption on the underlying category about the existence of the colimit any more. Furthermore, different from χ before, the components of the natural transformation μ are T-algebra structures, which allows us to extend the constructed B-coalgebras themselves into λ-bialgebras, as we shall demonstrate.

Lemma 4.3.3 *Let λ be a distributive law of a monad $\langle \mathrm{T}, \eta, \mu \rangle$ over a functor* B*, and let $\phi : X \to \mathrm{TB}X$ be a guarded recursive definition.*

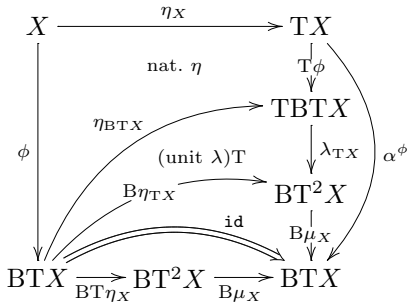
(i) *With $\alpha^\phi := \mathrm{B}\mu_X \circ \lambda_{\mathrm{T}X} \circ \mathrm{T}\phi$ and T_λ from Lemma 3.2.5 we get that μ_X is a* B*-coalgebra homomorphism from $\mathrm{T}_\lambda\langle X, \alpha^\phi \rangle$ to $\langle X, \alpha^\phi \rangle$. In other words, $\langle \mathrm{T}X, \mu_X, \alpha^\phi \rangle$ is a λ-bialgebra.*

(ii) *The arrow $\eta_X : X \to \mathrm{T}X$ is a solution of the guarded recursive definition ϕ in the above λ-bialgebra $\langle \mathrm{T}X, \mu_X, \alpha^\phi \rangle$.*

Proof: Item (i) is shown by the following diagram.



Item (ii) is proved by the commutative diagram below. The two lower triangles without inscription arise by applying the functor B to the two unit laws for T.



□

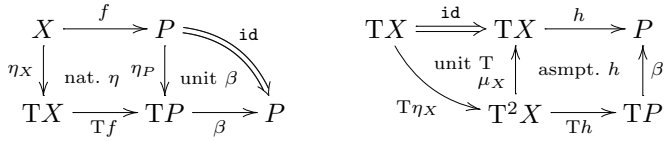
Note that item (i) above is actually an equivalence, since the following converse statement is true as well: whenever $\langle \mathrm{T}X, \mu_X, \alpha\rangle$ is a λ-bialgebra then $\alpha = \alpha^\phi$ for some $\phi : X \to \mathrm{BT}X$. For a proof we take $\phi = \phi^\alpha := \alpha \circ \eta_X$. It can easily be shown that the mappings $\phi \mapsto \alpha^\phi$ and $\alpha \mapsto \phi^\alpha$ (for α satisfying the condition above) are inverses of each other.

In the case of a distributive law λ of a monad over a functor every guarded recursive definition ϕ thus has what we can call a *free* λ-bialgebra $\langle \mathrm{T}X, \mu_X, \alpha^\phi\rangle$. And ϕ has a solution in its free λ-bialgebra, namely η_X – so we have shown that it has a solution even without assuming that a final B-coalgebra exists.

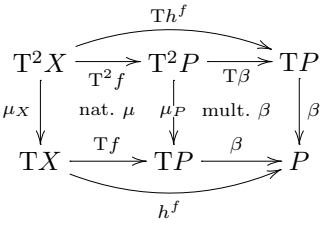
Moreover, this solution is minimal in the sense that every other solution in any λ-bialgebra factors through it. This is expressed by the following statement.

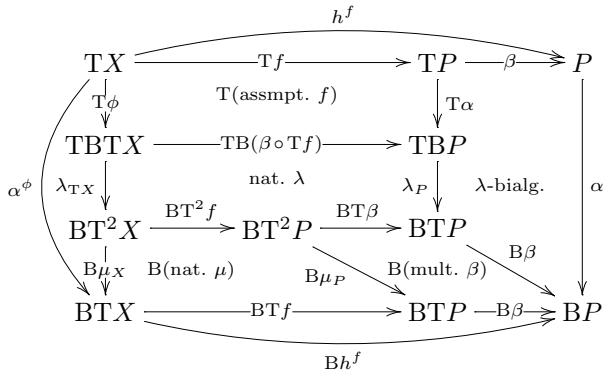
Lemma 4.3.4 *Let λ be a distributive law of a monad $\langle \mathrm{T}, \eta, \mu\rangle$ over a functor* B*, let $\phi : X \to \mathrm{BT}X$ be a guarded recursive definition, and let $\langle P, \beta, \alpha\rangle$ be a λ-bialgebra. There is a one-to-one correspondence between solutions $f : X \to P$ of ϕ in $\langle P, \beta, \alpha\rangle$ and bialgebra homomorphisms h from $\langle \mathrm{T}X, \mu_X, \alpha^\phi\rangle$ (as defined in Lemma 4.3.3) to $\langle P, \beta, \alpha\rangle$. The correspondence is given by $f \mapsto \beta \circ \mathrm{T}f$ and $h \mapsto h \circ \eta_X$.*

Proof: We first show that the two constructions are inverse, i.e. $f = \beta \circ \mathrm{T}f \circ \eta_X$ and $h = \beta \circ \mathrm{T}(h \circ \eta_X)$. The identities follow from the naturality of η, the unit law for β, one of the unit laws of the monad, and the assumption on h to be an algebra homomorphism:



That $h^f = \beta \circ \mathrm{T}f$ is a bialgebra homomorphism is proved by the following two commuting diagrams, the first for the algebra part, the second for the coalgebras. The second one, which corresponds to the diagram in the proof of Theorem 4.1.2, uses our assumption that f is a solution of ϕ in $\langle P, \beta, \alpha \rangle$ and that the latter is a λ-bialgebra.





Finally, Lemma 4.2.5 proves that $f^h = h \circ \eta_X$ is a solution of ϕ in $\langle P, \beta, \alpha \rangle$.

□

We can further use the above lemma to show the validity of a λ-coinduction proof principle (cf. Theorem 4.1.2).

Corollary 4.3.5 *Let λ be a distributive law of a monad $\langle \mathrm{T}, \eta, \mu \rangle$ over a functor* B. *Every λ-bisimulation $R \subseteq P \times Q$ between two λ-bialgebras $\langle P, \beta_P, \alpha_P \rangle$ and*

$\langle Q, \beta_Q, \alpha_Q\rangle$ is contained in some (standard) bisimulation between $\langle P, \alpha_P\rangle$ and $\langle Q, \alpha_Q\rangle$.

Proof: Let $\phi : R \to \mathrm{BT}R$ witness the λ-bisimulation property. Let $r_P : \mathrm{T}R \to P$ and $r_Q : \mathrm{T}R \to Q$ be the bialgebra homomorphisms from $\langle \mathrm{T}R, \mu_R, \alpha^\phi\rangle$ to $\langle P, \beta_P, \alpha_P\rangle$ and $\langle Q, \beta_Q, \alpha_Q\rangle$ respectively that correspond to $\pi_1 : R \to P$ and $\pi_2 : R \to Q$ by Lemma 4.3.4. The correspondence further yields

$$\pi_1 = r_P \circ \eta_R \quad \text{and} \quad \pi_2 = r_Q \circ \eta_R.$$

With these equations, η_R witnesses, according to Lemma 2.3.5, that the image of $\langle \mathrm{T}R, r_P, r_Q\rangle$ contains R. The latter relation is a bisimulation between $\langle P, \alpha_P\rangle$ and $\langle Q, \alpha_Q\rangle$, as we conclude from the homomorphism property of r_P and r_Q together with Lemma 2.3.4.

□

Compared to Theorem 4.1.2 this statement does not need an assumption on the underlying category. Moreover, the constructed bisimulation is better behaved: it is a λ-bialgebra itself and it is a congruence.

Our statement about the validity of the λ-coiteration principle (Theorem 4.2.2) can be rephrased for monads as well, and again we can dispense with the assumption on the underlying category.

Corollary 4.3.6 *Let λ be a distributive law of a monad $\langle \mathrm{T}, \eta, \mu\rangle$ over a functor* B *and let $\langle \mathcal{F}, \omega\rangle$ be a final* B*-coalgebra. Every arrow $\phi : X \to \mathrm{BT}X$ uniquely determines a λ-coiterative morphism $f : X \to \mathcal{F}$ (cf. Def. 4.2.1).*

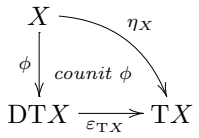
Proof: Let $h : \mathrm{T}X \to \mathcal{F}$ be the unique λ-bialgebra homomorphism from $\langle \mathrm{T}X, \mu_X, \alpha^\phi\rangle$ (cf. Lemma 4.3.3 (i)) to the final λ-bialgebra $\langle \mathcal{F}, \beta, \omega\rangle$ from Theorem 3.2.3 (ii) and Corollary 3.4.19. Since $\eta_X : X \to \mathrm{T}X$ is a solution of ϕ in the λ-bialgebra $\langle \mathrm{T}X, \mu_X, \alpha^\phi\rangle$ according to Lemma 4.3.3 (ii), we obtain from Lemma 4.2.5 that $f := h \circ \eta_X$ is a solution of ϕ in $\langle \mathcal{F}, \beta, \omega\rangle$. From Corollary 4.3.5 and the coinduction proof principle (cf. Corollary 2.3.9) it follows that this solution is unique.

□

4.3.2 Adding structure to the coalgebra functor

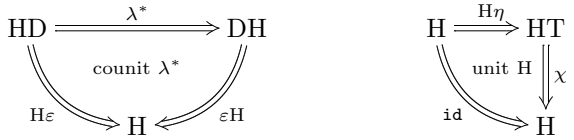
Next we will investigate how to adapt the λ-coinduction framework to the setting where the coalgebra functor comes as a copointed functor. The addition of extra structure on the coalgebra side is slightly more problematic than on the algebra side, because we now need to make sure that the B-coalgebras we construct from the guarded recursive definition ϕ satisfy the counit law. It turns out that for this we need to impose a suitable counit law on ϕ, and since this law involves a unit for the algebra functor, we need to assume that the latter comes as a pointed functor or monad.

Lemma 4.3.7 (i) *Let λ be a distributive law of a pointed functor $\langle \mathrm{T}, \eta \rangle$ over a copointed functor $\langle \mathrm{D}, \varepsilon \rangle$ (cf. Definitions 3.4.11 and 3.4.2) and let $\phi : X \to \mathrm{D}\mathrm{T}X$ be a guarded recursive definition. If ϕ satisfies the counit law pictured below then the* D*-coalgebra $\langle \mathrm{H}X, \alpha^{\phi} \rangle$ constructed in the proof of Theorem 4.1.2 with the modified definition of the functor* H *to be a colimit of the diagram (4.13) is a coalgebra for the copointed functor.*



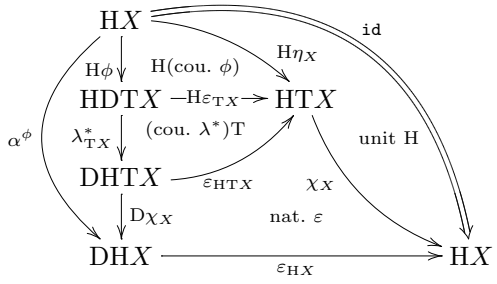
(ii) *The statement in* (i) *holds as well in the setting of a monad $\langle \mathrm{T}, \eta, \mu \rangle$ for the* B*-coalgebra $\langle \mathrm{T}X, \alpha^{\phi} \rangle$ constructed in Lemma 4.3.3.*

Proof: (i) To see that the coalgebra operation α^{ϕ} satisfies the counit law, we first claim the following two auxiliary identities for λ^* and χ from Def. 4.1.4 (with S replaced by T and B replaced by D).



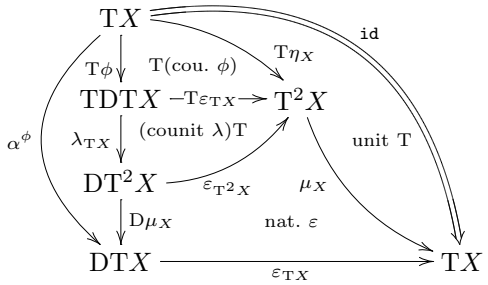
The left is the counit law for λ^* and the right bears similarity to one of the unit laws of a monad, so we call it the *unit law of* H. The counit law for λ^* follows immediately from corresponding counit laws for all λ^i from Def. 4.1.4, which are in turn easily obtained from the counit law of λ by induction on i. The unit law of H holds since both `id` and $\chi \circ \mathrm{H}\eta$ fit as $[\iota_i] : \mathrm{H} \Rightarrow \mathrm{H}$.

The counit law for α^{ϕ} is established by the diagram below.



(ii) That the construction in Lemma 4.3.3 defines a coalgebra for the copointed functor $\langle \mathrm{D}, \varepsilon \rangle$ is shown by a similar diagram. It is obtained from the one above essentially by replacing H with T and adapting the natural transformations

accordingly.



□

Note that the proof of the unit law for H in part (i) above relies on the identities $\iota_{i+1} \circ \mathrm{T}^i\eta_X = \iota_i$ given by the definition of H to be a colimit of the diagram (4.13). The argument does not work for the original definition of H to be just a coproduct in the proof of Theorem 4.1.2.[2] Once we put the arrows $\mathrm{T}^i\eta_X$ into the diagram of which $\mathrm{H}X$ is a colimit, we furthermore need to include all arrows $\mathrm{T}^i\eta_{\mathrm{T}^jX}$ for $i, j \in \mathbb{N}$ to be able to define χ, which in turn we needed for the definition of the coalgebra structure α^ϕ.

With the above lemma we can easily lift our statements about the λ-coinduction definition and proof principles to the setting of copointed functors. In the definition of a λ-bisimulation we now further assume that the required mediating DT-coalgebra structure satisfies the counit law from the above lemma.

Corollary 4.3.8 (i) *Let λ be a distributive law of the pointed functor $\langle \mathrm{T}, \eta\rangle$ over the copointed functor $\langle \mathrm{D}, \varepsilon\rangle$ on a category with colimits of the diagram (4.13). Every λ-bisimulation is contained in a (standard) bisimulation between the respective coalgebras. Moreover, if the copointed functor $\langle \mathrm{D}, \varepsilon\rangle$ has a final coalgebra, every guarded recursive specification $\phi : X \to \mathrm{DT}X$ satisfying the counit law from Lemma 4.3.7 uniquely defines a λ-coiterative arrow.*

(ii) *A corresponding statement is true in the case of a distributive law λ of a monad $\langle \mathrm{T}, \eta, \mu\rangle$ over a copointed functor $\langle \mathrm{D}, \varepsilon\rangle$ without any assumption on the underlying category.*

We have to admit that we do not have an elegant proof for the existence of a λ-coiterative arrow claimed in item (i) of the above corollary. The problem is that the arrow $[\mathrm{S}\iota_i]$ used in the proof of Theorem 4.2.2 cannot be defined with the modified definition of H by the diagram (4.13). The problem can be worked around by resorting to the original definition of H and applying a factorisation argument to cope with the counit law, but we feel that it should be possible to find a better argument.

In Section 3.4 our main candidate for a copointed functor $\langle \mathrm{D}, \varepsilon\rangle$ was the cofree copointed functor generated by a behaviour functor B (cf. Def. 3.4.7). Similar

[2] It neither does with Lenisa's construction [Len99a], where the diagram defining H just contains the arrows η_{T^iX} for $i \in \mathbb{N}$, so that we obtain the identities $\iota_{i+1} \circ \eta_{\mathrm{T}^iX} = \iota_i$ only.

to the situation there, we can characterise a guarded recursive definition $\tilde{\phi}$ satisfying the counit law by a plain function ϕ into a smaller set, and we can rephrase the definition of a solution and consequently the definition of a λ-bisimulation in terms of ϕ.

Lemma 4.3.9 *Let $\langle \mathrm{T}, \eta \rangle$ be a pointed functor, let* B *be a functor on a category* C *with binary products, and let λ be a distributive law of $\langle \mathrm{T}, \eta \rangle$ over the cofree copointed functor $\langle \mathrm{D}, \varepsilon \rangle$ generated by* B*.*

(i) *There is a one-to-one correspondence of guarded recursive definitions $\tilde{\phi} : X \to \mathrm{DT}X$ satisfying the counit law $\eta_X = \varepsilon_{\mathrm{T}X} \circ \tilde{\phi}$ and plain functions $\phi : X \to \mathrm{BT}X$.*

(ii) *A function $f : X \to P$ is a solution of the guarded recursive definition $\tilde{\phi} : X \to \mathrm{DT}X$ in the λ-bialgebra $\langle P, \beta, \tilde{\alpha} \rangle$ if and only if it makes the diagram below commute, where $\alpha : P \to \mathrm{B}P$ is the* B*-coalgebra operation corresponding to $\tilde{\alpha} : P \to \mathrm{D}P$ (cf. Corollary 3.4.8 (ii)).*

$$\begin{array}{ccccc} X & & \xrightarrow{\quad f \quad} & & P \\ {\scriptstyle\phi}\downarrow & & & & \downarrow{\scriptstyle\alpha} \\ \mathrm{BT}X & \xrightarrow[\mathrm{BT}f]{} & \mathrm{BT}P & \xrightarrow[\mathrm{B}\beta]{} & \mathrm{B}P \end{array}$$

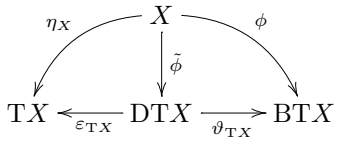
(iii) *Correspondingly, $R \subseteq P \times Q$ is a λ-bisimulation between the λ-bialgebras $\langle P, \beta_P, \tilde{\alpha}_P \rangle$ and $\langle Q, \beta_Q, \tilde{\alpha}_Q \rangle$ if and only if there exists a* BT*-coalgebra operation γ on R fitting into the diagram below, where α_P and α_Q are the* B*-coalgebras corresponding to $\tilde{\alpha}_P$ and $\tilde{\alpha}_Q$ respectively.*

$$\begin{array}{ccccc} P & \xleftarrow{\quad \pi_1 \quad} & R & \xrightarrow{\quad \pi_2 \quad} & Q \\ {\scriptstyle\alpha_P}\downarrow & & {\scriptstyle|}\,{\scriptstyle\gamma}\downarrow & & \downarrow{\scriptstyle\alpha_Q} \\ \mathrm{B}P & \xleftarrow[\mathrm{B}(\beta_P \circ \mathrm{T}\pi_1)]{} & \mathrm{BT}R & \xrightarrow[\mathrm{B}(\beta_Q \circ \mathrm{T}\pi_2)]{} & \mathrm{B}Q \end{array}$$

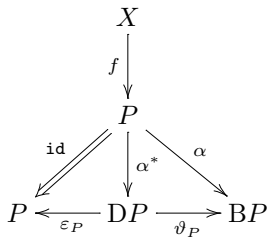
The diagrams in (ii) are precisely those from the setting of a plain behaviour functor B. The difference is that this time the bialgebras can be models of a specification given by a distributive law λ of the pointed functor $\langle \mathrm{T}, \eta \rangle$ (or monad $\langle \mathrm{T}, \eta, \mu \rangle$) over the *cofree copointed functor* $\langle \mathrm{D}, \varepsilon \rangle$. As we explained in Section 3.6, using distributive laws over the cofree copointed functor instead of those over the plain functor B we extend the class of specifications. So the above statement generalises the plain λ-coiteration framework.

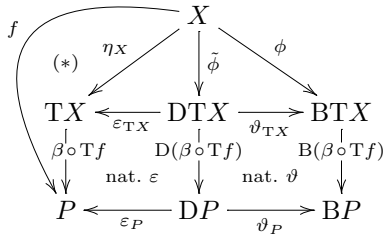
Of course the statement holds as well for monads instead of pointed functors, just that the extra structure will not be used.

Proof: Item (i) immediately follows from the definition principle in Corollary 3.4.8 (i):



For item (ii) we show that the composites appearing in the condition for a solution and the diagram in the statement, i.e. $\alpha \circ f$ and $\alpha^* \circ f$ as well as $\mathrm{B}(\beta \circ \mathrm{T}f) \circ \phi$ and $\mathrm{D}(\beta \circ \mathrm{T}f) \circ \tilde{\phi}$, correspond to each other under f (cf. Remark 3.4.9):





That we indeed have $f = \beta \circ \mathrm{T}f \circ \eta_X$, as exploited in $(*)$ in the second diagram, easily follows from the unit law for β and the naturality of η (as in the proof of Lemma 4.3.4).

The proof of item (iii) is similar.

□

4.4 Instances of λ-coinduction

In this section we present some concrete instances of the λ-coinduction schemata. We discuss several known as well as a new definition and proof principle. For the former our framework yields a uniform and more simple justification. The new definition principle guarantees unique solutions to guarded recursive definitions involving operators defined by the abstract GSOS format. The corresponding proof principle justifies a bisimulation up-to-context technique for the same operators. All examples involve algebra functors carrying a monad structure which is respected by the distributive law.

Before we consider more interesting instances, note the following trivial one: the identity functor Id also carries the structure of a monad, namely $\langle \mathrm{Id}, \mathtt{id}, \mathtt{id} \rangle$, which distributes via the identity natural transformation $\lambda = \mathtt{id}\mathrm{B}$ over every behaviour functor B. With this instance, the λ-coiteration schema simplifies to the coiteration schema and λ-bisimulations are just ordinary bisimulations. So the standard coinduction principles are an instance of λ-coinduction themselves.

4.4.1 Primitive corecursion

For this instance we assume that we are given a behaviour functor B on a category C with binary coproducts such that B has a final coalgebra $\langle \mathcal{F}, \omega \rangle$. The following defines a monad $\langle \mathrm{T}, \eta, \mu \rangle$.

$$\begin{array}{lcl} \mathrm{T} & := & \mathrm{Id} + \underline{\mathcal{F}} \\ \eta & := & \iota_1 : \mathrm{Id} \Rightarrow \mathrm{Id} + \underline{\mathcal{F}} \\ \mu & := & [\mathtt{id}, \iota_2] : (\mathrm{Id} + \underline{\mathcal{F}}) + \underline{\mathcal{F}} \Rightarrow \mathrm{Id} + \underline{\mathcal{F}} \end{array} \tag{4.15}$$

This monad distributes over B via $\lambda : \mathrm{TB} \Rightarrow \mathrm{BT}$ defined by

$$\lambda := [\mathrm{B}\iota_1, \mathrm{B}\iota_2 \circ \omega] : \mathrm{B} + \underline{\mathcal{F}} \Rightarrow \mathrm{B}(\mathrm{Id} + \underline{\mathcal{F}}). \tag{4.16}$$

It is straightforward to check that λ respects the monad structure, i.e. it satisfies the corresponding unit and multiplication law.

For a B-coalgebra $\langle P, \alpha \rangle$ and T_λ from Lemma 3.2.5 we get

$$\mathrm{T}_\lambda \langle P, \alpha \rangle = \langle P + \mathcal{F}, [\mathrm{B}\iota_1, \mathrm{B}\iota_2] \circ (\alpha + \omega) \rangle.$$

This is the categorical coproduct of $\langle P, \alpha \rangle$ and $\langle \mathcal{F}, \omega \rangle$ in $\mathsf{Coalg}_{\mathrm{B}}$. A homomorphism from $\mathrm{T}_\lambda \langle P, \alpha \rangle$ to another B-coalgebra $\langle Q, \alpha_Q \rangle$ is thus of the shape $[f, g]$ for two homomorphisms $f : \langle P, \alpha \rangle \to \langle Q, \alpha_Q \rangle$ and $g : \langle \mathcal{F}, \omega \rangle \to \langle Q, \alpha_Q \rangle$. This implies that the unique T-algebra operation extending the final coalgebra $\langle \mathcal{F}, \omega \rangle$ to a final λ-bialgebra is $[\mathtt{id}, \mathtt{id}] : \mathcal{F} + \mathcal{F} \to \mathcal{F}$. From this we immediately obtain the known definition principle of *primitive corecursion* [Geu92]:

Corollary 4.4.1 *Let* B *be a functor on a category with binary coproducts which has a final coalgebra* $\langle \mathcal{F}, \omega \rangle$*. Every arrow* $\phi : X \to \mathrm{B}(X + \mathcal{F})$ *uniquely determines a morphism* $f : X \to \mathcal{F}$ *fitting into the diagram below.*

$$\begin{array}{ccc} X & \overset{\exists! f}{- - - \succ} & \mathcal{F} \\ {\scriptstyle \forall \phi}\downarrow & \textit{primitive corecursion} & \downarrow{\scriptstyle \omega} \\ \mathrm{B}(X + \mathcal{F}) & \underset{\mathrm{B}[f, \mathtt{id}]}{- - \succ} & \mathrm{B}\mathcal{F} \end{array}$$

We say that ϕ *defines* f *by* primitive corecursion.

The principle arises as the dual of (a categorical presentation of) the *primitive recursion* schema for arrows out of an initial algebra. It is studied e.g. by Vene and Uustalu [VU98].

Proof: The principle arises as the instance of Corollary 4.3.6 with the monad and distributive law from (4.15) and (4.16). We use that, as mentioned above, the T-algebra operation β on the final B-coalgebra is $\beta = [\mathtt{id}, \mathtt{id}]$, which yields $\beta \circ \mathrm{T}f = [f, \mathtt{id}]$.

□

Application

Primitive corecursion extends the coiteration schema (cf. Def. 2.3.13) in that it offers the additional possibility to directly specify elements from the final coalgebra to be the successors. As it were, we can step out of the recursion at any point. To illustrate this, consider the example of the function

$$\mathtt{insert} : \mathbb{R} \times \mathbb{R}^\omega \to \mathbb{R}^\omega$$

that is supposed to insert a real number into a stream at the first position such that the next element in the stream will be greater. It can be specified as follows, where $r \in \mathbb{R}$ and $\mathbf{s} = s_0 : \mathbf{s}' \in \mathbb{R}^\omega$:

$$\mathtt{insert}(r, \mathbf{s}) = \begin{cases} s_0 : \mathtt{insert}(r, \mathbf{s}') & \text{if } s_0 \leq r, \\ r : \mathbf{s} & \text{otherwise.} \end{cases} \tag{4.17}$$

In the second case we step out of the recursion, because we can give the whole tail after the insertion of r at once.

The specification is justified by primitive recursion, because, with the final stream coalgebra $\langle \mathbb{R}^\omega, \langle h, t \rangle \rangle$ (cf. equation (2.7) on page 32), the equation can equivalently be expressed by the following diagram.

$$\begin{array}{ccc} \mathbb{R} \times \mathbb{R}^\omega & \xrightarrow{\quad\mathtt{insert}\quad} & \mathbb{R}^\omega \\ {\scriptstyle\phi}\downarrow & & \downarrow{\scriptstyle\langle h,t\rangle} \\ \mathbb{R} \times ((\mathbb{R} \times \mathbb{R}^\omega) + \mathbb{R}^\omega) & \xrightarrow[\mathtt{id}\times[\mathtt{insert},\mathtt{id}]]{} & \mathbb{R} \times \mathbb{R}^\omega \end{array}$$

The function ϕ is defined, for all $r \in \mathbb{R}$ and $\mathbf{s} = s_0 : \mathbf{s}' \in \mathbb{R}^\omega$, as

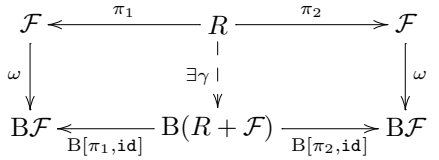
$$\phi(r, \mathbf{s}) := \begin{cases} \langle s_0, \iota_1(r, \mathbf{s}') \rangle & \text{if } s_0 \leq r, \\ \langle r, \iota_2(\mathbf{s}) \rangle & \text{otherwise.} \end{cases}$$

The corresponding proof principle

The λ-coinduction proof principle instantiated with the monad $\langle \mathrm{T}, \eta, \mu \rangle$ and the distributive law λ from (4.15) and (4.16) is most useful for relations on the

final B-coalgebra $\langle \mathcal{F}, \omega \rangle$ itself. We will call such a λ-bisimulation a *bisimulation up-to-equality*.

Definition 4.4.2 *Let* B *be a functor on a category with binary coproducts and let* $\langle \mathcal{F}, \omega \rangle$ *be a final* B*-coalgebra. We call* $R \subseteq \mathcal{F} \times \mathcal{F}$ *a* **bisimulation up-to-equality** *if there is a function* $\gamma : R \to \mathrm{B}(R + \mathcal{F})$ *making both parts of the diagram below commute.*



From Corollary 4.3.5 we immediately get a proof principle based on bisimulations up-to-equality:

Corollary 4.4.3 *Every bisimulation up-to-equality is contained in some (standard) bisimulation on* $\langle \mathcal{F}, \omega \rangle$.

Analysing the proof of Corollary 4.3.5 we find that the standard bisimulation constructed for a bisimulation up-to-equality $R \subseteq \mathcal{F} \times \mathcal{F}$ is actually $R \cup \Delta_{\mathcal{F}}$, where $\Delta_{\mathcal{F}} := \{\langle x, x \rangle \mid x \in \mathcal{F}\}$ is the diagonal of $\mathcal{F}$ (or the equality relation). If we use $R \cup \Delta_{\mathcal{F}}$ in the first place, we have to check the bisimulation condition for the pairs in $\Delta_{\mathcal{F}}$ as well, but for them it is always satisfied. This principle is a rather simple extension of the standard one, but it is still quite useful for many examples.

Application

As a simple example, we can use the principle to prove that `insert` from (4.17) commutes with the elementwise application of any strictly monotonic function $g : \mathbb{R} \to \mathbb{R}$, i.e. for all $r \in \mathbb{R}$ and $\mathbf{s} \in \mathbb{R}^\omega$ we have

$$\mathtt{insert}\big(g(r), \mathtt{map}_g(\mathbf{s})\big) = \mathtt{map}_g\big(\mathtt{insert}(r, \mathbf{s})\big), \tag{4.18}$$

where $\mathtt{map}_g$ is defined as in equation (2.8) on page 32.

We again use that $\langle \mathbb{R}^\omega, \langle h, t \rangle \rangle$ is a final coalgebra for $\mathrm{B} := \underline{\mathbb{R}} \times \mathrm{Id}$. For this instance, a bisimulation up-to-equality from Def. 4.4.2 is a relation $R \subseteq \mathbb{R}^\omega \times \mathbb{R}^\omega$ such that for all $\langle \mathbf{s}, \mathbf{t} \rangle \in R$ with $\mathbf{s} = s_0 : \mathbf{s}'$ and $\mathbf{t} = t_0 : \mathbf{t}'$ we have that

$$s_0 = t_0 \quad \text{and} \quad \langle \mathbf{s}', \mathbf{t}' \rangle \in R \cup \Delta_{\mathbb{R}^\omega}.$$

To prove (4.18) we show that the relation

$$R := \big\{ \langle \mathtt{insert}(g(r), \mathtt{map}_g(\mathbf{s})), \mathtt{map}_g(\mathtt{insert}(r, \mathbf{s})) \rangle \;\big|\; r \in \mathbb{R},\ \mathbf{s} \in \mathbb{R}^\omega \big\}$$

has this property. Take any pair

$$\langle \mathtt{insert}(g(r), \mathtt{map}_g(s_0 : \mathbf{s}')), \mathtt{map}_g(\mathtt{insert}(r, s_0 : \mathbf{s}'))\rangle \in R.$$

We first consider the case $s_0 \leq r$. The monotonicity of g implies $g(s_0) \leq g(r)$, so we obtain

$$\begin{array}{rcl} \mathtt{insert}(g(r), \mathtt{map}_g(s_0 : \mathbf{s}')) & = & \mathtt{insert}(g(r), g(s_0) : \mathtt{map}_g(\mathbf{s}')) \\ & = & g(s_0) : \mathtt{insert}(g(r), \mathtt{map}_g(\mathbf{s}')), \\ \mathtt{map}_g(\mathtt{insert}(r, (s_0 : \mathbf{s}'))) & = & \mathtt{map}_g(s_0 : \mathtt{insert}(r, \mathbf{s}')) \\ & = & g(s_0) : \mathtt{map}_g(\mathtt{insert}(r, \mathbf{s}')). \end{array}$$

We find that the two heads are equal and the tails are related again, namely

$$\langle \mathtt{insert}(g(r), \mathtt{map}_g(\mathbf{s}')), \mathtt{map}_g(\mathtt{insert}(r, \mathbf{s}'))\rangle \in R,$$

so the pair satisfies the bisimulation up-to-equality condition.

In case $s_0 > r$, which implies $g(s_0) > g(r)$ by strict monotonicity of g, we find

$$\begin{array}{rcl} \mathtt{insert}(g(r), \mathtt{map}_g(s_0 : \mathbf{s}')) & = & \mathtt{insert}(g(r), g(s_0) : \mathtt{map}_g(\mathbf{s}')) \\ & = & g(r) : (g(s_0) : \mathtt{map}_g(\mathbf{s}')), \\ \mathtt{map}_g(\mathtt{insert}(r, (s_0 : \mathbf{s}'))) & = & \mathtt{map}_g(r : (s_0 : \mathbf{s}')) \\ & = & g(r) : \mathtt{map}_g(s_0 : \mathbf{s}') \\ & = & g(r) : (g(s_0) : \mathtt{map}_g(\mathbf{s}')). \end{array}$$

The two heads are equal again, and the requirement on the tails, which are both $g(s_0) : \mathtt{map}_g(\mathbf{s}')$, is satisfied as well: they are not related by R, as the definition of a standard bisimulation would require, but they are equal.

4.4.2 The dual of course-of-value iteration

Another interesting instance of the λ-coinduction framework arises from the trivial observation that every functor B distributes over itself via the identity natural transformation $\lambda := \mathtt{id} : \mathrm{B}^2 \Rightarrow \mathrm{B}^2$. For this distributive law, the λ-bialgebras are triples $\langle P, \beta, \alpha\rangle$ where β is a B-algebra, and α a B-coalgebra operation on P such that the following diagram commutes:

$$\begin{array}{ccc} \mathrm{B}P & \xrightarrow{\beta} & P \\ {\scriptstyle \mathrm{B}\alpha}\downarrow & & \downarrow{\scriptstyle \alpha} \\ \mathrm{B}^2P & \xrightarrow[\mathrm{B}\beta]{} & \mathrm{B}P \end{array}$$

The unique B-algebra operation on the carrier of a final coalgebra $\langle \mathcal{F}, \omega \rangle$ satisfying the above is the inverse ω^{-1} of the final coalgebra map, which exists by Lambek's Lemma (cf. Lemma 2.3.10). So a final λ-bialgebra is $\langle \mathcal{F}, \omega^{-1}, \omega \rangle$.

The λ-coiteration schema arising directly from this instance is of rather limited use. As it were, for any argument of the function to be defined we would have to give precisely two steps of the behaviour of the result. The schema becomes more flexible when we consider the free monad generated by B instead, provided it exists, together with a lifting of the above distributive law. So let us assume that B freely generates a monad $\langle \mathrm{T}, \eta, \mu \rangle$ (note that before we were always considering free monads of another functor S, usually regarded as a signature functor). As explained in Section 3.6 (cf. Lemmata 3.6.1 and 3.6.3), the above distributive law λ of B over B lifts to a distributive law λ^* of $\langle \mathrm{T}, \eta, \mu \rangle$ over B such that a $\langle \mathrm{B}, \mathrm{B} \rangle$-bialgebra $\langle P, \beta, \alpha \rangle$ is a λ-bialgebra if and only if $\langle P, \beta^*, \alpha \rangle$ is a λ^*-bialgebra, where β^* is the inductive extension of β (cf. Lemma 2.2.10). This means that the final λ^*-bialgebra will be $\langle \mathcal{F}, (\omega^{-1})^*, \omega \rangle$.

The corresponding instance of Corollary 4.3.6 yields a coinductive definition principle, which, for every arrow $\phi : X \to \mathrm{BT}X$, guarantees the unique existence of a morphism $f : X \to \mathcal{F}$ making the diagram below commute.

$$\begin{array}{ccc} X & \overset{\exists! f}{- - - \succ} & \mathcal{F} \\ {\scriptstyle \forall \phi}\downarrow & & \downarrow{\scriptstyle \omega} \\ \mathrm{BT}X & \underset{\mathrm{B}((\omega^{-1})^* \circ \mathrm{T}f)}{- - - \succ} & \mathrm{B}\mathcal{F} \end{array} \tag{4.19}$$

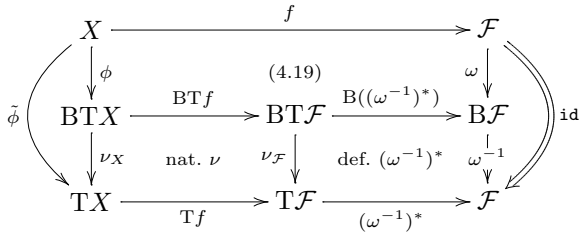
To obtain a slightly more intuitive presentation of this schema, we define the following notion: we call a specification $\tilde{\phi} : X \to \mathrm{T}X$ *guarded* if it factors as $\tilde{\phi} = \nu_X \circ \phi$ for some $\phi : X \to \mathrm{BT}X$, where $\nu : \mathrm{BT} \Rightarrow \mathrm{T}$ is the natural transformations from the Definition 2.2.7 of a free monad. So guarded specifications are those that assign to each element in X an arbitrary finite but positive number of B-steps.

