Functorial Operational Semantics

and its Denotational Dual

Ψ

Daniele Turi

PhD Thesis
Free University, Amsterdam
June 1996

Author's e-mail address: dt@dcs.ed.ac.uk

VRIJE UNIVERSITEIT

Functorial Operational Semantics

and its Denotational Dual

ACADEMISCH PROEFSCHRIFT

ter verkrijging van de graad van doctor aan
de Vrije Universiteit te Amsterdam,
op gezag van de rector magnificus
prof.dr E. Boeker,
in het openbaar te verdedigen
ten overstaan van de promotiecommissie
van de faculteit der wiskunde en informatica
op donderdag 6 juni 1996 te 15.45 uur
in het hoofdgebouw van de universiteit, De Boelelaan 1105

door

Daniele Turi

geboren te Salerno, Italië

Promotor:	prof.dr J.W. de Bakker
Copromotoren:	dr B.P.F. Jacobs
	dr J.J.M.M. Rutten
Referent:	dr A.M. Pitts

Research supported by:

- The "Stichting Informatica Onderzoek in Nederland"
  (Project: *Non-well-founded sets and semantics of programming languages*)
- The CWI, Amsterdam
- The Laboratory for Foundations of Computer Science, Edinburgh
- The European Institute in the Logical Foundations of Computer Science (*EuroFOCS*)
- The European Community SCIENCE programme
  (Project: *Mathematical Structures in Semantics of Concurrency*)

a Lucia D'Agosto, per la sua fiducia fatta di universali

e

alla memoria di Giovanni Turi – e della sua filosofia

met dank aan mijn paranimfen Maurizio Gabbrielli en Roos Vogel

Preface

The notion of 'functorial operational semantics' introduced in this thesis is a categorical formulation (and generalization) of 'well-behaved' structural operational semantics based on labelled transition systems. This notion has several desirable properties (such as congruence of the associated strong bisimilarity, and existence of a dual denotational semantics) and it subsumes existing, concrete schemes (such as GSOS) for guaranteeing such good behaviour – at least in the case of languages extending 'basic process algebra'. All this is achieved via use of the category theory of monads and comonads. The thesis also contains a coalgebraic treatment of the theory of non-well-founded sets which simplifies and improves some aspects of Peter Aczel's original presentation.

Non-well-founded sets have played an important rôle in the development of the whole thesis: by working within Jan Rutten and Jaco de Bakker's project 'non-well-founded sets and programming languages semantics', I have had the opportunity of distilling the mathematical foundations for the main contribution of the thesis, the introduction of the functorial approach to operational semantics.

Most of the research presented here has been conducted at the CWI, in Amsterdam. I can hardly imagine a better place to work on a thesis: the serene atmosphere, the international contacts, the superb library, the efficient organization, and the building itself, with quiet, balanced rooms, have made of this institute an ideal place for conducting pure research.

Jaco de Bakker's department at the CWI is part of EuroFOCS, the European institute in the logical foundations of computer science. This has offered me the opportunity of spending six, most profitable months at LFCS, Edinburgh, visiting Gordon Plotkin, one of whose many contributions to the theory of computer science has been the introduction of the *structural* approach to operational semantics.

When, in the early 80's, it was introduced, the novelty of structural operational semantics was that of bringing the mathematics of (structural) induction in the operational description of the behaviour of programming languages, providing a powerful formal tool for reasoning about programs. The present *functorial* approach can be seen as one step further in that direction: based on a suitable interplay between inductive and (dual) coinductive principles, it provides a mathematical definition and treatment of 'well-behaved' structural operational semantics.

The contact with Gordon Plotkin has been crucial both for this thesis and for my general development. Particularly vivid in my memory is the image of a beautiful February of two years ago, when, during some discussions with him, the blackboard looked like self-drawing; the last picture he drew, with "algebras over coalgebras", has been decisive for formulating the notion of functorial operational semantics.

The development of this notion, in Edinburgh, has been influenced by exciting discussions with Marcelo Fiore and Alex Simpson. More generally, Marcelo has been precious for my whole research activity.

Conceived, for the functorial part, in Edinburgh, this thesis has been written in Amsterdam. Thanks to very frequent reviewing sessions with my supervisors, Jaco de Bakker, Bart Jacobs, and Jan Rutten, the writing has rapidly converged to its final form, in a natural and serene rhythm.

Jaco, one of the pioneers of the mathematical approach to the semantics of programming languages which inspires this thesis, has granted me the room to develop the mathematics I felt most suitable, free from any prejudice. Almost without realizing it, I have written a much more thorough thesis than I had imagined, thanks to his gentle, but steady influence.

Jan, who brought me to the CWI, has collaborated to the development of coalgebraic methods in semantics which has been the basis for the research presented here. Bart, with his secure knowledge of category theory, has been a constant source of suggestions, corrections, and improvements. His limpid mind has always been available for discussions. Like Jan, he has shown great interest in and has collaborated to the foundational work on coalgebras.

The last step in the preparation of this thesis, the refereeing process, is due to Andy Pitts, who has been very sympathetic to the problems tackled and the methods used in this thesis. In this preface, I have used many expressions plundered from his precise summarizing words.

The 'palaestra' for my early scientific development has been the 'Amsterdam Concurrency Group' led by Jaco and including Marcello Bonsangue, Frank de Boer, Franck van Breugel, Arie de Bruin, Joost Kok, Erik de Vink, and Herbert Wiklicky. Nostalgically, I remember the first three-sessions talk I gave there, a promising winter of four years ago.

Marcello "kamergeno(o)t" Bonsangue, together with Franck room-mate in the beloved M335, has shared these early developments and my growing interest in category theory. He is one of the extraordinarily many Italians who, from Catuscia Palamidessi on, have been at the CWI over the years. One of the persons who are most 'responsible' for this Italian 'colonization' is Krzysztof Apt; he was also the supervisor of my "tesi di laurea" for the University of Pisa, in my 'prehistorical' time at the CWI.

Also at LFCS I have been surrounded by Italians or Italian speakers. One of them, Pietro 'everywhere' Di Gianantonio, has also been my colleague at the CWI and in the European SCIENCE project 'Mathematical Structures in Semantics of Concurrency'. This project has been an important forum for discussions to me; apart from the CWI, the sites involved have been the university of Koblenz (Lutz Priese), Mannheim (Mila Majster-Cederbaum), Pisa (Ugo Montanari), and Udine (Furio Honsell), and the IRISA-INRIA of Rennes, where, in particular, I have had

fruitful contacts with Eric Badouel and Philippe Darondeau.

At the CWI, I have enjoyed discussions with Fer-Jan de Vries, Tim Fernando, and Femke van Raamsdonk, the efficient secretarial support by Mieke Bruné and Marja Hegt, the technical support by the Computer Help Information Desk, and the outstanding library service. My visit to Edinburgh has been arranged thanks to George Cleland and Monika Lekuse's help at LFCS.

Most of the economic support for this thesis has been provided by the "Stichting Informatica Onderzoek in Nederland" of the Dutch organization for scientific research (NWO); my grant has been handled in a particularly friendly way by Richard Kellermann Deibel and Virginie Meijer-Mes. The remainder of the support has come from the SCIENCE project and from EuroFOCS.

I have tried to write this thesis in the most unassuming way, trying to communicate m-my p-personal experience of discovering, through elementary problems, the beauty and necessity of the universals of category theory, a discovery which has turned my mathematical activity into a "fröhliche Wissenschaft".

Daniele Turi – Amsterdam, April 1996

Contents

Introduction

"It is all very well to aim for a more 'abstract' and a 'cleaner' approach to semantics, but if the plan is to be any good, the operational aspects cannot be completely ignored. The reason is obvious: in the end the program still must be run on a machine – a machine which does not possess the benefit of 'abstract' human understanding, a machine that must operate with finite configurations. Therefore, a mathematical semantics, which will represent the first major segment of the complete, rigourous definition of a programming language, must lead naturally to an operational simulation of the abstract entities, which – if done properly – will establish the practicality of the language, and which is necessary for a full presentation."

Dana Scott, *Outline of a Mathematical Theory of Computation*

"Many modern programming languages are inconsistent with standard mathematical foundations. The task of finding sound interpretations for what it is that computer scientists do strikes this writer as, perhaps, the highest type of applied mathematics. It is akin to the process that has been going on throughout the 20th Century with respect to physics. The interaction between the mathematicians and the practitioners in each case has resulted in the growth of both subjects."

Peter Freyd, *Computer Science Contradicts Mathematics*
lecture at the Int'l Conf. on Category Theory
held in Como, Italy, July 1990 (see [Fre91])

The *operational semantics* of a programming language accounts for a formal description of the *behaviour* of the programs, specifying the way programs should be executed and the kind of behaviour which should be observable. The operational semantics is usually contrasted with the *mathematical* interpretation of the programs called *denotational semantics*.

This thesis presents a new mathematical approach to the semantics of programming languages aimed at bridging the gap between the operational and the denotational aspects of semantics. This is based on a suitable interplay between the standard *induction principle* which pervades modern mathematics, and the dual '*coinduction principle*' which has led to *non-standard* mathematical foundations.

In order to introduce coinduction as the dual of induction, it is convenient to move from the traditional presentation of induction in the language of *set theory* to a presentation in the language of *category theory*. The primitive notions of category theory are those of composition and equality of abstract functions called *arrows*, like the notions of membership and equality of those abstract collections called sets are the primitives notions of set theory. Now, every statement expressible in the language of category theory can be straightforwardly *dualized* by 'reversing the arrows'. (*Duality principle*.)

Induction. In set theory, mathematical induction is based on the notion of a *well-founded* relation, that is, a relation R such that, for every set x, there is *no* infinitely descending chain

$$\ldots R\, x_2\, R\, x_1\, R\, x_0 = x$$

For instance, one can perform induction on the set $N = \{0,\, 1 = s(0),\, 2 = s^2(0),\, \ldots\}$ of natural numbers by using the well-foundedness of the order relation on them

$$0 < s(0) < s^2(0) < \cdots < s^n(0) = n$$

as follows.

> *Recursion Theorem.* Given a set X, an element $e \in X$ and a function $g : X \to X$, there *exists* a *unique* function $f : N \to X$ from the set of natural numbers to the given set such that
>
> $$f(0) = e \qquad \text{and} \qquad f(s(n)) = g(f(n))$$
>
> for all numbers $n \in N$.