Corollary 4.4.4 *Every guarded specification $\tilde{\phi} : X \to \mathrm{T}X$ determines a unique arrow $f : X \to \mathcal{F}$ such that the following diagram commutes:*

$$\begin{array}{ccc} X & \overset{\exists! f}{- - \succ} & \mathcal{F} \\ {\scriptstyle \forall \tilde{\phi}}\downarrow & & \uparrow{\scriptstyle (\omega^{-1})^*} \\ \mathrm{T}X & \underset{\mathrm{T}f}{- - \succ} & \mathrm{T}\mathcal{F} \end{array}$$

Proof: Let $\tilde{\phi}$ factor as $\nu_X \circ \phi$. The correspondence of this formulation with the one in (4.19) is explained by the diagram below. The proof of the converse

is similar.



□

Application

We will illustrate this definition schema again in the context of streams, this time over two sets A and B, i.e. A^ω and B^ω. Assume that A is a set of high-level instructions and B a set of micro-code instructions. Moreover, we are given a translation function t that tells us how to realise any high-level instruction by a nonempty sequence of low level instructions. We want to define the function $\mathtt{trans}_t$ that turns an infinite stream of high-level instructions into the corresponding stream of micro-code instructions. Let

$$B^+ := \{\langle b_1, \ldots, b_n\rangle \mid n > 0,\ b_i \in B\}$$

denote the set of nonempty, finite words over B. The single instruction translation function has the type $t : A \to B^+$ and its sought extension to infinite streams is a mapping $\mathtt{trans}_t : A^\omega \to B^\omega$. Writing $t(a_i) = \langle b^i_1, \ldots, b^i_{n_i}\rangle$ for $a_i \in A$ we want to obtain

$$\mathtt{trans}_t(\langle a_0, a_1, \ldots\rangle) = \langle b^0_1, \ldots, b^0_{n_0}, b^1_1, \ldots, b^1_{n_1}, \ldots\rangle.$$

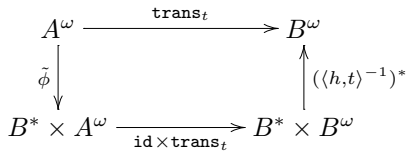
To obtain a coalgebraic justification of this specification, we rewrite the equation as

$$\mathtt{trans}_t(a : \mathbf{s}) = \langle b_1, \ldots, b_n\rangle : \mathtt{trans}(\mathbf{s}), \tag{4.20}$$

where $\langle b_1, \ldots, b_n\rangle : \mathbf{t}$ is a shorthand for $b_1 : (\ldots : (b_n : \mathbf{t})\ldots)$ and $t(a) = \langle b_1, \ldots, b_n\rangle$. This definition is not coiterative, because for $n > 1$ it specifies in one step a longer prefix of the resulting stream instead of only the first element. It fits into the schema from Corollary 4.4.4 though, as the following consideration shows. Similar to equation (2.7) on page 32, we obtain a final coalgebra $\langle B^\omega, \langle h, t\rangle\rangle$ of the behaviour functor $\mathrm{B} := \underline{B} \times \mathrm{Id}$. The free monad over this behaviour functor is $\mathrm{T} \simeq \underline{B^*} \times \mathrm{Id}$, where $B^* = B^+ \cup \{\varepsilon\}$ denotes the possibly empty, finite words over B. We find that $(\langle h, t\rangle^{-1})^* : \underline{B^*} \times B^\omega \to B^\omega$ is given, using again the shorthand above, by

$$(\langle h, t\rangle^{-1})^*(\langle b_1, \ldots, b_n\rangle, \mathbf{s}) \ = \ \langle b_1, \ldots, b_n\rangle : \mathbf{s}.$$

So we can equivalently express the specification (4.20) by the diagram below,



where $\tilde{\phi}(a : \mathbf{s}) := \langle t(a), \mathbf{s}\rangle$. The function $\tilde{\phi}$ is guarded, since $t(a)$ was assumed to be nonempty for all $a \in A$. Corollary 4.4.4 yields that this diagram characterises a function $\mathtt{trans}_t$ uniquely.

The corresponding proof principle

The λ^*-coinduction proof principle for λ^* as above is useful as well. We spell it out in the special case of λ^*-bisimulations on streams $\mathbb{R}^\omega$, which gave rise to the final coalgebra $\langle \mathbb{R}^\omega, \langle h, t\rangle\rangle$ of the functor $\mathrm{B} := \underline{\mathbb{R}} \times \mathrm{Id}$. We have that a relation $R \subseteq \mathbb{R}^\omega \times \mathbb{R}^\omega$ is a λ^*-bisimulation if for all $\langle \mathbf{s}, \mathbf{t}\rangle \in R$ we can find a nonempty, finite common prefix $\langle r_1, \ldots, r_n\rangle \in \mathbb{R}^+$ and a pair of streams $\langle \mathbf{u}, \mathbf{v}\rangle \in R$ such that $\mathbf{s} = \langle r_1, \ldots, r_n\rangle : \mathbf{u}$ and $\mathbf{t} = \langle r_1, \ldots, r_n\rangle : \mathbf{v}$.

Application

We next present an example statement that can be proved conveniently with this principle. It involves the functions

$$\mathtt{even}, \mathtt{odd}, \mathtt{exchange} : \mathbb{R}^\omega \to \mathbb{R}^\omega \quad \text{and} \quad \mathtt{zip} : \mathbb{R}^\omega \times \mathbb{R}^\omega \to \mathbb{R}^\omega$$

defined as

$$\begin{aligned}
\mathtt{zip}(r_0 : \mathbf{s}, \mathbf{t}) &:= r_0 : \mathtt{zip}(\mathbf{t}, \mathbf{s}),\\
\mathtt{even}(r_0 : (r_1 : \mathbf{s})) &:= r_0 : \mathtt{even}(\mathbf{s}),\\
\mathtt{odd}(r_0 : (r_1 : \mathbf{s})) &:= r_1 : \mathtt{odd}(\mathbf{s}),\\
\mathtt{exchange}(r_0 : (r_1 : \mathbf{s})) &:= \langle r_1, r_0\rangle : \mathtt{exchange}(\mathbf{s}).
\end{aligned}$$

The upper three lines are (standard) coiterative definitions, the last one fits in the definition schema from Corollary 4.4.4.

We now want to prove that we can implement the function `exchange` using the other three functions as

$$\mathtt{exchange} = \mathtt{zip} \circ \langle \mathtt{odd}, \mathtt{even}\rangle.$$

The natural candidate for a bisimulation relation to prove this equation is

$$R := \big\{ \langle \mathtt{exchange}(\mathbf{t}), \mathtt{zip}(\mathtt{odd}(\mathbf{t}), \mathtt{even}(\mathbf{t}))\rangle \;\big|\; \mathbf{t} \in \mathbb{R}^\omega \big\},$$

but it is not straightforward to prove that it is a standard bisimulation. However, it is simple to see that it satisfies the condition spelled out above for a λ^*-bisimulation with λ^* as in the beginning of this section:
for any $\mathbf{t} = (r_0 : (r_1 : \mathbf{s}))$ we calculate

$$\begin{aligned}
\mathtt{exchange}(\mathbf{t}) &= \langle r_1, r_0\rangle : \mathtt{exchange}(\mathbf{s}),\\
\mathtt{zip}(\mathtt{odd}(\mathbf{t}), \mathtt{even}(\mathbf{t})) &= \mathtt{zip}(r_1 : \mathtt{odd}(\mathbf{s}), r_0 : \mathtt{even}(\mathbf{s}))\\
&= r_1 : \mathtt{zip}(r_0 : \mathtt{even}(\mathbf{s}), \mathtt{odd}(\mathbf{s}))\\
&= r_1 : (r_0 : \mathtt{zip}(\mathtt{odd}(\mathbf{s}), \mathtt{even}(\mathbf{s})))\\
&= \langle r_1, r_0\rangle : \mathtt{zip}(\mathtt{odd}(\mathbf{s}), \mathtt{even}(\mathbf{s})).
\end{aligned}$$

So we find a nonempty common prefix $\langle r_1, r_0\rangle$ and a pair of streams related by R describing the remainders, namely $\langle \mathtt{exchange}(\mathbf{s}), \mathtt{zip}(\mathtt{odd}(\mathbf{s}), \mathtt{even}(\mathbf{s}))\rangle \in R$.

4.4.3 The language accepted by a nondeterministic automaton

In this section we show that a classical construction in theoretical computer science arises as a λ-coiterative arrow, namely the function that assigns to every state of a nondeterministic automaton the language it accepts. This consideration leads to an example of a distributive law of a monad over a functor where the monad is not freely generated by a signature.

In Section 2.3.3 we have defined deterministic automata over an alphabet L as coalgebras of the functor $\mathrm{B} = \underline{2} \times \mathrm{Id}^L$ and mentioned that we can build a final B-coalgebra on the set of languages $\mathcal{L} = \{M \mid M \subseteq L^*\}$: we equip it with the automaton structure $\langle o, t\rangle : \mathcal{L} \to 2 \times \mathcal{L}^L$ defined as $o(M) := (\varepsilon \in M)$ and $t(M) := [a \mapsto M_a]$, where $M_a := \{w \mid aw \in M\}$ denotes the a-derivative of M.

Consider the powerset monad $\langle \mathcal{P}, \eta, \mu\rangle$ where $\eta_X(x) := \{x\}$ forms singleton sets and $\mu_X(Q) := \bigcup_{X' \in Q} X'$ yields unions of sets $Q \subseteq \mathcal{P}X$ of subsets of X. We can define a distributive law of this monad over the functor B from above, i.e. a natural transformation $\lambda : \mathcal{P}(\underline{2} \times \mathrm{Id}^L) \Rightarrow \underline{2} \times \mathcal{P}^L$ satisfying the unit and multiplication law:

$$\lambda := \langle \bigvee \circ \mathcal{P}\pi_1, \delta \circ \mathcal{P}\pi_2\rangle, \tag{4.21}$$

where $\bigvee : \mathcal{P}2 \to 2$ computes the disjunction of truth values and $\delta : \mathcal{P}(\mathrm{Id}^L) \Rightarrow \mathcal{P}^L$ is given by

$$\delta_X(Q) := [\, a \mapsto \{\phi(a) \mid \phi \in Q\}\,].$$

It is not difficult to show that the unit and multiplication law hold. The following structural lemma, which may be interesting as such, can be used to simplify the task. Its proof is straightforward.

Lemma 4.4.5 *Let λ^1 and λ^2 be distributive laws of the monad $\langle \mathrm{T}, \eta, \mu\rangle$ over the functors B_1 and B_2 respectively. Then by setting*

$$\lambda := (\lambda^1 \times \lambda^2) \circ \langle \mathrm{T}\pi_1, \mathrm{T}\pi_2\rangle : \mathrm{T}(\mathrm{B}_1 \times \mathrm{B}_2) \Rightarrow \mathrm{B}_1\mathrm{T} \times \mathrm{B}_2\mathrm{T}$$

we define a distributive law of the monad $\langle \mathrm{T}, \eta, \mu\rangle$ over the functor $\mathrm{B}_1 \times \mathrm{B}_2$.

We claim that the unique $\mathcal{P}$-algebra operation extending the final B-coalgebra $\langle \mathcal{L}, \langle o, t\rangle\rangle$ from above to a λ-bialgebra is the union $\bigcup : \mathcal{P}\mathcal{L} \to \mathcal{L}$ of sets of languages, which means that the following diagram commutes:

$$\begin{array}{ccc} \mathcal{P}\mathcal{L} & \xrightarrow{\quad\bigcup\quad} & \mathcal{L} \\ {\scriptstyle \mathcal{P}\langle o,t\rangle}\downarrow & & \\ \mathcal{P}(2 \times \mathcal{L}^L) & & \downarrow{\scriptstyle \langle o,t\rangle} \\ {\scriptstyle \lambda_{\mathcal{L}}}\downarrow & & \\ 2 \times (\mathcal{P}\mathcal{L})^L & \xrightarrow[\mathtt{id}_2 \times (\bigcup)^L]{} & 2 \times \mathcal{L}^L \end{array}$$

This is indeed the case, since for any set of languages $Q \subseteq \mathcal{L}$ we obtain

$$\begin{aligned} (\langle o, t\rangle \circ \bigcup)(Q) &= \langle o, t\rangle(\bigcup_{M \in Q} M) \\ &= \langle \varepsilon \in \bigcup_{M \in Q} M, [a \mapsto (\bigcup_{M \in Q} M)_a]\rangle \\ &= \langle \bigvee_{M \in Q} (\varepsilon \in M), [a \mapsto \bigcup_{M \in Q} M_a]\rangle \end{aligned}$$

and

$$\begin{aligned} &\big((\mathtt{id}_2 \times (\textstyle\bigcup)^L) \circ \lambda_{\mathcal{L}} \circ \mathcal{P}\langle o, t\rangle\big)(Q) \\ &\quad = \big((\mathtt{id}_2 \times (\textstyle\bigcup)^L) \circ \lambda_{\mathcal{L}}\big)\big(\{\langle \varepsilon \in M, [a \mapsto M_a]\rangle \mid M \in Q\}\big) \\ &\quad = \big(\mathtt{id}_2 \times (\textstyle\bigcup)^L\big)\big(\langle \bigvee_{M \in Q} (\varepsilon \in M), [a \mapsto \{M_a \mid M \in Q\}]\rangle\big) \\ &\quad = \langle \bigvee_{M \in Q} (\varepsilon \in M), [a \mapsto \bigcup_{M \in Q} M_a]\rangle. \end{aligned}$$

For $\mathrm{B} = \underline{2} \times \mathrm{Id}^L$ and $\mathrm{T} = \mathcal{P}$ a guarded recursive definition from Definition 4.2.3 is an arrow $\langle \tilde{o}, \tilde{t}\rangle : X \to 2 \times (\mathcal{P}X)^L$. Such an arrow defines a *nondeterministic automaton*: to each state $x \in X$ it assigns an acceptance attribute $\tilde{o}(x) \in 2$ and for each letter a of the alphabet L a set of successor states $\tilde{t}(x)(a) \in \mathcal{P}X$.

Corollary 4.3.6 and the finality of the λ-bialgebra $\langle \mathcal{L}, \bigcup, \langle o, t\rangle\rangle$ yield the following corollary.

Corollary 4.4.6 *Every nondeterministic automaton*

$$\langle X, \langle \tilde{o}, \tilde{t} \rangle : X \to 2 \times (\mathcal{P}X)^L \rangle$$

induces a unique arrow $f : X \to \mathcal{L}$ fitting into the diagram below.

$$\begin{array}{ccc} X & \overset{f}{\dashrightarrow} & \mathcal{L} \\ {\scriptstyle \langle \tilde{o}, \tilde{t} \rangle}\downarrow & & \downarrow{\scriptstyle \langle o, t \rangle} \\ 2 \times (\mathcal{P}X)^L & \underset{\mathtt{id}_2 \times (\bigcup \circ \mathcal{P}f)}{\dashrightarrow} & 2 \times \mathcal{L}^L \end{array}$$

In other words, for all $x \in X$ and $a \in L$ the arrow f satisfies

$$\varepsilon \in f(x) \iff \tilde{o}(x) \quad \textit{and} \quad f(x)_a = \bigcup_{y \in \tilde{t}(x)(a)} f(y).$$

With the definition of the a-derivative the latter equation is equivalent to

$$aw \in f(x) \iff w \in f(y) \text{ for some } y \in \tilde{t}(x)(a).$$

This characterisation of $f(x)$ is equivalent to the definition of the language accepted by x as a state of the nondeterministic automaton $\langle X, \langle \tilde{o}, \tilde{t} \rangle \rangle$. The model constructed in the proof of Lemma 4.3.3 is the automaton that results from the known powerset construction turning a nondeterministic automaton into a deterministic one.

As an aside, we mention a variation of the above construction. In his work on stream calculus, Rutten [Rut01] uses *weighted stream automata* to conveniently characterise certain streams. These automata are BT-coalgebras, where $\mathrm{B} = \underline{\mathbb{R}} \times \mathrm{Id}$ is the functor for stream systems, and

$$\mathrm{T}X := \{\mu : X \to \mathbb{R} \mid \mathtt{supp}(\mu) \text{ is finite}\}$$

with $\mathtt{supp}(\mu) = \{x \in X \mid \mu(x) \neq 0\}$ is a functor that bears similarity to the (finite) powerset functor. The streams represented by the states of a weighted stream automaton are defined through a function into the final stream coalgebra. This function turns out to be λ-coiterative in a similar way as the function assigning the accepted language to the states of a nondeterministic automaton.

4.4.4 λ-coinduction and abstract GSOS

In this section we spell out our results about λ-coinduction for the abstract GSOS specification format from Def. 3.5.3. So we assume that in addition to the behaviour functor B we are given a finitary signature Σ with associated signature functor S (cf. equation (2.2) on page 20). Let $\langle \mathrm{T}, \eta, \mu \rangle$ denote the

term monad generated by Σ. A specification for the operators in Σ is given in abstract GSOS, i.e. by a natural transformation

$$\rho : \mathrm{S}(\mathrm{Id} \times \mathrm{B}) \Rightarrow \mathrm{BT}.$$

In Section 3.5 we have given rule formats that correspond to the abstract GSOS format for several kinds of systems. In the setting of labelled transition systems we recovered the known GSOS format [BIM95]. We also spelled out the format for streams and deterministic automata. The examples in this section also refer to the latter settings.

In the following we shall show that the λ-coinduction specification and definition principle here yields unique solutions for sets of guarded recursive equations and a bisimulation up-to-context proof principle.

Guarded recursive equations

In process algebra [BW90, Fok00], processes (i.e. labelled transition systems with a designated current state) are often defined by sets of recursive equations of the form

$$x = t_x \quad (x \in X),$$

where X is a set of process names and $t_x \in \mathrm{T}X$ for each $x \in X$ is a term involving the operators of the process algebra and again the process names in X. We express such a specification by a function $\psi : X \to \mathrm{T}X$ with $\psi(x) := t_x$.

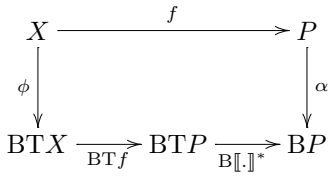
To guarantee uniqueness of solutions, one usually imposes a (syntactic) *guardedness condition* on the terms. One for instance assumes that the terms are of the shape

$$t_x = \sum_{i \in I} a_i.t_i$$

or can be transformed into it, where $a_i.t_i$ denotes the process t_i prefixed with a transition labelled by $a_i \in L$ and the sum denotes nondeterministic choice. This restriction essentially guarantees that each term t_x defines one layer of the behaviour of x independent of the interpretation of the process names appearing in t_x. To achieve this in a more semantic fashion – independent of the existence of certain basic operators in Σ that allow to construct these immediate transitions – we can require that the specification is given by a function $\phi : X \to \mathrm{BT}X$. The additional application of the behaviour functor B enforces that ϕ provides one layer of the behaviour of each $x \in X$. This is the reason why we called functions of that type *guarded recursive definitions* before.[3]

[3] Note that the notion of guardedness used by Aczel et al. [AAMV03], which looks different from ours, is actually a special case of the above definition: in loc. cit. a system of equations is called guarded if for none of the equations $x = t_x$ we have that t_x is a single variable. Expressed in our framework, Aczel el al. use the functor S associated to a signature both as the algebra and the coalgebra functor (i.e. B = S). So a guarded recursive equation as we define it becomes a function $\phi : X \to \mathrm{ST}X$. The elements of its codomain are precisely terms which are not single variables.

Let $\langle P, [\![.]\!], \alpha\rangle$ be an $\langle \mathrm{S}, \mathrm{B}\rangle$-bialgebra which is a model of the operator specification ρ in abstract GSOS (cf. Def. 3.5.3). Although in this setting we actually deal with cofree copointed behaviour functors, Lemma 4.3.9 allows us to still consider a plain arrow $\phi : X \to \mathrm{BT}X$ as a guarded recursive definition. The solutions of ϕ in the model $\langle P, [\![.]\!], \alpha\rangle$ are morphisms $f : X \to P$ making the diagram below commute, where $[\![.]\!]^*$ is the inductive extension of $[\![.]\!] : \mathrm{S}P \to P$ to terms (cf. Lemma 2.2.10).



The following statement is an immediate consequence of Corollary 4.3.8.

Corollary 4.4.7 *Every guarded recursive definition $\phi : X \to \mathrm{BT}X$ has a solution in some model of the specification ρ in abstract GSOS. Moreover, the solutions are determined up to bisimilarity.*

We will spell out the definition of a guarded recursive definition and a solution in the case of labelled transition systems, i.e. for the functor $\mathrm{B} := (\mathcal{P}_\omega)^L$. A *guarded recursive definition* $\phi : X \to (\mathcal{P}_\omega \mathrm{T}X)^L$ can be described as a set Tr of "transitions" of the shape

$$Tr \subseteq \{x \xrightarrow{a} t \mid x \in X,\ a \in L,\ t \in \mathrm{T}X\},$$

such that Tr contains only finitely many transitions from any $x \in X$ with the same label $a \in L$. A *solution* of Tr in the $\langle \mathrm{S}, \mathrm{B}\rangle$-bialgebra $\langle P, [\![.]\!], \alpha\rangle$ is an assignment of a state $p_x \in P$ to every $x \in X$ such that for all $x \in X$ and $a \in L$ we have $p_x \xrightarrow{a} q$ in the LTS $\langle P, \alpha\rangle$ if and only if there exists a transition $x \xrightarrow{a} t$ in Tr such that $q = [\![\ t[y := p_y | y \in X]\ ]\!]^*$. By $t[y := p_y | y \in X] \in \mathrm{T}P$ here we mean the term resulting after the replacement of every $y \in X$ in t by p_y.

The above corollary states that if the operators in Σ are specified by GSOS rules, than any set of transitions Tr as above has solutions, and that those solutions are determined up to bisimilarity.

Note that for this result we did not exploit the existence of a final coalgebra. A corresponding statement holds, for instance, also for labelled transition systems without the image finiteness assumption, for which no final coalgebra exists. For these systems unique (up to bisimilarity) solutions of guarded recursive equations are guaranteed as well, if the involved operators are specified in a correspondingly adapted GSOS format.

Application

Most of the standard operators for LTSs do not need the full expressive power of the GSOS format, because the rules specifying them have targets that consist of at most one operator application. Consequently, the existence of solutions of the standard examples of guarded recursive definitions follows from statements simpler then the one we have just given. In other settings we have seen operators defined by rules with more complex targets, like for instance the sequential composition of automata (see equation (3.23) on page 64) or the shuffle product of streams (see equation (3.27) on page 80). We present an example involving the latter operation: we want to specify the stream

$$\tau = \langle 0, 1, 0, 2, 0, 16, 0, 272, 0, 7936, \ldots \rangle \in \mathbb{R}^\omega$$

of Taylor coefficients of the tangent function, the so-called *tangent numbers*. We quote a definition by Rutten [Rut01], which uses streams $[r] = \langle r, 0, 0, \ldots \rangle$ representing real numbers $r \in \mathbb{R}$ and as auxiliary operators the sum $+$ and the shuffle product $\otimes$ (cf. equations (3.26) and (3.27) on page 80). We recall that for all $r \in \mathbb{R}$ and streams $\mathbf{s} = s_0 : \mathbf{s}'$ and $\mathbf{t} = t_0 : \mathbf{t}'$ they satisfy the equations

$$\begin{aligned} [r] &= r : [0], \\ \mathbf{s} + \mathbf{t} &= (s_0 + t_0) : (\mathbf{s}' + \mathbf{t}'), \\ \mathbf{s} \otimes \mathbf{t} &= (s_0 \cdot t_0) : ((\mathbf{s}' \otimes \mathbf{t}) + (\mathbf{s} \otimes \mathbf{t}')). \end{aligned}$$

The identities $\tan(0) = 0$ and $\tan' = 1 + \tan \cdot \tan$ motivate the following definition of the the stream τ of tangent numbers (for more details see [Rut01]):

$$\tau = 0 : ([1] + (\tau \otimes \tau)) \tag{4.22}$$

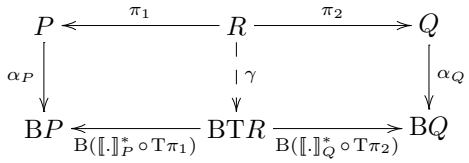
In order to show that this equation has a unique solution, Rutten builds a *weighted stream automaton* from this equation an uses a statement that every state of such an automaton uniquely characterises a stream (cf. the remark at the end of Section 4.4.3).

Alternatively, we can prove the validity of (4.22) using λ-coinduction: the specification of τ gives rise to a guarded recursive equation using auxiliary operators that can be defined within the abstract GSOS format. So by Corollary 4.4.7 the specification of τ has a unique solution in the final model of the specification of the auxiliary operators.

The bisimulation up-to-context proof principle

For distributive laws λ arising from operator specifications in abstract GSOS, a λ-bisimulation is a *bisimulation up-to-context* (see e.g. [San98]), as we shall explain.

Let the two $\langle \mathrm{S}, \mathrm{B}\rangle$-bialgebras $\langle P, [\![.]\!]_P, \alpha_P\rangle$ and $\langle Q, [\![.]\!]_Q, \alpha_Q\rangle$ be models of a specification ρ in abstract GSOS and let λ be the distributive law of the free monad over the cofree copointed functor corresponding to ρ (cf. Lemma 3.5.2 (i)). Lemma 4.3.9 stated that a relation $R \subseteq P \times Q$ is a λ-bisimulation between those bialgebras if there is an operation $\gamma : R \to \mathrm{BT}R$ making the two parts of the diagram below commute, where $[\![.]\!]_P^*$ and $[\![.]\!]_Q^*$ denote the inductive extensions of $[\![.]\!]_P$ and $[\![.]\!]_Q$ respectively to terms (cf. Lemma 2.2.10).

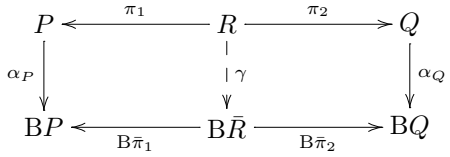


The important observation here is that the span

$$\langle \mathrm{T}R, [\![.]\!]_P^* \circ \mathrm{T}\pi_1, [\![.]\!]_Q^* \circ \mathrm{T}\pi_2\rangle,$$

the B-image of which appears in this definition, expresses the *congruence closure* $\bar{R}$ of R with respect to $[\![.]\!]_P$ and $[\![.]\!]_Q$. By this we mean the smallest congruence $\bar{R} \subseteq P \times Q$ (cf. Def 2.2.2) containing R. Using this, we can rephrase the condition on a λ-bisimulation as follows.

Definition 4.4.8 *A relation $R \subseteq P \times Q$ is a* **bisimulation up-to-context** *between two $\langle \mathrm{S}, \mathrm{B}\rangle$-bialgebras $\langle P, [\![.]\!]_P, \alpha_P\rangle$ and $\langle Q, [\![.]\!]_Q, \alpha_Q\rangle$ if there exists an arrow $\gamma : R \to \mathrm{B}\bar{R}$ fitting into the diagram below, where $\bar{R} \subseteq P \times Q$ (with projections written as $\bar{\pi}_1$ and $\bar{\pi}_2$ for clarity) is the congruence closure of R with respect to $[\![.]\!]_P$ and $[\![.]\!]_Q$.*



Corollary 4.4.9 *Every bisimulation up-to-context between two models of a specification in abstract GSOS (cf. Def. 3.5.3) is contained in some standard bisimulation between the respective coalgebras.*

For labelled transition systems Sangiorgi [San98] proves that a corresponding statement holds for operator specifications in the De Simone format [Sim85]. Corollary 4.4.9 generalises this result not only to the more expressive GSOS specification format [BIM95] for LTSs, but also to coalgebras for an arbitrary functor B and specifications in abstract GSOS.

Application

To illustrate the last point, we present an example of the use of the bisimulation up-to-context principle in the setting of deterministic automata. Following Rutten [Rut98, Equation (11) in Section 6], we establish the identity

$$A.(B \cup C) = A.B \cup A.C \tag{4.23}$$

for all languages $A, B, C \in \mathcal{L}$, where we use the union and concatenation of languages as defined respectively by the rules in (3.20) on page 52 and (3.23) on page 64.

Using the final $(\underline{2} \times \mathrm{Id}^L)$-coalgebra $\langle \mathcal{L}, \langle o, t \rangle \rangle$ defined through the clauses (2.9) on page 33, we can prove (4.23) using the coinduction proof principle of Corollary 2.3.9. With Corollary 4.4.9 it suffices to show that the relation

$$R = \{\langle A.(B \cup C), A.B \cup A.C \rangle \mid A, B, C \in \mathcal{L}\} \cup \{\langle A, A \rangle \mid A \in \mathcal{L}\}$$

is a bisimulation up-to-context. With the definition of $\langle o, t \rangle$ this is the case if for all $\langle A, B \rangle \in R$ we have $\varepsilon \in A \iff \varepsilon \in B$ and for all $a \in L$ we have $\langle A_a, B_a \rangle \in \bar{R}$, where we write again $A_a = \{w \in L^* \mid aw \in A\}$ for the a-derivative of A and $\bar{R}$ is the congruence closure of R under concatenation and union of languages. The pairs $\langle A, A \rangle \in R$ trivially satisfy these conditions, so let us check a pair $\langle A.(B \cup C), A.B \cup A.C \rangle \in R$. As needed, we find

$$\varepsilon \in A.(B \cup C) \iff (\varepsilon \in A) \wedge ((\varepsilon \in B) \vee (\varepsilon \in C)) \iff \varepsilon \in A.B \cup A.C,$$

so the first condition is satisfied. For the second condition, we let $a \in L$ and distinguish two cases: if $\varepsilon \notin A$ then

$$(A.(B \cup C))_a = A_a.(B \cup C) \quad \text{and} \quad (A.B \cup A.C)_a = A_a.B \cup A_a.C,$$

so with $\langle A_a.(B \cup C), A_a.B \cup A_a.C \rangle \in R \subseteq \bar{R}$ the condition is easily satisfied; if $\varepsilon \in A$ then

$$(A.(B \cup C))_a = A_a.(B \cup C) \cup (B_a \cup C_a)$$

and

$$\begin{aligned} (A.B \cup A.C)_a &= (A_a.B \cup B_a) \cup (A_a.C \cup C_a) \\ &= (A_a.B \cup A_a.C) \cup (B_a \cup C_a). \end{aligned}$$

Since $\langle A_a.(B \cup C), A_a.B \cup A_a.C \rangle \in R$ and $\langle B_a \cup C_a, B_a \cup C_a \rangle \in R$ implies

$$\langle A_a.(B \cup C) \cup (B_a \cup C_a), (A_a.B \cup A_a.C) \cup (B_a \cup C_a) \rangle \in \bar{R},$$

the condition is satisfied as well.

Note that the ideal proof principle for this example is a combination of the up-to-equality (cf. Def. 4.4.2) and up-to-context principles. We leave the study of such combined principles to future work.

To show the limits of the bisimulation up-to-context principle, we quote an example of Sangiorgi's [San98, Section 2.1.3] in the setting of LTSs, for which it is not valid. It involves unary operators σ and $a.(-)$ (action prefix) for $a \in L$ and the constant 0 for the inactive process. The transition rules are the ones below for all $a \in L$:

$$\frac{x \xrightarrow{a} x' \quad x' \xrightarrow{a} x''}{\sigma(x) \xrightarrow{a} x''} \qquad \frac{}{a.x \xrightarrow{a} x}$$

In the final model of this specification we have $a.0 \not\sim a.a.0$ although the relation $R = \{\langle a.0, a.a.0\rangle\}$ is a bisimulation up-to-context. To show the latter, we easily establish $\sigma(a.0) \sim 0$ and $\sigma(a.a.0) \sim a.0$, which implies $\sigma(a.0) = 0$ and $\sigma(a.a.0) = a.0$ in the final model. The relation R is a bisimulation up-to-context because the only transitions that the two related states can make are $a.0 \xrightarrow{a} 0$ and $a.a.0 \xrightarrow{a} a.0$, and we have $\langle 0, a.0\rangle = \langle \sigma(a.0), \sigma(a.a.0)\rangle \in \bar{R}$.

Note that the example involves rules with *lookahead*, i.e. rules with chained premises. Such rules do not fit into the GSOS format, which corresponds to distributive laws of the free monad over the cofree copointed functor. But, as Turi and Plotkin show [TP97], they can be expressed by formats involving a cofree comonad generated by the behaviour functor. So this counterexample supports our conjecture that no (simple) λ-coinduction proof principle can be given in a setting involving a cofree comonad. This in turn justifies our explicit treatment of the less expressive formats in Chapter 3.

4.5 Comparison with related work

As already stated in the introduction of this chapter, the idea to use a distributive law λ of an additional pointed functor $\langle \mathrm{T}, \eta\rangle$ over the behaviour functor under consideration in order to obtain extensions of the coiteration schema was taken from Lenisa [Len99a, Len99b, LPW00]. She studies a class of $\langle \mathrm{T}, \eta\rangle$-coiterative arrows, which she defines to be the morphisms of the type $h \circ \eta_X$, where h is a final homomorphism from some coalgebra on the carrier $\mathrm{T}X$. She shows for instance that the primitive corecursive arrows are $\langle \mathrm{T}, \eta\rangle$-coiterative for the pointed functor contained in the monad defined in equation (4.15) on page 112.

Lemma 4.3.4 shows that, in the setting of a monad $\langle \mathrm{T}, \eta, \mu\rangle$, every λ-coiterative arrow is also $\langle \mathrm{T}, \eta\rangle$-coiterative. So one may argue that in that case the class of $\langle \mathrm{T}, \eta\rangle$-coiterative arrows is larger than that of λ-coiterative arrows. But Lenisa's approach is only seemingly wider in scope: her main result [Len99a, Theorem 3.5] is applicable only for a subclass of the class of $\langle \mathrm{T}, \eta\rangle$-coiterative arrows, and this subclass turns out to be closely related to the λ-coiterative arrows.

Our definition of a λ-coiterative arrow to be a function making a diagram commute is more easy to work with than Lenisa's factorisation property. To obtain the known extended coiteration schemata, all we need to do is to simplify the

concrete instances of the λ-coiteration diagram. So our framework yields an alternative justification of the schemata. This is not the case for $\langle \mathrm{T}, \eta \rangle$-coiterative arrows. There the relation to the known schemata is obtained from the individual justifications of the latter. As another improvement, λ-coiteration can be instantiated also in settings where no unit natural transformation is given, as we have shown.

Moss [Mos01] also develops generalised coinduction schemata. He presents, amongst others, the *flattening lemma* (Lemma 2.1 in loc. cit.) and the *parametric corecursion schema* (Theorem 2.11 in loc. cit.). The first schema is a variation of the primitive corecursion schema from Section 4.4.1. It arises directly from the λ-coiteration schema for $\langle \mathrm{T}, \eta, \mu \rangle$ and λ as in equations (4.15) and (4.16) on page 112, but with the final coalgebra $\langle \mathcal{F}, \omega \rangle$ generalised to be any B-coalgebra. With this generalisation we moreover obtain a bisimulation up-to-equality proof principle on every B-coalgebra. The idea underlying Moss' parametric corecursion can be covered by λ-coiteration as well: the approach is similar to the one for the dual of course-of-value iteration in Section 4.4.2, but with the free monad generated by the behaviour functor B replaced by a monad built on what we call "free coalgebras". There $\mathrm{T}X$ is the carrier of a final $(\mathrm{B}+\underline{X})$-coalgebra. The same monad is also used by Moss. This functor T is bigger than the free algebra functor in that it also contains behaviours with infinite branches. Consequently, the resulting λ-coiteration principle is more powerful and generalises both primitive corecursion and the dual of course-of-value iteration.[4] Moss studies another interesting aspect, which we do not consider here, namely the relation of the extended coiteration schemata for two different behaviour functors with a natural transformation between them.