The value e of the function f at (the least element) 0 (wrt the order relation) is the 'base' of the induction and g defines the 'inductive step'.

The fact that standard mathematical constructions are inductive is mirrored by the common assumption that the axioms of set theory include the *axiom of foundation* which postulates that the set-membership relation '$\in$' is well-founded: for every set x, there exists no infinitely descending chain

$$\ldots \in x_2 \in x_1 \in x_0 = x$$

The axiom of foundation allows an inductive (idealized) construction of sets starting from the empty set (the base) and recursively applying the power-set operator mapping a set to the set of its subsets. The induction is on those generalized natural numbers which are the ordinal numbers.

In this thesis, an equivalent categorical formulation of the foundation axiom is given which allows for a straightforward dualization. This is best illustrated starting from the above recursion theorem:

The recursion theorem can be taken as the *definition* of natural numbers. That is, every set N with a distinguished element $0 \in N$ and a unary operation $s : N \to N$ such that the recursion theorem holds, is *isomorphic* to the natural numbers. (See, eg, [Mac86, Chapter 2].) As pointed out by Lawvere, the existence/uniqueness statement of the recursion theorem asserts the **universal property** characterizing the natural numbers: *initiality*. This property underlies induction, not only on the natural numbers, but in general.

Category Theory. The mathematical study of universal properties is called category theory. It is based on an abstract notion of function called *arrow*

$$f : X \to Y$$

which formally is a triple: name (f), domain (X), and codomain (Y).

A category is a collection of arrows with a *composition* operation '$\circ$' which obeys generalized monoidal laws: any two arrows $f : X \to Y$ and $g : Y \to Z$ which 'match' in the sense that the codomain of f is the same as the domain of g can be composed

$$\begin{tikzcd} X \arrow[r, "f"] \arrow[rr, bend right, "g \circ f"'] & Y \arrow[r, "g"] & Z \end{tikzcd}$$

to form the arrow $g \circ f : X \to Z$; the composition of arrows is associative, ie $f \circ (g \circ h) = (f \circ g) \circ h$; the domains and codomains of the arrows are called the *objects* of the category and for every object X there exists an *identity arrow* $\mathsf{id}_X : X \to X$ which is both a left and a right unit for the composition, ie $\mathsf{id}_Y \circ f = f = f \circ \mathsf{id}_X$.

The archetypal category is **Set**, having sets as objects and functions as arrows. However, it is very misleading (especially at the beginning!) to try and understand the *universals* of category theory in terms of **Set**.

The most elementary universal property which an object of a category can enjoy is initiality: an object X is initial in a category if, for every object Y of the category, there *exists* an arrow $f : X \to Y$ from X to Y and, moreover, this arrow is *unique*.

The basic way of understanding the natural numbers as an initial object is by regarding them as an object $\langle N, 0, s\rangle$ in the category having as objects triples $\langle X, e, t\rangle$, where X is a set with a distinguished element $e \in X$ and a function $t : X \to X$ on it. The arrows $f : \langle X, e, t\rangle \to \langle X', e', t'\rangle$ of the category are functions $f : X \to X'$ such that

$$f(e) = e' \qquad \text{and} \qquad f(t(x)) = t'(f(x))$$

(It is easy to verify that the above objects and arrows form a category with composition and identities as in **Set**.) Then the recursion theorem says exactly that the triple $\langle N, 0, s\rangle$ is initial in this category. (Notice that in the category **Set** the initial object is the trivial empty set.) Conversely, since initial objects, like all universals, are unique up to isomorphism, the initial object of this category defines the natural numbers up to isomorphism.

Next, a series of abstractions is necessary in order to generalize this specific form of initiality.

Firstly, notice that the element $e \in X$ of a set X can be written as a function from the one-element set $1 = \{*\}$ to the set X; that is, one can identify a function $e : 1 \to X$ from the one-element set 1 to a set X with its value $e(*) \in X$ at the unique element $*$ of 1. Then the recursion theorem amounts to having an object $1 \xrightarrow{0} N \xrightarrow{s} N$ such that for every object $1 \xrightarrow{e} X \xrightarrow{g} X$, there exists a unique function $f : N \to X$ with

$$f \circ 0 = e \qquad \text{and} \qquad f \circ s = g \circ f$$

Diagrammatically, using dashed arrows to denote arrows given by universal properties, one has that the following diagram commutes.

$$\begin{tikzcd}
1 \arrow[r, "0"] \arrow[d, equal] & N \arrow[r, "s"] \arrow[d, dashed, "f"] & N \arrow[d, dashed, "f"] \\
1 \arrow[r, "e"'] & X \arrow[r, "g"'] & X
\end{tikzcd}$$

Secondly, every pair of functions with the same codomain (thus, eg, $e : 1 \to X$ and $g : X \to X$) can be made into a single arrow with as domain the disjoint union of the domains. This holds in general in every category with *coproducts*: given two objects X and Y in a category, their coproduct, if it exists, is an object $X + Y$ with two arrows $\mathsf{inl}_X : X \to X + Y$ and $\mathsf{inl}_Y : Y \to X + Y$ which is *universal* in the sense that for every pair of arrows $f : X \to Z$ and $f : Y \to Z$ there *exists* a *unique* arrow $[f, g] : X + Y \to Z$, making the following diagram commute.

$$\begin{tikzcd}
X \arrow[r, "\mathsf{inl}_X"] \arrow[dr, "f"'] & X + Y \arrow[d, dashed, "{[f, g]}"] & Y \arrow[l, "\mathsf{inr}_Y"'] \arrow[dl, "g"] \\
& Z &
\end{tikzcd}$$

(The dual of the coproduct $X + Y$ is the product $X \times Y$: its *projections* $\mathsf{fst}_X : X \times Y \to X$ and $\mathsf{snd}_Y : X \times Y \to Y$ are universal among all pairs of arrows $f : Z \to X$ and $g : Z \to Y$.)

In **Set** the disjoint union, together with the corresponding injection functions, is a coproduct. Hence, one can write $[e, g] : 1 + X \to X$ instead of $1 \xrightarrow{e} X \xrightarrow{g} X$. Correspondingly, the initiality of the natural numbers can be expressed by saying that for every function $h : 1 + X \to X$ there exists a unique arrow $f : N \to X$ such that the following diagram commutes.

$$\begin{tikzcd}
1 + N \arrow[r, "1+f"] \arrow[d, "{[0,s]}"'] & 1 + X \arrow[d, "h"] \\
N \arrow[r, dashed, "f"'] & X
\end{tikzcd}$$

The arrow $1 + f : 1 + N \to 1 + X$ is defined by universality:

$$1 + f = [\mathsf{inl}_1 \circ \mathsf{id}_1,\ \mathsf{inr}_X \circ f] = [\mathsf{inl}_1,\ \mathsf{inr}_X \circ f] : 1 + N \to 1 + X$$

Thus the operation $X \mapsto 1 + X$ on objects extends to an operation $f \mapsto 1 + f$ on arrows: this defines a *functor* from **Set** to **Set**.

Functors are arrows between categories (regarded as objects!). A general criterion for forming a category from a collection of objects is to take as arrows the 'homomorphisms', that is, the morphisms which preserve the structure of the objects. Now, the structure of a category is given by composition and identities, and functors preserve it: a functor $F : \mathbf{C} \to \mathbf{D}$ from a category $\mathbf{C}$ to a category $\mathbf{D}$ maps every object X of $\mathbf{C}$ to an object FX of $\mathbf{D}$ and every arrow $f : X \to Y$ of $\mathbf{C}$ to an arrow $Ff : FX \to FY$ of $\mathbf{D}$ in such a way that

$$F(\mathsf{id}_X) = \mathsf{id}_{FX} \qquad \text{and} \qquad F(g \circ f) = Fg \circ Ff$$

The composition of functors can be then defined 'pointwise'.

Universal definitions are always functorial. For instance, given two functions $f : X \to Y$ and $g : Y \to Z$ one defines $1 + f : 1 + X \to 1 + Y$ by

$$\begin{tikzcd}
1 \arrow[r, "\mathsf{inl}_1"] \arrow[d, equal] & 1 + X \arrow[d, dashed, "{1+f = [\mathsf{inl}_1,\ \mathsf{inr}_Y \circ f]}"'] & X \arrow[l, "\mathsf{inr}_X"'] \arrow[d, "f"] \\
1 \arrow[r, "\mathsf{inl}_1"'] & 1 + Y & Y \arrow[l, "\mathsf{inr}_Y"]
\end{tikzcd}$$

and then $1 + (g \circ f)$ is, by uniqueness, necessarily equal to $(1 + g) \circ (1 + f)$.

Algebras and Coalgebras. The third step of abstraction is now to move from the above (endo) functor $FX = 1 + X$ on **Set** to arbitrary endofunctors $F : \mathbf{C} \to \mathbf{C}$ and, correspondingly, to consider initial objects in categories of structures $h : FX \to X$ rather than $h : 1 + X \to X$.

Given an endofunctor $F : \mathbf{C} \to \mathbf{C}$ on a category $\mathbf{C}$ one can form the category of F-*algebras* having as objects pairs $\langle X, h\rangle$ with X an object and $h : FX \to X$ an arrow of $\mathbf{C}$. An arrow $f : \langle X, h\rangle \to \langle X', h'\rangle$ between F-algebras is an arrow $f : X \to X'$ between their 'carriers' such that

$$\begin{tikzcd} FX \arrow[r, "Ff"] \arrow[d, "h"'] & FX' \arrow[d, "h'"] \\ X \arrow[r, "f"'] & X' \end{tikzcd}$$

commutes, that is, $f \circ h = h' \circ Ff$. Therefore, the natural numbers can also be understood as the initial algebra of the endofunctor $FX = 1 + X$ on $\mathbf{Set}$. Similarly, the axiom of foundation can be understood as postulating the initiality of an algebra as follows.

Form the *class* (ie large set) V of all sets, namely the *universe of sets*. This class is a (strict) fixed point $V = \mathcal{P}_S V$ of the operator $\mathcal{P}_S$ mapping a class (ie a possibly large set) to the class of all its (small) subsets. This operator can be extended to an endofunctor $\mathcal{P}_S : \mathbf{SET} \to \mathbf{SET}$ on the (superlarge!) category $\mathbf{SET}$ of classes and class-functions. Thus the identity function given by the equality $\mathcal{P}_S V = V$ can be seen as an algebra structure of this endofunctor.

Now, it is shown in this thesis that the axiom of foundation is equivalent to postulating that 'the universe $\mathcal{P}_S V = V$ is an initial $\mathcal{P}_S$-algebra'. This gives the formal link between initiality and (generalized) induction (on well-founded relations). Most importantly, in this form the foundation axiom is easily dualized:

The dual of the notion of initiality is the notion of *finality*: an object X is *final* (or *terminal*) in a category when *from* every object of the category there is a unique arrow *to* X. And the dual of the notion of an algebra of an endofunctor F on a category C is the notion of an F-*coalgebra*, that is, a pair $\langle X, k\rangle$ with X an object and $k : X \to FX$ an arrow of $\mathbf{C}$; the arrows $f : \langle X, k\rangle \to \langle X', k'\rangle$ between coalgebras are those arrows $f : X \to X'$ between their carriers such that

$$\begin{tikzcd} X \arrow[r, "f"] \arrow[d, "k"'] & X' \arrow[d, "k'"] \\ FX \arrow[r, "Ff"'] & FX' \end{tikzcd}$$

commutes, ie $Ff \circ k = k' \circ f$. Therefore, the dual of foundation amounts to postulating that 'the universe $V = \mathcal{P}_S V$ is a final $\mathcal{P}_S$-coalgebra', which, as shown in this thesis, is equivalent to Peter Aczel's '*anti-foundation axiom*' yielding *non-well-founded sets*.

Coinduction with non-well-founded sets.

> "The original stimulus for my own interest in the notion of a non-well-founded set came from a reading of the work of Robin Milner in connection with his development of a mathematical theory of concurrent processes. This topic in theoretical computer science is one of a number of such topics that are generating exciting new ideas and intuitions that are in need of suitable mathematical expression."
>
> Peter Aczel, *Non-Well-Founded Sets*

Aczel's theory of non-well-founded sets was driven by the quest for a set-theoretic foundation for the (abstract) semantics of Milner's *Calculus of Communicating Systems* (CCS). In CCS, the behaviour of a program t is given by the set

$$\{t \xrightarrow{a_i} t_i\}$$

of *transitions* $t \xrightarrow{a_i} t_i$ which the program can perform, producing an observable *action* a_i and becoming t_i. The *non-deterministic* nature of the calculus is expressed by the fact that a program t can *choose* among a set of transitions.

The meaning $[\![t]\!]$ of a program t should abstract from the name of the programs involved in the transitions and focus to the actions which can be performed, together with the choices which can be made. It should then be the following 'coinductively' defined set.

$$[\![t]\!]^{@} = \{<a, [\![t']\!]^{@}> \mid t \xrightarrow{a} t'\}$$

(The superscript '@' is used in this thesis to denote coinductive definitions in general; its dual is the superscript '#' used for inductive definitions.) Now, in general, the transition relation is not well-founded, since, for instance, cyclic programs $t \xrightarrow{a} t$ are allowed. Therefore, the above meaning $[\![t]\!]^{@}$ can be a non-well-founded set.

Traditionally, this 'problem' has been overcome by imposing either an order or a metric on the transition relation and then defining $[\![t]\!]^{@}$ as a suitable *limit*. (See, eg, [Win93] for the order-theoretic and [BV96] for the metric-theoretic approach.) Aczel, instead, chose to look for new foundations allowing for non-well-founded sets and then replaced the foundation axiom by the anti-foundation axiom [Acz88]. But one does not need to resort to non-standard foundations: as already clear in [Acz88], coinductive definitions can be founded on final coalgebras and these exist also in the standard category of ordinary sets (and in many other categories).

What the anti-foundation axiom gives is the non-standard fact that the greatest (strict) fixed point

$$gfp(F) = F(gfp(F))$$

of an endofunctor F on **SET** is a final F-coalgebra, provided F satisfies some mild conditions. This theorem [Acz88, "Special Final Coalgebra Theorem"] is the 'dual' of the standard fact (holding also without anti-foundation) that the least fixed points of most endofunctors on **SET** are initial algebras.

In particular, the special final coalgebra theorem holds for the endofunctor mapping a class X to the class $\mathcal{P}_S(A \times X)$ having as elements (small) sets of pairs $<a, x>$, with $a \in A$ and $x \in X$. Now, the behaviour of CCS programs can be seen as a coalgebra of this endofunctor by taking for A the set *Act* of actions performable by the programs, for X the set *Prog* of programs, and for coalgebra structure the function $[\![\text{-}]\!] : \mathit{Prog} \to \mathcal{P}_S(\mathit{Act} \times \mathit{Prog})$ defined for every program $t \in \mathit{Prog}$ as follows.

$$[\![t]\!] = \{<a, t'> \mid t \xrightarrow{a} t'\}$$

Then the function $[\![\text{-}]\!]^@$ mapping a program to its abstract meaning can be defined as the *coinductive extension* of this coalgebra structure, that is, as the unique coalgebra arrow from the coalgebra of programs to the greatest fixed point of the '*behaviour endofunctor*'

$$BX = \mathcal{P}_S(\mathit{Act} \times X)$$

which, by the special final coalgebra theorem, is a final coalgebra:

$$\begin{tikzcd}
\mathit{Prog} \arrow[r, dashed, "{[\![\text{-}]\!]^@}"] \arrow[d, "{[\![\text{-}]\!]}"'] & \mathit{gfp}(B) \arrow[d, equal] \\
B(\mathit{Prog}) \arrow[r, "{B([\![\text{-}]\!]^@)}"'] & B(\mathit{gfp}(B))
\end{tikzcd}$$

That is, for every program $t \in P$, $[\![t]\!]^@ = \{<a, [\![t']\!]^@> \mid t \xrightarrow{a} t'\}$.

The special final coalgebra theorem is stated in terms of the "Solution Lemma" [Acz88]. The final coalgebra presentation of anti-foundation introduced in this thesis makes the solution lemma (and its equivalence with anti-foundation) trivial. Correspondingly, the 'uniformity on maps' condition – which an endofunctor has to satisfy in order for the special final coalgebra theorem to hold – can be formulated in a more transparent way than in [Acz88].

Structural Operational Semantics. The operational semantics of CCS, that is, the definition of the transition relation between CCS programs, is given using Gordon Plotkin's *structural* approach to *operational semantics* [Plo81b]. In structural operational semantics both the programs and their behaviour are defined by induction on the basic program constructs – the *structure* of the programs. In particular, the behaviour of the programs is defined as the least transition relation closed under some conditional operational rules.

Since its inception, the structural approach has rapidly become the predominant approach to operational semantics. The two main reasons are that (i) it is *universal*, in the sense that all existing languages can be described this way, and (ii) it comes with a *structural induction principle* for reasoning about programs.

In this thesis, a mathematical theory of 'well-behaved' operational semantics is introduced which arises from a suitable interplay between the inductive (ie algebraic) aspects of the structural approach and the coinductive (ie coalgebraic) aspects present in Aczel's work on CCS.

Let us focus on the inductive aspects first. In the structural approach, programs are inductively defined in terms of some basic constructs $\sigma \in \Sigma$ from a *signature* Σ. Every signature can be seen as an endofunctor mapping a set X to the coproduct

$$\Sigma X = \coprod_{\sigma} X^{\mathrm{arity}(\sigma)}$$

indexed by the constructs σ of the language. The programs form then the (unique up to isomorphism) initial algebra of this endofunctor. In particular, by taking as constructs a constant (arity $= 0$) and a unary operator (arity $= 1$) one obtains the equivalence between the natural numbers (as inductively defined from zero and successor) and the initial algebra of the endofunctor $X \mapsto 1 + X$.

The initial Σ-algebra gives the set of *closed* programs, that is, programs without variables. In order to adjoin variables from a set *Var* it is sufficient to take the initial algebra of the endofunctor

$$X \mapsto \mathit{Var} + \Sigma X$$

(In particular, if *Var* is empty then one gets back the original Σ.) This initial algebra is also called the *free* Σ*-algebra over Var*.

It is worthwhile to make one more step of abstraction and introduce the notion of a *monad*.

Monads. Given a signature Σ, let $X \mapsto TX$ be the operation mapping a set X, regarded as a set *Var* of variables, to the free Σ-algebra over X (ie the initial $(X + \Sigma)$-algebra). By universality, this operation extends to an endofunctor $T : \mathbf{Set} \to \mathbf{Set}$ on **Set**. This endofunctor T comes equipped with two 'operations': the 'insertion-of-the-variables' $\eta_X : X \to TX$ and a 'multiplication' $\mu_X : T^2X \to TX$ for plugging programs into contexts. These operations are 'natural' in X and the triple $T{=}{<}T, \eta, \mu{>}$ is a *monad* on **Set**.

In general, a monad $T =< T, \eta, \mu >$ on a category $\mathbf{C}$ can be understood as a monoid in a category of endofunctors on $\mathbf{C}$, the 'operation' μ being the associative multiplication of the monoid and η its unit.

The notion of a monad is one of the most general mathematical notions. For instance, *every* algebraic theory, that is, every set of operations satisfying equational laws, can be seen as a monad; thus the monoid laws of the monad do subsume all possible algebraic laws! And algebraic theories are only a minor source of monads. In fact, every 'canonical' construction between two categories gives rise to a monad: the free Σ-algebra construction from **Set** to the category of Σ-algebras is one such canonical construction.

Next, there is a notion of a T-algebra which subsumes the notion of an algebra and, in particular, of a Σ-algebra. (Σ-algebras can be understood as algebras in which the operators (of the signature) are not subject to any law.) In particular, the monad T freely generated by a signature Σ is such that its category of algebras is isomorphic to the category of Σ-algebras. Therefore, the *syntax* of a programming language can be identified with a monad, the *syntactical monad* T freely generated by the program constructs Σ.

Now that the *syntax* is understood as a monad T and the *behaviour* as an endofunctor B whose coalgebras can be regarded as operational models (eg $BX = \mathcal{P}(Act \times X)$) the new notion of a 'functorial operational semantics' can be introduced.

Functorial Operational Semantics.

> A *functorial operational semantics* for a syntax T and a behaviour B is a monad Φ which 'lifts' the syntactical monad T to the coalgebras of the behaviour endofunctor B.

The *operational monad* Φ inherits the operations η and μ of the syntactical monad T; as a functor it maps a coalgebra structure $k : X \to BX$ to a structure $\Phi k : TX \to BTX$ which can be seen as the *operational model* on the set of programs TX given by the semantics Φ starting from the 'assumptions' $k : X \to BX$.