In a recently published article, Cancila, Honsell, and Lenisa [CHL03] give a collection of function definitions that do not fit into the standard coiteration schema, and they present a framework for extended schemata to accommodate these examples. The approach is based on bialgebras for what they call *generalised distributive laws*. In most of the cases, the use of these generalised distributive laws can be avoided by working with a cofree copointed functor, as we suggest here. Still, their approach has the potential to capture definitions that cannot be handled by λ-coiteration. It remains to be seen whether one can distill interesting schemata from such examples.

At the same time that we first published the main ideas discussed in this chapter [Bar00, Bar03], but independently, Pardo, Uustalu, and Vene [UVP01] proposed a generalised inductive definition principle parametric in a distributive law of the algebra functor under consideration over a comonad. This schema turns out to be the categorical dual of our λ-coiteration definition principle in the setting of a distributive law of a monad over the behaviour functor.

Generalised proof techniques for bisimilarity, mainly in the shape of bisimula-

[4] We know from private communication that Vene and Uustalu were aware of this generalisation as well when they published their paper on the course-of-value principle and its dual [UV99].

tion up-to-... principles, have also been studied for the concrete case of labelled transition systems in isolation, i.e. without using coalgebraic techniques. One early example is the bisimulation up-to-bisimilarity method of Milner [Mil89]. Sangiorgi [San98] developed a methodology to derive various such bisimulation up-to-... principles, among them the up-to-context technique that we studied in Section 4.4.4. Notably, Sangiorgi studies the combination of the different extended proof principles. We view a generalisation of the latter to the λ-coinduction framework as an interesting direction for future work.

We conclude our discussion of related work by mentioning that there is another approach to generalise coinductive definition principles based on distributive laws. It is discussed for instance by Power and Turi [PT99] (a dual approach for inductive definitions is presented by Pardo [Par00]). It assumes a distributive law $\tilde{\lambda}$ of a behaviour functor B over a monad $\langle \mathrm{T}, \eta, \mu \rangle$, i.e. a natural transformation of the opposite direction compared to the ones studied here. This distributive law lifts the behaviour functor B to a functor $\mathrm{B}_{\tilde{\lambda}}$ on the *Kleisli category* of the monad instead of the *Eilenberg-Moore* category of algebras for the monad. Again, a final B-coalgebra $\langle \mathcal{F}, \omega \rangle$ extends to a final $\mathrm{B}_{\tilde{\lambda}}$-coalgebra, but the final morphisms in the Kleisli Category are morphisms into $\mathrm{T}\mathcal{F}$ in the base category. This way, by choosing the stream functor and the powerset monad, Power and Turi obtain a definition principle for functions into sets of infinite streams. With another monad one can, for instance, obtain functions that return an infinite stream or an exception. We stress that this approach serves a different purpose than the one presented here.

Chapter 5

Specification formats for probabilistic systems

Probabilistic systems arise from nondeterministic ones by adding probabilities to the different outgoing transitions of a state. This information is needed, for instance, to model probabilistic algorithms, which can "flip a coin" to make decisions (like e.g. in [Har02, Sto02a]). Or it is added to compute quantitative information, such as the failure probability of a system with faulty system components. Therefore probabilistic systems have widely been studied also in performance analysis [BHK01].

In the literature, various types of probabilistic systems have been introduced and equipped with corresponding notions of probabilistic bisimilarity [LS91, SL94, Seg95b, HJ94]. Composition operators have been specified and studied for several of these systems as well. Since one needs to check that the specified transition probabilities form probability distributions indeed, the validity of a probabilistic operator specification is less obvious than that of a nondeterministic one, where no such check is needed. Showing that the operators preserve behavioural equivalence is more complex than in the nondeterministic case too, since it is harder to work with probabilistic bisimulations than with the ordinary bisimulations.

Knowing that for nondeterministic systems the development of specification formats has simplified the task of showing validity and well-behavedness of operator specifications considerably, we immediately ask for corresponding probabilistic formats. Surprisingly, none have been proposed in the literature yet. So in this chapter we introduce the first well-behaved SOS-like specification formats for probabilistic systems.

We distill our concrete rule-shapes from abstract GSOS (cf. Def. 3.5.3). The main work will be to establish a correspondence between specifications in abstract GSOS – recall that these were natural transformations of a certain type

– and sets of transition rules in the proposed shape. Once this is done, concrete well-behavedness statements immediately result as instances of the well-behavedness of the abstract format. As explained in Chapter 3, the most important results are: any specification has two canonical models, one on the set of terms of the signature and one on the set of possible system behaviours (cf. Corollary 3.5.4); probabilistic bisimilarity is a congruence on all models of the specification (cf. Corollary 3.5.5); and, as established in Chapter 4, guarded recursive definitions have solutions which are determined up to probabilistic bisimilarity (cf. Corollary 4.4.7), and the bisimulation up-to-context proof principle is valid (cf. Corollary 4.4.9).

Our decompositional method to analyse natural transformations has the advantage that one can handle various systems of a similar type with the same structural lemmata. For probabilistic systems, this flexibility is important because the variety of different system types considered in the literature is much larger:

In nondeterministic modelling, most of the work involves labelled transition systems as introduced in Def. 2.3.12 or some minor variant thereof. The variety mainly involves different bounds on the nondeterminism (finitely branching, image finite as considered here, or no bound at all), label sets of different size, and possibly additional attributes associated to the states of the system, like for instance a termination predicate.[1] From our point of view, all these systems are rather similar at heart: it is straightforward to adapt our study of congruence formats for the simple type of labelled transition systems considered here to any of these variants.

In probabilistic modelling, however, a large number of rather different system types are considered in the literature (see e.g. [SV04] for a survey): there are generative and reactive types of systems [GSS95], depending on whether the probability distributions range over all outgoing transitions of a state or only those with the same label; in addition to purely probabilistic systems, one can also consider types that allow nondeterministic and probabilistic choice; among the systems which allow for both types of choice we find some that may execute probabilistic and nondeterministic steps in an arbitrary order whereas others are constrained to alternate them; and then the probabilistic transitions may be labelled or the nondeterministic ones or both.

Here we develop specification formats for two important and representative classes in this variety. Other types can be dealt with similarly.

First, we treat *probabilistic transition systems (PTS)* of Larsen and Skou [LS91], which are also known as *reactive systems* [GSS95]. They belong to the simple types and arise by imposing probability distributions on all sets of transitions leaving one state of an LTS with the same label. Similar to our treatment of abstract GSOS for labelled transition systems in Chapter 3.5.1, we obtain a

[1] As we have argued in Section 2.3.3, the latter variation can be easily dealt with coalgebraically.

probabilistic rule-style format that we call *probabilistic GSOS*, or *PGSOS* for short. It precisely corresponds to specifications in abstract GSOS. The introduction of probabilities makes it considerably more complicated though to establish this correspondence. Like the GSOS format itself, its new probabilistic variant, PGSOS, has all the well-behavedness properties of abstract GSOS listed above. At the same time, as we illustrate with a number of example specifications, it is rather powerful.

Second, as an example of a more complex model which combines nondeterministic and probabilistic choice, we consider the probabilistic automata of Segala [Seg95b, Seg95a, SL95]. These *Segala systems*, as we call them here, have received considerable attention recently (excellent introductions can be found in the work of Stoelinga [Sto02b, Sto02a]). Also for Segala systems we propose a powerful well-behaved rule format for operator specifications, Segala-GSOS. We do not know yet whether our format covers all the expressiveness offered by abstract GSOS. But we state one direction of the correspondence, namely that specifications in Segala-GSOS give rise to natural transformations of the type of abstract GSOS. This direction is the easier but still the more important one for applications, because it allows us to conclude that also this new format inherits the well-behavedness of abstract GSOS summarised above. To illustrate the expressiveness of Segala-GSOS, we show again that various composition operators for Segala systems can be specified within the format.

The chapter is organised as follows: first we introduce PTS and Segala systems together with the corresponding notions of probabilistic bisimilarity; in Sections 5.2 and 5.3 we introduce PGSOS and Segala GSOS; in Section 5.4 we conclude by mentioning some related work.

5.1 Probabilistic systems

In this section we define the two types of probabilistic systems we are going to study in this chapter, namely probabilistic transition systems (PTS) [LS91] and (simple) Segala systems [SL94]. Both are defined as B-coalgebras for some functor B constructed from the following probability functor.

Definition 5.1.1 *We define the* **probability distribution functor**

$$\mathcal{D}_\omega : \mathsf{Set} \to \mathsf{Set}$$

for any set X and any function $f : X \to Y$ as

$$\begin{array}{rcl} \mathcal{D}_\omega X & := & \big\{\mu : X \to \mathbb{R}_0^+ \mid \mathtt{supp}(\mu) \text{ is finite, } \mu[X] = 1\big\}, \\ (\mathcal{D}_\omega f)(\mu) & := & y \mapsto \mu[f^{-1}(y)], \end{array}$$

where for a function $\mu : X \to \mathbb{R}_0^+$ and a subset $X' \subseteq X$ we write

$$\mathtt{supp}(\mu) := \{x \in X \mid \mu(x) \neq 0\} \quad \text{and} \quad \mu[X'] := \sum_{x \in X'} \mu(x).$$

The set $\mathtt{supp}(\mu)$ *is called the* **support** *of* μ.

It has been proved by De Vink and Rutten [VR99] and Moss [Mos99] that the above functor preserves weak pullbacks. This is important to obtain that for systems constructed with this functor the coalgebraic notion of a bisimulation is well-behaved in many respects. Most importantly, a final coalgebra is a fully abstract domain for bisimilarity (cf. Theorem 2.3.8), and the composition of bisimulations is a bisimulation again.

5.1.1 PTS

A classical definition of a *probabilistic transition system (PTS)* arises from that of a labelled transition system by imposing a probability distribution on the set of a-successors of each state for each label $a \in L$ for which this set is nonempty. In the original definition of Larsen and Skou [LS91], for a given set of labels L, a PTS is a triple

$$\langle P, (C_a \subseteq P)_{a\in L}, (\mu_{p,a} \in \mathcal{D}_\omega P)_{a\in L, p\in C_a}\rangle, \tag{5.1}$$

where P is a set of states, for all $a \in L$ the subset C_a contains the states which can do an a-transition, and $\mu_{p,a}$ is a probability distribution on a-successors of the state p.

The transition structure of a PTS defines for each state and each label enabled in this state a probability distribution over successor states. So the probability distributions do not range over transitions with different labels. Therefore the labels can best be viewed as an input to which the system reacts by choosing a successor state probabilistically. This is why a PTS is also called a *reactive system* elsewhere [GSS95].

To define PTS coalgebraically, we argue that the data in (5.1) can equivalently be expressed by the pair $\langle P, \alpha : P \to (\mathcal{D}_\omega P + 1)^L\rangle$, where $1 = \{*\}$ is a singleton set needed to model disabled labels: given a PTS as in (5.1), we define the corresponding transition function α for any $p \in P$ as

$$\alpha(p) := \left[\, a \mapsto \begin{cases} \iota_1(\mu_{p,a}) & \text{if } p \in C_a, \\ \iota_2(*) & \text{otherwise.} \end{cases} \,\right]$$

It is easy to see that this construction defines an isomorphism between the transition structure in (5.1) and the transition functions α, so we can equivalently adopt the following coalgebraic definition.

Definition 5.1.2 *Given a set of labels* L*, a* **probabilistic transition system (PTS)** *is a pair*

$$\langle P, \alpha : P \to (\mathcal{D}_\omega P + 1)^L\rangle,$$

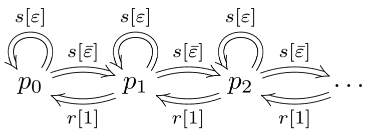
for a singleton set $1 = \{*\}$. In other words, a PTS is a coalgebra of the functor $\mathrm{B} = (\mathcal{D}_\omega + \underline{1})^L$. If no confusion about the coalgebra structure α is likely to arise, we write for $p, q \in P$ and $a \in L$

$$\begin{array}{llll}
p \stackrel{a}{\not\longrightarrow} & \textit{if} & \alpha(p)(a) = \iota_2(*), & \\
p \stackrel{a}{\longrightarrow} \mu & \textit{if} & \alpha(p)(a) = \iota_1(\mu), & \\
p \stackrel{a}{\longrightarrow} & \textit{if} & p \stackrel{a}{\longrightarrow} \mu & \textit{for some } \mu \in \mathcal{D}_\omega P, \\
p \stackrel{a[r]}{\Longrightarrow} q & \textit{if} & p \stackrel{a}{\longrightarrow} \mu & \textit{for some } \mu \in \mathcal{D}_\omega P \textit{ with } \mu(q) = r \in (0, 1].
\end{array}$$

Technically, PTSs arise by replacing the (finite) powerset functor $\mathcal{P}_\omega$ in the behaviour functor $\mathrm{B} = {\mathcal{P}_\omega}^L$ defining LTSs (cf. Def. 2.3.12) by $\mathcal{D}_\omega + \underline{1}$. The correspondence becomes even more apparent if we write $\mathcal{P}_\omega$ equivalently as $\mathcal{P}^+_\omega + \underline{1}$, where $\mathcal{P}^+_\omega$ is the nonempty, finite powerset functor.

Example 5.1.3 *As an example of a PTS, we consider what can be called a* **lossy bag***: a system that can store (s) items and remove (r) them again from stock. So the number of removals is limited to the number of previous storages. But the system is lossy in the sense that a store operation fails to actually add something to the bag with a given probability $\varepsilon \in [0, 1]$. For simplicity we consider the case where all items are of the same kind, so we are just counting the number of items currently in stock.*

We model the bag as a probabilistic process p_0 in a PTS $\langle P, \alpha\rangle$ for the set of labels $L := \{s, r\}$. The set of states is $P := \{p_i \mid i \in \mathbb{N}\}$, where p_i is the state of the system with i items in storage. A store event can always be processed, and it will increase the number of stored items by one if everything works fine. But with probability ε an error occurs and the number stays the same. A remove event is possible if there is at least one item stored, and it will decrease the number of stored items by one. The system is pictured as follows, where we abbreviate $1 - \varepsilon$ to $\bar{\varepsilon}$:



The coalgebraic definition of a bisimulation can be spelled out as follows.

Definition 5.1.4 *For a relation $R \subseteq P \times Q$ write $\equiv_R \subseteq \mathcal{D}_\omega P \times \mathcal{D}_\omega Q$ for the relation with $\phi \equiv_R \psi$ if and only if there exists a distribution $\mu \in \mathcal{D}_\omega R$ such that for all $p \in P$ and $q \in Q$ we have*

$$\phi(p) = \mu[R \cap (\{p\} \times Q)] \quad \textit{and} \quad \psi(q) = \mu[R \cap (P \times \{q\})]$$

(or, equivalently, $\phi = (\mathcal{D}_\omega \pi_1)(\mu)$ and $\psi = (\mathcal{D}_\omega \pi_2)(\mu)$).

A **probabilistic bisimulation** *between two probabilistic transition systems $\langle P, \alpha_P\rangle$ and $\langle Q, \alpha_Q\rangle$ is a relation $R \subseteq P \times Q$ such that for all pairs $\langle p, q\rangle \in R$*

and labels $a \in L$ we have $p \xrightarrow{a}$ if and only if $q \xrightarrow{a}$, and if the transitions exist, with $p \xrightarrow{a} \phi$ for $\phi \in \mathcal{D}_\omega P$ and $q \xrightarrow{a} \psi$ for $\psi \in \mathcal{D}_\omega Q$, we moreover have $\phi \equiv_R \psi$.

Restricted to equivalence relations $R \subseteq P \times P$ on the carrier of one PTS $\langle P, \alpha \rangle$, we can characterise $\equiv_R$ by $\mu \equiv_R \nu$ if and only if $\mu[E] = \nu[E]$ for all equivalence classes $E \in P/R$. The latter condition was given by Larsen and Skou [LS91] in their widely accepted definition of a probabilistic bisimulation. This notion was compared by De Vink and Rutten [VR99] with the coalgebraic one, where it turned out that both define the same bisimilarity relation.

Lemma 5.1.5 *There exists a final probabilistic transition system, i.e. a final* B*-coalgebra for* $\mathrm{B} = (\mathcal{D}_\omega + \underline{1})^L$.

Proof: The statement can be proved with Theorem 2.3.11. We need to argue that the functor $(\mathcal{D}_\omega + \underline{1})^L$ is bounded. Since we consider probability distributions with finite support, from any state we can reach only finitely many possible successor states with a transition for a given label $l \in L$. The size of the set of possible successor states after one transition is thus bounded by $L \times \mathbb{N}$. Continuing we get that with n transitions we can reach no more than $(L \times \mathbb{N})^n$ states. So the size of a one generated B-coalgebra is bounded by the set $\coprod_{n \in \mathbb{N}} (L \times \mathbb{N})^n$.

□

5.1.2 Segala systems

The second class of probabilistic systems we consider are the *simple probabilistic automata* studied by Segala and Lynch [SL95], which we will refer to as *(simple) Segala systems*. For a set of labels L, such a system is given by a pair

$$\langle P, (\xrightarrow{a} \subseteq P \times \mathcal{D}_\omega P)_{a \in L} \rangle,$$

where, as also done by Stoelinga [Sto02b], we simplified the original presentation to discrete probability distributions with finite sets of support. The behaviour of such a system is more complex than that of a PTS: in a state p the system can potentially choose between several transitions for the same label a, each of which leads to a distribution μ of successor states, from which the actual successor p' is chosen probabilistically.

We will be working with the coalgebraic definition below. It contains an assumption about finite branching. We add it here only to stay closer to our definition of an LTS and to obtain a final coalgebra. Note that this is not essential for the development of the format in Section 5.3.

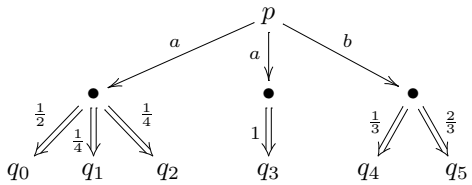
Definition 5.1.6 *For a set of labels L, a* **(simple) Segala system** *is a pair*

$$\langle P, \alpha : P \to (\mathcal{P}_\omega \mathcal{D}_\omega P)^L \rangle,$$

i.e. a coalgebra of the functor $\mathrm{B} := (\mathcal{P}_\omega \mathcal{D}_\omega)^L$. *If no confusion about the coalgebra structure* α *is likely to arise, for* $p, q \in P$, $\mu \in \mathcal{D}_\omega P$, *and* $a \in L$ *we write*

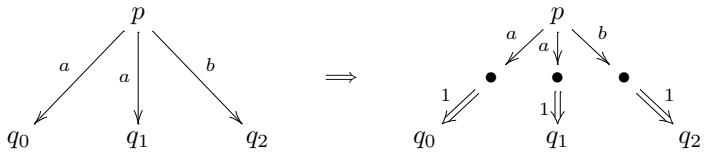
$$
\begin{array}{lll}
p \xrightarrow{a}\!\!\!\!\!\!\rightarrow & \text{if} & \alpha(p)(a) = \emptyset, \\
p \xrightarrow{a} & \text{if} & \alpha(p)(a) \neq \emptyset, \\
p \xrightarrow{a} \mu & \text{if} & \mu \in \alpha(p)(a), \\
p \xrightarrow{a} \bullet \overset{r}{\Rightarrow} q & \text{if} & p \xrightarrow{a} \mu \quad \text{with} \quad \mu(q) = r > 0.
\end{array}
$$

When a Segala system $\langle P, \alpha \rangle$ in a state $p \in P$ receives a label $a \in L$, it first chooses nondeterministically one probability distribution $\mu \in \alpha(p)(a)$, provided that this set is nonempty. Then it moves to a successor state $q \in P$ chosen with probability $\mu(q)$. We picture the behaviour of a state p in a simple Segala system as in the following example.

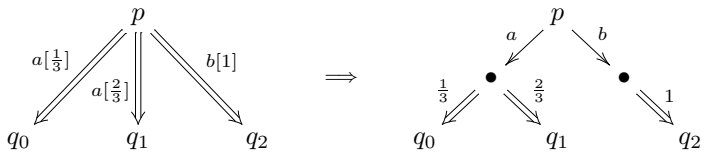


We call these systems *simple* in order to distinguish them from systems of a slightly more expressive type, which are studied by Segala and Lynch as well [Seg95b, SL95]. Those systems have labels associated to the probabilistic choices instead of the nondeterministic choices.

Note that simple Segala systems generalise both LTS and PTS: An LTS can be viewed as a Segala system where all probability distributions are Dirac distributions (i.e. one single state is chosen with probability 1).



A PTS is a Segala system $\langle P, \alpha \rangle$ where at most one distribution is contained in each set $\alpha(p)(a)$ for $p \in P$ and $a \in L$. From this point of view, probabilistic transition systems as defined above arise as *deterministic* (simple) Segala systems (cf. [JLY01]).



When we instantiate the coalgebraic definition of a bisimulation with the functor defining Segala systems, we obtain a notion that can equivalently be expressed

as in the definition below. The characterisation combines elements from the corresponding notions for LTSs and PTSs (cf. equation (2.6) on page 25 and Def. 5.1.4), and it is equivalent to the one used in the literature [BSV04].

Definition 5.1.7 *A* **bisimulation** *between two Segala systems* $\langle P, \alpha_P \rangle$ *and* $\langle Q, \alpha_Q \rangle$ *is a relation* $R \subseteq P \times Q$ *such that for all* $\langle p, q \rangle \in R$ *and all* $a \in L$ *we have that*

- $p \xrightarrow{a} \phi$ *implies* $q \xrightarrow{a} \psi$ *for some* $\psi \in \mathcal{D}_\omega Q$ *with* $\phi \equiv_R \psi$*, and*
- $q \xrightarrow{a} \psi$ *implies* $p \xrightarrow{a} \phi$ *for some* $\phi \in \mathcal{D}_\omega P$ *with* $\phi \equiv_R \psi$*,*

for the relation $\equiv_R$ *introduced in Definition 5.1.4.*

With an argument similar to the one for Lemma 5.1.5 we can show the following statement.

Lemma 5.1.8 *There exists a final Segala system, i.e. a final* B*-coalgebra for* $\mathrm{B} = (\mathcal{P}_\omega \mathcal{D}_\omega)^L$*.*

5.2 A specification format for PTS

We are now going to derive a concrete well-behaved specification format for PTS from Definition 5.1.2 as a representation of specifications in abstract GSOS instantiated with the behaviour functor modelling PTSs, i.e. with $\mathrm{B} := (\mathcal{D}_\omega + \underline{1})^L$. Letting S and T be the signature and term functor associated to a finitary signature Σ, we are dealing with specifications given as natural transformations

$$\rho : \mathrm{S}\big(\mathrm{Id} \times (\mathcal{D}_\omega + \underline{1})^L\big) \Rightarrow (\mathcal{D}_\omega \mathrm{T} + \underline{1})^L. \tag{5.2}$$

Because of the similar structure of the functors defining LTSs and PTSs, these natural transformations can be decomposed in the same way as in the case of LTSs (cf. Sections 3.3.1 and 3.5.1). An outline is given in Figure 5.1. The natural transformations encountered in this decomposition differ from the ones in the nondeterministic setting in that the occurrences of the functor $\mathcal{P}_\omega$ are replaced by $\mathcal{D}_\omega + \underline{1}$ (and $\mathcal{P}^+_\omega$ by $\mathcal{D}_\omega$). The probabilistic nature does not come into play until we arrive at the bottom of the table, where we apply our main representation result. The technicalities are again deferred to the appendix. The main result, Theorem A.3.5, again bears similarity to the corresponding Theorem A.2.4 from the nondeterministic setting, but the proof is considerably more involved. Assembling the representations along the previous decomposition steps, we obtain a representation for the natural transformations ρ in (5.2). This representation is a probabilistic variant of the GSOS format, which we call *probabilistic GSOS*, or *PGSOS* for short. In the following two sections we

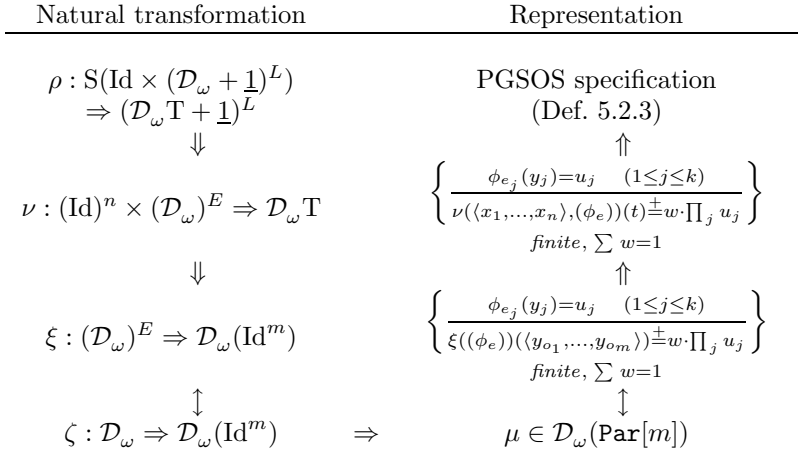


Figure 5.1: The outline of the approach in the probabilistic setting.

explain the details of the derivation of PGSOS. Readers who want to skip this rather technical part may want to jump immediately to Section 5.2.3, where the resulting format is described.

5.2.1 Top-down: decomposing natural transformations

First of all, by Lemma A.1.1 and the adjunction $\mathrm{Id} \times \underline{L} \dashv (\mathrm{Id})^L$, natural transformations in (5.2) are in one-to-one correspondence with those of the shape

$$\tilde{\rho} : \mathrm{S}\big(\mathrm{Id} \times (\mathcal{D}_\omega + \underline{1})^L\big) \times \underline{L} \Rightarrow \mathcal{D}_\omega \mathrm{T} + \underline{1}. \tag{5.3}$$

For a moment we abbreviate the functor describing the domain of these natural transformations as

$$\mathrm{F} := \mathrm{S}\big(\mathrm{Id} \times (\mathcal{D}_\omega + \underline{1})^L\big) \times \underline{L}.$$

With Lemma A.1.4, a statement telling how to write a functor as a coproduct, we obtain $\mathrm{F} \simeq \coprod_{z \in \mathrm{F}1} \mathrm{F}|_z$, where

$$\mathrm{F}|_z X := \{\phi \in \mathrm{F}X \mid (\mathrm{F}!)(\phi) = z\}$$

for the unique arrow $! : X \to 1$ into the final set $1 = \{*\}$ and $z \in \mathrm{F}1$. We shall derive a concrete description of $\mathrm{F}|_z$ below.

According to Lemma A.1.2, giving a natural transformation whose domain is a coproduct is equivalent to giving one natural transformation for each summand. So $\tilde{\rho}$ above can by be described by a family of natural transformations

$$(\nu^z : \mathrm{F}|_z \Rightarrow \mathcal{D}_\omega \mathrm{T} + \underline{1})_{z \in \mathrm{F}1}. \tag{5.4}$$

From the definition of $\mathrm{F}|_z$ we easily get $\mathrm{F}|_z 1 = \{z\}$, so the functor preserves finality. Lemma A.1.3 states that if a natural transformation with such a finality preserving domain functor has a coproduct as its codomain, then it touches one of the summands of that coproduct only. So each ν^z arises from a natural transformation either into $\mathcal{D}_\omega\mathrm{T}$ or into $\underline{1}$. In the latter case there is trivially precisely one choice for a natural transformation, so we need to record only those ν^z that map everything to $\mathcal{D}_\omega\mathrm{T}$, which means that the above representation is equivalent to giving

$$M \subseteq \mathrm{F}1 \quad \text{and} \quad (\tilde{\nu}^z : \mathrm{F}|_z \Rightarrow \mathcal{D}_\omega\mathrm{T})_{z\in M}. \tag{5.5}$$

To obtain a concrete description of $\mathrm{F}|_z$ for any $z \in \mathrm{F}1$ we make the calculation below. We use $\mathcal{D}_\omega 1 = \{\delta_*\} \simeq 1$, where δ_* with $\delta_*(*) = 1$ is the unique probability distribution on the singleton set $1 = \{*\}$, and write $2 := 1 + 1 \simeq \{\top, \bot\}$:

$$\begin{array}{rcl} \mathrm{F}1 & = & \mathrm{S}(1 \times (\mathcal{D}_\omega 1 + \underline{1})^L) \times L \\ & \simeq & \mathrm{S}(2^L) \times L \\ & \simeq & \{\langle \sigma(E_1, \ldots, E_n), a\rangle \mid \sigma \in \Sigma \text{ with arity } n, E_i \subseteq L, a \in L\}. \end{array}$$

For $z = \langle \sigma(E_1, \ldots, E_n), a\rangle \in \mathrm{F}1$ a calculation similar to the one in (3.6) yields

$$\mathrm{F}|_z \simeq (\mathrm{Id})^n \times (\mathcal{D}_\omega)^E,$$

where $E := E_1 + \cdots + E_n$. So each natural transformation $\tilde{\nu}^z$ for $z \in M$ appearing in (5.5) is, for a suitable number $n \in \mathbb{N}$ and set E, equivalent to a natural transformation

$$\nu : (\mathrm{Id})^n \times (\mathcal{D}_\omega)^E \Rightarrow \mathcal{D}_\omega\mathrm{T}. \tag{5.6}$$

With $N := \{1, \ldots, n\}$, Lemma A.1.7 states that a factor $(\mathrm{Id})^n \simeq (\mathrm{Id})^N$ in the domain of a natural transformation can be eliminated by precomposing the codomain functor with $\underline{N} + \mathrm{Id}$. So each natural transformation ν above is equivalent to a natural transformation

$$\tilde{\nu} : (\mathcal{D}_\omega)^E \Rightarrow \mathcal{D}_\omega\mathrm{T}(\underline{N} + \mathrm{Id}). \tag{5.7}$$

The correspondence between $\tilde{\nu}$ and ν is as follows: to transform a result of $\tilde{\nu}$ into one of ν we replace any leaf $i \in N$ appearing in the terms of the result of $\tilde{\nu}$ by the i-th additional argument of ν.

At this point, we need the following statement, which allows us to distribute the probability functor over a coproduct (cf. Lemma A.1.5).

Lemma 5.2.1 *For functors* $\mathrm{G}^i : \mathsf{C} \to \mathsf{Set}$ *(*$i \in I$*) we have*

$$\mathcal{D}_\omega\Big(\coprod_{i\in I} \mathrm{G}^i\Big) \simeq \coprod_{\mu\in\mathcal{D}_\omega I} \Big(\prod_{j\in \mathtt{supp}(\mu)} \mathcal{D}_\omega\mathrm{G}^j\Big).$$

Proof: For all sets X we have an equivalence of sets

$$\mathcal{D}_\omega\big(\coprod_{i\in I} \mathrm{G}^i X\big) \simeq \coprod_{\mu\in\mathcal{D}_\omega I}\ \Big(\prod_{j\in\mathtt{supp}(\mu)} \mathcal{D}_\omega \mathrm{G}^j X\Big)$$

given from left to right by

$$\phi \in \mathcal{D}_\omega\big(\coprod_{i\in I} \mathrm{G}^i X\big) \mapsto \iota_{\mu^\phi}\big((\phi_j)_{j\in\mathtt{supp}(\mu)}\big) \in \coprod_{\mu\in\mathcal{D}_\omega I}\ \Big(\prod_{j\in\mathtt{supp}(\mu)} \mathcal{D}_\omega \mathrm{G}^j X\Big)$$

where

$$\mu^\phi(i) := \phi[\iota_i\big[\mathrm{G}^i X\big]] \quad \text{and} \quad \phi_j(\alpha) := \frac{\phi\big(\iota_j(\alpha)\big)}{\mu^\phi(j)}$$

for all $j \in \mathtt{supp}(\phi)$ and $\alpha \in \mathrm{G}^j X$. The equivalence commutes with the action of the respective functors on arrows.

□

Writing again $|t|_*$ for the number of occurrences of the variable $* \in 1$ in a term $t \in \mathrm{T}(N+1)$, we get

$$\begin{aligned}\mathcal{D}_\omega \mathrm{T}(\underline{N} + \mathrm{Id}) &\overset{\text{cf. (3.25)}}{\simeq} \mathcal{D}_\omega\Big(\coprod_{t\in\mathrm{T}(N+1)} \mathrm{Id}^{|t|_*}\Big)\\ &\overset{\text{Lemma 5.2.1}}{\simeq} \coprod_{\mu\in\mathcal{D}_\omega\mathrm{T}(N+1)}\ \Big(\prod_{t\in\mathtt{supp}(\mu)} \mathcal{D}_\omega(\mathrm{Id}^{|t|_*})\Big).\end{aligned}$$

The codomain of the natural transformation $\tilde{\nu}$ can thus be written as a coproduct, and we can again apply Lemma A.1.3 to find that all results will be in the same μ-summand for some $\mu \in \mathcal{D}_\omega \mathrm{T}(N+1)$. This summand in turn is a product. So a natural transformation into it can be split according to Lemma A.1.2 (ii). Together we find that any natural transformation $\tilde{\nu}$ in (5.7) can be characterised by

$$\mu \in \mathcal{D}_\omega \mathrm{T}(N+1) \quad \text{and} \quad \big(\xi^t : (\mathcal{D}_\omega)^E \Rightarrow \mathcal{D}_\omega(\mathrm{Id}^{|t|_*})\big)_{t\in\mathtt{supp}(\mu)}. \tag{5.8}$$

We can stop the decomposition here, because for natural transformations of the type

$$\xi : (\mathcal{D}_\omega)^E \Rightarrow \mathcal{D}_\omega(\mathrm{Id}^m) \tag{5.9}$$

(for some $m \in \mathbb{N}$) as they appear in the representation (5.8), Corollary A.3.10 in the appendix provides a direct characterisation.

5.2.2 Bottom-up: constructing representations

We now develop representations – expressed in terms of derivation rules – for natural transformations of all types encountered in the above decomposition of abstract GSOS for probabilistic transition systems. We compose the representations along the previous decomposition steps, starting with natural transformations of the simplest type, namely ξ in (5.9). As already mentioned, we can apply Corollary A.3.10 to them. It states that we can characterise the natural transformation ξ by a set of transition rules as below.

$$\xi \doteq \left\{ \frac{\phi_{e_j}(y_j) = u_j \quad (1 \leq j \leq k)}{\xi\big((\phi_e)\big)(\langle y_{o_1}, \ldots, y_{o_m}\rangle) \stackrel{+}{=} w \cdot u_1 \cdot \ldots \cdot u_k} \right\}_{\textit{finite},\ \sum w=1} \tag{5.10}$$

Each rule mentions distinct element variables $y_1, \ldots, y_k$ and probability variables $u_1, \ldots, u_k$ for some $k \in \mathbb{N}$, where each y_j has an associated origin $e_j \in E$, so that y_j is instantiated with an element chosen according to the probability distribution ϕ_{e_j}. Moreover, the indices $o_1, \ldots, o_m \in \{1, \ldots, k\}$ determine which element variable appears in which position of the resulting tuple. The real number $w \in (0, 1]$ appearing in each rule is called the *weight* of the rule and the weights of all rules in the set are required to sum up to 1. We write a plus above the equality sign in the conclusion to express that after instantiating one of the rules, the real value calculated in the conclusion does not necessarily denote the whole probability assigned to the corresponding tuple, but the rule's contribution to it. The resulting probability of the tuple is defined to be the sum of all contributions derivable from different instances of the rules.

The following example is intended to illustrate how a natural transformation ξ is represented by such a set of rules.

Example 5.2.2 *Suppose in the case $E = \{1, 2\}$ and $m = 3$ that ξ is represented by the following two rules.*

$$\frac{\phi_1(x) = u \quad \phi_2(z) = v}{\xi(\langle\phi_1, \phi_2\rangle)(\langle x, z, z\rangle) \stackrel{+}{=} \frac{1}{5}\, u\, v} \qquad \frac{\phi_1(x) = u \quad \phi_1(y) = v \quad \phi_2(z) = w}{\xi(\langle\phi_1, \phi_2\rangle)(\langle x, y, z\rangle) \stackrel{+}{=} \frac{4}{5}\, u\, v\, w}$$

For a set $P = \{p, q\}$, and distributions $\psi_1, \psi_2 \in \mathcal{D}_\omega P$ with $\psi_1(p) = \frac{1}{3}$, $\psi_1(q) = \frac{2}{3}$, and $\psi_2(p) = \psi_2(q) = \frac{1}{2}$, we calculate the probability of $\langle p, q, q\rangle$ in $\xi_P(\psi_1, \psi_2)$. The rules can be instantiated to contribute to the probability of $\langle p, q, q\rangle$ as

$$\frac{\psi_1(p) = \frac{1}{3} \quad \psi_2(q) = \frac{1}{2}}{\xi_P(\psi_1, \psi_2)(\langle p, q, q\rangle) \stackrel{+}{=} \frac{1}{5} \cdot \frac{1}{3} \cdot \frac{1}{2} = \frac{1}{30}}$$

and

$$\frac{\psi_1(p) = \frac{1}{3} \quad \psi_1(q) = \frac{2}{3} \quad \psi_2(q) = \frac{1}{2}}{\xi_P(\psi_1, \psi_2)(\langle p, q, q\rangle) \stackrel{+}{=} \frac{4}{5} \cdot \frac{1}{3} \cdot \frac{2}{3} \cdot \frac{1}{2} = \frac{4}{45}}$$

We conclude $\xi_P(\psi_1, \psi_2)(\langle p, q, q\rangle) = \frac{1}{30} + \frac{4}{45} = \frac{11}{90}$.