There are many possible liftings Φ of the same syntax T, each giving a different operational interpretation of the programs corresponding to T.

The novelty of this approach to operational semantics is that it captures in terms of abstract notions of *syntax* and *behaviour* the essence of 'well-behaved' operational semantics.

A condition which a well-behaved operational semantics should satisfy is *compositionality*: To every behaviour B there corresponds a notion of *observational equivalence* called B-bisimulation [AM89] (which for the behaviour $BX = \mathcal{P}(Act \times X)$ corresponds to Park and Milner's (strong) bisimulation – the finest notion of observational equivalence for transition relations); if this observational equivalence is a congruence wrt the constructs of the syntax, then the operational semantics is compositional. This means that programs with the same observable behaviour can be interchanged in any context without affecting the overall observable behaviour. Now, as shown in this thesis, *every* functorial operational semantics enjoys the property of being compositional.

Previous general results on compositional operational semantics stem from the theory of concurrent processes: the operational semantics is then assumed to be structural and the behaviour is fixed to be $BX = \mathcal{P}(Act \times X)$ (ie the notion of observational equivalence is (strong) bisimulation). The compositionality is ensured by imposing some restrictions on the *syntactic* format of the operational rules. Several formats have been proposed [dS85, BIM88, GV92, Gro93] and one of the most general is '*GSOS*' [BIM88], suitable to model most of the imperative or concurrent languages, including Milner's CCS.

Another result in this thesis is that every set $\mathcal{R}$ of GSOS rules defines an 'action' of the syntactical monad T on the composite endofunctor BT; in turn, this action *induces* a functorial operational semantics observationally equivalent to the operational semantics induced by the rules $\mathcal{R}$. Hence the syntactic restrictions making GSOS well-behaved are explained mathematically in terms of abstract notions of syntax and behaviour.

Denotational Semantics. A more general way of understanding the compositionality (and 'well-behaviour') of an operational semantics is in terms of 'denotational models'. Given a syntactical monad T, a *denotational model* for the corresponding language is simply a T-algebra; if the monad T is freely generated by a signature Σ, then this is the same as a Σ-algebra, that is, a set and a 'denotation' on this set of each program construct in Σ.

(More structured denotational models can be obtained by 'interpreting' the syntactical monad T in categories of structured objects like partial orders or metric spaces, rather than simply sets.)

The unique algebra arrow from the initial algebra of programs to the denotational model gives an inductive interpretation mapping programs to elements of the model. (This is the well-known *initial algebra semantics* approach of the 'ADJ group' – cf, eg, [GTW78].) This interpretation is by definition compositional, but one has to establish its *adequacy*:

> A denotational model is *adequate* wrt an operational semantics if it determines the operational behaviour of the programs up to observational equivalence.

It is at this point that the coalgebraic (ie coinductive) aspects of the functorial approach to operational semantics start playing a rôle: one of the pleasing properties of functorial operational semantics is that they (canonically) coinduce adequate denotational models. In order to understand this property, let us first look at coinduction in the category of ordinary (ie well-founded) sets.

Coinduction with ordinary sets. One of the properties of Aczel's coinductive semantics for CCS is that it maps two programs to the same set if and only if they are observationally equivalent:

$$[\![t_1]\!]^{@} = [\![t_2]\!]^{@} \iff t_1 \sim t_2$$

That is, the coinductive extension of the operational model $[\![\text{-}]\!] : \mathit{Prog} \to B(\mathit{Prog})$ does preserve B-bisimulation and, conversely, it can be 'pulled back' to form the largest B-bisimulation relation.

The above is a property which holds in general for every coinductive extension of coalgebras of endofunctors B preserving categorical (weak) *pullbacks*, where the endofunctor B can be on any category. Therefore:

> One does not need to work with non-well-founded sets: all one needs is that there exists a final coalgebra (hence coinduction) for B. In particular, one can work in the category of *ordinary* sets.

If anti-foundation is not assumed, then one cannot apply the special final coalgebra theorem in order to obtain final coalgebras from greatest (strict) fixed points. (While initial algebras can still be obtained as least fixed points.) There are several categorical methods to obtain final coalgebras though. One is a simple generalization of the standard greatest fixed point construction (à la Tarski) but it does not hold for endofunctors like the power-set functor.

There is also a problem of size: the structure of a final coalgebra is an isomorphism, that is, if $\widehat{B}$ is the carrier of a final B-coalgebra then its structure is an isomorphism

$$\varphi : \widehat{B} \cong B\widehat{B}$$

(This fact, in its dual version for initial algebras, is known as "Lambek's lemma".) Therefore, there is no final coalgebra for the endofunctor $BX = \mathcal{P}(Act \times X)$ or just $\mathcal{P}$, because there is no set isomorphic to the set of its subsets.

Aczel overcomes this problem by moving to the superlarge category of classes and considering the endofunctor $\mathcal{P}_S$ mapping a class to the class of its (*small*) subsets. Another solution, adopted here, consists in taking the *finite* power-set endofunctor mapping a set X to the set $\mathcal{P}_{fi}(X)$ of its finite subsets.

In general for establishing the existence of a final object in a category one can use categorical theorems like the "Special Adjoint Functor Theorem". As shown in [Bar93] this applies also to the coalgebras of endofunctors like the finite power-set and the corresponding behaviour

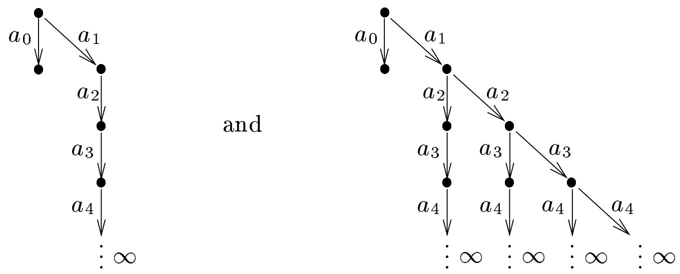
$$BX = \mathcal{P}_{fi}(Act \times X)$$

In particular, since CCS programs have only a *finite* degree of non-determinism, that is, each program can choose only among a finite set of transitions, the operational model of CCS is a coalgebra of this behaviour; its coinductive extension $[\![\text{-}]\!]^{@} : Prog \to \widehat{B}$ yields a semantics in the ordinary category of sets which is 'almost' the same as Aczel's one. The difference is in the fact that the final coalgebra structure is an isomorphism $\varphi : \widehat{B} \cong B\widehat{B}$ rather than an equality $\widehat{B} = B\widehat{B}$. Correspondingly, one has, for every program t,

$$[\![t]\!]^{@} = \varphi^{-1}\{<a, [\![t']\!]^{@}> \mid t \xrightarrow{a} t'\}$$

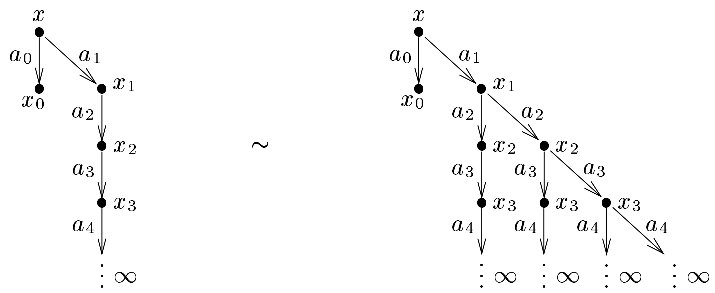
(In the sequel, for simplicity, the isomorphism φ is omitted.) This is the *final coalgebra semantics* corresponding to the operational model $[\![\text{-}]\!] : Prog \to B(Prog)$.

Concretely, the final coalgebra for the behaviour $BX = \mathcal{P}_{fi}(Act \times X)$ is the set of rooted, finitely branching trees, with branches labelled by the actions $a \in Act$, *quotiented* by the (largest) bisimulation relation. These (equivalence classes of) trees can be seen as the *abstract global behaviours* corresponding to $BX = \mathcal{P}_{fi}(Act \times X)$: the root of a tree τ is the starting point of an abstract computation c with behaviour B; the branching structure records the alternatives of the computation c and the labels of the branches are its observable actions; the quotient modulo bisimulation is needed in order to identify trees like



Notice that branches can be of infinite depth.

The fact that the nodes have no name reflects the abstractness of these global behaviours. This can be seen as a special case of the global behaviours observable with a set of 'states' X, which is obtained by labelling the nodes of the trees by elements x of X and, correspondingly, taking the quotient wrt a subtler form of bisimulation which takes into account the name of these states. For instance:



By putting $X = 1$, that is, by using the same label for all nodes, one gets back the abstract global behaviours.

Observational Comonads. The above operation $X \mapsto DX$ mapping a set X to the set of its global behaviours can be understood as a *cofree construction*, dual to the free construction of a monad from a signature. In general, given an endofunctor B on a category (with products) $\mathbf{C}$ a *cofree* B-coalgebra over an object X, if it exists, is the final coalgebra of the product endofunctor $X \times B$ mapping an object X' to the product $X \times BX'$. This generates a *comonad* $D{=}{<}D, \varepsilon, \delta{>}$, that is, an endofunctor $D : \mathbf{C} \to \mathbf{C}$ together with two 'operations' $\varepsilon_X : DX \to X$ and $\delta_X : DX \to D^2X$ 'natural' in X which make D a comonoid in a category of endofunctors on $\mathbf{C}$.

Comonads cofreely generated by behaviour endofunctors are called here *observational comonads*. Correspondingly, of the three conditions (implicitly) arisen so far which make of an endofunctor B a behaviour endofunctor, namely

1. the coalgebras of B have a computational interpretation as operational models,

2. B has a final coalgebra (hence coinduction),

3. B preserves weak pullbacks (hence coinduction can be 'pulled back' to B-bisimulation),

the second has to be generalized by requiring the existence of a final coalgebra of the product endofunctor $X \times B$ for every object X. Correspondingly, the category $\mathbf{C}$ should have finite products (including a final object 1). Since in every category $1 \times X \cong X$ holds, one has that the final coalgebra is isomorphic to the cofree coalgebra over 1.

As mentioned above, in the specific case of the behaviour $BX = \mathcal{P}_{\mathrm{fi}}(Act \times X)$, the value of the observational comonad D at a set X is a set of (equivalence classes of) rooted trees with nodes labelled by 'states' $x \in X$. The operations of the observational comonad $D {=} {<} D, \varepsilon, \delta {>}$ permit to visit these trees: the 'counit' ε is the operation which extracts the label of the root of a tree and the 'comultiplication' δ gives the remaining part of the tree.

One can form a category of *D-coalgebras* and, like for Σ-algebras and the algebras of the corresponding freely generated monad T, one can prove that if D is cofreely generated by an endofunctor B then this category is isomorphic to the category of B-coalgebras. Therefore, a functorial operational semantics can be seen as a lifting Φ of the syntactical monad T to the coalgebras of the observational comonad D. In this form, the notion of a functorial operational semantics can be readily dualized as follows.

Functorial Denotational Semantics.

> A *functorial denotational semantics* for a syntax T and a (global, observable) behaviour D is a comonad Ψ which 'lifts' the observational comonad D to the algebras of the syntactical monad T.

The *denotational comonad* Ψ inherits the operations ε and δ of the observational comonad D. In terms of Σ-algebras, the endofunctor Ψ maps a structure $h : \Sigma X \to X$ to a structure $\Psi h : \Sigma DX \to DX$ which can be seen as the *denotational model* on the set of global behaviours DX given by the semantics Ψ starting from the 'assumptions' $k : \Sigma X \to X$.

Operational is Denotational. Now, the abstract property showing that functorial operational semantics are well-behaved is that there is a one-to-one correspondence between operational monads Φ and denotational comonads Ψ (over the same syntax and behaviour). Symbolically:

$$
\begin{tikzcd}
 & \mathbf{C}_D \arrow[r, "\Phi"] \arrow[d, "U_D"'] & \mathbf{C}_D \arrow[d, "U_D"] & \\
\Phi \arrow[d, mapsto] & \mathbf{C} \arrow[r, "T"'] & \mathbf{C} & \Psi^{\#} \\
\Phi^{@} & \mathbf{C}^T \arrow[r, "\Psi"] \arrow[d, "U^T"'] & \mathbf{C}^T \arrow[d, "U^T"] & \Psi \arrow[u, mapsto] \\
 & \mathbf{C} \arrow[r, "D"'] & \mathbf{C} &
\end{tikzcd}
$$

The category $\mathbf{C}_D$ is the category of coalgebras of a comonad D over $\mathbf{C}$ and the 'forgetful' functor $U_D : \mathbf{C}_D \to \mathbf{C}$ forgets the coalgebra structure mapping a coalgebra to its carrier. The dual holds for $\mathbf{C}^T$ and $U^T : \mathbf{C}^T \to \mathbf{C}$.

The mapping $\Phi \mapsto \Phi^{@}$ is defined by coinduction. In particular, the value of the comonad $\Phi^{@}$ at the (trivial) final T-algebra is the coinductive extension of the coalgebra structure obtained by applying the given operational monad Φ to the final D-coalgebra. The resulting T-algebra is the 'canonical' denotational model coinduced by the operational semantics Φ.

The essence of the above coinductive construction was already presented in [RT94], but there the assumption was needed that observational equivalence be a congruence (hence compositionality had already to be known) and, in order to ensure this fact, the operational semantics was assumed to be à la GSOS. Instead, here the functoriality of Φ ensures that the construction can always take place. Moreover, the fact that the mapping $\Phi \mapsto \Phi^{@}$ is a bijection immediately gives that $\Phi^{@}$ is *adequate* wrt Φ, that is, one can recover the operational semantics from the denotational one. Compositionality becomes here a corollary.

The bijection 'operational $\longleftrightarrow$ denotational' can be used also in the reverse direction. The mapping $\Psi \mapsto \Psi^{\#}$ gives an *inductive* construction of *operational* models from denotational ones. This is a new principle which had been forecasted in [RT94]. It is used here to show that *basic process algebra* – the 'minimal' language corresponding to the behaviour $BX = \mathcal{P}_{\mathrm{fi}}(Act \times X)$ – is functorial. This is an important result because the proof given here that GSOS is functorial is based on the (mild) assumption that every set of GSOS rules embeds basic process algebra. Correspondingly, the syntactical monad is assumed to correspond to an algebra containing an associative, commutative, and absorptive binary operator of non-deterministic choice. (This is one example of the advantage of working with the T-algebras rather than with algebras of a signature.)

Φ-algebras are $\Phi^@$-coalgebras. Another way of understanding the above adequacy result is by considering the category of algebras of the operational monad Φ. It is shown in this thesis that the category of Φ-algebras is the same as the category of coalgebras of its coinduced denotational comonad $\Phi^@$. One can take this category as the category of *models of* Φ: its objects carry both a T-algebra and a D-coalgebra structure which are suitably related via Φ. (Thus a Φ-model carries both a denotational and an operational structure.) The arrows of the category are those which preserve both the algebraic and the coalgebraic structure.

The category of Φ-models has both an initial and a final object: the initial Φ-model is the initial algebra of closed programs corresponding to the syntactical monad T, together with the operational model obtained by applying Φ to the (trivial) initial D-coalgebra; dually, the final Φ-model is the final coalgebra of abstract global behaviours corresponding to the observational comonad D, together with the denotational model obtained by applying $\Phi^@$ to the (trivial) final T-algebra.

Now, the (both by initiality and finality) unique arrow from the initial to the final Φ-model is a mapping going from the closed program $T0$ to the abstract global behaviours $D1$ and it necessarily is both an initial algebra semantics and a final coalgebra semantics. This is the categorical formulation of adequacy.

Interestingly, if Φ is the operational monad corresponding to a set of GSOS rules, then the notion of a Φ-model cuts down to the notion of a *GSOS-model* independently introduced by Alex Simpson in [Sim95].

Adjunctions subsume induction and coinduction. It should be stressed that, categorically, induction and coinduction are just two instances of the same notion, namely the one of an *adjunction*:

If the forgetful functor mapping the algebras of an arbitrary monad T to their carriers has a *left adjoint*, then the T-algebras come with an induction principle; the monad T itself is defined by this adjunction. Dually, if the forgetful functor mapping the coalgebras of a comonad D to their carriers has a *right adjoint*, then the D-coalgebras come with a coinduction principle.

Every 'canonical' construction between two categories defines an adjunction and every adjunction defines both a monad and a comonad. It is in this sense that canonical constructions give rise to monads (and comonads).

Sets like recursive processes. Finally, one remark on the title of the part of this thesis dedicated to non-well-founded sets.

It is shown in this thesis that recursive programs can be seen as coalgebras having as carrier the set of variables involved in the recursion. As a consequence, no (explicit) binding operator (like the operator "fix" in GSOS) is needed and the solution of a recursive program is (a *recursive process*) defined by coinduction. This subsumes standard fixed point methods like least fixed points in categories of complete partial orders [Plo76] or unique fixed points in categories of complete metric spaces [Niv79, BZ82].

Now, the same method is used here to treat (and trivialize!) the "Solution Lemma" [Acz88] for defining non-well-founded sets as solution of recursive equations involving exclusively (variables and) well-founded sets.

Historical Notes. The study of adequate denotational models for structural operational semantics has been, from [BZ82] on, the central topic of Jaco de Bakker's Amsterdam school of semantics based on the use of metric spaces. (See [BR92, BV96] for overviews.) The present functorial approach harvests the fruits of that work.

The main mathematical tool available in (complete) metric spaces is "Banach's theorem" ensuring the existence of unique fixed points of 'contracting' functions. Like coalgebraic finality, Banach's theorem, especially in its higher-order form, can be used both for dealing with coinductive *definitions* and for *proving* adequacy results. (Cf [KR90].)

In particular, Banach's theorem is used in [Rut90] for coinductively deriving denotational models from structural operational semantics. The assumption is that the operational rules are 'well-behaved' in the sense that they are in (a sub-format of) the GSOS format [BIM88] and this implies that the coinduced models are adequate. (A precursor of this method is presented in [Bad87], which, in turn, has been inspired by [DG87].)

A considerable improvement of the above method is achieved in [Rut92] by treating the semantic domain of abstract global behaviours (ie the set of *processes*) as a transition system and subsequently applying the operational rules to it, that is, by treating "processes as terms". Coinduction is dealt there by means of non-well-founded sets and of the corresponding solution lemma; the operational rules are in the "tyft/tyxt" format of [GV92], a more general format than the positive GSOS used in [Rut90].

An explicit use of the finality of the greatest fixed point of the endofunctor $BX = \mathcal{P}_S(Act \times X)$ (under the anti-foundation axiom) is made in [Acz88] for coinductively defining a *denotational* model for CCS. That example has led the author of this thesis to try and understand the mathematics behind the "processes as terms" method in terms of an interplay between algebraic and coalgebraic aspects. The article [RT94] contains preliminary results in this sense, but the actual derivation of models, although formulated coalgebraically, still relies there on the use of 'well-behaved' structural operational rules à la GSOS and on regarding the final coalgebra (ie the abstract global behaviours) as a transition system.

The abstraction step from well-behaved transition systems to operational monads has come only after Gordon Plotkin's suggestion of working with algebras *over* coalgebras rather that with algebras *and* coalgebras: that has proved to be the extra 'dimension' needed for formulating the present functorial approach to operational semantics.

Algebraic Compactness. Another way of looking at initial algebras and final coalgebras of endofunctors F is as *data types*: the initial F-algebra is the *inductive* data type corresponding to the 'type constructor' F, while the final F-coalgebra is the *coinductive* one. For instance, the type constructor $FX = 1 + X$ yields, in **Set**, the natural numbers N as inductive data type and the 'extended natural numbers' $N \cup \{\infty\}$ as coinductive one.

Studies on coinductive types in **Set** date back at least to [AM80]. A more recent view, put forward by Peter Freyd in [Fre91], is that data types should be defined in *algebraically compact categories*, that is, in categories where endofunctors have both initial algebras and final coalgebras which, moreover, do *coincide* in the sense that they are 'canonically isomorphic'. (See also [Fre90, Fre92].)

The archetypal example of an algebraically compact category is the category **pCpo** of complete partial orders and partial 'Scott-continuous' functions: regarded as an 'order-enriched' category, it has as endofunctors the 'locally continuous' ones, which, as shown in [SP82], make it algebraically compact indeed. (See [Bar92] for more examples.)

Instead, algebraic compactness fails in the category of sets, no matter whether ordinary or non-well-founded sets are considered. The absence of algebraic compactness in **Set** motivated Peter Freyd's remark on the need for non-standard mathematical foundations in computer science quoted at the beginning of this introduction.