According to (5.8), the representation of $\tilde{\nu}$ in (5.7) is now given by a distribution $\mu \in \mathcal{D}_\omega \mathrm{T}(N+1)$ and for each $t \in \mathtt{supp}(\mu)$, as a representation of ξ^t, a collection of rules as in (5.10) with $m = |t|_*$ (which was the number of occurrences of $*$ in t). To write this representation as one collection of rules, we need to do two things: First we change the rule notation such that each rule from the description of ξ^t for any $t \in \mathtt{supp}(\mu)$ mentions the respective term t. To this end we replace the vector $\langle y_{o_1}, \ldots, y_{o_m} \rangle$ in each rule by $t_Y \in \mathrm{T}(N+Y)$, where $Y = \{y_1, \ldots, y_k\}$. The term t_Y arises after replacing, for all i, the i-th occurrence of $*$ in the corresponding $t \in \mathrm{T}(N+1)$ by y_{o_i}. Second, we need to take into account the overall contribution of each ξ^t as given by the distribution μ. This is easily achieved by multiplying the weight w of each individual rule in the representation of ξ^t with $\mu(t)$ for the corresponding $t \in \mathrm{T}(N+1)$. As a result, we obtain a collection of rules whose weights again sum up to 1.

So a representation of $\tilde{\nu}$ in (5.7) is given by a finite set of rules as below with the global condition that all their weights w should sum up to 1.

$$\tilde{\nu} \doteq \left\{ \quad \frac{\phi_{e_j}(y_j) = u_j \quad (1 \leq j \leq k)}{\tilde{\nu}((\phi_e))(t_Y) \stackrel{+}{=} w \cdot u_1 \cdot \ldots \cdot u_k} \quad \right\}_{\textit{finite},\, \sum w=1} \tag{5.11}$$

For the step from $\tilde{\nu} : (\mathcal{D}_\omega)^E \Rightarrow \mathcal{D}_\omega \mathrm{T}(\underline{N} + \mathrm{Id})$ in (5.7) to $\nu : (\mathrm{Id})^n \times (\mathcal{D}_\omega)^E \Rightarrow \mathcal{D}_\omega \mathrm{T}$ in (5.6) each variable $i \in N = \{1, \ldots, n\}$ appearing in the target term $t_Y \in \mathrm{T}(N+Y)$ of any rule is replaced by x_i, where $X := \{x_1, \ldots, x_n\}$ is a set of n variable names different from those in Y. This yields sets of rules as below where $t \in \mathrm{T}(X+Y)$.

$$\nu \doteq \left\{ \quad \frac{\phi_{e_j}(y_j) = u_j \quad (1 \leq j \leq k)}{\nu(\langle x_1, \ldots, x_n \rangle, (\phi_e))(t) \stackrel{+}{=} w \cdot u_1 \cdot \ldots \cdot u_k} \quad \right\}_{\textit{finite},\, \sum w=1} \tag{5.12}$$

A natural transformation $\tilde{\rho} : \mathrm{S}\big(\mathrm{Id} \times (\mathcal{D}_\omega + \underline{1})^L\big) \times \underline{L} \Rightarrow \mathcal{D}_\omega \mathrm{T} + \underline{1}$ in (5.3) was characterised by a set

$$M \subseteq \{\langle \sigma(E_1, \ldots, E_n), a \rangle \mid \sigma \in \Sigma \text{ with arity } n, E_i \subseteq L, a \in L\},$$

together with natural transformations $\tilde{\nu}^z : \mathrm{F}|_z \Rightarrow \mathcal{D}_\omega \mathrm{T}$ for all $z \in M$. Each $\tilde{\nu}^z$ for $z = \langle \sigma(E_1, \ldots, E_n), a \rangle$ was of the type $(\mathrm{Id})^n \times (\mathcal{D}_\omega)^E \Rightarrow \mathcal{D}_\omega \mathrm{T}$ as characterised above, where $E = E_1 + \cdots + E_n$. We recapitulate how we calculate

$$\tilde{\rho}_X\big(\sigma(\langle x_1, \theta_1 \rangle, \ldots, \langle x_n, \theta_n \rangle), a\big) \tag{5.13}$$

for $\sigma \in \Sigma$ with arity n, $x_i \in X$, $\theta_i : L \to \mathcal{D}_\omega X + 1$, and $a \in L$ with our current representation. We first check whether or not $z = \langle \sigma(E_1, \ldots, E_n), a \rangle$ is in M, where $E_i := \{l \in L \mid \theta_i(l) \neq \iota_2(*)\}$. In case $z \notin M$ the result of (5.13) is $\iota_2(*)$, otherwise it is

$$\iota_1\big(\underbrace{\tilde{\nu}^z_X(\langle x_1, \ldots, x_n \rangle, (\phi_{i,l})_{1 \leq i \leq n, l \in E_i})}_{=:\psi}\big),$$

where $\theta_i(l) = \iota_1(\phi_{i,l})$ for $1 \leq i \leq n$ and $l \in E_i$. The value of ψ in turn is derived from the rule representation of $\tilde{\nu}^z$ in the format (5.12).

To obtain a direct representation of $\tilde{\rho}$, we collect all rules from the representations of all $\tilde{\nu}^z$ for $z \in M$ modified in two respects:

First, we need to make sure that each rule is applied in the right situation only, which was determined above by σ, a, and $E_1, \ldots, E_n$. The information about σ and a is put in the conclusion of the rule in the straightforward way. The sets $E_i \subseteq L$ give rise to new premises: for $b \notin E_i$ we add $\theta_i(b) = \iota_2(*)$, and for $b \in E_i$ we put $\theta_i(b) = \iota_1(\phi_{i,b})$.

Second, we need to account for the conversion of parameters θ_i into $\phi_{i,l}$. So we replace a distribution ϕ_{e_j} with $e_j = \iota_{i_j}(l_j) \in E = E_1 + \cdots + E_n$ as it appeared in a premise before by ϕ_{i_j,l_j}.

This yields rules of the following shape:

$$\frac{\begin{array}{ll} \theta_i(b) = \iota_2(*) & b \notin E_i, 1 \leq i \leq n \\ \theta_i(b) = \iota_1(\phi_{i,b}) & b \in E_i, 1 \leq i \leq n \\ \phi_{i_j,l_j}(y_j) = u_j & 1 \leq j \leq k \end{array}}{\tilde{\rho}\big(\sigma(\langle x_1, \theta_1\rangle, \ldots, \langle x_n, \theta_n\rangle), a\big)(t) \stackrel{+}{=} w \cdot u_1 \cdot \ldots \cdot u_k} \tag{5.14}$$

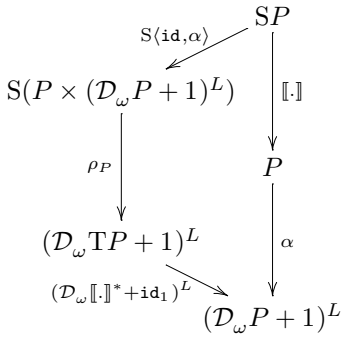
Note that the result of $\tilde{\rho}_P(\sigma(\langle x_1, \theta_1\rangle, \ldots, \langle x_n, \theta_n\rangle), a)$ needs to be in the coproduct $\mathcal{D}_\omega \mathrm{T}P + 1$, although no coproduct injections are written in the rules. We keep those injections implicit by the following convention for the reading of the conclusion of the rules: if any rules apply to the above expression then, for the distribution $\psi \in \mathcal{D}_\omega \mathrm{T}P$ these rules define on terms, the value of the expression is $\iota_1(\psi)$. If no rules apply (which means $z \notin M$), the expression evaluates to $\iota_2(*)$.

The condition on the original sets of rules translates into the following one: for any given $\sigma \in \Sigma$ with arity n, $a \in L$, and $E_1, \ldots, E_n \subseteq L$, the specification contains only finitely many rules mentioning σ, a, and $E_1, \ldots, E_n$, and the weights w of all these rules sum up to 1, if there are any.

From the above characterisation of $\tilde{\rho}$ we immediately get one for ρ in (5.2): we only need to replace $\tilde{\rho}\big(\sigma(\langle x_1, \theta_1\rangle, \ldots, \langle x_n, \theta_n\rangle), a\big)$ in the conclusion of each rule by $\rho\big(\sigma(\langle x_1, \theta_1\rangle, \ldots, \langle x_n, \theta_n\rangle)\big)(a)$.

According to Definition 3.5.3, the models of a specification ρ in abstract GSOS are the ρ-bialgebras, which are the $\langle \mathrm{S}, \mathrm{B}\rangle$-bialgebras $\langle P, [\![.]\!], \alpha\rangle$ – where $\mathrm{B} = (\mathcal{D}_\omega + \underline{1})^L$ is the functor describing probabilistic transition systems – such that the diagram below commutes. We write again $[\![.]\!]^*$ for the inductive extension

of $[\![.]\!]$ to terms (cf. Lemma 2.2.10).



We want to turn our characterisation of ρ in terms of rules as in (5.14) into a direct characterisation of the corresponding ρ-bialgebras. The arguments of ρ in this setting will be $\sigma(\langle p_1, \alpha(p_1)\rangle, \ldots, \langle p_n, \alpha(p_n)\rangle)$ for $\sigma(p_1, \ldots, p_n) \in \mathrm{S}P$, so the rules will instantiate to

$$\frac{\begin{array}{ll}\alpha(p_i)(b) = \iota_2(*) & b \notin E_i, 1 \leq i \leq n\\ \alpha(p_i)(b) = \iota_1(\phi_{i,b}) & b \in E_i, 1 \leq i \leq n\\ \phi_{i_j,l_j}(y_j) = u_j & 1 \leq j \leq k\end{array}}{\rho_P\big(\sigma(\langle p_1, \alpha(p_1)\rangle, \ldots, \langle p_n, \alpha(p_n)\rangle)\big)(a)(t) \stackrel{+}{=} w \cdot u_1 \cdot \ldots \cdot u_k}$$

Using the arrow notation for α as introduced in Definition 5.1.2, we can replace the premises

- $\alpha(p_i)(b) = \iota_2(*)$ by $p_i \xrightarrow{b}\!\!\!\!\!\!\rightarrow$ (the *negative applicability premise*),
- $\alpha(p_i)(b) = \iota_1(\phi_{i,b})$ by $p_i \xrightarrow{b}$ (the *positive applicability premises*), and
- $\phi_{i_j,l_j}(y_j) = u_j$ – which required a premise $\alpha(p_{i_j})(l_j) = \iota_1(\phi_{i_j,l_j})$ – by $p_{i_j} \stackrel{l_j[u_j]}{\Longrightarrow} y_j$.

Moreover, since the result of the application of ρ to the above arguments should – after the terms are evaluated by $[\![.]\!]^*$ – equal $\alpha([\![\sigma(p_1, \ldots, p_n)]\!])$, we can rewrite the conclusion of the above instance of a rule to

$$[\![\sigma(p_1, \ldots, p_n)]\!] \stackrel{a[w \cdot u_1 \cdot \ldots \cdot u_k]}{\Longrightarrow} [\![t]\!]^*.$$

So the ρ-bialgebras are directly characterised by rules of the shape

$$\frac{\begin{array}{ll}x_i \xrightarrow{b} & b \in E_i,\ 1 \leq i \leq n\\ x_i \xrightarrow{b}\!\!\!\!\!\!\rightarrow & b \notin E_i,\ 1 \leq i \leq n\\ x_{i_j} \stackrel{l_j[u_j]}{\Longrightarrow} y_j & 1 \leq j \leq k\end{array}}{[\![\sigma(x_1, \ldots, x_n)]\!] \stackrel{a[w \cdot u_1 \cdot \ldots \cdot u_k]}{\Longrightarrow} [\![t]\!]^*}$$

For simplicity, we omit the application of $[\![.]\!]$ and $[\![.]\!]^*$. Moreover, we relax the condition that for each $1 \leq i \leq n$ and $b \in L$ a rule needs to contain either a positive or negative applicability premise (depending on whether or not b is in E_i). Allowing that neither of them is present amounts to replacing each set E_i of *enabled* labels by two disjoint subsets $R_i, P_i \subseteq L$ of *requested* and *prohibited* labels of the i-th argument. We call a rule where for some i and b neither a positive nor negative applicability premise is present *incomplete* and view it as an abbreviation for the set of rules that arises when for each missing one either the corresponding positive or negative applicability premise is added.

5.2.3 The PGSOS format

The above consideration yields a novel rule-style specification format for PTS. Because of its relation to the GSOS format, we call it *probabilistic GSOS*.

Definition 5.2.3 *Given a signature Σ, a* **rule in probabilistic GSOS (PGSOS)** *has the shape*

$$\frac{\begin{array}{ll} x_i \xrightarrow{b} & b \in R_i,\ 1 \leq i \leq n \\ x_i \xrightarrow{b} \!\!\!\!\!\!\rightarrow & b \in P_i,\ 1 \leq i \leq n \\ x_{i_j} \overset{l_j[u_j]}{\Longrightarrow} y_j & 1 \leq j \leq J \end{array}}{\sigma(x_1, \ldots, x_n) \overset{a[w \cdot u_1 \cdot \ldots \cdot u_J]}{\Longrightarrow} t} \tag{5.15}$$

where

- $\sigma \in \Sigma$ *(with arity $n \in \mathbb{N}$) is the* type *of the rule,*
- $x_1, \ldots, x_n$ *are distinct state variables (we set $X := \{x_1, \ldots, x_n\}$),*
- $R_i, P_i \subseteq L$ *with $R_i \cap P_i = \emptyset$ are the sets of* requested *and* prohibited *labels for the i-th argument x_i ($1 \leq i \leq n$),*
- $y_1, \ldots, y_J$ *for some $J \in \mathbb{N}$ are distinct state such that $Y \cap X = \emptyset$ for $Y := \{y_1, \ldots, y_J\}$, where each y_j is tagged as a successor of argument x_{i_j} with $i_j \in \{1, \ldots, n\}$ for a requested label $l_j \in R_{i_j}$ ($1 \leq j \leq J$),*
- $u_1, \ldots, u_J$ *are distinct* probability variables,
- $a \in L$ *is the* label *of the rule,*
- $w \in (0, 1]$ *is the* weight *of the rule.*
- $t \in \mathrm{T}(X \cup Y)$ *such that $Y \subseteq \mathtt{vars}(t)$ is the* target *of the rule.*

A PGSOS rule as above is **triggered** *by sets of enabled labels $E_1, \ldots, E_n \subseteq L$ if $R_i \subseteq E_i$ and $E_i \cap P_i = \emptyset$ for all $1 \leq i \leq n$.*

A **PGSOS specification** *is a collection $\mathcal{R}$ of PGSOS rules such that for all $\sigma \in \Sigma$ (with arity n), $a \in L$, and $E_1, \ldots, E_n \subseteq L$ only finitely many rules with type σ and label a in $\mathcal{R}$ are triggered by $E_1, \ldots, E_n$, and in case there are any, the weights of all these rules sum up to 1.*

To define the models of PGSOS specification, we use the following auxiliary notion: Let $\langle P, \alpha \rangle$ be a PTS, let R be the PGSOS rule in (5.15), let $p_1, \ldots, p_n \in P$ be states, let $t_P \in \mathrm{T}P$ be a term with states as variables, and let $c \in (0, 1]$ be a (positive) probability. We write

$$R \vdash \sigma(p_1, \ldots, p_n) \stackrel{a[c]}{\Longrightarrow} t_P$$

if there are states $q_1, \ldots, q_J \in P$ and positive probabilities $r_1, \ldots, r_J \in (0, 1]$ such that

- $p_i \stackrel{b}{\longrightarrow}$ for all $b \in R_i$ and $1 \leq i \leq n$,
- $p_i \stackrel{b}{\not\longrightarrow}$ for all $b \in P_i$ and $1 \leq i \leq n$,
- $p_{i_j} \stackrel{l_j[r_j]}{\Longrightarrow} q_j$ for all $1 \leq j \leq J$,
- $c = w \cdot r_1 \ldots r_J$, and
- t_P arises from t after replacing each variable x_i by p_i and each variable y_j by q_j.

Definition 5.2.4 *A* **model** *of a PGSOS specification $\mathcal{R}$ is a triple $\langle P, [\![.]\!], \alpha \rangle$ consisting of a PTS $\langle P, \alpha \rangle$ and an interpretation $[\![.]\!]$ of the operators in Σ on P such that the following holds: for all $\sigma \in \Sigma$ (with arity n), $a \in L$, and $p_1, \ldots, p_n, q \in P$ we have*

$$[\![\sigma(p_1, \ldots, p_n)]\!] \stackrel{a[r]}{\Longrightarrow} q$$

according to α if and only if

$$r = \sum \big(c \;\big|\; R \vdash \sigma(p_1, \ldots, p_n) \stackrel{a[c]}{\Longrightarrow} t_P,\ [\![t_P]\!]^* = q,\ t_P \in \mathrm{T}P,\ R \in \mathcal{R} \big),$$

where $[\![.]\!]^*$ is the inductive extension of $[\![.]\!]$ to terms.

At first sight, PGSOS rules offer a rather limited control over the transition probabilities: we cannot constrain the probabilities in the premises to equal an absolute value or lie inside a certain range, and the probability of the generated transition has to depend linearly on the probabilities from the premises. Still it turns out that the rules are sufficiently powerful, as we shall demonstrate with a list of examples in the next section. There we also argue that rules with premises that do constrain the actual probabilities can specify non-well-behaved operators, which convinces us that this limitation is the "right" one.

When asked to design a probabilistic specification format, one would probably not come up with similar or the same restrictions immediately. This underlines the usefulness of the categorical approach of Turi and Plotkin combined with our precise method to analyse the corresponding natural transformations.

5.2.4 Examples of PGSOS specifications

To illustrate the PGSOS format, we present specifications for some basic probabilistic constructs.

(i) Consider the atomic action constant $a \in \Sigma$ for $a \in L$. The associated process should have a as its only enabled label and the a-transition should lead with probability 1 to an idle state, i.e. one that cannot do any transitions. The latter is obtained as the interpretation of the constant $0 \in \Sigma$ for which no transition rules are given. We specify the constant $a \in \Sigma$ by the following single rule without premises.

$$\frac{}{a \overset{a[1]}{\Longrightarrow} 0}$$

(ii) Next we consider a binary probabilistic choice operator $\oplus_r \in \Sigma$ for $r \in [0,1]$. For processes x and y we want $x \oplus_r y$ to be a process behaving either as x or as y, depending on the first input label and the probability r. In case the input can only be processed by x, the system should behave like x, and similar for y. If both can react, the decision should be made in favour of x with probability r and in favour of y with probability $\bar{r} := 1-r$. This is captured by the following set of PGSOS rules, each for all $a \in L$:

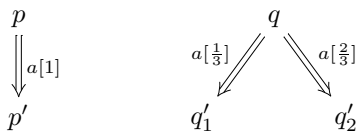
$$\frac{x \overset{a[u]}{\Longrightarrow} x' \qquad y \overset{a}{\longrightarrow\!\!\!\!\!\!\rightarrow}}{x \oplus_r y \overset{a[u]}{\Longrightarrow} x'} \qquad \frac{x \overset{a}{\longrightarrow\!\!\!\!\!\!\rightarrow} \qquad y \overset{a[v]}{\Longrightarrow} y'}{x \oplus_r y \overset{a[v]}{\Longrightarrow} y'}$$

$$\frac{x \overset{a[u]}{\Longrightarrow} x' \qquad y \overset{a}{\longrightarrow}}{x \oplus_r y \overset{a[r\cdot u]}{\Longrightarrow} x'} \qquad \frac{x \overset{a}{\longrightarrow} \qquad y \overset{a[v]}{\Longrightarrow} y'}{x \oplus_r y \overset{a[\bar{r}\cdot v]}{\Longrightarrow} y'}$$

To see that these rules satisfy the global constraints from Def. 5.2.3, for all $a \in L$ and sets of enabled labels $E_x, E_y \subseteq L$ for x and y we have to inspect the rules for $\oplus_r$ and a which are triggered according to the premises: in case a is neither enabled in x nor in y (i.e. $a \notin E_x \cup E_y$) no rule is triggered; in case a is enabled by just one of x or y (i.e. $a \in (E_x \setminus E_y) \cup (E_y \setminus E_x)$) it is one of the upper rules, each of which has weight 1; if a is enabled by both x and y (i.e. $a \in E_x \cap E_y$), both lower rules are triggered, the weights of which sum up to $r + \bar{r} = 1$.

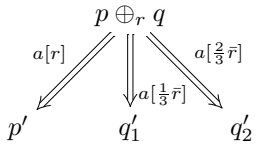
To illustrate Def. 5.2.4, we spell out what the requirement on a model $\langle P, [\![.]\!], \alpha \rangle$ amounts to in a concrete case. Let $\langle P, \alpha \rangle$ contain the two processes drawn below, where we assume that p', q'_1, and q'_2 do not have

outgoing transitions.



(5.16)

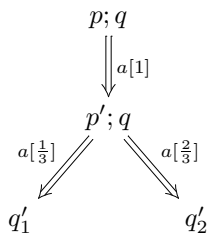
Both can make an a-transition, so the third and fourth rule are applicable to $p \oplus_r q$ (for readability, here and in the following, we leave the interpretation of operators in our model implicit, i.e. we omit the application of $[\![.]\!]$ or $[\![.]\!]^*$). They derive an a-transition which leads to the a-successor of p with probability r and to an a-successor of q otherwise. The relative probability of moving from $p \oplus_r q$ to q'_i given that a successor of q is chosen is the same as the absolute probability of moving from q to q'_i.



(iii) Another useful operator is the sequential composition $x; y$. It behaves like x until a situation is reached where no more transitions are possible. Then the process y takes over. We define the operator by the following transition rules for all $a \in L$.

$$\frac{x \overset{a[u]}{\Longrightarrow} x'}{x; y \overset{a[u]}{\Longrightarrow} x'; y} \qquad \frac{x \overset{l}{\not\longrightarrow} \ (\forall l \in L) \qquad y \overset{a[v]}{\Longrightarrow} y'}{x; y \overset{a[v]}{\Longrightarrow} y'}$$

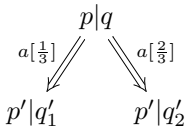
Note that for an infinite set of labels L the rules of the second type have infinitely many premises, which is allowed by the PGSOS format. (Various other notions of a sequential composition arise from different definitions of the moment when the second component takes over. Baeten [Bae03] for instance introduces an explicit termination predicate. The same idea can be applied here, but we would have to modify our definition of a PTS to include this predicate.) With our example processes p and q from (5.16) we get the following picture for $p; q$:



(iv) Furthermore, we define the synchronous parallel composition $x|y$ modelling a process with two components x and y waiting for input side by side. The enabled labels are those that are enabled for both of x and y. On such a label each component will independently make a move according to its own transition probability, and the whole process will become the synchronous parallel composition of the two resulting states. The operation is defined by the following set of rules for all $a \in L$:

$$\frac{x \overset{a[u]}{\Longrightarrow} x' \qquad y \overset{a[v]}{\Longrightarrow} y'}{x|y \overset{a[u\cdot v]}{\Longrightarrow} x'|y'}$$

Considering again p and q from (5.16) we get that $p|q$ is the process below.



Note that the (implicitly applied) term evaluation $[\![.]\!]^*$ of our model may identify $p'|q'_1$ and $p'|q'_2$. In that case the two arrows above actually represent one arrow with probability $\frac{1}{3} + \frac{2}{3} = 1$.

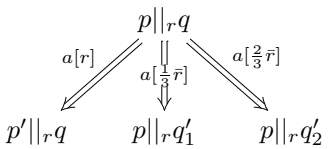
(v) For any $r \in [0, 1]$, the asynchronous parallel composition $x||_r y$ of the two processes x and y is intended to behave as follows: an input label a can be processed if it is enabled in x or in y. The input is always handled by one of them only, the other stays unchanged. If both components are able to deal with the input, then x is chosen with the probability r.

The operator is specified by the rules below, each for all $a \in L$ and with $\bar{r} := 1 - r$.

$$\frac{x \overset{a[u]}{\Longrightarrow} x' \qquad y \overset{a}{\not\longrightarrow}}{x||_r y \overset{a[u]}{\Longrightarrow} x'||_r y} \qquad \frac{x \overset{a}{\not\longrightarrow} \qquad y \overset{a[v]}{\Longrightarrow} y'}{x||_r y \overset{a[v]}{\Longrightarrow} x||_r y'}$$

$$\frac{x \overset{a[u]}{\Longrightarrow} x' \qquad y \overset{a}{\longrightarrow}}{x||_r y \overset{a[r\cdot u]}{\Longrightarrow} x'||_r y} \qquad \frac{x \overset{a}{\longrightarrow} \qquad y \overset{a[v]}{\Longrightarrow} y'}{x||_r y \overset{a[\bar{r}\cdot v]}{\Longrightarrow} x||_r y'}$$

Again for p and q from (5.16) we get the following transitions:



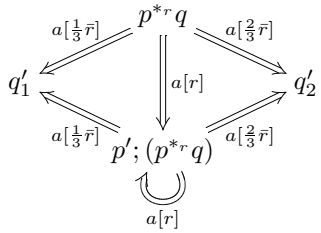
(vi) All the examples so far were simple in the sense that they did not involve target terms with more than one operator application. As a more complex

example we specify a probabilistic variant of the Kleene-Star operator $x^{*_r}y$ for $r \in [0,1]$. It uses the sequential composition from above. The operator is specified by the following rules, each for all $a \in L$, where again $\bar{r}$ abbreviates $1-r$.

$$\frac{x \overset{a[u]}{\Longrightarrow} x' \qquad y \overset{a}{\longrightarrow}}{x^{*_r}y \overset{a[r\cdot u]}{\Longrightarrow} x';(x^{*_r}y)} \qquad \frac{x \overset{a[u]}{\Longrightarrow} x' \qquad y \overset{a}{\not\longrightarrow}}{x^{*_r}y \overset{a[u]}{\Longrightarrow} x';(x^{*_r}y)}$$

$$\frac{x \overset{a}{\longrightarrow} \qquad y \overset{a[v]}{\Longrightarrow} y'}{x^{*_r}y \overset{a[\bar{r}\cdot v]}{\Longrightarrow} y'} \qquad \frac{x \overset{a}{\not\longrightarrow} \qquad y \overset{a[v]}{\Longrightarrow} y'}{x^{*_r}y \overset{a[v]}{\Longrightarrow} y'}$$

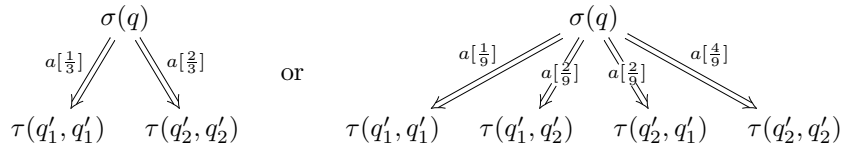
For p and q from (5.16) we get the picture below. Note again that $p^{*_r}q$ and $p';(p^{*_r}q)$ may be the same state according to the (implicitly applied) interpretation of terms $[\![.]\!]^*$.



One aspect of the format is not illustrated by the examples above, namely the possibility that the target term of a rule may have several occurrences of variables for successors of the same argument via transitions with the same label. We give an artificial example to show that in such a situation it makes a difference whether the same or different successor variables are used. Consider the following alternative rules for a signature Σ containing the two operator symbols σ (unary) and τ (binary).

$$\frac{x \overset{a[u]}{\Longrightarrow} x'}{\sigma(x) \overset{a[u]}{\Longrightarrow} \tau(x',x')} \qquad \text{or} \qquad \frac{x \overset{a[u]}{\Longrightarrow} x_1' \qquad x \overset{a[v]}{\Longrightarrow} x_2'}{\sigma(x) \overset{a[u\cdot v]}{\Longrightarrow} \tau(x_1',x_2')}$$

For q from (5.16) the two rules will generate the following a-transitions for $\sigma(q)$ respectively:



To conclude this example section, we also want to remark on the limits of the format by discussing an operation that is not covered, but which may appear

reasonable at first sight. The operation, however, turns out not to be well-behaved, which perfectly justifies that it cannot be specified within the format.

Consider the unary operation $\delta \in \Sigma$, which is intended to extract the deterministic part of the behaviour of a process by eliminating all transitions leading to distributions of more than one successor state. It is specified by the following rules for all $a \in L$:

$$\frac{x \stackrel{a[1]}{\Longrightarrow} x'}{\delta(x) \stackrel{a[1]}{\Longrightarrow} \delta(x')} \tag{5.17}$$

These rules are not in the PGSOS format, because it does not allow to prescribe absolute probabilities in the premises, as it is done here. We can only put probability variables in the premises.

To see that this specification is troublesome, assume that $\langle P, [\![.]\!], \alpha \rangle$ is a model containing the two processes p and q from (5.16). Note that p and q are probabilistically bisimilar: for both the only enabled label is a and the a-transition leads to an inert state with probability one (more precisely, it is easy to check that the relation $R = \{\langle p, q\rangle, \langle p', q'_1\rangle, \langle p', q'_2\rangle\}$ is a probabilistic bisimulation). Still, $\delta(p)$ and $\delta(q)$ are not bisimilar, because $\delta(p)$ can do an a-transition while $\delta(q)$ cannot. So in the interpretation $[\![.]\!]$ of the model the operator δ does not preserve bisimilarity, one of our basic requirements for a well-behaved model.

5.2.5 Properties of PGSOS

In Sections 5.2.1 through 5.2.3 we have derived the PGSOS format (cf. Def. 5.2.3) as a concrete characterisation of classes of models definable by specifications in abstract GSOS (cf. Def. 3.5.3) instantiated with probabilistic transition systems. As a consequence, the properties established for abstract GSOS specifications in Chapters 3 and 4 instantiate to properties of PGSOS. We spell them out in this section. First we state the correspondence established in the mentioned sections formally.

Theorem 5.2.5 *For a signature Σ with associated functor S and $\mathrm{B} = (\mathcal{D}_\omega + \underline{1})^L$, each subclass of $\langle \mathrm{S}, \mathrm{B}\rangle$-bialgebras definable as ρ-bialgebras for some specification ρ in abstract GSOS arises as a class of models of some PGSOS specification and vice versa. Moreover, this correspondence is one-to-one up to the renaming of variables in the rules, the splitting of rules, and equivalent abbreviations of sets of complete rules by incomplete ones.*

With this statement, Corollary 3.5.4 instantiates as follows.

Corollary 5.2.6 *Every specification $\mathcal{R}$ in PGSOS has an initial and a final model. In particular, the set of closed terms for the signature Σ carries a unique structure of a PTS extending it to a model of $\mathcal{R}$, and this model is initial. Dually,*

the final PTS (cf. Lemma 5.1.5) can uniquely be equipped with an interpretation of the operators in Σ such that we obtain a model of $\mathcal{R}$, and this model is final.

From Corollary 3.5.5 we obtain the following.

Corollary 5.2.7 *Probabilistic bisimilarity (cf. Def. 5.1.4) between two models of a PGSOS specification is a congruence for the operator interpretations.*

From the theory developed in Chapter 4 we learn that the bisimulation up-to-context proof principle is valid for operators defined by a PGSOS specification (see Corollary 4.4.9). However, our experiments indicate that in the probabilistic setting the bisimulation up-to-context technique is less useful than in the nondeterministic one for the following reason. Since with the additional information about transition probabilities we can distinguish processes more easily, many familiar process equivalences for nondeterministic systems do not have an equivalent in the world of PTSs. But the success of the bisimulation up-to-context proof principle depends on such laws to hold, because we usually first need to rewrite the given process terms in order to make a common context visible.

To give an example, consider the associative law for nondeterministic choice. We cannot establish a similar law for the probabilistic choice operator from above: for no $u, v \in (0,1)$ we can find values $u', v' \in [0,1]$ such that

$$(p \oplus_u q) \oplus_v r \sim p \oplus_{u'} (q \oplus_{v'} r)$$

for all states p, q, r in any model of the specification. To sketch the argument, let us study the a-transitions from

$$s = (p \oplus_u q) \oplus_v r \quad \text{and} \quad t = p \oplus_{u'} (q \oplus_{v'} r).$$

Assume that p, q, and r can all do a-transitions. To guarantee that an a-transition from both s and t leads to an a-successor of p with the same probability, we find that we need $u' = u \cdot v$. But under the assumption that only p and q can do an a-transition, the same analysis yields $u' = u$, a contradiction.

The specification of processes by guarded recursive definitions, however, is useful in the probabilistic setting. We first spell out the relevant definitions.

Definition 5.2.8 *A* **(probabilistic) guarded recursive specification** *over the signature Σ is a pair $\langle X, Tr \rangle$ consisting of a set of variables X and a set of transitions*

$$Tr \subseteq \big\{\, x \xrightarrow{a[u]} t \;\big|\; x \in X,\ a \in L,\ u \in (0,1],\ t \in \mathrm{T}X \big\},$$

where T is the term functor over Σ, such that for all $x \in X$ and $a \in L$ the set Tr contains finitely many transitions from x with label a only, the probabilities u of which sum up to 1 if there are any. A **solution** *of $\langle X, Tr \rangle$ in a model*

$\langle P, [\![.]\!], \alpha \rangle$ of a PGSOS specification $\mathcal{R}$ is given by an assignment of variables $h : X \to P$ such that for all $x \in X$, $a \in L$, and $q \in P$ we have

$$h(x) \stackrel{a[r]}{\Longrightarrow} q$$

if and only if

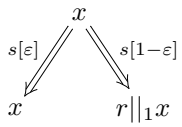
$$r = \sum\big(u \mid (x \stackrel{a[u]}{\Longrightarrow} t) \in Tr,\ [\![\, t[\, y := h(y) \mid y \in X \,]\,]\!]^* = q\big).$$

Corollary 4.4.7 instantiates to the following statement.

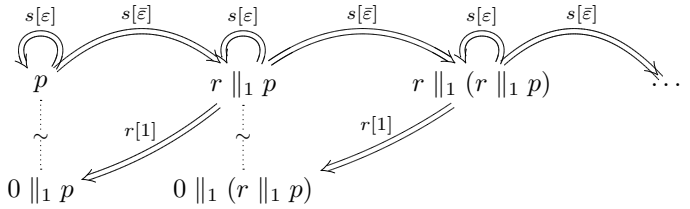
Corollary 5.2.9 *Every (probabilistic) guarded recursive definition $\langle X, Tr \rangle$ over Σ has solutions in some model of the PGSOS specification $\mathcal{R}$ for the operators in Σ. These solutions are determined up to bisimilarity. In particular, $\langle X, Tr \rangle$ has a unique solution in a final model of $\mathcal{R}$ (which exists by Corollary 5.2.6).*

The following simple example demonstrates the use of this principle.

Example 5.2.10 *We can alternatively specify a* **lossy bag** *(cf. Example 5.1.3) as a state x in some probabilistic transition system with the following behaviour: it can perform a store action (s) which keeps it unchanged with probability ε or otherwise leads to a state behaving like x except that it can do one additional remove action (r) at an arbitrary moment in the future. Using the operators specified in Section 5.2.4 this can be expressed by the guarded recursive specification $\langle \{x\}, Tr \rangle$ where the set Tr contains the two transitions drawn below.*



Corollary 5.2.9 says that this specification has solutions, determined up to bisimilarity. Such a solution is given by a model $\langle P, [\![.]\!], \alpha \rangle$ of the operators from Section 5.2.4 and a state $p \in P$ (the state that x is mapped on) which exhibits the behaviour shown below. We again omitted the application of the operator interpretation, i.e. $[\![.]\!]^*$. Also, the transitions from the states in the lower row are not drawn.



The states p and $0 \parallel_1 p$ (as well as $r \parallel_1 p$ and $0 \parallel_1 (r \parallel_1 p)$ and so forth) are not necessarily identical, but they are bisimilar. From this we conclude that the state p in any such solution is bisimilar to the state p_0 from Example 5.1.3.