Algebraic compactness is one of the axioms of Fiore and Plotkin's *axiomatic domain theory* [FP92, FP94, Fio96] which aims at isolating the abstract properties which a category should satisfy for hosting interpretations of programming languages. In particular, the semantic domain of a language – in the present setting the final coalgebra of the behaviour – should 'live' in such a category, typically **pCpo**. In contrast, the operational model of a language should carry only the structure imposed by syntax and behaviour and thus live in a simpler category, typically **Set**. This raises the problem of how to extend/lift a functorial operational semantics from an unstructured category like **Set** to a category of domains like **pCpo**.

Towards a mathematical operational semantics

> "The motivation for trying to formulate a mathematical theory of computation is to give *mathematical semantics* for high-level computer languages. The word 'mathematical' is to be contrasted in this context with some such term as 'operational'."
>
> Dana Scott, *Outline of a Mathematical Theory of Computation*

The present functorial approach shows that 'operational' and 'mathematical' are no longer necessarily contrasting attributes for a semantics. This is achieved by defining operational semantics in terms of abstract, mathematical notions of syntax and behaviour. Yet, considerable work remains to be done before this conceptual achievement will be of any 'practical' relevance.

Firstly, the examples of behaviour considered here are all minor variations of the endofunctor $BX = \mathcal{P}_{\mathrm{fi}}(Act \times X)$, with (strong) bisimulation as the corresponding observational equivalence. Among the other behaviours which can be described functorially and will be treated in future work there are those for side effects, for probabilistic computation, for *trace equivalence*, and for applicative languages like the *untyped lambda calculus*.

The first two behaviours are similar to the one for bisimulation, while a treatment of trace equivalence and of the lambda calculus require, for different reasons, the ability of extending or lifting an operational monad from **Set** to a more structured category, namely **pCpo** for the lambda calculus [Plo85] and the category of semi-lattices and join-preserving functions for trace equivalence [HP79]. Preliminary results on a coalgebraic treatment of trace equivalence and of the lambda calculus are presented in [TJ93, RT94].

Secondly, a more refined notion of syntactical monads is needed in order to deal with typed terms and with higher-order terms as introduced, eg, by *variable binding* in the *lambda calculus* and in many imperative and concurrent languages. For typed terms one can easily adapt the above approach using multi-sorted algebras. (Categorically, it means to deal with a power of **Set**.) For higher-order terms the plan is to consider signatures on variable sets (*presheaves*) rather than simple sets. Correspondingly, one has for a function(al) not an arity but a list of numbers. The length of the list is the number of arguments; the i-th number is the number of variables the function(al) binds at its i-th argument. (This notion of signature is considered, for semantics, in [Acz80], and, for syntax, in [Plo90]. Associated ideas are the work on higher-order rewriting [Klo80], and the work on higher-order algebra [Mei92].)

Thirdly, the above adequacy result should be strengthened by dealing also with *non-termination*: when, like in the untyped lambda calculus, programs might not terminate, adequacy imposes further requirements. For example, by using partial functions for the denotational semantics, the interpretation of a term should be undefined if and only if it does not terminate. This property is hard to verify and much work has been devoted to introduce methods for simplifying this kind of proofs. (See, eg, [Pit94b].) Therefore, a 'meta' adequacy result would be of a great relevance. (A related point still to be investigated is whether there exist some extra conditions which make a functorial operational semantics *fully-abstract*, but this is much harder a result to obtain.)

Finally, the present functorial approach seems closely related to Eugenio Moggi's monadic approach to operational semantics [Mog91]. His examples of *computational monads* do all qualify as *behaviours* and it would be interesting to incorporate their extra monadic structure in this functorial framework. As a result, a general notion of operational semantics for computational monads and a corresponding adequacy theorem could be obtained.

Synopsis

This thesis is divided in five parts: the first four parts are devoted to the functorial approach to operational semantics, while Part V (*Sets like Recursive Processes*) is a new presentation of Peter Aczel's theory of non-well-founded sets.

In Part I, after some preliminaries, the definition of functorial operational semantics is introduced. As an example, a simple deterministic language is treated with $BX = 1 + Act \times X$ as behaviour. Final coalgebras and recursive programs are also treated.

In Part II, the general properties of functorial operational semantics are illustrated. In Section 6 it is shown that every operational monad coinduces an adequate denotational model. This construction is explained in Section 7 in terms of the notion of functorial denotational semantics, dual to the operational one: every operational monad Φ coinduces a denotational comonad $\Phi^{@}$. This is the *basic property* of the functorial approach to operational semantics.

Section 8 shows that the mapping $\Phi \mapsto \Phi^{@}$ is a bijection between operational monads and denotational comonads, which implies that $\Phi^{@}$ is always adequate wrt Φ. This adequacy result is rephrased in Section 9, where it is shown that the algebras of an operational monad Φ are the same as the coalgebras of its coinduced comonad $\Phi^{@}$. The category of Φ-algebras (alias $\Phi^{@}$-coalgebras) is then taken as the category of Φ-*models*, and the unique arrow from the initial to the final Φ-model is both the initial algebra and the final coalgebra semantics corresponding to Φ.

Part III is dedicated to the non-deterministic behaviour $BX = \check{\mathcal{P}}(1 + Act \times X)$. Correspondingly, the simple deterministic language used as example in the two previous parts can be enriched with a non-deterministic choice construct à la CCS. In Section 10, following [HP79, Plo81a], the (non-empty) finite power-set $\check{\mathcal{P}}$ is introduced as the semi-lattice monad. Next, a functorial *denotational* semantics is 'naturally' associated to the behaviour $BX = \check{\mathcal{P}}(1 + Act \times X)$ and its induced operational semantics is shown to be *basic process algebra* [BW90]. This is used in Section 11 to prove that GSOS is functorial, under the mild assumption that GSOS embeds basic process algebra.

In Section 12, the observational equivalences corresponding to (arbitrary) behaviours B are treated using the notion of a relation lifting to a 'B-bisimulation' introduced in [AM89], which, for $BX = \check{\mathcal{P}}(1 + Act \times X)$, cuts down to Park and Milner's notion of a bisimulation. If the endofunctor B preserves (weak) pullbacks, then every coinductive definition of type B can be 'pulled back' to a relation lifting to a B-bisimulation, which fact is useful to reason about coinductively defined entities. Here it is shown that, as a corollary of adequacy, for every functorial operational semantics, bisimulation (wrt to the behaviour B) is a congruence (wrt the syntax T).

Section 13 treats the construction of cofree coalgebras for the finite power-set functor $\mathcal{P}_{\mathrm{fi}}$ and for the behaviour $BX = \check{\mathcal{P}}(1 + Act \times X)$. It is based on material in [AM89] and [Bar93].

Part IV consists of a technical summary (with proofs) of the first three parts phrased in terms of adjunctions rather than in terms of induction and coinduction.

Basic Universal Constructions

Category theory is the mathematical study of *universal* entities: an entity x is universal among a family $\mathcal{F}$ of entities if all entities of $\mathcal{F}$ can be 'reduced' to x. Formally, this can be expressed in a very general form by considering the family of arrows determined by a functor $F : \mathbf{C} \to \mathbf{D}$ and an object Y of the codomain category $\mathbf{D}$ of F. The family of entities is the set

$$\mathcal{F} = \{f : FX \to Y \in \mathbf{D} \mid X \in \mathbf{C}\}$$

of *arrows from* F *to* Y. (Alternatively, the dual case of arrows *from* Y *to* F can also be considered.)

The universal among the arrows of $\mathcal{F}$ (if it exists!) is an arrow $\varepsilon_Y : FGY \to Y$ such that, for every $f : FX \to Y$, there *exists* a *unique* arrow $f^\flat : X \to GY$ such that f factorizes through ε_Y as follows:

$$f = \varepsilon_Y \circ F(f^\flat)$$

Diagrammatically:

$$\begin{tikzcd}
\mathbf{C} \arrow[r, "F"] & \mathbf{D} & \\
X \arrow[d, dashed, "f^\flat"'] & FX \arrow[d, "F(f^\flat)"'] \arrow[dr, "f"] & \\
GY & FGY \arrow[r, "\varepsilon_Y"'] & Y
\end{tikzcd}$$

The object GY is unique up to isomorphism and so is the arrow ε_Y (in a suitable sense).

Particularly interesting is the case when a universal arrow from F to Y exists for every object Y of $\mathbf{D}$: then, by universality, the operation $Y \mapsto GY$ extends to a functor $G : \mathbf{D} \to \mathbf{C}$ by putting, for every $k : Y \to Y'$ in $\mathbf{D}$,

$$\begin{tikzcd}
GY \arrow[d, dashed, "Gk\ ="', "(k \circ \varepsilon_X)^\flat"] & FGY \arrow[r, "\varepsilon_Y"] \arrow[d, "FGk"'] & Y \arrow[d, "k"] \\
GY' & FGY' \arrow[r, "\varepsilon_{Y'}"'] & Y'
\end{tikzcd}$$

Moreover, one can check that, in this case, the arrow

$$\eta_X = (\mathsf{id}_{FX})^\flat : X \to GFX$$

obtained by 'reducing' the identity on FX to ε_{FX}, is a universal arrow from X to G, for every object X of $\mathbf{C}$:

$$\begin{tikzcd} \mathbf{C} \arrow[r, shift left, "F"] & \mathbf{D} \arrow[l, shift left, "G"] \end{tikzcd}$$

$$\begin{tikzcd} X \arrow[r, "\eta_X"] \arrow[dr, "g"'] & GFX \arrow[d, "G(g^\sharp)"] & & FX \arrow[d, dashed, "g^\sharp"] \\ & GY & & Y \end{tikzcd}$$

Dually, a universal arrow from X to a functor $G : \mathbf{D} \to \mathbf{C}$ for every object X of $\mathbf{C}$, defines a functor $F : \mathbf{C} \to \mathbf{D}$ and a universal arrow from F to Y, for every Y in $\mathbf{C}$. There is thus a hidden symmetry behind the notion of a universal arrow, a symmetry which is captured by the notion of an 'adjunction'.