Note that in order to obtain the well-behavedness results about PGSOS stated in this section, most of the effort we spent establishing the correspondence of abstract GSOS and PGSOS is not necessary: it is sufficient to know that a specification in PGSOS can be captured by a natural transformation ρ as in (5.2). We do not need to prove that all natural transformations ρ arise in such a way, which is actually the hard part. We tackled both directions in order to determine the exact position of PGSOS in Turi and Plotkin's framework. When we describe a specification format for Segala systems in the next section, we content ourselves with arguing that it is well behaved. We shall not establish a one-to-one correspondence with abstract GSOS in that setting. We expect that this correspondence is harder to obtain because we would have to deal with natural transformations involving a nested application of the the powerset and distribution functor. We do not have the tools yet to decompose such natural transformations.

5.3 A specification format for Segala systems

In this section we introduce Segala-GSOS, a well-behaved operator specification format for (simple) Segala systems from Def. 5.1.6. Admittedly, we do not yet understand the situation for these systems as well as we understand it for the less complex PTS. Here we give a preliminary report only in order to emphasise the strength of the categorical approach to yield formats for different system types.

Whereas PGSOS was developed as a concrete representation of specifications in abstract GSOS, we define our format for Segala systems in an ad-hoc way. We do not know whether all specifications in abstract GSOS can be expressed by rules in Segala-GSOS. But we argue that the converse is true, which is enough to conclude that Segala-GSOS is well-behaved in all respects studied here.

We start by defining the new format and its models. Next, we give a number of examples in order to illustrate both the meaning of the rules and their expressive power. Then we spell out the well-behavedness properties.

5.3.1 The Segala-GSOS format

The format we propose for Segala systems from Def. 5.1.6 is the following.

Definition 5.3.1 *Given a signature Σ, a* **Segala-GSOS rule** *has the following*

shape:

$$\frac{\begin{array}{ll} x_i \xrightarrow{b} & (b \in R_i; 1 \le i \le n) \\ x_i \xrightarrow{b} \!\!\!\!\rightarrow & (b \in P_i; 1 \le i \le n) \\ x_{i_j} \xrightarrow{l_j} \mu_j & (1 \le j \le J) \\ \mu_{j_k} \Longrightarrow z_k & (1 \le k \le K) \end{array}}{\sigma(x_1, \ldots, x_n) \xrightarrow{a} w_1 \cdot t_1 + \cdots + w_M \cdot t_M} \tag{5.18}$$

where

- $\sigma \in \Sigma$ *is an operator symbol with arity* n *that we call the* type *of the rule, and* $X = \{x_1, \ldots, x_n\}$ *is a set of* n *state variables;*
- $a \in L$ *is the* label *of the rule;*
- $R_i, P_i \subseteq L$ *with* $R_i \cap P_i = \emptyset$ *are sets of* requested *and* prohibited *labels for* x_i *for* $1 \le i \le n$*;*
- $Y = \{\mu_1, \ldots, \mu_J\}$ *is a set of* $J \in \mathbb{N}$ *probability distribution variables,* $1 \le i_j \le n$ *and* $l_j \in R_{i_j}$ *for all* $1 \le j \le J$*;*
- $Z = \{z_1, \ldots, z_K\}$ *is a set of* $K \in \mathbb{N}$ *state variable names with* $Z \cap X = \emptyset$*,* $1 \le j_k \le J$ *for* $1 \le k \le K$*;*
- *for* $M \in \mathbb{N}$*,* $t_1, \ldots, t_M \in \mathrm{T}(X + \tilde{Y} + Z)$ *are* target terms*, each with an associated weight* $w_m \in (0, 1]$ *such that* $w_1 + \cdots + w_M = 1$*. Here* T *is the term functor generated by* Σ *(cf. Lemma 2.2.10) and* $\tilde{Y} = \{\mu_j^{(o)} \mid 1 \le o \le O_j, 1 \le j \le J\}$ *for multiplicities* $O_1, \ldots, O_J > 0$ *is a set of probability variables derived by possibly duplicating some of the ones in* Y *(for* $M = 1$ *we omit the weight* $w_1 = 1$ *and for* $O_j = 1$ *we omit the upper index of* $\mu_j^{(1)}$*).*

A **specification in Segala-GSOS** *for the operators in* Σ *is a set* $\mathcal{R}$ *of Segala-GSOS rules such that for all* $\sigma \in \Sigma$*, with arity* n*, all labels* $a \in L$*, and all sets of enabled labels* $E_1, \ldots, E_n \subseteq L$ *there are only finitely many rules in* $\mathcal{R}$ *of type* σ *with label* a *triggered by* $E_1, \ldots, E_n$*. The rule in (5.18) is triggered by* $E_1, \ldots, E_n$ *if* $R_i \subseteq E_i$ *and* $E_i \cap P_i = \emptyset$ *for* $1 \le i \le n$*.*

To define the models of a Segala-GSOS specification, we need the following definitions. Let $V = \{v_1, \ldots, v_J\}$ be a *finite* set of variables, and let $t_V \in \mathrm{T}V$ be a term in which each variable in V occurs. A mapping $d : V \to \mathcal{D}_\omega P$ determines a distribution $\tilde{d}(t_V) \in \mathcal{D}_\omega \mathrm{T}P$ defined for all $t \in \mathrm{T}P$ as

$$\tilde{d}(t_V)(t) := \begin{cases} \prod_{j=1}^{J} d(v_j)(p_j) & \text{if } t = t_V[v_j := p_j] \text{ for } p_1, \ldots, p_J \in P, \\ 0 & \text{otherwise.} \end{cases} \tag{5.19}$$

The distribution $\tilde{d}(t_V)$ is generated by the following probabilistic procedure: for every variable v_j we independently draw an element p_j according to the distribution $d(v_j)$ and return the term t_V with the variables instantiated with the corresponding elements. Note that it is essential to consider a set of variables V instead of starting right away with a term that has distributions as variables, i.e. with $(\mathrm{T}d)(t_V) \in \mathrm{T}\mathcal{D}_\omega P$: if the same distribution μ appears in several variable positions of the latter term, we do not know whether to choose an element for each position separately or whether to draw just one element and put it in all positions in which μ occurs. We introduced the multiplicities O_j in the definition of a Segala-GSOS rule for a similar reason – see the explanation in the example section below.

Moreover, for distributions $\phi_1, \ldots, \phi_M \in \mathcal{D}_\omega P$ and weights $w_1, \ldots, w_M \in (0, 1]$ with $w_1 + \cdots + w_M = 1$ we write $w_1 \cdot \phi_1 + \cdots + w_M \cdot \phi_M \in \mathcal{D}_\omega P$ for the *convex combination* of the ϕ_m defined pointwise. For $p \in P$ we define the *Dirac distribution* $\delta_p \in \mathcal{D}_\omega P$ by $\delta_p(p) = 1$ and $\delta_p(q) = 0$ for $q \neq p$.

Definition 5.3.2 *A triple $\langle P, [\![.]\!], \alpha\rangle$, where $\langle P, \alpha\rangle$ is a Segala-system and $[\![.]\!]$ is an interpretation of the operator symbols in the signature Σ, is a* **model** *of a specification $\mathcal{R}$ in Segala-GSOS if for all operators $\sigma \in \Sigma$ with arity n and states $p_1, \ldots, p_n \in P$ the system $\langle P, \alpha\rangle$ allows a transition*

$$[\![\sigma(p_1, \ldots, p_n)]\!] \xrightarrow{a} \phi$$

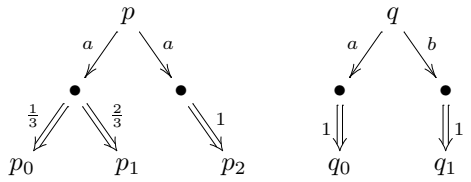
for $\phi \in \mathcal{D}_\omega P$ if and only if there is a rule with type σ and label a in $\mathcal{R}$ and – using the names from (5.18) for this rule – there are $\nu_1, \ldots, \nu_J \in \mathcal{D}_\omega P$ and $q_1, \ldots, q_K \in P$ satisfying the following:

- $p_i \xrightarrow{b}$ *for all $b \in R_i$ and $1 \leq i \leq n$,*
- $p_i \xrightarrow{b}\!\!\!\!\!\!\rightarrow$ *for all $b \in P_i$ and $1 \leq i \leq n$,*
- $p_{i_j} \xrightarrow{l_j} \nu_j$ *for all $1 \leq j \leq J$,*
- $q_k \in \mathtt{supp}(\nu_{j_k})$ *for all $1 \leq k \leq K$, and*
- $\phi = (\mathcal{D}_\omega [\![.]\!]^*)(w_1 \cdot d(t_1) + \cdots + w_M \cdot d(t_M))$, *where $[\![.]\!]^*$ is the inductive extension of $[\![.]\!]$ to terms (cf. Lemma 2.2.10) and $\tilde{d}(t_i)$ is defined as in equation (5.19) from $d : X + \tilde{Y} + Z \to \mathcal{D}_\omega P$ which assigns to every $x_i \in X$ and $z_k \in Z$ the Dirac distributions δ_{p_i} and δ_{q_k}, respectively, and to $\mu_j^{(o)} \in \tilde{Y}$ the distribution ν_j.*

The rules in Segala-GSOS are rather complex to describe, although for most applications simpler subformats are sufficient. For instance, we did not find a natural example yet that uses the multiplicities O_j. Since we present the format here mainly to illustrate the power of abstract GSOS, we define the format as expressive as we can make it. Still, we do not know whether we captured all freedom the abstract format allows. In the next section we explain the Segala-GSOS rules with their different premises by means of several examples.

5.3.2 Examples of Segala-GSOS specifications

We define a list of useful operators for Segala systems below. To illustrate their effect, we will occasionally refer to the processes p and q as drawn below. Here and in the following pictures we omit further transition from the successor states p_i and q_j.



(5.20)

(i) The constants $a \in L$ and 0 are specified as for PTS, namely by no rule for 0 and the following single rule for each constant $a \in L$:

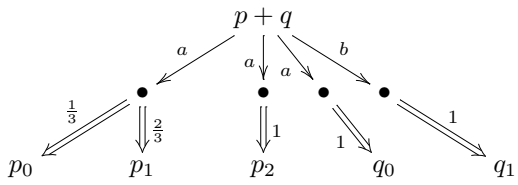
$$\frac{}{a \xrightarrow{a} 0}$$

Note that the definition of $\tilde{d}(t_V)$ in equation (5.19) yields the Dirac distribution δ_{t_V} if t_V does not contain variables. So, according to Definition 5.3.2, the target 0 of the above rule describes the Dirac distribution δ_0, which means that we obtain the following process:

(ii) A nondeterministic choice is specified by two rules for each $a \in L$:

$$\frac{x \xrightarrow{a} \mu}{x + y \xrightarrow{a} \mu} \qquad \frac{y \xrightarrow{a} \nu}{x + y \xrightarrow{a} \nu}$$

For p and q above we obtain the following process:



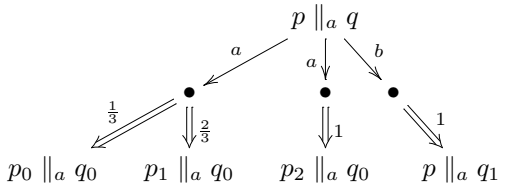
(iii) Our specification of a parallel composition of two processes is parameterised by a set $C \subseteq L$ of labels on which the two processes are bound to synchronise. It is given by the following rules for all $b \in L \setminus C$ and $c \in C$:

$$\frac{x \xrightarrow{b} \mu}{x \parallel_C y \xrightarrow{b} \mu \parallel_C y} \qquad \frac{y \xrightarrow{b} \nu}{x \parallel_C y \xrightarrow{b} x \parallel_C \nu}$$

$$\frac{x \xrightarrow{c} \mu \quad y \xrightarrow{c} \nu}{x \parallel_C y \xrightarrow{c} \mu \parallel_C \nu}$$

By Definition 5.3.2, every non-probabilistic variable occurring in a target of a rule – like y in $\mu \parallel_C y$ from the first rule above – is replaced by the corresponding Dirac probability. So $\mu \parallel_C y$ describes a distribution which assigns to the process $x' \parallel_C y$ the same probability as μ assigns to x'.

Below we draw the process that results if we put p and q in parallel such that they have to synchronise on a (we abbreviate $\parallel_{\{a\}}$ by $\parallel_a$).



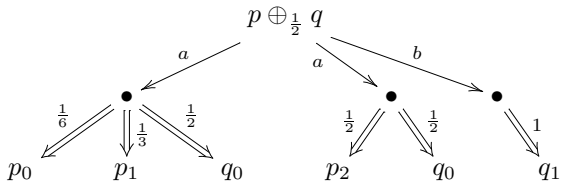
(iv) The specification of a probabilistic choice operator, parametric in a probability $r \in [0,1]$, can be lifted from PTS to Segala systems by giving the following rules for all $a \in L$:

$$\frac{x \xrightarrow{a} \mu \quad y \xrightarrow{a} \!\!\!\!\!\rightarrow}{x \oplus_r y \xrightarrow{a} \mu} \qquad \frac{x \xrightarrow{a} \!\!\!\!\!\rightarrow \quad y \xrightarrow{a} \nu}{x \oplus_r y \xrightarrow{a} \nu}$$

$$\frac{x \xrightarrow{a} \mu \quad y \xrightarrow{a} \nu}{x \oplus_r y \xrightarrow{a} r \cdot \mu + (1-r) \cdot \nu}$$

Different from the other rules shown here, the target of the last rule is a nontrivial convex combination of distributions. Note that in all other rules we exploited our convention to omit the weight 1 of the trivial convex combination.

Below we draw $p \oplus_{\frac{1}{2}} q$ for p and q from (5.20).



(v) Given a function $f : L \to L$, we can also define a relabelling operator r_f by the following rules for all $a \in L$:

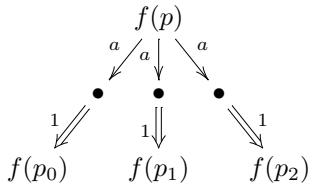
$$\frac{x \xrightarrow{a} \mu}{r_f(x) \xrightarrow{f(a)} r_f(\mu)}$$

(vi) To illustrate the use of the premises $\mu \Longrightarrow z$, we define an operation f that forgets the probabilistic information. It is given by the following rules for all $a \in L$:

$$\frac{x \xrightarrow{a} \mu \quad \mu \Longrightarrow z}{f(x) \xrightarrow{a} f(z)}$$

Note that in order to compute the resulting distributions for those transitions we again first replace the process substituted for z by the corresponding Dirac distribution.

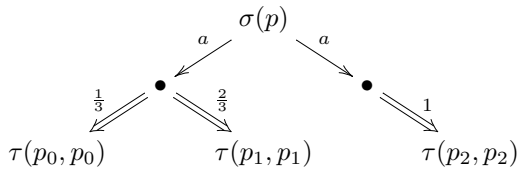
Applied to the p from (5.20), we obtain the process below.



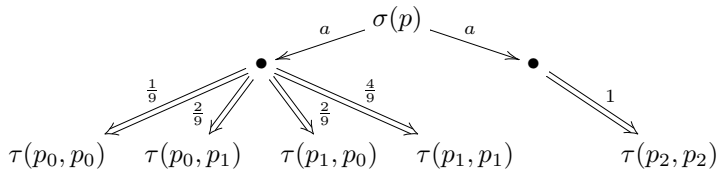
Since we did not yet encounter a natural example for a rule using more than one copy of a probability variable ν_j in a target term, we illustrate the idea with an artificial one. Consider the following three alternative rules (i), (ii), and (iii) for a unary operator symbol σ. They mention a binary operator τ whose semantics is to be defined by further rules not relevant for this example.

$$(i)\ \frac{x \xrightarrow{a} \mu}{\sigma(x) \xrightarrow{a} \tau(\mu,\mu)} \qquad (ii)\ \frac{x \xrightarrow{a} \mu}{\sigma(x) \xrightarrow{a} \tau(\mu^{(1)},\mu^{(2)})} \qquad (iii)\ \frac{x \xrightarrow{a} \mu \quad x \xrightarrow{a} \nu}{\sigma(x) \xrightarrow{a} \tau(\mu,\nu)}$$

With rule (i) we obtain the following process with p as in (5.20).

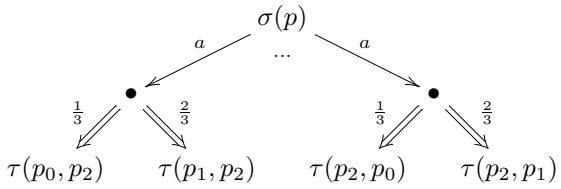


Rule (ii) generates the transitions pictured below instead.



With rule (iii), the process $\sigma(p)$ can do the transitions as with rule (ii) plus the following two. They can be derived from the rule when μ and ν are instantiated

with different transitions of p, which was not possible with rule (ii).



5.3.3 Properties of Segala-GSOS

The well-behavedness results we have for Segala-GSOS are based on the following statement.

Theorem 5.3.3 *Every specification $\mathcal{R}$ is Segala-GSOS gives rise to a specification in abstract GSOS, i.e. a natural transformation*

$$\rho^{\mathcal{R}} : \mathrm{S}(\mathrm{Id} \times (\mathcal{P}_\omega \mathcal{D}_\omega)^L) \Rightarrow (\mathcal{P}_\omega \mathcal{D}_\omega \mathrm{T})^L,$$

where S *is the functor corresponding to the signature Σ under consideration and* T *is the term functor (cf. equation (2.2) on page 20 and Lemma 2.2.10).*

The triple $\langle P, [\![.]\!], \alpha \rangle$ is a model of $\mathcal{R}$ according to Def. 5.3.2 if and only if it is a model of $\rho^{\mathcal{R}}$ according to Def. 3.5.3.

The proof of the theorem is straightforward. A central observation is that for a fixed $t_V \in \mathrm{T}V$ for some finite set V the definition of $d(t_V)$ from equation (5.19) gives rise to a natural transformation $\phi : (\mathcal{D}_\omega)^V \Rightarrow \mathcal{D}_\omega \mathrm{T}$.

With Theorem 5.3.3, the properties of abstract GSOS (cf. Corollaries 3.5.4, 3.5.5, 4.4.7, and 4.4.9) specialise to properties of Segala-GSOS:

Corollary 5.3.4 *Let $\mathcal{R}$ be a specification in Segala-GSOS of the operators in a signature Σ.*

(i) *Let $[\![.]\!] : \mathrm{ST}\emptyset \to \mathrm{T}\emptyset$ be the term construction on the set of closed terms $\mathrm{T}\emptyset$ of the signature Σ. There exists a unique structure α of a Segala system on $\mathrm{T}\emptyset$ such that $\langle \mathrm{T}\emptyset, [\![.]\!], \alpha \rangle$ is a model or $\mathcal{R}$. This model is initial.*

(ii) *Dually, the final Segala system $\langle \mathcal{F}, \omega \rangle$, which exists by Lemma 5.1.8, can be equipped uniquely with an operator interpretation $[\![.]\!] : \mathrm{S}\mathcal{F} \to \mathcal{F}$ such that $\langle \mathcal{F}, [\![.]\!], \omega \rangle$ is a model of $\mathcal{R}$. This is a final model.*

(iii) *Bisimilarity between two models of $\mathcal{R}$ is a congruence for the corresponding operator interpretations.*

(iv) *Every bisimulation up-to-context (cf. Def. 4.4.8) between two models of $\mathcal{R}$ is contained in some standard bisimulation between the two Segala systems.*

(v) *Every guarded recursive definition (cf. Definition 4.2.3 and Section 4.4.4) has a solution in a model of $\mathcal{R}$. Such a solution is determined up to bisimilarity.*

5.4 Concluding remarks and related work

We have presented well-behaved, expressive specification formats for two out of a large number of different probabilistic system types. We emphasise that our categorical approach allows to adapt the results to other variants easily, due to the following three properties: first, with the coalgebraic means we can describe different types of probabilistic systems uniformly as coalgebras of behaviour functors built from similar ingredients (probabilistic systems were modelled coalgebraically by De Vink and Rutten [VR99] and Moss [Mos99], a coalgebraic comparison of different probabilistic systems was made by Bartels, Sokolova, and De Vink [BSV03, BSV04]); second, once a suitable behaviour functor is found, Turi and Plotkin's bialgebraic approach to SOS specifications [TP97] immediately yields a well-behaved specification format phrased in terms of natural transformations; third, our decompositional technique to analyse natural transformations allows us to derive concrete rule-style representations of the categorical specification format in many cases.

To give an example, recall that we considered PTS as the system type that arises when we add probabilities to image finite LTSs as in Def. 2.3.12. If we had, instead, started with finitely branching LTSs, i.e. coalgebras of the functor $\mathcal{P}_\omega(\underline{L} \times \mathrm{Id})$, then we had obtained probabilistic systems as pairs

$$\langle P, \alpha : P \to \mathcal{D}_\omega(L \times P) + 1 \rangle.$$

Such a coalgebra is of a rather different nature, since the labels are involved in the probabilistic choice. So they are a part of the probabilistic output, which is why these systems are called *generative systems* in the literature [GSS95]. With the lemmata from the appendix one can easily turn abstract GSOS for generative systems into a concrete rule-style format as well.

The only attempt towards a more general theory of well-behaved operator specifications for probabilistic systems that we are aware of is made by Jonssen, Larsen, and Yi [JLY01]. They explain how to interpret nondeterministic operator specifications in the De Simone format [Sim85] as specifications for (a variant of) simple Segala systems from Def. 5.1.6. But their proposal is not an intrinsically probabilistic format, because it does not allow to specify genuine probabilistic operators except for a built-in probabilistic choice.

Apart from this, several authors specify concrete operators for different types of probabilistic systems and give elementary well-behavedness proofs. Among them are Larsen and Skou [LS92] and Van Glabbeek, Smolka, and Steffen [GSS95], who work with PTS as in Def. 5.1.2. With the exception of the recursion operator treated by Van Glabbeek et al., their congruence results are

subsumed by our general result, since the specifications fit in our PGSOS format.

We find two different approaches to define probabilistic operators using transition rules in the literature:

In one approach, one writes rules as for nondeterministic systems, i.e. without mentioning probabilities, in a first step. In a second step, the probabilities of the declared transitions are defined separately. Specifications of this type are used by Andova [And99, And02], who works with the alternating systems of Hansson [Han94].[2] With our formats, such a two-step definition is not necessary.

In the other approach, the probabilities are defined by the transition rules as well, as we do it. But then, in order to separate transitions generated by different rules, since they may go from one state with the same label to the same successor state, the transitions are tagged with an additional index. Specifications of this type are given, e.g., by Van Glabbeek, Smolka, and Steffen [GSS95], who work with different types of systems including PTS, and by D'Argenio, Hermanns, and Katoen [DHK99], who use bundle systems as defined in the same article[3]. We did not introduce these additional indices. Instead we argued that when several transitions to the same successor are generated they are to be viewed as one transition with the sum of the probabilities. Although this complicates the description of the format, we still prefer this solution because it does not force us to include artefacts in the description of the model, which are not part of the intended system behaviour (correspondingly, the definition of bisimilarity in the mentioned papers ignores the indices).

Finally we want to mention further attempts to apply Turi and Plotkin's bialgebraic modelling of SOS specifications [TP97] to concrete system types. The articles do not consider probabilistic systems though.

Immediately after the publication of their approach, Turi provided a few case studies [Tur97] for the abstract GSOS format. He did not study formats for one particular type of system, but illustrated the generality of the approach by showing how the same specification can be interpreted for different types of systems living even in different categories.

Kick [Kic02a, Kic02b, Kic03] also uses abstract GSOS to obtain concrete specification formats: he instantiates the framework for *timed systems*. One interesting aspect of his study is the use of a comonad which is not cofreely generated by a behaviour functor. This comonad arises when he models systems for continuous time.

[2] The behaviour of these systems is similar to that of Segala systems except that between the nondeterministic and probabilistic choice the systems move to an intermediate probabilistic state. As it were, these states correspond to the bullets we drew in our pictures of Segala systems.

[3] These systems are again similar to Segala systems, but the order of probabilistic and nondeterministic choices is reversed.

Chapter 6

Future work

We conclude by listing some open questions and directions for future work related to the subjects studied in this thesis.

Bialgebraic approach to SOS

We start with the categorical modelling of well-behaved operator specification formats. From that framework we have mainly focused on the abstract GSOS format [TP97], and so have most authors who contributed to the area (see e.g. [Tur97, LPW00, Wat02, Pow03, Kli04b]). The categorical approach, however, provides other formats as well. In particular, there is a categorical dual of abstract GSOS, which captures formats that allow *look-ahead*, i.e. the premises of the transition rules may refer to several successive transitions of the arguments instead of just the immediate outgoing transitions. Similar to our analysis of abstract GSOS, one could try and derive rule formats for various kinds of systems from this dual abstract format. We assume that it has not received much attention yet because the cofree comonad of a behaviour functor, on which the format is based, is much more difficult to work with in the case of LTSs than the free monad generated by a signature. As an indication for this, note that the *(safe) tree rules* [Fok94, WFG96], which Turi and Plotkin [TP97] presented in this context and which are already more difficult to describe than GSOS rules, do not have the full power of the abstract format. But the same is not true for other system types: for streams and deterministic automata, for instance, the cofree comonad can be described rather easily, and we expect that corresponding formats can be given.

The most general categorical format, namely that of distributive laws of the free monad generated by the signature over the cofree comonad generated by the behaviour functor, can justify specifications that go beyond both abstract GSOS and its dual. The interested reader may check that Rutten's specification of the Δ-operator [Rut00a, page22] on infinite streams of real numbers is an

example: it neither fits in the format corresponding to abstract GSOS nor its dual, but it gives rise to a distributive law of a term monad over a cofree comonad generated by the behaviour functor for streams. So the categorical approach has the potential to justify specification formats that go beyond the ones that were studied in this context so far. We view their identification as an interesting direction for future work.

The abstract GSOS format is based on the free monad generated by a signature. More generally, we obtain monads from signatures with equations. With these equations one could lay down desired properties of the operators under consideration in the beginning: one could for instance postulate that a choice operator should be commutative and associative. Such an approach is taken by Milner [Mil99], who defines the transitions of the terms in the π-calculus with respect to "structural congruence". Studying distributive laws of the monads arising this way over behaviour functors one may get rule formats respecting such equations. Such a format would have to guarantee in some way that the transitions derivable for different terms correspond to each other if the terms are to be identified according to the equations. In loc. cit., Milner for instance guarantees this by adding a rule stating that when one term can do a transition, every congruent term can do a corresponding transition, but such a rule is troublesome in general.

Another suggestion for an extension of the categorical theory of well-behaved operator specification formats concerns the notion of behavioural equivalence. In this thesis we have worked with bisimilarity. Although the operators and the behaviour, modelled as algebras and coalgebras, form dual concepts in this theory, the notions of a congruence and a bisimulation are not dual. This fact breaks the typical symmetry that we find in this context elsewhere. In a recent project [BSV03] we have convinced ourselves that another notion of behavioural equivalence is actually better suited as a general coalgebraic notion. It is based on *cocongruences* [Kur00], which are strictly dual to congruences. We found that the well-behavedness of the categorical formats with respect to this notion of behavioural equivalence can be proved as well, giving rise again to symmetric statements for the algebra and coalgebra part. We leave a presentation of this result to future work.

Generalised coinduction

One subject we did not study in our work on generalised coinduction schemata is the combination of different extensions. In Section 4.4.4 for instance we treated an example that actually needed a combination of the bisimulation up-to-context and the bisimulation up-to-equality technique. We expect that such combinations can be justified by an abstract framework to be found. For the special case of LTSs and strong bisimulation, Sangiorgi [San98] develops means to combine various generalised principles. We would like to reveal the categorical machinery underlying his approach to express combined principles and to prove

them valid on our level of abstraction.

Sangiorgi [San98] also includes a bisimulation up-to-bisimilarity [Mil89] principle, which allows to use previously proved bisimilarity results. It is not obvious how this principle can be captured by the λ-coinduction framework and we are not sure whether it can be done at all. The problem is that a formulation of this principle seems to require information about the particular coalgebras under consideration, which is not available at the level of abstraction of our approach.

As mentioned earlier, Pardo, Uustalu, and Vene [UVP01] have proposed the categorical dual of λ-coiteration independent of our work. Two aspects they do not consider are proof principles and the dual of our results on λ-coinduction for specifications in abstract GSOS (cf. Section 4.4.4). The latter would again involve the dual of the abstract GSOS format discussed above. This instance may lead to new inductive definition principles.

Formats for probabilistic systems

In our study of operator specification formats for probabilistic systems we feel that we understand the situation for the simple PTS [LS91] rather well. But our statements are not yet powerful enough to completely analyse the more complex system types, such as the probabilistic automata of Segala [Seg95b, Sto02b]. To this end we mainly need to extend our tools to characterise natural transformations involving the powerset and the distribution functor (cf. Theorems A.2.4 and A.3.5). We have added Section A.4 to give a few more details about how the statements could be generalised.

Coming back to an earlier point, we expect that for PTSs it possible to derive a format corresponding to the dual of the abstract GSOS format. This was difficult for LTSs mainly because the usual negative premises in the rules are not powerful enough to postulate the absence of transitions to states with a certain behaviour. With probabilities the problem can be remedied as follows: to express that a transition to a state with a certain behaviour is not possible, we can alternatively make sure that the probability to move to other states is 1. Note that here we recover a phenomenon which makes probabilistic systems better behaved in other aspects as well: probabilistic bisimilarity can be characterised by a surprisingly weak, namely negation free, logic [DEP02].

Appendix A

Representation results for natural transformations

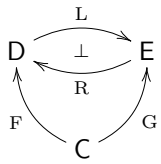
In this appendix we state and prove several statements that we use to analyse natural transformations. In the first section we give a collection of simple statements that can be used to decompose natural transformations of a complex type into possibly several ones of a simpler type. Later we develop direct representations for natural transformations of particular types. In the second and third section these natural transformations are between functors built from the powerset and distribution functor respectively.

The lemmata are of a rather technical nature, and we deferred them to the appendix in order not to disturb the flow of reading too much in the main chapters. Note that this does not mean that the material is folklore or straightforward. On the contrary, we consider in particular the statements in Sections A.2 and A.3 as one of the main contributions of this thesis.

A.1 Some structural lemmata

The first statement helps in dealing with natural transformations with constant exponents as codomain along the adjunction $\mathrm{Id} \times L \dashv \mathrm{Id}^L$.

Lemma A.1.1 *Consider categories and functors as pictured below, where* L *is left adjoint to* R*:*



There is a one-to-one correspondence between natural transformations

$$\nu : \mathrm{F} \Rightarrow \mathrm{RG} \quad \textit{and} \quad \xi : \mathrm{LF} \Rightarrow \mathrm{G}$$

given by $\nu \mapsto \varepsilon\mathrm{G} \circ \mathrm{L}\nu$ *and* $\xi \mapsto \mathrm{R}\xi \circ \eta\mathrm{F}$*, where* $\eta : \mathrm{Id} \Rightarrow \mathrm{RL}$ *and* $\varepsilon : \mathrm{LR} \Rightarrow \mathrm{Id}$ *are the unit and counit of the adjunction.*

Proof: To show that the two constructions are inverses of each other, we calculate, using (i) naturality of η and (ii) the adjunction law $\mathrm{R}\varepsilon \circ \eta\mathrm{R} = \mathtt{id}$,

$$\begin{aligned} \mathrm{R}(\varepsilon\mathrm{G} \circ \mathrm{L}\nu) \circ \eta\mathrm{F} &= \mathrm{R}\varepsilon\mathrm{G} \circ \mathrm{RL}\nu \circ \eta\mathrm{F} \\ &\overset{(i)}{=} \mathrm{R}\varepsilon\mathrm{G} \circ \eta\mathrm{RG} \circ \nu \\ &= (\mathrm{R}\varepsilon \circ \eta\mathrm{R})\mathrm{G} \circ \nu \\ &\overset{(ii)}{=} \nu \end{aligned}$$

and similarly, using (i) naturality of ε and (ii) the adjunction law $\varepsilon\mathrm{L} \circ \mathrm{L}\eta = \mathtt{id}$

$$\begin{aligned} \varepsilon\mathrm{G} \circ \mathrm{L}(\mathrm{R}\xi \circ \eta\mathrm{F}) &= \varepsilon\mathrm{G} \circ \mathrm{LR}\xi \circ \mathrm{L}\eta\mathrm{F} \\ &\overset{(i)}{=} \xi \circ \varepsilon\mathrm{LF} \circ \mathrm{L}\eta\mathrm{F} \\ &= \xi \circ (\varepsilon\mathrm{L} \circ \mathrm{L}\eta)\mathrm{F} \\ &\overset{(ii)}{=} \xi. \end{aligned}$$

□

If the domain or codomain of a natural transformation is given by a coproduct or product of functors respectively, then we can decompose it into a collection of natural transformations with simpler type, as the following simple lemma shows. It is actually a special case of the fact that point-wise (co)limits of any type in D yield (co)limits of that type in D^{C}:

Lemma A.1.2 *Let* $\mathrm{F}^i, \mathrm{G} : \mathsf{C} \to \mathsf{D}$ *for* $i \in I$ *be functors.*

(i) *Let the category* D *have* I*-indexed coproducts. There is a one-to-one correspondence between natural transformations* $\nu : \coprod_{i \in I} \mathrm{F}^i \Rightarrow \mathrm{G}$ *and families of natural transformations* $(\nu^i : \mathrm{F}^i \Rightarrow \mathrm{G})_{i \in I}$*.*

(ii) *Dually, let the category* D *have* I*-indexed products. There is a one-to-one correspondence between natural transformations* $\nu : \mathrm{G} \Rightarrow \prod_{i \in I} \mathrm{F}^i$ *and families of natural transformations* $(\nu^i : \mathrm{G} \Rightarrow \mathrm{F}^i)_{i \in I}$*.*

The next lemma states that under certain conditions we can also easily deal with codomains given as a coproduct.

Lemma A.1.3 *Let* C *be a category with a final object* 1_{C} *and let* $\mathrm{F}, \mathrm{G}^i : \mathsf{C} \to \mathsf{Set}$ *(*$i \in I$*) be functors such that* F *preserves finality, i.e.* $\mathrm{F}1_{\mathsf{C}} \simeq 1$ *for the terminal*

object $1 = \{*\}$ *in* Set. *Every natural transformation*

$$\nu : \mathrm{F} \Rightarrow \coprod_{i \in I} \mathrm{G}^i$$

factors as $\nu = \iota_j \circ \nu^j$ *for some* $j \in I$ *and natural transformation* $\nu^j : \mathrm{F} \Rightarrow \mathrm{G}^j$, *where* $\iota_j : \mathrm{G}^j \Rightarrow \coprod_{i \in I} \mathrm{G}^i$ *is the coproduct injection.*

Proof: Let $j \in I$ be such that $\nu_{1_\mathsf{C}}(\phi) = \iota_j(\psi)$ for some $\psi \in \mathrm{G}^j 1_\mathsf{C}$, where ϕ is the unique element of $\mathrm{F}1_\mathsf{C}$. It suffices to show that for all sets X and $\phi_X \in \mathrm{F}X$ we have that $\nu_X(\phi_X) = \iota_j(\psi_X)$ for some $\psi_X \in \mathrm{G}^j X$. This is equivalent to saying that $(\coprod_{i \in I} \mathrm{G}^i !)\nu_X(\phi_X) = \iota_j(\psi')$ for some $\psi' \in \mathrm{G}^j 1_\mathsf{C}$, where $! : X \to 1_\mathsf{C}$ is the unique map given by finality of 1_C. But this is the case, since by naturality of ν we have

$$(\coprod_{i \in I} \mathrm{G}^i !)(\nu_X(\phi_X)) = \nu_{1_\mathsf{C}}((\mathrm{F}!)(\phi_X)) = \nu_{1_\mathsf{C}}(\phi) = \iota_j(\psi).$$

$$\begin{array}{ccc} \mathrm{F}X & \xrightarrow{\nu_X} & \prod_{i \in I} \mathrm{G}^i X \\ {\scriptstyle \mathrm{F}!}\downarrow & \text{nat. } \nu & \downarrow{\scriptstyle (\prod_{i \in I} \mathrm{G}^i f)(!)} \\ \mathrm{F}1_\mathsf{C} = \{\phi\} & \xrightarrow[\nu_{1_\mathsf{C}}]{} & \prod_{i \in I} \mathrm{G}^i 1_\mathsf{C} \end{array} \qquad \begin{array}{ccc} \phi_X & \xmapsto{\nu_X} & \iota_j(\psi_X) \\ {\scriptstyle \mathrm{F}!}\downarrow & & \downarrow{\scriptstyle (\prod_{i \in I} \mathrm{G}^i f)(!)} \\ \phi & \xmapsto[\nu_{1_\mathsf{C}}]{} & \iota_j(\psi) \end{array}$$

□

In the view of Lemma A.1.2 (i) and Lemma A.1.3 we are interested in writing the functors from the domain and codomain of the natural transformation under consideration as coproducts. The following lemma states that for functors into Set there is a finest such splitting.

Lemma A.1.4 *Let* C *be a category with a final object* 1_C. *Every functor* $\mathrm{F} : \mathsf{C} \to \mathsf{Set}$ *can be written as*

$$\mathrm{F} \simeq \coprod_{z \in \mathrm{F}1_\mathsf{C}} \mathrm{F}|_z$$

where for $z \in \mathrm{F}1_\mathsf{C}$ *we define*

$$\mathrm{F}|_z X := (\mathrm{F}!)^{-1}(z) = \{\phi \in \mathrm{F}X \mid (\mathrm{F}!_X)(\phi) = z\}$$

for a C*-object* X, *where* $! : X \to 1_\mathsf{C}$ *is the unique arrow into the final object, and* $\mathrm{F}|_z f : \mathrm{F}|_z X \to \mathrm{F}|_z Y$ *for an arrow* $f : X \to Y$ *is the restriction of* $\mathrm{F}f : \mathrm{F}X \to \mathrm{F}Y$ *to* $\mathrm{F}|_z X$.

Proof: Obviously we have $\mathrm{F}X \simeq \coprod_{z \in \mathrm{F}1_\mathsf{C}} \mathrm{F}|_z X$ for any set X. We need to show that this splitting is respected by the action of F on arrows, which is to say that for any $f : X \to Y$ and $x \in \mathrm{F}|_z X$ we get $(\mathrm{F}f)(x) \in \mathrm{F}|_z Y$. This follows from finality, as the computation below shows. We write $!_X$ and $!_Y$ for the unique arrows from X and Y respectively into 1_C.

$$\mathrm{F}!_Y((\mathrm{F}f)(x)) = (\mathrm{F}(!_Y \circ f))(x) = (\mathrm{F}!_X)(x) = z.$$

□

The following fact helps us to write functors as coproducts as well. It applies in the special case of the (finite) powerset functor – or rather the nonempty version $\mathcal{P}^+_\omega$, since $\mathcal{P}_\omega \simeq 1 + \mathcal{P}^+_\omega$.

Lemma A.1.5 *For functors* $\mathrm{G}^i : \mathsf{C} \to \mathsf{Set}$ *(*$i \in I$*) we have*

$$\mathcal{P}^+_\omega(\coprod_{i\in I} \mathrm{G}^i) \simeq \coprod_{M\in\mathcal{P}^+_\omega I} (\prod_{i\in M} \mathcal{P}^+_\omega \mathrm{G}^i).$$

Proof: For all sets X we have an equivalence of sets

$$\mathcal{P}^+_\omega(\coprod_{i\in I} \mathrm{G}^i X) \simeq \coprod_{M\in\mathcal{P}^+_\omega I} (\prod_{i\in M} \mathcal{P}^+_\omega \mathrm{G}^i X)$$

given from left to right by $X' \mapsto \iota_M((X'_i)_{i\in M})$ where

$$M := \{i \in I \mid X' \cap \iota_i[\mathrm{G}^i X] \neq \emptyset\} \quad \text{and} \quad X'_i = \{\alpha \in \mathrm{G}^i X \mid \iota_i(\alpha) \in X'\}.$$

The equivalence easily extends to one between functors.

□

We continue with a few simple statements about natural transformations involving constant and identity functors.

Lemma A.1.6 *Let* $\mathrm{F} : \mathsf{C} \to \mathsf{D}$ *be a functor and let* A *be an object of* D*.*

(i) *If* C *has an initial object* 0*, then natural transformations* $\eta : \underline{A} \Rightarrow \mathrm{F}$ *are in one-to-one correspondence with arrows* $h : A \to \mathrm{F}0$*.*

(ii) *Dually, if* C *has an final object* 1*, then natural transformations* $\eta : \mathrm{F} \Rightarrow \underline{A}$ *are in one-to-one correspondence with arrows* $h : \mathrm{F}1 \to A$*.*

Note that as a special case of the first item we get that for a Set-functor F natural transformations $\eta : 1 \Rightarrow \mathrm{F}$ are given by the elements of $\mathrm{F}\emptyset$.

Proof: The bijection for item (i) is given by $\eta \mapsto \eta_0$ (where $\eta_0 : A \to \mathrm{F}0$ as always denotes the component of η at the initial object 0) and $h \mapsto \eta^h$ where $\eta^h_X = \mathrm{F}! \circ h$ and $! : 0 \to X$ is the unique initial arrow. It is straightforward to check that the second mapping defines a natural transformation indeed and that both constructions are inverses of each other. The proof of item (ii) is dual.

□

Lemma A.1.7 *Let* $\mathrm{F}, \mathrm{G} : \mathsf{Set} \to \mathsf{Set}$ *be functors and let* A *be a set. There is a one-to-one correspondence between natural transformations*

$$\nu : (\mathrm{Id})^A \times \mathrm{F} \Rightarrow \mathrm{G} \quad \textit{and} \quad \xi : \mathrm{F} \Rightarrow \mathrm{G}(\underline{\mathrm{A}} + \mathrm{Id})$$

given by $\nu \mapsto \xi^\nu$ *and* $\xi \mapsto \nu^\xi$ *defined for any set* X, $\alpha \in \mathrm{F}X$, *and* $f : A \to X$ *as*

$$\xi^\nu_X(\alpha) := \nu_{A+X}(\iota_1, (\mathrm{F}\iota_2)(\alpha)) \quad \textit{and} \quad \nu^\xi_X(f, \alpha) := (\mathrm{G}[f, \mathtt{id}_X] \circ \xi_X)(\alpha).$$

Proof: It is easy to check that the two constructions define natural transformations. Moreover, they are each others inverses, as the calculations below for all sets X, $\alpha \in \mathrm{F}X$, and $f : A \to X$ show. Using the naturality of ξ in the step marked by $(*)$,we have

$$\begin{array}{rcl}
\xi^{\nu^\xi}_X(\alpha) & = & \nu^\xi_{A+X}(\iota_1, (\mathrm{F}\iota_2)(\alpha)) \\
& = & (\mathrm{G}[\iota_1, \mathtt{id}_{A+X}] \circ \xi_{A+X} \circ \mathrm{F}\iota_2)(\alpha) \\
& \overset{(*)}{=} & (\mathrm{G}[\iota_1, \mathtt{id}_{A+X}] \circ \mathrm{G}(\mathtt{id}_A + \iota_2) \circ \xi_X)(\alpha) \\
& = & (\mathrm{G}\underbrace{[\iota_1, \iota_2]}_{=\mathtt{id}_{A+X}} \circ \xi_X)(\alpha) \\
& = & \xi_X(\alpha).
\end{array}$$

Using the naturality of ν in the step marked by $(*)$ below, we obtain the second identity.

$$\begin{array}{rcl}
\nu^{\xi^\nu}_X(f, \alpha) & = & (\mathrm{G}[f, \mathtt{id}_X] \circ \xi^\nu_X)(\alpha) \\
& = & (\mathrm{G}[f, \mathtt{id}_X] \circ \nu_{A+X})(\iota_1, (\mathrm{F}\iota_2)(\alpha)) \\
& \overset{(*)}{=} & (\nu_X \circ ([f, \mathtt{id}_X]^A \times \mathrm{F}[f, \mathtt{id}_X]))(\iota_1, (\mathrm{F}\iota_2)(\alpha)) \\
& = & \nu_X(\underbrace{([f, \mathtt{id}_X]^A)(\iota_1)}_{=[f,\mathtt{id}_X]\circ\iota_1 = f}, (\mathrm{F}(\underbrace{[f, \mathtt{id}_X] \circ \iota_2}_{=\mathtt{id}_X}))(\alpha)) \\
& = & \nu_X(f, \alpha).
\end{array}$$

□

Lemma A.1.8 *Let* C *and* D *be categories with* I*-indexed coproducts and products respectively and let* $\mathrm{F}^i, \mathrm{G} : \mathsf{C} \to \mathsf{D}$ *for* $i \in I$ *be functors. There is a one-to-one correspondence between natural transformations of the type*

$$\nu : \prod_{i \in I} \mathrm{F}^i \Rightarrow \mathrm{G} \quad \textit{and} \quad \xi_{(X_i)_{i\in I}} : \prod_{i \in I} \mathrm{F}^i X_i \Rightarrow \mathrm{G}(\coprod_{i \in I} X_i).$$

(Note that ξ *is a natural transformation between functors from* C^I *to* D*.) The correspondence is given by* $\nu \mapsto \nu\Lambda \circ \prod_{i\in I} \mathrm{F}^i\iota_i$ *and* $\xi \mapsto \mathrm{G}[\mathrm{Id}]_{i\in I} \circ \xi\Delta$*, where* $\Delta : \mathsf{C} \to \mathsf{C}^I$ *is the diagonal functor mapping* X *to* $(X)_{i\in I}$ *and* $\Lambda : \mathsf{C}^I \to \mathsf{C}$ *is its left adjoint, i.e. the functor mapping the tuple* $(X_i)_{i\in I}$ *to the coproduct* $\coprod_{i\in I} X_i$*.*

More precisely, we should have written the natural transformation ξ as

$$\xi : \prod_{i \in I} \mathrm{F}^i\pi_i \Rightarrow \mathrm{G}(\coprod_{i \in I} \pi_{\mathrm{i}}),$$

where $\pi_i : \mathsf{C}^I \to \mathsf{C}$ for $i \in I$ is the projection functor mapping $(X_j)_{j\in I}$ to X_i. We prefer the above notation since we consider it more readable.

Proof: The statement follows from the dual of Lemma A.1.1 when instantiated with Δ and its left adjoint, which exists by the assumption that C has I-indexed coproducts.

□

A.2 The powerset functor

In Chapter 3 we used distributive laws of a signature functor over the behaviour functor as an abstract specification format. In order to obtain practical formats from instances of this framework, we need to characterise the corresponding natural transformations in more concrete terms. In the case of labelled transition systems the natural transformations are between two functors constructed from the finite powerset functor. After some structural simplification using the lemmata from the previous section, we are left with natural transformations of a type similar to

$$\zeta : \mathcal{P}_\omega^+ \Rightarrow \mathcal{P}_\omega^+(\mathrm{Id}^m) \tag{A.1}$$

for some natural number m. In this section, we will show that any such transformation can be described by a set of derivation rules of a certain shape.

To motivate the development we first look at natural transformations of this type for small m.

$m = 0$**:** We have $\mathcal{P}_\omega^+(\mathrm{Id}^m) \simeq \underline{1}$, so the only candidates for the components of ζ are the unique functions into the final object, and this definition is indeed natural.

$m = 1$**:** It is not so difficult to see that the only natural transformation $\zeta : \mathcal{P}_\omega^+ \Rightarrow \mathcal{P}_\omega^+$ is the identity on each component.

$m = 2$**:** This case is more interesting. The following two transformations $\zeta^1, \zeta^2 : \mathcal{P}_\omega^+ \Rightarrow \mathcal{P}_\omega^+(\mathrm{Id} \times \mathrm{Id})$ can easily be seen to be natural:

$$\begin{aligned} \zeta^1_X(X') &:= \{\langle x, y\rangle \mid x, y \in X'\}, \\ \zeta^2_X(X') &:= \{\langle x, x\rangle \mid x \in X'\}. \end{aligned}$$

The result we are about to develop will show that the two transformations in the last case are the only possibilities. It will turn out that, for any m, the natural transformations ζ can be written as pointwise unions of certain basic ones which are defined in a way similar to the definition of ζ^1 and ζ^2: Each basic transformation β^Γ is defined by an equivalence relation Γ on the set of the m positions of the tuple (we call it a *partition* here) where β^Γ_X maps a set

$X' \subseteq X$ to the set of all m-tuples in $(X')^m$ which have identical elements in all positions belonging to the same equivalence class.

Definition A.2.1 *Let $m \in \mathbb{N}$.*

- *By $\mathtt{Par}[m]$ we denote the set of all* **partitions of** $\{1, \ldots, m\}$*, i.e. all sets Γ of nonempty, disjoint subsets of $\{1, \ldots, m\}$ such that $\bigcup \Gamma = \{1, \ldots, m\}$.*
- *For $\Gamma \in \mathtt{Par}[m]$ and $1 \leq i \leq m$ we denote by $[i]_\Gamma$ the* **equivalence class of i in Γ***, which is the unique $c \in \Gamma$ such that $i \in c$.*
- *We write $\sim_\Gamma$ for the equivalence relation on $\{1, \ldots, m\}$ induced by the partition $\Gamma \in \mathtt{Par}[m]$, i.e. $i \sim_\Gamma j$ just in case $[i]_\Gamma = [j]_\Gamma$. (Since partitions and equivalence relations are in one-to-one correspondence, we can define one in terms of the other, as we will do below.)*
- *There is an order of partitions defined for $\Gamma, \Gamma' \in \mathtt{Par}[m]$ as $\Gamma \preceq \Gamma'$ if and only if $\sim_\Gamma \subseteq \sim_{\Gamma'}$, which means that for all $1 \leq i, j \leq m$ we have that $i \sim_\Gamma j$ implies $i \sim_{\Gamma'} j$. We write $\Gamma \prec \Gamma'$ if $\Gamma \preceq \Gamma'$ and $\Gamma \neq \Gamma'$.*
- *Given a vector $\vec{x} = \langle x_1, \ldots, x_m \rangle \in X^m$ we define the* **partition** $\mathtt{par}(\vec{x}) \in \mathtt{Par}[m]$ **induced by** *$\vec{x}$ to satisfy $i \sim_{\mathtt{par}(\vec{x})} j$ just in case $x_i = x_j$.*
- *For $\Gamma \in \mathtt{Par}[m]$ and $c \in \Gamma$ we write $c{\downarrow} \in \{1, \ldots, m\}$ for an arbitrary element in c. This notation will be used in cases only where no ambiguity arises. As an example, for $\vec{x} \in X^m$, $\Gamma \in \mathtt{Par}[m]$ with $\Gamma \preceq \mathtt{par}(\vec{x})$, and $c \in \Gamma$ we may write $x_{c\downarrow}$. This is unambiguous because for $i, j \in \{1, \ldots, m\}$ we have*

$$i, j \in c \;\Rightarrow\; i \sim_\Gamma j \;\Rightarrow\; i \sim_{\mathtt{par}(\vec{x})} j \;\Rightarrow\; x_i = x_j.$$

Definition A.2.2 *For $m \in \mathbb{N}$ and $\Gamma \in \mathtt{Par}[m]$ define the* **basic transformation**

$$\beta^\Gamma : \mathcal{P}^+_\omega \Rightarrow \mathcal{P}^+_\omega(\mathrm{Id}^m)$$

for a set X, subset $X' \in \mathcal{P}^+_\omega X$, and $\vec{x} = \langle x_1, \ldots, x_m \rangle \in X^m$ as

$$\vec{x} \in \beta^\Gamma_X(X') \iff \Gamma \preceq \mathtt{par}(\vec{x}) \;\wedge\; \forall i \in \{1, \ldots, m\} :\; x_i \in X'.$$

The tuples $\vec{x}$ in $\beta^\Gamma_X(X')$ can be viewed to arise from the following procedure. With $\Gamma = \{c_1, \ldots, c_k\}$ we draw an element y_j from X' for each $1 \leq j \leq k$ and put it in all positions $i \in c_j$ of $\vec{x}$. For all $1 \leq i \leq m$ let $o_i \in \{1, \ldots, k\}$ denote the unique index such that $i \in c_{o_i}$. Then the definition of β^Γ can equivalently be expressed as follows: let $\beta^\Gamma_X(X')$ for a set X and subset $X' \in \mathcal{P}^+_\omega X$ contain precisely the tuples arising from instantiations of the rule below.

$$\frac{y_j \in X' \; (1 \leq j \leq k)}{\langle y_{o_1}, \ldots, y_{o_m} \rangle \in \beta^\Gamma_X(X')} \tag{A.2}$$

We will later use such rules as a more intuitive notation. Moreover, the rule representation allows us to show easily that the basic transformations are natural indeed.

Lemma A.2.3 *For each* $\Gamma \in \mathtt{Par}[m]$ *the transformation* β^Γ *from Def. A.2.2 is natural.*

Proof: Take any function $f : X \to Y$. We have to show that the diagram below commutes.

$$\begin{array}{ccc} \mathcal{P}^+_\omega X & \xrightarrow{\beta^\Gamma_X} & \mathcal{P}^+_\omega(X^m) \\ {\scriptstyle \mathcal{P}^+_\omega f}\downarrow & & \downarrow{\scriptstyle \mathcal{P}^+_\omega(f^m)} \\ \mathcal{P}^+_\omega Y & \xrightarrow[\beta^\Gamma_Y]{} & \mathcal{P}^+_\omega(Y^m) \end{array}$$

Let β^Γ be described by a rule as in (A.2). To show that the naturality square above commutes, take any $X' \in \mathcal{P}^+_\omega X$. We get $\vec{y} \in \beta^\Gamma_Y((\mathcal{P}^+_\omega f)(X'))$ if and only if there exist $y_1, \ldots, y_k \in (\mathcal{P}^+_\omega f)(X')$ such that $\vec{y} = \langle y_{o_1}, \ldots, y_{o_m}\rangle$. This in turn is equivalent to saying that there exist $x_1, \ldots, x_k \in X'$ such that $\vec{y} = \langle f(x_{o_1}), \ldots, f(x_{o_m})\rangle = (f^m)(\langle x_{o_1}, \ldots, x_{o_m}\rangle)$. This now means that there exists $\vec{x} \in \beta^\Gamma_X(X')$ such that $\vec{y} = (f^m)(\vec{x})$, which is to say $\vec{y} \in (\mathcal{P}^+_\omega(f^m))(\beta^\Gamma_X(X'))$.

□

Our representation result states that any natural transformation ζ as in (A.1) arises as a (point-wise) union of some basic transformations β^Γ.

Theorem A.2.4 *A transformation* ζ *as in (A.1) is natural if and only if we can write is as*

$$\zeta = \bigcup_{\Gamma \in M} \beta^\Gamma \quad \textit{for some} \quad M \in \mathcal{P}^+_\omega(\mathtt{Par}[m]).$$

The union on the right hand side of the above equation is to be read pointwise, i.e. for any set X and $X' \in \mathcal{P}^+_\omega X$ we have

$$\big(\bigcup_{\Gamma \in M} \beta^\Gamma\big)_X(X') = \bigcup_{\Gamma \in M} \beta^\Gamma_X(X').$$

Such a union always yields a natural transformation, as the lemma below states. Its proof is straightforward.

Lemma A.2.5 *Let* $\mathrm{F}, \mathrm{G} : \mathsf{C} \to \mathsf{Set}$ *be functors, and let* $\nu^i : \mathrm{F} \Rightarrow \mathcal{P}^+_\omega \mathrm{G}$ *for* $i \in I$ *be functors, where* I *is a nonempty, finite set. Then their pointwise union*

$$\bigcup_{i \in I} \nu^i : \mathrm{F} \Rightarrow \mathcal{P}^+_\omega \mathrm{G}$$

is a natural transformation of the same type as well.

Before we give the proof of Theorem A.2.4 we remark that the mentioned representation is not unique in general, due to the following fact about the natural transformations β^Γ, which immediately follows from their definition.

Lemma A.2.6 *For $\Gamma, \Gamma' \in \mathtt{Par}[m]$ with $\Gamma \preceq \Gamma'$ we have $\beta^{\Gamma'} \subseteq \beta^\Gamma$, where the subset relation is to be read point-wise, i.e. $\beta^{\Gamma'}_X(X') \subseteq \beta^\Gamma_X(X')$ for all sets X and $X' \in \mathcal{P}^+_\omega X$.*

Let M be the representation from Theorem A.2.4. With the above lemma, for $\Gamma \in M$ and $\Gamma' \in \mathtt{Par}[m]$ with $\Gamma \prec \Gamma'$ the union on the right hand side of the equation in Theorem A.2.4 does not depend on whether Γ is in M as well or not. We will therefore call Γ' *redundant* in this setting. This means that the union is solely determined by the minimal elements of M. On the other hand, it is easy to verify that the resulting natural transformations differ for two sets with different minimal elements. So the representation is unique up to the inclusion or omission of redundant partitions.

For the proof of Theorem A.2.4 we need the following lemma.

Lemma A.2.7 *Let ζ be a natural transformation as in (A.1). For a set X and $X' \in \mathcal{P}^+_\omega X$ we have that $\vec{x} \in \zeta_X(X')$ implies $x_i \in X'$ for all $1 \leq i \leq m$.*

Proof: Let $\mathtt{in} : X' \hookrightarrow X$ be the subset inclusion and consider the following naturality square:

$$\begin{array}{ccc} \mathcal{P}^+_\omega X' & \xrightarrow{\zeta_{X'}} & \mathcal{P}^+_\omega({X'}^m) \\ {\scriptstyle \mathcal{P}^+_\omega \mathtt{in}}\downarrow & & \downarrow{\scriptstyle \mathcal{P}^+_\omega(\mathtt{in}^m)} \\ \mathcal{P}^+_\omega X & \xrightarrow[\zeta_X]{} & \mathcal{P}^+_\omega(X^m) \end{array} \qquad \begin{array}{ccc} X' & \xmapsto{\zeta_{X'}} & \zeta_{X'}(X') \ni \vec{x}' \\ {\scriptstyle \mathcal{P}^+_\omega \mathtt{in}}\downarrow & & \downarrow{\scriptstyle \mathcal{P}^+_\omega(\mathtt{in}^m)} \quad \downarrow{\scriptstyle \mathtt{in}^m} \\ X' & \xmapsto[\zeta_X]{} & \zeta_X(X') \ni \vec{x} \end{array}$$

We can read off that for every $\vec{x} \in \zeta_X(X') \subseteq (X')^m$ there has to be $\vec{x}' \in \zeta_{X'}(X')$ with $\vec{x} = \mathtt{in}^m(\vec{x}')$. We get $x_i = \mathtt{in}(x'_i) = x'_i \in X'$ for all i as wanted.

□

Proof: [Theorem A.2.4] We claim that the statement holds for

$$M := \{\Gamma \in \mathtt{Par}[m] \mid \beta^\Gamma \subseteq \zeta\},$$

which is to say that

$$\zeta = \bigcup\{\beta^\Gamma \mid \Gamma \in \mathtt{Par}[m], \beta^\Gamma \subseteq \zeta\}$$

Since the other inclusion is immediate, we need to show $\zeta \subseteq \bigcup\{\beta^\Gamma \mid \Gamma \in \mathtt{Par}[m], \beta^\Gamma \subseteq \zeta\}$ only, which is to say that for any set X, subset $X' \in \mathcal{P}^+_\omega X$, and $\vec{x} \in \zeta_X(X')$ we have to find $\Gamma \in \mathtt{Par}[m]$ such that $\beta^\Gamma \subseteq \zeta$ and $\vec{x} \in \beta^\Gamma_X(X')$. We show that we can take $\Gamma = \mathtt{par}(\vec{x})$. From $\vec{x} \in \zeta_X(X')$ it follows with

Lemma A.2.7 that $x_i \in X'$ for all i. With $\mathtt{par}(\vec{x}) \preceq \mathtt{par}(\vec{x})$ this yields $\vec{x} \in \beta_X^{\mathtt{par}(\vec{x})}(X')$ as needed (cf. Def. A.2.2). It remains to be shown that $\beta^{\mathtt{par}(\vec{x})} \subseteq \zeta$.

Without loss of generality we can assume that X, X', and $\vec{x}$ are such that $\mathtt{par}(\vec{x})$ is minimal with respect to the order $\prec$. By this we mean that there are no Y, $Y' \in \mathcal{P}_\omega^+ Y$, and $\vec{y} \in \zeta_Y(Y')$ with $\mathtt{par}(\vec{y}) \prec \mathtt{par}(\vec{x})$. Otherwise, we choose Y, Y', and $\vec{y}$ as above such that $\mathtt{par}(\vec{y})$ is minimal and carry out the argument below for them instead to obtain $\beta^{\mathtt{par}(\vec{y})} \subseteq \zeta$. With Lemma A.2.6 we have $\beta^{\mathtt{par}(\vec{x})} \subseteq \beta^{\mathtt{par}(\vec{y})}$ and thus $\beta^{\mathtt{par}(\vec{x})} \subseteq \zeta$ as needed.

To prove $\beta^{\mathtt{par}(\vec{x})} \subseteq \zeta$ under the minimality assumption, we take arbitrary sets Y and $Y' \in \mathcal{P}_\omega^+ Y$ and show $\beta_Y^{\mathtt{par}(\vec{x})}(Y') \subseteq \zeta_Y(Y')$. For any $\vec{y} \in \beta_Y^{\mathtt{par}(\vec{x})}(Y')$, i.e. $\vec{y} \in Y^m$ with $y_i \in Y'$ for all i and $\mathtt{par}(\vec{x}) \preceq \mathtt{par}(\vec{y})$, we derive $\vec{y} \in \zeta_Y(Y')$ as follows:

For $Z := X' \times Y'$ and $\overrightarrow{\langle x, w\rangle} := \big\langle \langle x_1, w_1\rangle, \ldots, \langle x_m, w_m\rangle \big\rangle$ we find

$$
\begin{array}{lcl}
\vec{x} \in \zeta_X(X') & \iff & \vec{x} \in \underbrace{\zeta_X\big((\mathcal{P}_\omega^+ \pi_1)(Z)\big)}_{\overset{\text{nat. }\zeta}{=} (\mathcal{P}_\omega^+(\pi_1^m))(\zeta_Z(Z))} \\
& \iff & \exists \vec{z} \in \zeta_Z(Z) : \vec{x} = \pi_1^m(\vec{z}) \\
& \iff & \exists \vec{w} \in (Y')^m : \overrightarrow{\langle x, w\rangle} \in \zeta_Z(Z) \\
& \overset{(*)}{\iff} & \overrightarrow{\langle x, y\rangle} \in \zeta_Z(Z) \\
& \implies & \underbrace{\pi_2^m(\overrightarrow{\langle x, y\rangle})}_{=\vec{y}} \in \underbrace{\big(\mathcal{P}_\omega^+(\pi_2^m)\big)\big(\zeta_Z(Z)\big)}_{\overset{\text{nat. }\zeta}{=} \zeta_Y((\mathcal{P}_\omega^+ \pi_2)(Z)) = \zeta_Y(Y')} \\
& \iff & \vec{y} \in \zeta_Y(Y').
\end{array}
$$

The implication "$\implies$" in step $(*)$ remains to be explained: We easily find $\mathtt{par}(\overrightarrow{\langle x, w\rangle}) \preceq \mathtt{par}(\vec{x})$, but with $\overrightarrow{\langle x, w\rangle} \in \zeta_Z(Z)$ the above minimality assumption on $\vec{x}$ rules out that $\mathtt{par}(\overrightarrow{\langle x, w\rangle})$ is strictly smaller than $\mathtt{par}(\vec{x})$. So we find $\mathtt{par}(\overrightarrow{\langle x, w\rangle}) = \mathtt{par}(\vec{x})$, which implies $\mathtt{par}(\vec{x}) \preceq \mathtt{par}(\vec{w})$. Together with the assumption $\mathtt{par}(\vec{x}) \preceq \mathtt{par}(\vec{y})$ this means that $x_i = x_j$ implies $w_i = w_j$ as well as $y_i = y_j$. With this observation the function $f : Z \to Z$ which exchanges $\langle x_i, w_i\rangle$ and $\langle x_i, y_i\rangle$ for all $i \in \{1, \ldots, m\}$ is well defined (in the sense that whenever multiple cases in the definition apply, then they all determine the same result) by

$$
f(x, y) := \begin{cases} \langle x_i, y_i\rangle & \text{if } \langle x, y\rangle = \langle x_i, w_i\rangle \text{ for some } 1 \le i \le m, \\ \langle x_i, w_i\rangle & \text{if } \langle x, y\rangle = \langle x_i, y_i\rangle \text{ for some } 1 \le i \le m, \\ \langle x, y\rangle & \text{otherwise.} \end{cases}
$$

The function f is self inverse and thus bijective, so that we find $(\mathcal{P}_\omega^+ f)(Z) = Z$.

Knowing this we reason as follows:

$$\begin{aligned}\overrightarrow{\langle x,w\rangle} \in \zeta_Z(Z) \quad &\Longrightarrow \quad \underbrace{f^m(\overrightarrow{\langle x,w\rangle})}_{=\overrightarrow{\langle x,y\rangle}} \in \underbrace{\big(\mathcal{P}^+_\omega(f^m)\big)\big(\zeta_Z(Z)\big)}_{\stackrel{\text{nat. }\zeta}{=}\zeta_Z((\mathcal{P}^+_\omega f)(Z))=\zeta_Z(Z)} \\ &\Longleftrightarrow \quad \overrightarrow{\langle x,y\rangle} \in \zeta_Z(Z).\end{aligned}$$

This concludes the proof of Theorem A.2.4.

□

As the next step, we generalise the above result to natural transformations of the slightly more complex type

$$\tilde{\zeta} : (\mathcal{P}^+_\omega)^E \Rightarrow \mathcal{P}^+_\omega(\mathrm{Id}^m) \tag{A.3}$$

for an arbitrary set E and a natural number m. Again it can be shown that such a natural transformation can be written as the union of finitely many basic ones. Before, the basic transformations β^Γ were determined by a partition $\Gamma = \{c_1, \ldots, c_k\} \in \mathtt{Par}[m]$. To generate the tuples in $\beta^\Gamma_X(X')$, for every equivalence class c_j we draw an element y_j from X' and place it in all positions $i \in c_j$ of the tuple. Now the procedure is similar, but, since we are given a family of subsets $(X'_e)_{e\in E}$, we further need to know for each $1 \leq j \leq k$ from which subset X'_{e_j} (for $e_j \in E$) we have to draw y_j.

Since this makes the definition of the basic transformations in the style of Definition A.2.2 a bit unwieldy, we will instead use derivation rules similar to those in (A.2) for the formulation of our representation result.

Corollary A.2.8 *Every natural transformation $\tilde{\zeta}$ as in (A.3) can be characterised by a finite set of derivation rules of the shape*

$$\frac{y_j \in X'_{e_j}\ (1 \leq j \leq k)}{\langle y_{o_1}, \ldots, y_{o_m}\rangle \in \tilde{\zeta}\big((X'_e)\big)} \tag{A.4}$$

for some $k \in \mathbb{N}$; $e_1, \ldots, e_k \in E$; $1 \leq o_i \leq k$ $(1 \leq i \leq m)$ as follows: For every set X and subsets X'_e $(e \in E)$ we have that $\tilde{\zeta}_X\big((X'_e)_{e\in E}\big)$ contains $\vec{x} \in X^m$ just in case this can be inferred from an instance of any of the rules.

Proof: [sketch] Applying Lemma A.1.8 we find that $\tilde{\zeta}$ from (A.3) is equivalent to a natural transformation

$$\xi_{(X_e)_{e\in E}} : \prod_{e\in E} \mathcal{P}^+_\omega X_e \Rightarrow \mathcal{P}^+_\omega\big((\coprod_{e\in E} X_e)^m\big) : \mathsf{Set}^E \to \mathsf{Set} \tag{A.5}$$

The functor describing the codomain of ξ can be manipulated as follows

$$\begin{aligned}\mathcal{P}^+_\omega\big((\coprod_{e\in E} X_e)^m\big) \quad &\simeq \quad \mathcal{P}^+_\omega\big(\coprod_{\vec{e}\in E^m} (X_{e_1} \times \cdots \times X_{e_m})\big) \\ &\simeq \quad \coprod_{M\in\mathcal{P}^+_\omega(E^m)} \big(\prod_{\vec{e}\in M} \mathcal{P}^+_\omega(X_{e_1} \times \cdots \times X_{e_m})\big),\end{aligned}$$

where the first equivalence uses distributivity and the second Lemma A.1.5. With the last representation we can apply Lemma A.1.3 and Lemma A.1.2 (ii) to find that ξ can be characterised by some set $M \in \mathcal{P}^+_\omega(E^m)$ together with natural transformations

$$\big(\xi^{\vec{e}}_{(X_e)_{e\in E}} : \prod_{e\in E} \mathcal{P}^+_\omega X_e \Rightarrow \mathcal{P}^+_\omega(X_{e_1} \times \cdots \times X_{e_m})\big)_{\vec{e}\in M}. \tag{A.6}$$

For the individual natural transformations $\xi^{\vec{e}}$ we get a representation using a variant of Lemma A.2.4. We decided not to treat the statement in this more general version explicitly, because it needs a more complicated notation although the idea behind the proof is the same.

We get the desired representation of $\tilde{\zeta}$ after collecting the resulting derivation rules from $\xi^{\vec{e}}$ for all $\vec{e} \in M$.

□

A.3 The distribution functor

In this section we will adapt the above representation results for natural transformations arising in the nondeterministic setting to the probabilistic one, i.e. we will reprove them after replacing the (nonempty) powerset functor $\mathcal{P}^+_\omega$ by the distribution functor $\mathcal{D}_\omega$ from Def. 5.1.1. The statements are rather similar in structure, but the proofs here are technically more complicated.

First we develop a representation result for natural transformations of the type

$$\zeta : \mathcal{D}_\omega \Rightarrow \mathcal{D}_\omega(\mathrm{Id}^m) \tag{A.7}$$

for some natural number m. It turns out that any such transformation can be described uniquely as a convex combination of certain basic ones.

Examining these natural transformations for small m we again find that the case $m = 0$ is trivial and that for $m = 1$ we get the identity only. Two candidates for $m = 2$ are given, for any set X and $\phi \in \mathcal{D}_\omega X$, as

$$\begin{aligned}
\zeta^1_X(\phi) &:= \Big[\ \langle x, y\rangle \mapsto \phi(x) \cdot \phi(y)\ \Big] \in \mathcal{D}_\omega(X \times X),\\
\zeta^2_X(\phi) &:= \Big[\ \langle x, y\rangle \mapsto \begin{cases} \phi(x) & \text{if } x = y \\ 0 & \text{otherwise} \end{cases}\ \Big] \in \mathcal{D}_\omega(X \times X).
\end{aligned}$$

But unlike the nondeterministic setting, where the corresponding two transformations were the only ones, moreover every (pointwise) convex combination of these two basic transformations above is natural as well, i.e. transformations of the shape $r \cdot \zeta^1_X + (1 - r) \cdot \zeta^2_X$ for $r \in (0, 1)$.

The definition of the basic natural transformations generalises to arbitrary m as follows, where we use the notation for partitions as introduced in Definition A.2.1.

Definition A.3.1 *Let $m \in \mathbb{N}$. For $\Gamma \in \mathtt{Par}[m]$ we define the* **basic transformations**

$$\beta^\Gamma : \mathcal{D}_\omega \Rightarrow \mathcal{D}_\omega(\mathrm{Id}^m)$$

for any set X and $\phi \in \mathcal{D}_\omega X$ as

$$\beta^\Gamma_X(\phi) := \Big[\, \langle x_1, \ldots, x_m\rangle \mapsto \begin{cases} \prod_{c\in\Gamma} \phi(x_{c\downarrow}) & \textit{if } \Gamma \preceq \mathtt{par}(\langle x_1, \ldots, x_m\rangle) \\ 0 & \textit{otherwise} \end{cases} \Big].$$

Note that this definition is actually closely related to the corresponding Definition A.2.2 for the powerset functor. To make this apparent, we can write the latter equivalently as follows, where we view a subset $X' \subseteq X$ as a function $X' : X \to 2$ with $2 := \{\bot, \top\}$:

$$\beta^\Gamma_X(X') := \Big[\langle x_1, \ldots, x_m\rangle \mapsto \begin{cases} \bigwedge_{c\in\Gamma} X'(x_{c\downarrow}) & \text{if } \Gamma \preceq \mathtt{par}(\langle x_1, \ldots, x_m\rangle) \\ \bot & \text{otherwise} \end{cases} \Big].$$

The probability distribution $\beta^\Gamma_X(\phi)$ is generated by the following probabilistic procedure: For each equivalence class c_j in $\Gamma = \{c_1, \ldots, c_k\}$ we draw an element y_j from X according to the probability distribution ϕ. The chosen y_j is put in all positions $i \in c_j$ of the resulting vector. So the probability that one particular vector appears is the probability that all k independent choices are made appropriately, i.e. the product of the probabilities for all individual choices. Writing $o_i \in \{1, \ldots, k\}$ to denote the index of the equivalence class of Γ containing i for $1 \le i \le m$ (i.e. $i \in c_{o_i}$), we can define β^Γ equivalently by the following rule, where X denotes any set:

$$\frac{\phi(y_j) = u_j \ (1 \le j \le k)}{\beta^\Gamma_X(\phi)(\langle y_{o_1}, \ldots, y_{o_m}\rangle) = u_1 \cdot \ldots \cdot u_k} \tag{A.8}$$

Similar to the proof of Lemma A.2.3, we can use this rule notation for an easy proof of the following statement.