Formally, an *adjunction from a category* $\mathbf{C}$ *to a category* $\mathbf{D}$ is given by a pair of functors $F : \mathbf{C} \to \mathbf{D}$ and $G : \mathbf{D} \to \mathbf{C}$ in opposite direction and by a 'natural' bijection between the arrows of type $FX \to Y$ and those of type $X \to GY$, for every X in $\mathbf{C}$ and Y in $\mathbf{D}$:

$$\begin{array}{ccc} f & FX \longrightarrow Y & g^\sharp \\ \downarrow & \overline{\overline{\qquad\qquad\qquad\qquad}} & \uparrow \\ f^\flat & X \longrightarrow GY & g \end{array}$$

The *naturality* of the mapping $f \mapsto f^\flat$ amounts to the fact that it is 'well-behaved' wrt both pre- and post-composition; that is, for all arrows $h : X' \to X$ in $\mathbf{C}$ and $k : Y \to Y'$ in $\mathbf{D}$, the following two equations hold.

$$(f \circ Fh)^\flat = f^\flat \circ h \qquad (k \circ f)^\flat = Gk \circ f^\flat$$

By duality, this is equivalent to the following.

$$(g \circ h)^\sharp = g^\sharp \circ Fh \qquad (Gk \circ g)^\sharp = k \circ g^\sharp$$

One usually writes the above adjunction as

$$F \dashv G$$

and says that G is a *right adjoint* for F; dually, F is a *left adjoint* for G. Correspondingly, $f^\flat$ is the *right adjunct of* f and $g^\sharp$ is the *left adjunct of* g.

Now, if there exists a universal arrow $\eta_X : X \to GFX$ from every object X of a category $\mathbf{C}$ to a functor $G : \mathbf{D} \to \mathbf{C}$, then G has a left adjoint, the functor F which, by universality, extends the operation $X \mapsto FX$. (And the dual holds for universal arrows from F to the objects of $\mathbf{D}$.) Conversely, every adjunction determines two families of universal arrows

$$\{\eta_X = (\mathsf{id}_{FX})^\flat : X \to GFX\}_{X\in\mathbf{C}} \qquad \{\varepsilon_Y = (\mathsf{id}_{GY})^\sharp : FGY \to Y\}_{Y\in\mathbf{D}}$$

(See, eg, [Mac71, §IV.1, Theorems 1 and 2].)

The description of an adjunction in terms of universal arrows is *procedurally* very important for the actual 'construction' of adjunctions. Usually, one has a simple functor at hand, like an inclusion functor or a a functor forgetting some structure, and one investigates the problem of the existence of a right or left adjoint to it: if this problem can be solved then the result can be a complex construction. For instance, the left adjoint of the forgetful functor from a category of algebras to sets maps a set to the free algebra over it. (Adjoints, like all universals, are unique up to isomorphism, thus one can speak of *the* left adjoint of a functor.) The advantage is that a complex construction is reduced to the notion of an adjoint to a simple construction and, moreover, in this form, the same result can be understood in different categories. For instance, one can consider algebras over complete partial orders rather than over sets and the left adjoint to the corresponding forgetful functor gives the free algebras over cpos rather than over sets. Similarly, various topological completions like the one of metric spaces can all be understood as left adjoints of inclusion functors. In general, every 'canonical' construction *arises* from an adjunction.

The family $\{\varepsilon_Y : FGY \to Y\}_{Y\in\mathbf{D}}$ of universal arrows determined by an adjunction has the property that, for all arrows $k : Y \to Y'$ in $\mathbf{D}$, the following diagram commutes.

$$\begin{tikzcd} FGY \arrow[r, "\varepsilon_Y"] \arrow[d, "FGk"'] & Y \arrow[d, "k"] \\ FGY' \arrow[r, "\varepsilon_{Y'}"'] & Y' \end{tikzcd}$$

(And similarly for the family $\{\eta_X : X \to GFX\}_{X\in\mathbf{C}}$.) This gives a 'natural transformation' from the composite functor FG on $\mathbf{D}$ to the identity functor $I_{\mathbf{D}}$.

In general, given two functors $F_1, F_2 : \mathbf{E} \to \mathbf{D}$, a *natural transformation*

$$\vartheta : F_1 \Rightarrow F_2$$

from F_1 to F_2 is a family $\{\vartheta_Y : F_1X \to F_2X \in \mathbf{D} \mid X \in \mathbf{E}\}$ of arrows of $\mathbf{D}$ indexed by the objects of $\mathbf{E}$ such that, for every arrow $f : X \to X'$ in $\mathbf{E}$ the square in the following diagram commutes.

$$
\begin{tikzcd}
Y \arrow[d, "f"'] & F_1Y \arrow[r, "\vartheta_Y"] \arrow[d, "F_1f"'] & F_2Y \arrow[d, "F_2f"] \\
Y' & F_1Y' \arrow[r, "\vartheta_{Y'}"'] & F_2Y'
\end{tikzcd}
$$

For every two categories $\mathbf{D}$ and $\mathbf{E}$ one can form the *functor category* $\mathbf{D}^{\mathbf{E}}$ having as objects the functors from $\mathbf{E}$ to $\mathbf{D}$ and as arrows the natural transformations between them. Identities and composition are obtained 'pointwise'. Thus: natural transformations are arrows between functors, which, in turn, are arrows between categories.

One usually omits the subscript under the identity functors and writes

$$\eta : I \Rightarrow GF \qquad \text{and} \qquad \varepsilon : FG \Rightarrow I$$

for the two natural transformations defined by an adjunction $F \dashv G$; these are the *unit* and the *counit* of the adjunction, respectively.

Initial and final objects can be described in terms of adjunctions as follows. Consider the trivial category $\mathbf{1}$ with only one object and one (identity) arrow. From every category $\mathbf{C}$ there is a unique functor

$$\mathbf{C} \to \mathbf{1}$$

to $\mathbf{1}$. Now, this functor has a left adjoint if and only if $\mathbf{C}$ has an initial object: this left adjoint maps the unique object of $\mathbf{1}$ to the initial object of $\mathbf{C}$; the counit of the adjunction at an object X of $\mathbf{C}$ gives the unique arrow from the initial object to X. Dually, the functor $\mathbf{C} \to \mathbf{1}$ has a right adjoint if and only if $\mathbf{C}$ has a final object and the unit of the adjunction gives the unique arrows to this final object.

Also coproducts and products can be described in terms of adjunctions. Consider the *product category* $\mathbf{C} \times \mathbf{C}$ having as objects and arrows pairs $<X, X'>$ of objects and pairs $<f, f'>$ of arrows of $\mathbf{C}$, with componentwise composition. There is a *diagonal functor*

$$\Delta : \mathbf{C} \to \mathbf{C} \times \mathbf{C} \qquad X \mapsto <X, X> \qquad f \mapsto <f, f>$$

'duplicating' the objects and the arrows of $\mathbf{C}$. This diagonal functor has a left adjoint if and only if $\mathbf{C}$ has (binary) coproducts; this left adjoint maps a pair $<X, Y>$ of objects of $\mathbf{C}$ to their *binary coproduct* $X + Y$ and the value of the unit at $<X, Y>$ is the corresponding pair of *injections* $<\mathsf{inl}_X, \mathsf{inr}_Y>$. Dually, the right adjoint, if it exists, gives *binary products* and the counit gives the corresponding *projections*.

The above binary product and coproduct adjunctions are instances of the following. Consider an arbitrary *small category* J, that is, a category with a (small) set of objects and a (small) set of arrows. (Counterexample: **Set** is not small.) Next, take the functor category

$$\mathbf{C}^J$$

having as objects the functors from J to $\mathbf{C}$ and as arrows the natural transformations between them. By putting J in $\mathbf{C}^J$ equal to the empty category $\mathbf{0}$ with no objects one obtains a category isomorphic to $\mathbf{1}$; similarly, by putting J equal to the category

$$\cdot \;\; \cdot$$

with two objects and no arrows other than the identities, one obtains a category isomorphic to $\mathbf{C} \times \mathbf{C}$:

$$\mathbf{C}^{\mathbf{0}} \cong \mathbf{1} \qquad \mathbf{C}^{\cdot\;\cdot} \cong \mathbf{C} \times \mathbf{C}$$

Correspondingly, the two functors $\mathbf{C} \to \mathbf{1}$ and $\Delta : \mathbf{C} \to \mathbf{C} \times \mathbf{C}$ can be seen as instances of a general notion of a diagonal functor

$$\Delta : \mathbf{C} \to \mathbf{C}^J$$

This diagonal functor maps an object X of $\mathbf{C}$ to a functor from J to $\mathbf{C}$ which, in turn, maps every object of J to X and all arrows of J to the identity on X. The left adjoint to this Δ give the 'colimits' of functors $D : J \to \mathbf{C}$ and the right adjoint gives the 'limits'. Thus initial objects and coproducts on the one hand and final objects and products on the other hand are, respectively, special cases of colimits and limits, which are the most common form of universals.

As an example, consider the category J with three objects and, apart from the identities, two arrows connecting one object to the other two:

$$\cdot \leftarrow \cdot \rightarrow \cdot$$

A functor $D : J \to \mathbf{C}$ from J to $\mathbf{C}$ can be seen as a *diagram* D in $\mathbf{C}$ of 'shape' J:

$$\begin{tikzcd} D: & Y_1 & Y_0 \arrow[l, "f"'] \arrow[r, "g"] & Y_2 \end{tikzcd}$$

A natural transformation $\vartheta : D \Rightarrow \Delta X$ from such a diagram D to the constant diagram $\Delta X : J \to \mathbf{C}$ obtained by applying Δ to an object X of $\mathbf{C}$

$$\begin{tikzcd} D: & Y_1 \arrow[d, "\vartheta_1"'] & Y_0 \arrow[l, "f"'] \arrow[r, "g"] \arrow[d, "\vartheta_0"'] & Y_2 \arrow[d, "\vartheta_2"] \\ \Delta X: & X \arrow[r, equal] & X \arrow[r, equal] & X \end{tikzcd}$$

can be collapsed into a '*cocone*' over D having X as 'vertex':

$$\begin{tikzcd} D: & Y_1 \arrow[dr, "\vartheta_1"'] & Y_0 \arrow[l, "f"'] \arrow[r, "g"] \arrow[d, "\vartheta_0"'] & Y_2 \arrow[dl, "\vartheta_2"] \\ & & X & \end{tikzcd}$$

that is, a family of arrows from the objects of the diagram D to X making everything in sight commute. (Notice that the middle arrow $\nu_0 : Y_0 \to X$ is superfluous because it factorizes (both) as $\nu_1 \circ f$ (and as $\nu_2 \circ g$).)