Lemma A.3.2 *For each $\Gamma \in \mathtt{Par}[m]$ the transformation β^Γ from Def. A.3.1 is natural.*

Our representation result to be developed states that all natural transformations ζ in (A.1) arise uniquely as convex combinations of these basic transformations. Before we state and prove it, we present two technical lemmata that will be useful in the proof.

A.3.1 Simple statements about real valued functions

Below we present two facts about real valued functions.

Lemma A.3.3 *For $u \in \mathbb{R}_0^+$ let $f : [0,u] \to \mathbb{R}$ be a function with a bounded range satisfying*

$$f(r+s) = f(r) + f(s)$$

for all $r, s \in \mathbb{R}_0^+$ such that $r + s \in [0,u]$. Then for all $r \in [0,u]$ and $c \in [0,1]$ we find

$$f(c \cdot r) = c \cdot f(r).$$

Proof: By induction on $p \in \mathbb{N}$ we easily get

$$f(p \cdot r) = p \cdot f(r) \tag{A.9}$$

for all $r \in [0,u]$ with $p \cdot r \in [0,u]$. This further implies $f(r/q) = f(r)/q$ for all $q \in \mathbb{N}$ with $q > 0$ and $r \in [0,u]$. So the statement is true for $c = p/q$, i.e. for rational c. For an arbitrary c choose a sequence of rational numbers $(c_n)_{n\in\mathbb{N}}$ with $c_n \leq c$ and $c_n \to c$ for $n \to \infty$. We calculate

$$c \cdot f(r) = (\lim_{n\to\infty} c_n) \cdot f(r) = \lim_{n\to\infty} (c_n \cdot f(r)) = \lim_{n\to\infty} f(c_n \cdot r) \overset{(*)}{=} f(c \cdot r).$$

For the step marked with $(*)$ we instantiate the following calculation with $d_n = c_n \cdot r$ and $d = c \cdot r$: for any sequence $(d_n)_{n\in\mathbb{N}}$ and $d \in [0,u]$ with $d_n \to d$ for $n \to \infty$ and $d_n \leq d$ we have

$$\begin{aligned} f(d) &= \lim_{n\to\infty} f\big(d_n + (d - d_n)\big) \\ &= \lim_{n\to\infty} \big(f(d_n) + f(d - d_n)\big) \\ &= \lim_{n\to\infty} f(d_n) + \underbrace{\lim_{n\to\infty} f(d - d_n)}_{=0}. \end{aligned}$$

To see that the last addend is zero indeed, note first that $d - d_n$ converges to zero. Now the identity follows with $f(e_n) \to 0$ for $e_n \to 0$. This is because otherwise there exists $\varepsilon > 0$ such that arbitrary close to zero we can still find values $e \in \mathbb{R}_0^+$ with $\varepsilon < |f(e)|$, which is in conflict with our assumption on f being bounded: to arrive at a contradiction, take a bound $b > 0$; let $k = \lceil \frac{b}{\varepsilon} \rceil$ and choose $e \in [0, u/k]$ such that $\varepsilon < |f(e)|$; this implies

$$k \cdot e \in [0,u] \quad \text{and} \quad b \leq k \cdot \varepsilon < k \cdot |f(e)| \overset{(A.9)}{=} |f(k \cdot e)|.$$

□

For the next statement we first introduce some notation about vectors in $(\mathbb{R}_0^+)^M$ for a finite set M. We define the order $\preceq$ on $\mathbb{R}_0^+$ pointwise, i.e. $\vec{u} \preceq \vec{v}$ if and only

if $\vec{u}(i) \leq \vec{v}(i)$ for all $i \in M$. Moreover, for $\vec{v} \in (\mathbb{R}_0^+)^M$, $r \in \mathbb{R}_0^+$, and some index i, which may or may not be an element M, we write $\vec{v}[i := r] \in (\mathbb{R}_0^+)^{M \cup \{i\}}$ for an update or extension of $\vec{v}$ at index i, i.e. the vector with $(\vec{v}[i := r])(i) = r$ and $(\vec{v}[i := r])(j) = \vec{v}(j)$ for all $j \in M \setminus \{i\}$. For $i \neq j$ we abbreviate $\vec{v}[i := r][j := s]$ to $\vec{v}[i := r, j := s]$.

Lemma A.3.4 *For a finite set M let $C \subseteq (\mathbb{R}_0^+)^M$ be downward closed, i.e. $\vec{u} \preceq \vec{v}$ and $\vec{v} \in C$ imply $\vec{u} \in C$, and let $h : C \to \mathbb{R}$ be a function which is componentwise linear, i.e. for all $\vec{v} \in C$, $i \in M$, and $c \in [0, 1]$ we have $h(\vec{v}[i := c \cdot \vec{v}(i)]) = c \cdot h(\vec{v})$. Then there exists $\tau \in \mathbb{R}$ with*

$$h(\vec{v}) = \tau \cdot \prod_{i \in M} \vec{v}(i) \quad \textit{for all } \vec{v} \in C.$$

The statement is rather obvious. Still we give an explicit proof, because we have to be a bit careful about the domain restriction.

Proof: Assume that there exists $\vec{u} \in C$ such that $\vec{u}(i) > 0$ for all $i \in M$. Note that the statement is trivially true in case such a $\vec{u}$ does not exist, because with $\vec{u}(i) = 0$ for some i the linearity assumption easily implies $h(\vec{u}) = 0$. We show that the statement holds for

$$\tau := \frac{h(\vec{u})}{\prod_{i \in M} \vec{u}(i)}.$$

Take any $\vec{v} \in C$. Setting $I := \{i \in M \mid \vec{v}(i) > \vec{u}(i)\}$ we can apply the linearity assumption $|I|$ and $|M \setminus I|$ times respectively to get

$$(\prod_{i \in I} \frac{\vec{u}(i)}{\vec{v}(i)}) \cdot h(\vec{v}) = h(\mathrm{glb}(\vec{u}, \vec{v})) = (\prod_{i \in M \setminus I} \frac{\vec{v}(i)}{\vec{u}(i)}) \cdot h(\vec{u})$$

where by $\mathrm{glb}(\vec{u}, \vec{v})$ we denote the *greatest upper bound* of the two vectors, i.e. $\mathrm{glb}(\vec{u}, \vec{v})(i) = \min(\vec{u}(i), \vec{v}(i))$. This implies

$$h(\vec{v}) = (\prod_{i \in M} \frac{\vec{v}(i)}{\vec{u}(i)}) \cdot h(\vec{u}) = \tau \cdot \prod_{i \in M} \vec{v}(i).$$

We use the step via $\mathrm{glb}(\vec{u}, \vec{v})$ to make sure that we do not run out of the domain of h on our way.

□

A.3.2 The representation theorem

Next we state and prove our representation theorem for natural transformations ζ as in (A.7).

Theorem A.3.5 *For $m \in \mathbb{N}$ a transformation $\zeta : \mathcal{D}_\omega \Rightarrow \mathcal{D}_\omega(\mathrm{Id}^m)$ is natural if and only if it can be represented as a convex combination of the basic ones from Definition A.3.1, i.e.*

$$\zeta = \sum_{\Gamma \in \mathtt{Par}[m]} \mu(\Gamma) \cdot \beta^\Gamma \quad \text{for some } \mu \in \mathcal{D}_\omega(\mathtt{Par}[m]).$$

It can easily be shown that for $\mu, \mu' \in \mathcal{D}_\omega(\mathtt{Par}[m])$ we have

$$\sum_{\Gamma \in \mathtt{Par}[m]} \mu(\Gamma) \cdot \beta^\Gamma \;=\; \sum_{\Gamma \in \mathtt{Par}[m]} \mu'(\Gamma) \cdot \beta^\Gamma \quad \text{if and only if} \quad \mu = \mu',$$

so the above representation of ζ by a distribution μ is unique. Different from the nondeterministic setting, in the probabilistic case there are no redundant partitions!

For the proof we need a few more lemmata. The first one narrows down the information needed from $\phi \in \mathcal{D}_\omega X$ and $\vec{x} \in X^m$ to determine the value of $\zeta_X(\phi)(\vec{x})$.

Lemma A.3.6 *Let ζ be a natural transformation as in (A.7), X and Y be sets, $\phi \in \mathcal{D}_\omega X$ and $\psi \in \mathcal{D}_\omega Y$ be distributions, and let $\vec{x} = \langle x_1, \ldots, x_m \rangle \in X^m$ and $\vec{y} = \langle y_1, \ldots, y_m \rangle \in Y^m$. If*

$$\mathtt{par}(\vec{x}) = \mathtt{par}(\vec{y}) \quad \text{and} \quad \phi(x_i) = \psi(y_i) \quad \text{for all} \quad 1 \le i \le m$$

then

$$\zeta_X(\phi)(\vec{x}) = \zeta_Y(\psi)(\vec{y}).$$

Proof: Let $\Gamma := \mathtt{par}(\vec{x})$ $(= \mathtt{par}(\vec{y}))$, $Z := \Gamma \cup \{*\}$, $\chi \in \mathcal{D}_\omega Z$ with $\chi(c) := \phi(x_{c\downarrow})$ $(= \psi(y_{c\downarrow}))$ for $c \in \Gamma$ and $\chi(*) := 1 - \chi[\Gamma]$, and let $\vec{z} := \langle [1]_\Gamma, \ldots, [m]_\Gamma \rangle$. With $f : X \to Z$ where

$$f(x) := \begin{cases} [i]_\Gamma & \text{if } x = x_i \text{ for some } i \in \{1, \ldots, m\}, \\ * & \text{otherwise,} \end{cases}$$

we find

$$\begin{array}{rcll} \zeta_X(\phi)(\vec{x}) & = & \zeta_X(\phi)[(f^m)^{-1}(\vec{z})] & \big\{(f^m)^{-1}(\vec{z}) = \{\vec{x}\}\big\} \\ & = & \big((\mathcal{D}_\omega(f^m))(\zeta_X(\phi))\big)(\vec{z}) & \big\{\text{Def. } \mathcal{D}_\omega\big\} \\ & = & \zeta_Z\big((\mathcal{D}_\omega f)(\phi)\big)(\vec{z}) & \big\{\text{nat. } \zeta\big\} \\ & = & \zeta_Z(\chi)(\vec{z}). & \big\{(\mathcal{D}_\omega f)(\phi) = \chi\big\} \end{array}$$

In the same way we obtain $\zeta_Y(\psi)(\vec{y}) = \zeta_Z(\chi)(\vec{z})$, which implies the statement.

□

The above lemma states that all we need to know to determine $\zeta_X(\phi)(\vec{x}) \in [0,1]$ for $\vec{x} = \langle x_1, \ldots, x_m \rangle \in X^m$ is $\mathtt{par}(\vec{x})$ and the probabilities $\phi(x_1), \ldots, \phi(x_m)$. In $\vec{x}$ there occur precisely $k = |\mathtt{par}(\vec{x})|$ different elements of X. So we actually need the k probabilities $\phi(x_{c\downarrow})$ for all $c \in \mathtt{par}(\vec{x})$, which sum up to at most 1. We will talk about ζ in a representation that takes exactly this data.

Definition A.3.7 *For* $\Gamma \in \mathtt{Par}[m]$ *let*

$$\mathbb{C}^\Gamma := \{\vec{u} \in (\mathbb{R}_0^+)^\Gamma \mid |\vec{u}| \leq 1\},$$

where $|\vec{u}| = \sum_{c\in\Gamma} \vec{u}(c)$. *For every natural transformation* ζ *as in* (A.7) *we define a family of functions*

$$\big(\gamma^\Gamma : \mathbb{C}^\Gamma \to [0,1]\big)_{\Gamma\in\mathtt{Par}[m]}$$

by $\gamma^\Gamma(\vec{u}) := \zeta_X(\phi)(\vec{x})$ *where* X, ϕ, *and* $\vec{x}$ *are such that* $\Gamma = \mathtt{par}(\vec{x})$ *and* $\vec{u}(c) = \phi(x_{c\downarrow})$ *for* $c \in \Gamma$.

For the validity of the definition note that according to Lemma A.3.6 the definition is independent of the particular choice of X, $\vec{x}$, and ϕ, and that for all $\Gamma \in \mathtt{Par}[m]$ and $\vec{u} \in \mathbb{C}^\Gamma$ we can find suitable X, ϕ, and $\vec{x}$ (take e.g. $X := \Gamma \cup \{*\}$, $\vec{x} := \langle [1]_\Gamma, \ldots, [m]_\Gamma\rangle$, $\phi(c) := \vec{u}(c)$ for $c \in \Gamma$ and $\phi(*) := 1 - |\vec{u}|$).

For later use we check what the above family of functions looks like in the case of our basic transformations: for $\Gamma' \in \mathtt{Par}[m]$ we find that $\beta^{\Gamma'}$ induces a family of functions $\big(\gamma^\Gamma : \mathbb{C}^\Gamma \to [0,1]\big)_{\Gamma\in\mathtt{Par}[m]}$ with

$$\gamma^\Gamma(\vec{u}) = \begin{cases} \displaystyle\prod_{c'\in\Gamma'} \vec{u}([c'\downarrow]_\Gamma) = \prod_{c\in\Gamma} \vec{u}(c)^{|l(\Gamma',c)|} & \text{if } \Gamma' \preceq \Gamma, \\ 0 & \text{otherwise.} \end{cases} \tag{A.10}$$

where $l(\Gamma', c) := \{c' \in \Gamma' \mid c' \subseteq c\}$.

The family of functions (γ^Γ) induced by a natural transformation ζ has the following property:

Lemma A.3.8 *For* $\Gamma \in \mathtt{Par}[m]$, $d \in \Gamma$, $\vec{u} : \Gamma \setminus \{d\} \to \mathbb{R}_0^+$, *and* $r, s \in \mathbb{R}_0^+$ *such that* $|\vec{u}| + r + s \leq 1$ *we have*

$$\begin{aligned} \gamma^\Gamma(\vec{u}[d := r+s]) \;=\; & \gamma^\Gamma(\vec{u}[d := r]) + \gamma^\Gamma(\vec{u}[d := s]) + \\ & \sum_{\emptyset\subset d'\subset d} \gamma^{\Gamma(d')}(\vec{u}[d' := r, (d \setminus d') := s]), \end{aligned}$$

where $\Gamma(d') \in \mathtt{Par}[m]$ *for* $\emptyset \subset d' \subset d$ *results from* Γ *by splitting* d *into* d' *and* $d \setminus d'$, *i.e.*

$$\Gamma(d') := (\Gamma \setminus \{d\}) \cup \{d', d \setminus d'\} \prec \Gamma.$$

Proof: The statement follows from the following consideration:

Let Y be a set with $p \notin Y$. Set $X := Y \cup \{p\}$ and let $\phi \in \mathcal{D}_\omega X$ and $\vec{x} = \langle x_1, \ldots, x_m\rangle \in X^m$ be such that p occurs in $\vec{x}$, i.e. $d := \{i \mid x_i = p\} \neq \emptyset$. We can "split" the state p into two, say q_1 and q_2 (for $q_i \notin Y$), and distribute the original probability of p as $\phi(p) = r + s$ on the two copies. This yields $X' := Y \cup \{q_1, q_2\}$

and $\phi' \in \mathcal{D}_\omega X'$ with $\phi'(q_1) := r$, $\phi'(q_2) := s$, and $\phi'(y) = \phi(y)$ for $y \in Y$. Define $f : X' \to X$ by $f(q_i) := p$ and $f(y) := y$ for $y \in Y$. From the naturality square of ζ and f below we read off that $\zeta_X(\phi)(\vec{x})$ is the sum of all $\zeta_{X'}(\phi')(\vec{x}')$ such that $\vec{x}'$ arises by replacing in $\vec{x}$ any occurrence of p by either q_1 or q_2.

$$\begin{array}{ccc} \phi' & \xmapsto{\zeta_{X'}} & \zeta_{X'}(\phi') \\ {\scriptstyle \mathcal{D}_\omega f}\downarrow & \text{nat. } \zeta & \downarrow{\scriptstyle \mathcal{D}_\omega(f^m)} \\ \phi & \xmapsto[\zeta_X]{} & \zeta_X(\phi) \end{array}$$

Formally, for $d' \subseteq d$ set

$$\vec{x}^{d'} = \langle x_1^{d'}, \ldots, x_m^{d'} \rangle \quad \text{with} \quad x_i^{d'} = \begin{cases} q_1 & \text{if } i \in d', \\ q_2 & \text{if } i \in d \setminus d', \\ x_i & \text{otherwise.} \end{cases}$$

Then we calculate as follows:

$$\begin{aligned} &\zeta_X(\phi)(\vec{x}) \\ &\quad = \zeta_X\big((\mathcal{D}_\omega f)(\phi')\big)(\vec{x}) && \{(\mathcal{D}_\omega f)(\phi') = \phi\} \\ &\quad = \big((\mathcal{D}_\omega(f^m))(\zeta_{X'}(\phi'))\big)(\vec{x}) && \{\text{nat. } \zeta\} \\ &\quad = \zeta_{X'}(\phi')[(f^m)^{-1}(\vec{x})] && \{\text{def. } \mathcal{D}_\omega\} \\ &\quad = \zeta_{X'}(\phi')[\{\vec{x}^{d'} \mid d' \subseteq d\}] && \{f^{-1}(x_i) = \begin{cases} \{p,q\} & \text{if } i \in d, \\ \{x_i\} & \text{else.} \end{cases}\} \\ &\quad = \sum_{d' \subseteq d} \zeta_{X'}(\phi')(\vec{x}^{d'}) \\ &\quad = \zeta_{X'}(\phi')(\vec{x}^{\emptyset}) + \zeta_{X'}(\phi')(\vec{x}^{d}) + \\ &\qquad \sum_{\emptyset \subset d' \subset d} \zeta_{X'}(\phi')(\vec{x}^{d'}). \end{aligned}$$

This idea leads to the statement through an application of Lemma A.3.6 to both ends of the computation, together with the observation that for $\Gamma = \mathtt{par}(\vec{x})$ (which yields $d \in \Gamma$) we have $\mathtt{par}(\vec{x}^{\emptyset}) = \Gamma = \mathtt{par}(\vec{x}^{d})$ and $\mathtt{par}(\vec{x}^{d'}) = \Gamma(d')$ for $\emptyset \subset d' \subset d$. (Of course we again need to show that for all suitable Γ and $\vec{u}$ we can find appropriate X, ϕ, and $\vec{x}$. This can be done as suggested in the remark after Def. A.3.7.)

□

Lemma A.3.9 *Let ζ be a natural transformation as in (A.7) inducing the family (γ^Γ) from Definition A.3.7. For every downwards closed set $M \subseteq \mathtt{Par}[m]$ there exist weights $(\tau_\Gamma \in \mathbb{R}_0^+)_{\Gamma \in M}$ such that for all $\Gamma \in M$ and $\vec{u} \in \mathbb{C}^\Gamma$ we have*

$$\gamma^\Gamma(\vec{u}) = \sum_{\Gamma' \preceq \Gamma} \tau_{\Gamma'} \cdot \prod_{c \in \Gamma} \vec{u}(c)^{|l(\Gamma', c)|}, \tag{A.11}$$

where again $l(\Gamma', c) := \{c' \in \Gamma' \mid c' \subseteq c\}$.

Proof: The statement is proved by induction on the size of M. For $M = \emptyset$ there is nothing to do. For nonempty M choose a maximal element $\hat{\Gamma} \in M$. Take $(\tau_\Gamma)_{\Gamma \in \hat{M}}$ as given by the induction hypothesis for $\hat{M} := M \setminus \{\hat{\Gamma}\}$. These coefficients satisfy equation (A.11) for all $\Gamma \in \hat{M}$ already. We have to find $\tau_{\hat{\Gamma}}$ so that it holds for $\hat{\Gamma}$ as well.

For all $\vec{v} \in \mathbb{C}^{\hat{\Gamma}}$ defining

$$f(\vec{v}) := \sum_{\Gamma' \prec \hat{\Gamma}} \tau_{\Gamma'} \cdot \prod_{c \in \hat{\Gamma}} \vec{v}(c)^{|l(\Gamma', c)|} \quad \text{and} \quad h(\vec{v}) := \gamma^{\hat{\Gamma}}(\vec{v}) - f(\vec{v}),$$

we need to show that there exists $\tau_{\hat{\Gamma}} \in \mathbb{R}_0^+$ such that

$$h(\vec{v}) = \tau_{\hat{\Gamma}} \cdot \prod_{c \in \hat{\Gamma}} \vec{v}(c).$$

The set $\mathbb{C}^{\hat{\Gamma}}$ and function h satisfy the assumption on C and h in Lemma A.3.4. Applying the lemma we get that it suffices to show that h is linear in all components. For any $d \in \hat{\Gamma}$ and $\vec{u} \in (\mathbb{R}_0^+)^{\hat{\Gamma} \setminus \{d\}}$ we define $h_{\vec{u}} : [0, 1 - |\vec{u}|] \to \mathbb{R}$ by $h_{\vec{u}}(r) := h(\vec{u}[d := r])$ and show that

$$h_{\vec{u}}(c \cdot r) = c \cdot h_{\vec{u}}(r)$$

for all $c \in [0, 1]$ and $r \in [0, 1 - |\vec{u}|]$. Since $h_{\vec{u}}$ is bounded (because $\gamma^{\hat{\Gamma}}$ and f are), we can apply Lemma A.3.3 for this task. With this statement, it remains to be shown that

$$h_{\vec{u}}(r + s) = h_{\vec{u}}(r) + h_{\vec{u}}(s) \quad \text{for all } r, s \in \mathbb{R}_0^+ \text{ such that } r + s \leq 1 - |\vec{u}|.$$

Writing also $\gamma_{\vec{u}}^{\hat{\Gamma}}(r) := \gamma^{\hat{\Gamma}}(\vec{u}[d := r])$ and $f_{\vec{u}}(r) := f(\vec{u}[d := r])$ this is equivalent to

$$\gamma_{\vec{u}}^{\hat{\Gamma}}(r + s) - \gamma_{\vec{u}}^{\hat{\Gamma}}(r) - \gamma_{\vec{u}}^{\hat{\Gamma}}(s) = f_{\vec{u}}(r + s) - f_{\vec{u}}(r) - f_{\vec{u}}(s). \tag{A.12}$$

For the left hand side we compute with (a) Lemma A.3.8 and (b) the induction hypothesis

$$\begin{aligned}
&\gamma_{\vec{u}}^{\hat{\Gamma}}(r + s) - \gamma_{\vec{u}}^{\hat{\Gamma}}(r) - \gamma_{\vec{u}}^{\hat{\Gamma}}(s) \\
&\overset{(a)}{=} \sum_{\emptyset \subset d' \subset d} \gamma^{\hat{\Gamma}(d')}(\vec{u}[d' := r, (d \setminus d') := s]) \\
&\overset{(b)}{=} \sum_{\emptyset \subset d' \subset d} \Bigg(\sum_{\Gamma' \preceq \hat{\Gamma}(d')} \underbrace{\tau_{\Gamma'} \cdot \Big(\prod_{c \in \hat{\Gamma} \setminus \{d\}} \vec{u}(c)^{|l(\Gamma', c)|} \Big)}_{=: \tilde{\tau}_{\Gamma'}} \cdot r^{|l(\Gamma', d')|} \cdot s^{|l(\Gamma', d \setminus d')|} \Bigg) \\
&= \sum_{\Gamma' \prec \hat{\Gamma}} \tilde{\tau}_{\Gamma'} \cdot \sum_{\emptyset \subset d' \subset d, \Gamma' \preceq \hat{\Gamma}(d')} r^{|l(\Gamma', d')|} \cdot s^{|l(\Gamma', d)| - |l(\Gamma', d')|}.
\end{aligned} \tag{A.13}$$

By the definitions of f, $f_{\vec{u}}$, and $\tilde{\tau}_{\Gamma'}$ from above we have

$$f_{\vec{u}}(t) = \sum_{\Gamma' \prec \hat{\Gamma}} \tau_{\Gamma'} \cdot \Big(\prod_{c \in \hat{\Gamma} \setminus \{d\}} \vec{u}(c)^{|l(\Gamma', c)|} \Big) \cdot t^{|l(\Gamma', d)|} = \sum_{\Gamma' \prec \hat{\Gamma}} \tilde{\tau}_{\Gamma'} \cdot t^{|l(\Gamma', d)|}.$$

So for the right hand side of (A.12) we get

$$\begin{aligned} & f_{\vec{u}}(r+s) - f_{\vec{u}}(r) - f_{\vec{u}}(s) \\ &= \sum_{\Gamma' \prec \hat{\Gamma}} \tilde{\tau}_{\Gamma'} \cdot \big((r+s)^{|l(\Gamma', d)|} - r^{|l(\Gamma', d)|} - s^{|l(\Gamma', d)|} \big) \\ &= \sum_{\Gamma' \prec \hat{\Gamma}} \tilde{\tau}_{\Gamma'} \cdot \sum_{j=1}^{|l(\Gamma', d)|-1} \binom{|l(\Gamma', d)|}{j} \cdot r^j \cdot s^{|l(\Gamma', d)|-j}, \end{aligned} \tag{A.14}$$

where in the second step we used

$$(r+s)^k = \sum_{j=0}^{k} \binom{k}{j} \cdot r^j \cdot s^{k-j} = r^k + s^k + \sum_{j=1}^{k-1} \binom{k}{j} \cdot r^j \cdot s^{k-j}.$$

So we are done if for all $\Gamma' \prec \hat{\Gamma}$ we can show that the two inner sums of (A.13) and (A.14) are equal, i.e. the following equation holds with $k := |l(\Gamma', d)|$.

$$\sum_{\emptyset \subset d' \subset d, \Gamma' \preceq \hat{\Gamma}(d')} r^{|l(\Gamma', d')|} \cdot s^{k - |l(\Gamma', d')|} = \sum_{j=1}^{k-1} \binom{k}{j} \cdot r^j \cdot s^{k-j} \tag{A.15}$$

Let us investigate what the sum on the left hand side ranges over: For $d' \subseteq d$ we can rewrite the condition $\Gamma' \preceq \hat{\Gamma}(d')$ into $\Gamma' \preceq \hat{\Gamma}$ and $c' \subseteq d'$ or $c' \subseteq d \setminus d'$ for all $c' \in \Gamma'$ with $c' \subseteq d$, i.e. for all $c' \in l(\Gamma', d)$. The first part is implied by our assumption $\Gamma' \prec \hat{\Gamma}$. The second can be stated as $d' = \bigcup C$ for some $C \subseteq l(\Gamma', d)$. The condition $\emptyset \subset d' \subset d$ is satisfied just in case $\emptyset \subset C \subset l(\Gamma', d)$. So with $|l(\Gamma', d')| = |C|$ the sum on the left hand side of (A.15) rewrites to

$$\sum_{\emptyset \subset C \subset l(\Gamma', d)} r^{|C|} \cdot s^{k-|C|} = \sum_{j=1}^{|l(\Gamma', d)|-1} \underbrace{\big| \big\{ C \subseteq l(\Gamma', d) \;\big|\; |C| = j \big\} \big|}_{= \binom{|l(\Gamma', d)|}{j}} \cdot r^j \cdot s^{k-j}.$$

Remembering $k = |l(\Gamma', d)|$ this completes the proof of (A.15) and thus of (A.12).

We have demonstrated that there is a $\tau_{\hat{\Gamma}} \in \mathbb{R}$ such that equation (A.11) holds for $\Gamma = \hat{\Gamma}$. It remains to be shown that $\tau_{\hat{\Gamma}} \geq 0$. For $r \in \mathbb{R}_0^+$ let $\vec{r} \in (\mathbb{R}_0^+)^{\hat{\Gamma}}$ denote the vector with $\vec{r}(c) = r$ for all $c \in \hat{\Gamma}$. With $0 < r \leq \frac{1}{|\hat{\Gamma}|}$ we find $|\vec{r}| \leq 1$.

We have

$$\begin{aligned} 0 \;\leq\; \gamma^{\hat{\Gamma}}(\vec{r}) \;&=\; \sum_{\Gamma' \preceq \hat{\Gamma}} \tau_{\Gamma'} \cdot \prod_{c \in \hat{\Gamma}} r^{|l(\Gamma', c)|} \\ &=\; \sum_{\Gamma' \preceq \hat{\Gamma}} \tau_{\Gamma'} \cdot r^{|\Gamma'|} \\ &=\; r^{|\hat{\Gamma}|} \cdot (\tau_{\hat{\Gamma}} + \sum_{\Gamma' \prec \hat{\Gamma}} \tau_{\Gamma'} \cdot r^{|\Gamma'| - |\hat{\Gamma}|}). \end{aligned}$$

This implies $\tau_{\hat{\Gamma}} \geq -\sum_{\Gamma' \prec \hat{\Gamma}} \tau_{\Gamma'} \cdot r^{|\Gamma'|-|\hat{\Gamma}|}$. Since $|\Gamma'| > |\hat{\Gamma}|$ for all $\Gamma' \prec \hat{\Gamma}$ we have that the right hand side converges to 0 for $r \to 0$, and so $\tau_{\hat{\Gamma}} \geq 0$ as wanted.

□

Our main representation theorem easily follows from this lemma.

Proof: [Theorem A.3.5] Just take $\mu(\Gamma) = \tau_\Gamma$ for the values from Lemma A.3.9 for $M = \mathtt{Par}[m]$. Equations (A.11) and (A.10) together yield the equation in the statement. It remains to be shown that we get a probability distribution indeed, i.e. that all weights sum up to one. For an arbitrary set X and distribution $\phi \in \mathcal{D}_\omega X$ we have

$$1 = \zeta_X(\phi)[X^m] = \sum_{\Gamma \in \mathtt{Par}[m]} \tau_\Gamma \cdot \underbrace{\beta^\Gamma_X(\phi)[X^m]}_{=1} = \sum_{\Gamma \in \mathtt{Par}[m]} \tau_\Gamma.$$

□

The natural transformations we encountered in Section 5.2 are actually of a slightly more complicated type than those in (A.7), namely

$$\xi : (\mathcal{D}_\omega)^E \Rightarrow \mathcal{D}_\omega(\mathrm{Id}^m) \tag{A.16}$$

for an arbitrary set E and a natural number m. Such natural transformations can be written as (finite) convex combinations of basic ones as well. Since a definition of these basic transformations as a generalisation of Definition A.3.1 is hard to read, we prefer a rule-style presentation similar to the one in (A.8) in our representation result below.

Corollary A.3.10 *Every natural transformation ξ in (A.16) can be characterised by a finite set $\mathcal{R}$ of derivation rules of the shape*

$$\frac{\phi_{e_j}(y_j) = u_j \quad (1 \leq j \leq k)}{\xi\big((\phi_e)\big)(\langle y_{o_1}, \ldots, y_{o_m}\rangle) \stackrel{+}{=} w \cdot u_1 \cdot \ldots \cdot u_k}$$

for some $k \in \mathbb{N}$; $e_1, \ldots, e_k \in E$; $1 \leq o_i \leq k$ $(1 \leq i \leq m)$; and $w \in (0, 1]$. We call w the weight *of the rule and demand that the weights of all rules in $\mathcal{R}$ sum up to 1. Such a collection of rules determines ξ as follows: For every set X, distributions $\phi_e \in \mathcal{D}_\omega X$ $(e \in E)$, and $\vec{x} \in X^m$ we have that $\xi_X\big((\phi_e)_{e \in E}\big)(\vec{x})$*

is the sum of all contributions that can be inferred from instances of all of the rules. Such an instance associates to the variables y_j elements from X and to u_j probabilities such that the premises are satisfied and $\langle y_{o_1}, \ldots, y_{o_m} \rangle = \vec{x}$. The instance contributes a portion of $w \cdot u_1 \cdot \ldots \cdot u_k$ to the probability of $\vec{x}$ in $\xi_X\big((\phi_e)_{e \in E}\big)$.

The corollary arises as a straightforward generalisation of Theorem A.3.5 in a similar way as Corollary A.2.8 arises as a generalisation of Theorem A.2.4. Example 5.2.2 illustrates how to read these rules.

Each rule in the representation given by Corollary A.3.10 encodes a contribution of one basic transformation (corresponding to Definition A.3.1 in the case of ζ from (A.7)) to ξ. So, like Theorem A.3.5, the statement says that ξ can be written as a convex combination of basic transformations. And again – in contrast to the nondeterministic case – this convex combination is uniquely determined. However, the use of the rule notation in Corollary A.3.10 introduces redundancy, even if we look at the rules up to the renaming of variables. The reason is that we can write down more then one rule encoding the same basic transformation. The weights of these rules together give its contribution to ξ. We call this the *splitting of a rule* and we will not disallow it, since it does not harm (the above interpretation of the rules for instance still works fine.) So the representation is unique up to the renaming of variables and the splitting of rules.

A.4 Future work

The collection of technical lemmata presented in this section evolved from our efforts to analyse concrete natural transformations related to distributive laws of signature functors over behaviour functors describing the system types we considered. Although one can already handle a considerable number of natural transformation using these statements, we still consider the collection incomplete. To extend it, it may be worthwhile to proceed in a more systematic manner: one could for instance define an interesting class of natural transformations inductively and describe a procedure to obtain concrete representations for any transformation in this class.

Examples of natural transformations that we cannot tackle satisfactorily with our current tools are those involving nested applications of the powerset and distribution functor. We encountered these natural transformation while studying specification formats for Segala systems (cf. Definition 5.1.6). In order to give corresponding statements, it may be necessary to first understand the natural transformations better that we studied in Sections A.2 and A.3. We expect that it is possible to simplify and at the same time generalise the arguments we have given. The following observation may be a potential starting point of such an investigation:

The natural transformations under consideration in (A.1) and (A.7) have, respectively, the functors $\mathcal{P}_\omega \mathrm{F}$ and $\mathcal{D}_\omega \mathrm{F}$ for $\mathrm{F} = \mathrm{Id}^m$ as codomain. We found that any such natural transformation can be written as a (pointwise) union or convex combination of basic ones. Those are indexed by partitions $\Gamma \in \mathtt{Par}[m]$ of $\{1, \ldots, m\}$, i.e. the set of positions of the tuples in $\mathrm{F}X = X^m$ (we use again the notation from Def. A.2.1 to talk about these partitions). The same partitions also index the set of what we shall call the *subfunctors* of F, i.e. the functors F' such that for all sets X we have $\mathrm{F}'X \subseteq \mathrm{F}X$ and such that the collection of inclusions is natural: it can be shown that the subfunctors of $\mathrm{F} = \mathrm{Id}^m$ are $\{\mathrm{F}^\Gamma \mid \Gamma \in \mathtt{Par}[m]\}$, where

$$\mathrm{F}^\Gamma X := \{\vec{x} \in X^m \mid \Gamma \preceq \mathtt{par}(\vec{x})\}.$$

This observation suggests that there may be a characterisation of natural transformations with functors $\mathcal{P}_\omega \mathrm{F}$ or $\mathcal{D}_\omega \mathrm{F}$ as codomain in terms of the subfunctors of F, maybe with a suitable restriction on the class of functors F.

Bibliography

[AAMV03] Peter Aczel, Jiří Adámek, Stefan Milius, and Jiří Velebil. Infinite trees and completely iterative theories: a coalgebraic view. *Theoretical Computer Science*, 300:1–45, 2003.

[AFV01] Luca Aceto, Wan Fokkink, and Chris Verhoef. Structural operational semantics. In Bergstra et al. [BPS01], pages 197–292.

[AM89] Peter Aczel and Nax Mendler. A final coalgebra theorem. In D.H. Pitt, D.E. Rydeheard, P. Dybjer, A.M. Pitts, and A. Poigné, editors, *Proc. 3rd CTCS*, volume 389 of *Lecture Notes in Computer Science*, pages 357–365. Springer Verlag, 1989.

[And99] Suzana Andova. Process algebra with probabilistic choice. In J.-P. Katoen, editor, *Proc. ARTS'99*, volume 1601 of *Lecture Notes in Computer Science*, pages 111–129. Springer Verlag, 1999.

[And02] Suzana Andova. *Probabilistic process algebra*. PhD thesis, Technical University of Eindhoven, The Netherlands, 2002.

[Bae03] J.C.M. Baeten. Embedding untimed into timed process algebra: The case for explicit termination. *Journal of Mathematical Structures in Computer Science*, 13(4):589–618, 2003.

[Bar00] Falk Bartels. Generalised coinduction. Technical Report SEN-R0043, CWI, Amsterdam, December 2000.

[Bar01] Falk Bartels. Generalised coinduction. In Andrea Corradini, Marina Lenisa, and Ugo Montanari, editors, *Proc. Coalgebraic Methods in Computer Science (CMCS 2001)*, volume 44 of *Electronic Notes in Theoretical Computer Science*. Elsevier Science Publishers, 2001.

[Bar02a] Falk Bartels. GSOS for probabilistic transition systems. Technical Report SEN-R0221, CWI, Amsterdam, 2002.

[Bar02b] Falk Bartels. GSOS for probabilistic transition systems (extended abstract). In Moss [Mos02].

[Bar03] Falk Bartels. Generalised coinduction. *Journal of Mathematical Structures in Computer Science*, 13(2):321–348, April 2003.

[Ber99] Marco Bernardo. *Theory and application of extended markovian process algebra*. PhD thesis, University of Bologna, 1999.

[BHK01] Ed Brinksma, Holger Hermanns, and Joost-Pieter Katoen, editors. *Lectures on formal methods and performance analysis: first EEF/ Euro Summer School on Trends in Computer Science, Berg en Dal, The Netherlands, July 3–7, 2000: revised lectures*, volume 2090 of *Lecture Notes in Computer Science and Lecture Notes in Artificial Intelligence*. Springer Verlag, 2001.

[BIM95] Bard Bloom, Sorin Istrail, and Albert R. Meyer. Bisimulation can't be traced. *Journal of the ACM*, 42(1):232–268, January 1995.

[BPS01] Jan A. Bergstra, Alban Ponse, and Scott A. Smolka, editors. *Handbook of Process Algebra*. Elsevier Science Publishers, 2001.

[BSV03] Falk Bartels, Ana Sokolova, and Erik de Vink. A hierarchy of probabilistic system types. In Gumm [Gum03a].

[BSV04] Falk Bartels, Ana Sokolova, and Erik de Vink. A hierarchy of probabilistic system types. *Theoretical Computer Science*, 2004. to appear.

[BW90] J.C.M. Baeten and W.P. Weijand. *Process Algebra*, volume 18 of *Cambridge Tracts in Theoretical Computer Science*. Cambridge University Press, Cambridge, England, 1990.

[CHL03] Daniela Cancila, Furio Honsell, and Marina Lenisa. Generalized coiteration schemata. In Gumm [Gum03a].

[DEP02] Josée Desharnais, Abbas Edalat, and Prakash Panangaden. Bisimulation for labelled markov processes. *Information and Computation (formerly Information and Control)*, 179(2):163–193, December 2002.

[DHK99] Pedro R. D'Argenio, Holger Hermanns, and Joost-Pieter Katoen. On generative parallel composition. In Christel Baier, Michael Huth, Marta Kwiatkowska, and Mark Ryan, editors, *Proc. PROBMIV '98*, volume 22 of *Electronic Notes in Theoretical Computer Science*. Elsevier Science Publishers, 1999.

[Fok94] Wan Fokkink. The tyft/tyxt format reduces to tree rules. In M. Hagiya and J. C. Mitchell, editors, *Proceedings of the International Symposium on Theoretical Aspects of Computer Software (TACS'94)*, Sendai, Japan, volume 789 of *Lecture Notes in Computer Science*, pages 440–453. Springer Verlag, 1994.

[Fok00] Wan Fokkink. *Introduction to Process Algebra.* Springer Verlag, 2000.

[FT01] Marcelo Fiore and Daniele Turi. Semantics of name and value passing. In *Proc. 16th LICS Conf.*, pages 93–104. IEEE, Computer Society Press, 2001.

[Geu92] Herman Geuvers. Inductive and coinductive types with iteration and recursion. In B. Nordström, K. Pettersson, and G. Plotkin, editors, *Informal Proceedings Workshop on Types for Proofs and Programs, Båstad, Sweden, 8–12 June 1992*, pages 193–217. Dept. of Computing Science, Chalmers Univ. of Technology and Göteborg Univ., 1992. Available from the authors homepage at `http://www.cs.kun.nl/~herman/`.

[Gla90] Rob J. van Glabbeek. The linear time – branching time spectrum. Report CS-R9029, CWI, Amsterdam, 1990. Extended abstract in J.C.M. Baeten & J.W. Klop, editors: Proceedings *CONCUR '90*, Lecture Notes in Theoretical Computer Science 458, Springer Verlag, 1990, pages 278–297.

[Gla93] Rob J. van Glabbeek. The linear time – branching time spectrum II; the semantics of sequential systems with silent moves (extended abstract). In E. Best, editor, Proceedings *CONCUR'93,* 4^{th} International Conference on Concurrency Theory, Hildesheim, Germany, August 1993, volume 715 of *Lecture Notes in Computer Science*, pages 66–81. Springer Verlag, 1993.

[Gla01] Rob J. van Glabbeek. The linear time – branching time spectrum I. In Bergstra et al. [BPS01], pages 3–99.

[GSS95] Rob J. van Glabbeek, Scott A. Smolka, and Bernhard Steffen. Reactive, generative and stratified models of probabilistic processes. *Information and Computation*, 121(1):59–80, August 1995.

[Gum99] H. Peter Gumm. Elements of the general theory of coalgebras. Lecture notes for LUATCS'99, available from `http://www.mathematik.uni-marburg.de/~gumm/Papers/`, 1999.

[Gum03a] H. Peter Gumm, editor. *Proc. Coalgebraic Methods in Computer Science (CMCS 2003)*, volume 82 of *Electronic Notes in Theoretical Computer Science.* Elsevier Science Publishers, 2003.

[Gum03b] H. Peter Gumm. State based systems are coalgebras. *Cubo - Matemática Educacional*, 5(2):239–262, 2003. Available from `http://www.mathematik.uni-marburg.de/~gumm/Papers/`.

[GV89] Jan Friso Groote and Frits W. Vaandrager. Structural operational semantics and bisimulation as a congruence (extended abstract). In Giorgio Ausiello, Mariangiola Dezani-Ciancaglini, and Simona

Ronchi Della Rocca, editors, *Automata, Languages and Programming, 16th International Colloquium*, volume 372 of *Lecture Notes in Computer Science*, pages 423–438. Springer Verlag, July 1989.

[GV92] Jan Friso Groote and Frits Vaandrager. Structured operational semantics and bisimulation as a congruence. *Information and Computation*, 100(2):202–260, October 1992.

[Han94] Hans Hansson. *Time and Probability in Formal Design of Distributed Systems*. Series in Real-Time Safety Critical Systems. Elsevier Science Publishers, 1994.

[Har02] J.I. den Hartog. *Probabilistic extensions of semantical models*. PhD thesis, Vrije Universiteit Amsterdam, 2002.

[HJ94] Hans Hansson and Bengt Jonsson. A logic for reasoning about time and reliability. *Formal Aspects of Computing*, 836:481–535, 1994.

[HJ98] Ulrich Hensel and Bart Jacobs. Coalgebraic theories of sequences in PVS. *Journal of Logic and Computation*, 9(4):463–500, 1998.

[Jac02] Bart Jacobs. Exercises in coalgebraic specification. *Lecture Notes in Computer Science*, 2297:237–281, 2002.

[JLY01] Bengt Jonsson, Kim G. Larsen, and Wang Yi. Probabilistic extensions of process algebras. In Bergstra et al. [BPS01], pages 685–710.

[JR96] Bart Jacobs and Jan Rutten. A tutorial on (co)algebras and (co)induction. *Bulletin of the European Association for Theoretical Computer Science*, 62:222–259, 1996. Available from `http://www.cs.kun.nl/~bart/PAPERS/`.

[Kic02a] Marco Kick. Bialgebraic modelling of timed processes. In P. Widmayer, F. Triguero, R. Morales, M. Hennessy, S. Eidenbenz, and R. Conejo, editors, *Proceedings ICALP'02*, volume 2380 of *Lecture Notes in Computer Science*. Springer Verlag, 2002.

[Kic02b] Marco Kick. Rule formats for timed processes. In *Proceedings CMCIM'02*, volume 68(1) of *Electronic Notes in Theoretical Computer Science*. Elsevier Science Publishers, 2002.

[Kic03] Marco Kick. *Coalgebraic Modelling of Timed Processes*. PhD thesis, University of Edinburgh, 2003.

[Kli04a] Bartek Klin. *An abstract coalgebraic approach to process equivalence for well-behaved operational semantics*. PhD thesis, University of Aarhus, 2004.

[Kli04b] Bartek Klin. Adding recursive constructs to bialgebraic semantics. *Journal of Logic and Algebraic Programming*, 2004. Special issue devoted to Structural Operational Semantics, to appear.

[Kur00] Alexander Kurz. *Logics for Coalgebras and Applications to Computer Science*. PhD thesis, Ludwig-Maximilians-Universität München, 2000. Available from `http://www.informatik.uni-muenchen.de/~kurz`.

[Len99a] Marina Lenisa. From set-theoretic coinduction to coalgebraic coinduction: some results, some problems. In Bart Jacobs and Jan Rutten, editors, *Proc. Coalgebraic Methods in Computer Science (CMCS 1999)*, volume 19 of *Electronic Notes in Theoretical Computer Science*, pages 1–21. Elsevier Science Publishers, 1999.

[Len99b] Marina Lenisa. From set-theoretic coinduction to coalgebraic coinduction: Some results, some problems. Unpublished extension of [Len99a] (available through the author's homepage at `http://www.dimi.uniud.it/~lenisa/`), December 1999.

[LPW00] Marina Lenisa, John Power, and Hiroshi Watanabe. Distributivity for endofunctors, pointed and co-pointed endofunctors, monads and comonads. In Horst Reichel, editor, *Proc. Coalgebraic Methods in Computer Science (CMCS 2000)*, volume 33 of *Electronic Notes in Theoretical Computer Science*, pages 233–263. Elsevier Science Publishers, 2000.

[LPW04] Marina Lenisa, John Power, and Hiroshi Watanabe. Category theory for operational semantics. *Theoretical Computer Science*, 2004. to appear.

[LS91] Kim G. Larsen and Arne Skou. Bisimulation through probabilistic testing. *Information and Computation*, 94(1):1–28, September 1991.

[LS92] Kim G. Larsen and Arne Skou. Compositional verification of probabilistic processes. In W. R. Cleaveland, editor, *CONCUR '92: Third International Conference on Concurrency Theory*, volume 630 of *Lecture Notes in Computer Science*, pages 456–471. Springer Verlag, 1992.

[McI99] M.D. McIlroy. Functional pearl: Power series, power serious. *Journal of Functional Programming*, 9:323–335, 1999.

[Mil80] Robin Milner. *A Calculus for Communicating Systems*, volume 92 of *Lecture Notes in Computer Science*. Springer Verlag, 1980.

[Mil89] Robin Milner. *Communication and Concurrency*. International Series in Computer Science. Prentice Hall, 1989.

[Mil99] Robin Milner. *Communicating and Mobile Systems: The π-calculus*. Cambridge University Press, 1999.

[ML97] Saunders Mac Lane. *Categories for the Working Mathematician*, volume 5 of *Graduate Texts in Mathematics*. Springer Verlag, 2nd edition, 1997. (1st ed., 1971).

[Mos99] Lawrence S. Moss. Coalgebraic logic. *Annals of Pure and Applied Logic*, 96(1–3):277–317, 1999. Note that layout problems were corrected in volume 99 (1999).

[Mos01] Lawrence S. Moss. Parametric corecursion. *Theoretical Computer Science*, 260:139–163, 2001.

[Mos02] Lawrence S. Moss, editor. *Proc. Coalgebraic Methods in Computer Science (CMCS 2002)*, volume 65 of *Electronic Notes in Theoretical Computer Science*. Elsevier Science Publishers, 2002.

[Oos95] Jaap van Oosten. Basic category theory. BRICS Lecture Series LS-95-1, Dept. of Computer Science, Univ. of Århus, January 1995.

[Par81] David Park. Concurrency and automata on infinite sequences. In Peter Deussen, editor, *Theoretical Computer Science: 5th GI-Conference, Karlsruhe*, volume 104 of *Lecture Notes in Computer Science*, pages 167–183. Springer Verlag, 1981.

[Par00] Alberto Pardo. Towards merging recursion and comonads. In Johan Jeuring, editor, *Proc. WGP'2000*, Tech. Report UU-CS-2000-19, pages 50–68. Utrecht Univ., June 2000.

[Plo81] Gordon D. Plotkin. A structural approach to operational semantics. Technical Report DAIMI FN-19, Computer Science Department, Aarhus University, Aarhus, Denmark, September 1981.

[Pow03] John Power. Towards a theory of mathematical operational semantics. In Gumm [Gum03a].

[PT99] John Power and Daniele Turi. A coalgebraic foundation for linear time semantics. In M. Hofmann, D. Pavlović, and G. Rosolini, editors, *Proc. 8$^{\text{th}}$ CTCS Conf.*, volume 29 of *Electronic Notes in Theoretical Computer Science*. Elsevier Science Publishers, 1999.

[RT94] Jan Rutten and Daniele Turi. Initial algebra and final coalgebra semantics for concurrency. In Jaco de Bakker et al., editors, *Proc. of the REX workshop A Decade of Concurrency – Reflections and Perspectives*, volume 803 of *Lecture Notes in Computer Science*, pages 530–582. Springer Verlag, 1994.

[Rut96] Jan Rutten. Universal coalgebra: a theory of systems. Technical Report CS-R9652, CWI, Amsterdam, December 1996.

[Rut98] Jan Rutten. Automata and coinduction (an exercise in coalgebra). In *CONCUR'98: 9th International Conference on Concurrency Theory*, volume 1466 of *Lecture Notes in Computer Science*, pages 194–218. Springer Verlag, 1998.

[Rut00a] Jan Rutten. Behavioural differential equations: a coinductive calculus of streams, automata, and power series. Technical Report SEN-R0023, CWI, Amsterdam, 2000. Also appeared as [Rut03].

[Rut00b] Jan Rutten. Universal coalgebra: A theory of systems. *Theoretical Computer Science*, 249(1):3–80, October 2000.

[Rut01] Jan Rutten. Elements of stream calculus (an extensive exercise in coinduction). In S. Brookes and M. Mislove, editors, *Proc. of 17th Conf. on Mathematical Foundations of Programming Semantics, Aarhus, Denmark, 23–26 May 2001*, volume 45 of *Electronic Notes in Theoretical Computer Science*. Elsevier Science Publishers, 2001.

[Rut03] Jan Rutten. Behavioural differential equations: a coinductive calculus of streams, automata, and power series. *Theoretical Computer Science*, 308(1–3):1–53, 2003.

[San98] Davide Sangiorgi. On the bisimulation proof method. *Journal of Mathematical Structures in Computer Science*, 8:447–479, 1998.

[Seg95a] Roberto Segala. A compositional trace-based semantics for probabilistic automata. In Insup Lee and Scott A. Smolka, editors, *Proc. CONCUR '95*, volume 962 of *Lecture Notes in Computer Science*, pages 234–248. Springer Verlag, 1995.

[Seg95b] Roberto Segala. *Modeling and Verification of Randomized Distributed Real-Time Systems*. PhD thesis, MIT, 1995.

[Sim85] Robert de Simone. Higher-level synchronising devices in MEIJE–SCCS. *Theoretical Computer Science*, 37(3):245–267, December 1985. Fundamental studies.

[SL94] Roberto Segala and Nancy Lynch. Probabilistic simulations for probabilistic processes. In Bengt Jonsson and Joachim Parrow, editors, *CONCUR '94: Concurrency Theory, 5th International Conference*, volume 836 of *Lecture Notes in Computer Science*, pages 481–496, Uppsala, Sweden, August 1994. Springer Verlag.

[SL95] Roberto Segala and Nancy Lynch. Probabilistic simulations for probabilistic processes. *Nordic Journal of Computing*, 2(2):250–273, 1995. An extended abstract appeared as [SL94].

[Sto02a] Mariëlle Stoelinga. *Alea jacta est: verification of probabilistic, real-time and parametric systems*. PhD thesis, University of Nijmegen, the Netherlands, April 2002. Available via `http://www.soe.ucsc.edu/~marielle`.

[Sto02b] Mariëlle Stoelinga. An introduction to probabilistic automata. *Bulletin of the European Association for Theoretical Computer Science*, 78:176–198, 2002.

[SV04] Ana Sokolova and Erik de Vink. Classes of probabilistic automata, composition and comparison. In *Proc. Validation of Stochastic Systems, Dagstuhl, Germany, Dec. 2002*, Lecture Notes in Computer Science. Springer Verlag, 2004. To appear, available from `http://www.win.tue.nl/~ana/`.

[TP97] Daniele Turi and Gordon D. Plotkin. Towards a mathematical operational semantics. In *Proc. 12$^{\text{th}}$ LICS Conf.*, pages 280–291. IEEE, Computer Society Press, 1997.

[Tur96] Daniele Turi. *Functorial Operational Semantics and its Denotational Dual*. PhD thesis, Free University of Amsterdam, June 1996.

[Tur97] Daniele Turi. Categorical modelling of structural operational rules: case studies. In E. Moggi and G. Rosolini, editors, *Proc. 7$^{\text{th}}$ CTCS Conf.*, volume 1290 of *Lecture Notes in Computer Science*, pages 127–146. Springer Verlag, 1997.

[Tur01] Daniele Turi. Category theory lecture notes. Available from `http://www.dcs.ed.ac.uk/home/dt/`, 2001.

[UV99] Tarmo Uustalu and Varmo Vene. Primitive (co)recursion and course-of-value (co)iteration, categorically. *Informatica (IMI, Lithuania)*, 10(1):5–26, 1999.

[UVP01] Tarmo Uustalu, Varmo Vene, and Alberto Pardo. Recursion schemes from comonads. *Nordic Journal of Computing*, 8(3):366–390, 2001.

[VR99] Erik de Vink and Jan Rutten. Bisimulation for probabilistic transition systems: A coalgebraic approach. *Theoretical Computer Science*, 221:271–293, 1999.

[VU98] Varmo Vene and Tarmo Uustalu. Functional programming with apomorphisms (corecursion). *Proc. of of the Estonian Academy of Sciences: Physics, Mathematics*, 47(3):147–161, 1998. Selected Papers 9th Nordic Workshop on Programming Theory, NWPT'97, Tallinn, Estonia, 1997.

[Wat02] Hiroshi Watanabe. Well-behaved translations between structural operational semantics. In Moss [Mos02].

[WFG96] Rob J. van Wan Fokkink Glabbeek. Ntyft/ntyxt rules reduce to ntree rules. *Information and Computation*, 126(1):1–10, 1996.

[Wor00] James Worrell. *On coalgebras and final semantics*. PhD thesis, Oxford University, 2000.

Samenvatting

Over Gegeneraliseerde Co-Inductie en Probabilistische Specificatie Formaten

Distributieve wetten voor co-algebraïsche beschrijvingen

Co-algebra's van een gedragsfunctor worden gebruikt als wiskundige modellen van toestandgebaseerde dynamische systemen, zoals transitiesystemen, deterministische automaten, of oneindige gegevensstromen. Een finale co-algebra geeft een abstracte domein voor al het mogelijke gedrag dat een systeem van de gegeven soort kan weergeven. Om een bepaald gedrag te definiëren en daar eigenschappen, zoals gelijkheid, over te bewijzen, gebruiken we technieken die *co-inductief* heten. Het eenvoudigste principe is *co-iteratie*, en het volgt direct uit de definitie van finaliteit. Maar er zijn hiernaast ook meer uitgewerkte en expressieve schemata, zoals het duale (in een categoriale zin) van primitieve recursie of "*course-of-value*"-iteratie.

In dit proefschrift wordt het λ-co-iteratie schema geïntroduceerd. Dit schema krijgt een *distributieve wet* λ van een ander functor S over de bestudeerde gedragsfunctor B als parameter. Dit is een natuurlijke transformatie $\lambda : \mathrm{SB} \Rightarrow \mathrm{BS}$. Voor iedere λ levert het λ-co-iteratie schema een definitie- en een bewijsprincipe op, waaronder bijvoorbeeld de twee bovengenoemde schemata zijn en ook principes zoals het definieren van een taal met behulp van een niet-deterministische automaat.

In het geval dat de functoren S en B de structuur van een monade en co-monade hebben, wordt van een distributieve wet λ van S over B vereist dat deze de structuur in een bepaalde zin respecteert. Deze distributieve wetten worden door Turi en Plotkin gebruikt voor hun categorische aanpak van Plotkin's "*structural operational semantics*" (SOS). SOS specificaties zijn verzamelingen van regels om progammeertalen van een transitie systeem semantiek te voorzien. Het is bekend dat een goed gedrag van de specificatie kan worden gegarandeerd als men zich beperkt tot regels die aan bepaalde syntactische eisen voldoen. De categorische aanpak is gebaseerd op de observatie dat regels in sommige van deze SOS formaten ook tot uitdrukking kunnen worden gebracht door distributieve

wetten, en dat dit op zich voldoende is om de goede gedragseigenschappen to tonen.

Om een verband te kunnen leggen tussen gegeneraliseerde co-inductie en SOS formaten met een goed gedrag, probeerden we om λ-co-iteratie te gebruiken met distributieve wetten die SOS specificaties uitdrukken. Maar distributieve wetten over co-monaden zijn daar niet geschikt voor. Daarom hebben we de categorische aanpak van SOS specificaties uitgebreid naar distributieve wetten met minder structuur. De nieuwe versies van de theorie kunnen minder expressieve formaten aan maar zijn meer algemeen toepasbaar en garandeeren verdere belangrijke eigenschappen. Twee van deze eigenschappen komen uit ons λ-co-iteratie schema: bewaakte recursieve vergelijkingen (*guarded recursive equations*) hebben oplossingen en een bisimulatie "*up-to-context*" methode is geldig.

Een groot voordeel is dat de categorische aanpak van zich goed gedragende SOS specificaties kan worden toegepast op verschillende gedragfunctoren B. Daarom kan deze worden gebruikt om specificatie formaten voor verschillende soorten systemen te ontwerpen. Daarvoor is het belangrijk, dat distributive wetten in verband kunnen worden gebracht met regels in een bepaald formaat. Voor deze taak hebben we een de-compositionele aanpak ontwikkeld die natuurlijke transformaties precies analyseert. Hiermee kunnen we het eerder genoemde verband tussen GSOS regels en distributieve wetten in alle detail bewijzen. Verder leiden we formaten voor deterministische automaten en oneindige gegevensstromen af.

Onze belangrijkste toepassing analyseert distributieve wetten om de eerste zich goed gedragende en expressieve specificatie formaten voor operatoren voor probabilistische systemen te ontwikkelen. Deze zijn het *PGSOS* en het *Segala-GSOS* formaat. Specificaties in beide formaten hebben unieke canonieke modellen, probabilistische bisimulatie is een congruentie op alle modellen, bewaakte recursieve vergelijkingen hebben oplossingen, en het bisimulatie "*up-to-context*" principe is geldig.

Titles in the IPA Dissertation Series

J.O. Blanco. *The State Operator in Process Algebra*. Faculty of Mathematics and Computing Science, TUE. 1996-01

A.M. Geerling. *Transformational Development of Data-Parallel Algorithms*. Faculty of Mathematics and Computer Science, KUN. 1996-02

P.M. Achten. *Interactive Functional Programs: Models, Methods, and Implementation*. Faculty of Mathematics and Computer Science, KUN. 1996-03

M.G.A. Verhoeven. *Parallel Local Search*. Faculty of Mathematics and Computing Science, TUE. 1996-04

M.H.G.K. Kesseler. *The Implementation of Functional Languages on Parallel Machines with Distrib. Memory*. Faculty of Mathematics and Computer Science, KUN. 1996-05

D. Alstein. *Distributed Algorithms for Hard Real-Time Systems*. Faculty of Mathematics and Computing Science, TUE. 1996-06

J.H. Hoepman. *Communication, Synchronization, and Fault-Tolerance*. Faculty of Mathematics and Computer Science, UvA. 1996-07

H. Doornbos. *Reductivity Arguments and Program Construction*. Faculty of Mathematics and Computing Science, TUE. 1996-08

D. Turi. *Functorial Operational Semantics and its Denotational Dual*. Faculty of Mathematics and Computer Science, VUA. 1996-09

A.M.G. Peeters. *Single-Rail Handshake Circuits*. Faculty of Mathematics and Computing Science, TUE. 1996-10

N.W.A. Arends. *A Systems Engineering Specification Formalism*. Faculty of Mechanical Engineering, TUE. 1996-11

P. Severi de Santiago. *Normalisation in Lambda Calculus and its Relation to Type Inference*. Faculty of Mathematics and Computing Science, TUE. 1996-12

D.R. Dams. *Abstract Interpretation and Partition Refinement for Model Checking*. Faculty of Mathematics and Computing Science, TUE. 1996-13

M.M. Bonsangue. *Topological Dualities in Semantics*. Faculty of Mathematics and Computer Science, VUA. 1996-14

B.L.E. de Fluiter. *Algorithms for Graphs of Small Treewidth*. Faculty of Mathematics and Computer Science, UU. 1997-01

W.T.M. Kars. *Process-algebraic Transformations in Context*. Faculty of Computer Science, UT. 1997-02

P.F. Hoogendijk. *A Generic Theory of Data Types*. Faculty of Mathematics and Computing Science, TUE. 1997-03

T.D.L. Laan. *The Evolution of Type Theory in Logic and Mathematics*. Faculty of Mathematics and Computing Science, TUE. 1997-04

C.J. Bloo. *Preservation of Termination for Explicit Substitution*. Faculty of Mathematics and Computing Science, TUE. 1997-05

J.J. Vereijken. *Discrete-Time Process Algebra*. Faculty of Mathematics and Computing Science, TUE. 1997-06

F.A.M. van den Beuken. *A Functional Approach to Syntax and Typing*. Faculty of Mathematics and Informatics, KUN. 1997-07

A.W. Heerink. *Ins and Outs in Refusal Testing*. Faculty of Computer Science, UT. 1998-01

G. Naumoski and W. Alberts. *A Discrete-Event Simulator for Systems Engineering*. Faculty of Mechanical Engineering, TUE. 1998-02

J. Verriet. *Scheduling with Communication for Multiprocessor Computation*. Faculty of Mathematics and Computer Science, UU. 1998-03

J.S.H. van Gageldonk. *An Asynchronous Low-Power 80C51 Microcontroller*. Faculty of Mathematics and Computing Science, TUE. 1998-04

A.A. Basten. *In Terms of Nets: System Design with Petri Nets and Process Algebra*. Faculty of Mathematics and Computing Science, TUE. 1998-05

E. Voermans. *Inductive Datatypes with Laws and Subtyping – A Relational Model*. Faculty of Mathematics and Computing Science, TUE. 1999-01

H. ter Doest. *Towards Probabilistic Unification-based Parsing*. Faculty of Computer Science, UT. 1999-02

J.P.L. Segers. *Algorithms for the Simulation of Surface Processes*. Faculty of Mathematics and Computing Science, TUE. 1999-03

C.H.M. van Kemenade. *Recombinative Evolutionary Search*. Faculty of Mathematics and Natural Sciences, UL. 1999-04

E.I. Barakova. *Learning Reliability: a Study on Indecisiveness in Sample Selection*. Faculty of Mathematics and Natural Sciences, RUG. 1999-05

M.P. Bodlaender. *Schedulere Optimization in Real-Time Distributed Databases*. Faculty of Mathematics and Computing Science, TUE. 1999-06

M.A. Reniers. *Message Sequence Chart: Syntax and Semantics*. Faculty of Mathematics and Computing Science, TUE. 1999-07

J.P. Warners. *Nonlinear approaches to satisfiability problems*. Faculty of Mathematics and Computing Science, TUE. 1999-08

J.M.T. Romijn. *Analysing Industrial Protocols with Formal Methods*. Faculty of Computer Science, UT. 1999-09

P.R. D'Argenio. *Algebras and Automata for Timed and Stochastic Systems*. Faculty of Computer Science, UT. 1999-10

G. Fábián. *A Language and Simulator for Hybrid Systems*. Faculty of Mechanical Engineering, TUE. 1999-11

J. Zwanenburg. *Object-Oriented Concepts and Proof Rules*. Faculty of Mathematics and Computing Science, TUE. 1999-12

R.S. Venema. *Aspects of an Integrated Neural Prediction System*. Faculty of Mathematics and Natural Sciences, RUG. 1999-13

J. Saraiva. *A Purely Functional Implementation of Attribute Grammars*. Faculty of Mathematics and Computer Science, UU. 1999-14

R. Schiefer. *Viper, A Visualisation Tool for Parallel Progam Construction*. Faculty of Mathematics and Computing Science, TUE. 1999-15

K.M.M. de Leeuw. *Cryptology and Statecraft in the Dutch Republic*. Faculty of Mathematics and Computer Science, UvA. 2000-01

T.E.J. Vos. *UNITY in Diversity. A stratified approach to the verification of distributed algorithms*. Faculty of Mathematics and Computer Science, UU. 2000-02

W. Mallon. *Theories and Tools for the Design of Delay-Insensitive Communicating Processes*. Faculty of Mathematics and Natural Sciences, RUG. 2000-03

W.O.D. Griffioen. *Studies in Computer Aided Verification of Protocols*. Faculty of Science, KUN. 2000-04

P.H.F.M. Verhoeven. *The Design of the MathSpad Editor*. Faculty of Mathematics and Computing Science, TUE. 2000-05

J. Fey. *Design of a Fruit Juice Blending and Packaging Plant*. Faculty of Mechanical Engineering, TUE. 2000-06

M. Franssen. *Cocktail: A Tool for Deriving Correct Programs*. Faculty of Mathematics and Computing Science, TUE. 2000-07

P.A. Olivier. *A Framework for Debugging Heterogeneous Applications*. Faculty of Natural Sciences, Mathematics and Computer Science, UvA. 2000-08

E. Saaman. *Another Formal Specification Language*. Faculty of Mathematics and Natural Sciences, RUG. 2000-10

M. Jelasity. *The Shape of Evolutionary Search Discovering and Representing Search Space Structure*. Faculty of Mathematics and Natural Sciences, UL. 2001-01

R. Ahn. *Agents, Objects and Events a computational approach to knowledge, observation and communication*. Faculty of Mathematics and Computing Science, TU/e. 2001-02

M. Huisman. *Reasoning about Java programs in higher order logic using PVS and Isabelle*. Faculty of Science, KUN. 2001-03

I.M.M.J. Reymen. *Improving Design Processes through Structured Reflection*. Faculty of Mathematics and Computing Science, TU/e. 2001-04

S.C.C. Blom. *Term Graph Rewriting: syntax and semantics*. Faculty of Sciences, Division of Mathematics and Computer Science, VUA. 2001-05

R. van Liere. *Studies in Interactive Visualization*. Faculty of Natural Sciences, Mathematics and Computer Science, UvA. 2001-06

A.G. Engels. *Languages for Analysis and Testing of Event Sequences*. Faculty of Mathematics and Computing Science, TU/e. 2001-07

J. Hage. *Structural Aspects of Switching Classes*. Faculty of Mathematics and Natural Sciences, UL. 2001-08

M.H. Lamers. *Neural Networks for Analysis of Data in Environmental Epidemiology: A Case-study into Acute Effects of Air Pollution Episodes*. Faculty of Mathematics and Natural Sciences, UL. 2001-09

T.C. Ruys. *Towards Effective Model Checking*. Faculty of Computer Science, UT. 2001-10

D. Chkliaev. *Mechanical verification of concurrency control and recovery protocols*. Faculty of Mathematics and Computing Science, TU/e. 2001-11

M.D. Oostdijk. *Generation and presentation of formal mathematical documents*. Faculty of Mathematics and Computing Science, TU/e. 2001-12

A.T. Hofkamp. *Reactive machine control: A simulation approach using χ*. Faculty of Mechanical Engineering, TU/e. 2001-13

D. Bošnački. *Enhancing state space reduction techniques for model checking*. Faculty of Mathematics and Computing Science, TU/e. 2001-14

M.C. van Wezel. *Neural Networks for Intelligent Data Analysis: theoretical and experimental aspects*. Faculty of Mathematics and Natural Sciences, UL. 2002-01

V. Bos and J.J.T. Kleijn. *Formal Specification and Analysis of Industrial Systems*. Faculty of Mathematics and Computer Science and Faculty of Mechanical Engineering, TU/e. 2002-02

T. Kuipers. *Techniques for Understanding Legacy Software Systems*. Faculty of Natural Sciences, Mathematics and Computer Science, UvA. 2002-03

S.P. Luttik. *Choice Quantification in Process Algebra*. Faculty of Natural Sciences, Mathematics, and Computer Science, UvA. 2002-04

R.J. Willemen. *School Timetable Construction: Algorithms and Complexity*. Faculty of Mathematics and Computer Science, TU/e. 2002-05

M.I.A. Stoelinga. *Alea Jacta Est: Verification of Probabilistic, Real-time and Parametric Systems*. Faculty of Science, Mathematics and Computer Science, KUN. 2002-06

N. van Vugt. *Models of Molecular Computing*. Faculty of Mathematics and Natural Sciences, UL. 2002-07

A. Fehnker. *Citius, Vilius, Melius: Guiding and Cost-Optimality in Model Checking of Timed and Hybrid Systems*. Faculty

of Science, Mathematics and Computer Science, KUN. 2002-08

R. van Stee. *On-line Scheduling and Bin Packing*. Faculty of Mathematics and Natural Sciences, UL. 2002-09

D. Tauritz. *Adaptive Information Filtering: Concepts and Algorithms*. Faculty of Mathematics and Natural Sciences, UL. 2002-10

M.B. van der Zwaag. *Models and Logics for Process Algebra*. Faculty of Natural Sciences, Mathematics, and Computer Science, UvA. 2002-11

J.I. den Hartog. *Probabilistic Extensions of Semantical Models*. Faculty of Sciences, Division of Mathematics and Computer Science, VUA. 2002-12

L. Moonen. *Exploring Software Systems*. Faculty of Natural Sciences, Mathematics, and Computer Science, UvA. 2002-13

J.I. van Hemert. *Applying Evolutionary Computation to Constraint Satisfaction and Data Mining*. Faculty of Mathematics and Natural Sciences, UL. 2002-14

S. Andova. *Probabilistic Process Algebra*. Faculty of Mathematics and Computer Science, TU/e. 2002-15

Y.S. Usenko. *Linearization in μCRL*. Faculty of Mathematics and Computer Science, TU/e. 2002-16

J.J.D. Aerts. *Random Redundant Storage for Video on Demand*. Faculty of Mathematics and Computer Science, TU/e. 2003-01

M. de Jonge. *To Reuse or To Be Reused: Techniques for component composition and construction*. Faculty of Natural Sciences, Mathematics, and Computer Science, UvA. 2003-02

J.M.W. Visser. *Generic Traversal over Typed Source Code Representations*. Faculty of Natural Sciences, Mathematics, and Computer Science, UvA. 2003-03

S.M. Bohte. *Spiking Neural Networks*. Faculty of Mathematics and Natural Sciences, UL. 2003-04

T.A.C. Willemse. *Semantics and Verification in Process Algebras with Data and Timing*. Faculty of Mathematics and Computer Science, TU/e. 2003-05

S.V. Nedea. *Analysis and Simulations of Catalytic Reactions*. Faculty of Mathematics and Computer Science, TU/e. 2003-06

M.E.M. Lijding. *Real-time Scheduling of Tertiary Storage*. Faculty of Electrical Engineering, Mathematics & Computer Science, UT. 2003-07

H.P. Benz. *Casual Multimedia Process Annotation – CoMPAs*. Faculty of Electrical Engineering, Mathematics & Computer Science, UT. 2003-08

D. Distefano. *On Modelchecking the Dynamics of Object-based Software: a Foundational Approach*. Faculty of Electrical Engineering, Mathematics & Computer Science, UT. 2003-09

M.H. ter Beek. *Team Automata – A Formal Approach to the Modeling of Collaboration Between System Components*. Faculty of Mathematics and Natural Sciences, UL. 2003-10

D.J.P. Leijen. *The λ Abroad – A Functional Approach to Software Components*. Faculty of Mathematics and Computer Science, UU. 2003-11

W.P.A.J. Michiels. *Performance Ratios for the Differencing Method*. Faculty of Mathematics and Computer Science, TU/e. 2004-01

G.I. Jojgov. *Incomplete Proofs and Terms and Their Use in Interactive Theorem Proving*. Faculty of Mathematics and Computer Science, TU/e. 2004-02

P. Frisco. *Theory of Molecular Computing – Splicing and Membrane systems*. Faculty of Mathematics and Natural Sciences, UL. 2004-03

S. Maneth. *Models of Tree Translation*. Faculty of Mathematics and Natural Sciences, UL. 2004-04

Y. Qian. *Data Synchronization and Browsing for Home Environments*. Faculty of Mathematics and Computer Science and Faculty of Industrial Design, TU/e. 2004-05

F. Bartels. *On Generalised Coinduction and Probabilistic Specification Formats*. Faculty of Sciences, Division of Mathematics and Computer Science, VUA. 2004-06