The *colimit of the diagram* D is then the universal cocone over D, that is, a cocone $\nu : D \Rightarrow \mathrm{Colim}D$ such that every cocone over D factorizes uniquely through it:

$$\begin{tikzcd}
D: & Y_1 \arrow[dr, "\nu_1"] \arrow[ddr, bend right, "\vartheta_1"'] & Y_0 \arrow[l, "f"'] \arrow[r, "g"] & Y_2 \arrow[dl, "\nu_2"'] \arrow[ddl, bend left, "\vartheta_2"] \\
& & \mathrm{Colim}\, D \arrow[d, dashed] & \\
& & X &
\end{tikzcd}$$

The existence and uniqueness of the '*mediating arrow*' from the colimit of a diagram D to the vertex X of any cocone over D expresses the universal property of the colimit.

In general, the left adjoint of the diagonal functor $\Delta : \mathbf{C} \to \mathbf{C}^J$, if it exists, maps diagrams of shape J to their colimit object; the unit of the adjunction gives the corresponding (universal) colimiting cocone.

The study of colimits can be reduced to the study of initial objects and '*pushouts*', the latter being colimits of diagrams of shape $J = \cdot \leftarrow \cdot \rightarrow \cdot$. Indeed, the colimit of any (small) diagram can be expressed in terms of combinations of (generalized) pushouts and initial objects. For instance, the coproduct $X + Y$ is isomorphic to the pushout of the diagram

$$\begin{tikzcd}
X & 0 \arrow[l, dashed] \arrow[r, dashed] & Y
\end{tikzcd}$$

where 0 is the initial object. Alternatively, (small) colimits can also be described in terms of (generalized) coproducts and '*coequalizers*', the latter being colimits of diagrams of shape

$$J = \cdot \rightrightarrows \cdot$$

A generalized coproduct is obtained by generalizing the two objects and no arrows category $J = \cdot\ \cdot$ to a category with a (small) set I of objects and no arrows. One writes then

$$\coprod_I X_i$$

for the corresponding coproduct. (And, similarly, binary pushouts can be generalized by taking (small) sets of arrows with the same domain.)

By duality, *limits* are right adjoints to diagonal functors and the counit gives the *limiting cones* over diagrams $D : J \to \mathbf{C}$, that is, the universal among the cones

$\nu : \Delta X \to D$. Products are limits with $J = \cdot\ \cdot$, while the dual of coequalizers and pushouts are limits of

$$J = \cdot \rightrightarrows \cdot$$

and

$$J = \cdot \to \cdot \leftarrow \cdot$$

and are called *equalizers* and *pullbacks*, respectively. All limits can be described with products and equalizers only, as well as with final objects and pullbacks only.

Notice that equalizers are 'left-cancellable' in the sense that, given an equalizer $m : Y \to Z$ and two parallel arrrows $f, g : X \to Y$, if $m \circ f = m \circ g$ then $f = g$; in general, left-cancellable arrows are called *monic arrows*. Dually, coequalizers are *epi*, ie 'right-cancellable'. In **Set** the epi and the monic arrows are the surjective and the injective functions, respectively.

Some final notational remarks. The (standard) notation for pullbacks and pushouts is

$$\begin{tikzcd}
\cdot \arrow[r] \arrow[d] \arrow[dr, phantom, very near start, "\lrcorner"] & Y_2 \arrow[d, "g"] \\
Y_1 \arrow[r, "f"'] & Y_0
\end{tikzcd}
\qquad \text{and} \qquad
\begin{tikzcd}
Y_0 \arrow[r, "g"] \arrow[d, "f"'] & Y_2 \arrow[d] \\
Y_1 \arrow[r] & \cdot \arrow[ul, phantom, very near start, "\ulcorner"]
\end{tikzcd}$$

respectively. Also, it is useful to introduce a special (non-standard) notation for the injection arrows $\mathsf{inl}_X : X \to X + Y$ and $\mathsf{inr}_Y : Y \to X + Y$ into a coproduct, namely by adding a triangle to their 'tails':

$$\begin{tikzcd}
X \arrow[r, tail] \arrow[dr, "f"'] & X + Y \arrow[d, dashed, "{[f, g]}"] & Y \arrow[l, tail'] \arrow[dl, "g"] \\
& Z &
\end{tikzcd}$$

Thus the above '*copair*' $[f, g] : X + Y \to Z$ of $f : X \to Z$ and $g : Y \to Z$ is the universal mediating arrow from the coproduct $X + Y$ to the vertex Z of the cocone formed by f and g over X and Y.

Notes. The standard textbook of category theory is [Mac71], whose first six chapters include the basic category theory used in this thesis; a useful summary (with examples and exercises) of those chapters can be found in Part 0 of [LS86].

For an alternative, vivid presentation of category theory see [FS90]. Computer scientists might want to consult also [Poi92] and [Cro93].

For the philosophical import of category theory (and of the notion of adjointness in particular) one can consult [Law69] and other Lawvere's writings, which are rich of stimulating connections between disparate fields.

I

1 Initial Algebras, Induction and Program Syntax

The *syntax* of a programming language is usually defined by *induction* on some basic constructs $\sigma \in \Sigma$. Formally, Σ is a *signature* and the syntax is the initial Σ-algebra. Equivalently, the signature defines an endofunctor with action $X \mapsto \coprod_{\sigma \in \Sigma} X^{\mathrm{ar}(\sigma)}$, whose algebras are the same as the algebras of the signature. This leads to the standard categorical construction of initial Σ-algebras as suitable ω-colimits.

Consider, as an example, a simple imperative language whose constructs are some primitive *actions* $a \in \mathsf{Act}$, a *sequential composition* operator ' ; ', and an 'inert' program nil. Correspondingly, the (single-typed) **signature** Σ of the above language is given by a set $\mathsf{Act} + 1$ of constants (ie operators of arity 0) and an operator of arity two.

The **programs** or **terms** t induced by the above signature Σ and some variables $x \in X$ are given by the grammar:

$$t ::= x \mid a \mid \mathsf{nil} \mid (t \,;\, t)$$

Denote this set of programs by TX. In particular, for $X = 0$, ie the empty set, the set $T0$ gives the **closed terms** of the language:

$$t ::= a \mid \mathsf{nil} \mid (t \,;\, t)$$

An alternative way of describing the set $T0$ of closed terms is as the carrier of the initial algebra of the signature Σ, that is, the initial object in the category of Σ-algebras, where Σ is the above signature. In general, given a signature Σ, the category of Σ-algebras has as objects pairs $\langle X, h\rangle$, where the **carrier** X is a set, and the **structure** h is a function interpreting each operator σ in the signature as a function $h(\sigma) : X^{\mathrm{arity}(\sigma)} \to X$. An arrow $f : \langle X, h\rangle \to \langle Y, k\rangle$ in this category is a function $f : X \to Y$ between the underlying sets such that, for every operator σ in the signature, the following diagram commutes.

$$\begin{tikzcd}
X^{\mathrm{ar}(\sigma)} \arrow[r, "f^{\mathrm{ar}(\sigma)}"] \arrow[d, "h\sigma"'] & Y^{\mathrm{ar}(\sigma)} \arrow[d, "k\sigma"] \\
X \arrow[r, "f"'] & Y
\end{tikzcd}$$

That is,
$$f(h\sigma)(x_1,\ldots,x_{\mathrm{ar}(\sigma)}) = (k\sigma)(fx_1,\ldots,fx_{\mathrm{ar}(\sigma)})$$
Notice that if the arity of an operator σ is zero, then $X^{\mathrm{ar}(\sigma)}$ is simply 1, the singleton set. The corresponding function $h\sigma : 1 \to X$ maps $*$, the unique element of 1, into an element of X. This gives the interpretation of a constant σ in the algebra.

For any signature Σ, the initial algebra always exists. It is the *term algebra* having as carrier the set $T0$ of closed terms over the signature and as algebra structure the evident one which maps, for every operator σ, a tuple $(t_1,\ldots,t_{\mathrm{ar}(\sigma)})$ of terms into the term $\sigma(t_1,\ldots,t_{\mathrm{ar}(\sigma)})$. Indeed, given any Σ-algebra $\langle X,h\rangle$, there is a unique arrow from the term algebra into $\langle X,h\rangle$, namely the function, say,
$$h^{\#} : T0 \to X$$
inductively defined as follows.
$$h^{\#}(\sigma(t_1,\ldots,t_{\mathrm{ar}(\sigma)})) = (h\sigma)(h^{\#}t_1,\ldots,h^{\#}t_{\mathrm{ar}(\sigma)})$$
Notice that the term algebra is initial also in the category of *partial* Σ-algebras, that is, algebras where the operators of the signature might be interpreted not only as total but also as partial functions.

A more compact way of describing the category of Σ-algebras is by taking the coproduct $\coprod_{\sigma\in\Sigma} X^{\mathrm{ar}(\sigma)}$, that is, the disjoint union of the domains of the operations. More formally, every signature Σ can be seen as a functor $\Sigma : \mathbf{Set} \to \mathbf{Set}$ (thus an **endofunctor** on **Set**) defined on objects as follows.
$$X \longmapsto \coprod_{\sigma\in\Sigma} X^{\mathrm{ar}(\sigma)}$$
For example, the endofunctor corresponding to the above signature $\Sigma = \mathsf{Act} \cup \{\mathsf{nil},\ ;\ \}$ is
$$\Sigma X \ = \ 1 + (\coprod_{Act} 1) + X\times X \ \cong \ 1 + \mathsf{Act} + X\times X$$
The category of algebras of a signature is then an instance of the following more general notion.

> Let $\Sigma : \mathbf{Set} \to \mathbf{Set}$ be *any* endofunctor on **Set**. The **category of Σ-algebras**, denoted by $\mathbf{Set}^{\Sigma}$, has as objects pairs $\langle X,h\rangle$, with X a set and $h : \Sigma X \to X$ a function. The arrows of the category are functions between the underlying sets preserving the algebra structure, that is, making the following diagram commute.

$$\begin{tikzcd} \Sigma X \arrow[r, "\Sigma f"] \arrow[d, "h"'] & \Sigma Y \arrow[d, "k"] \\ X \arrow[r, "f"'] & Y \end{tikzcd}$$

That is,

$$f \circ h = k \circ \Sigma f$$

Even more generality can be achieved by considering also algebras of endofunctors on categories $\mathbf{C}$ other than $\mathbf{Set}$. For instance, since any endofunctor corresponding to a signature Σ extends to the category $\mathbf{pSet}$ of sets and partial functions, the category $\mathbf{pSet}^{\Sigma}$ can be considered: this is the same as the category of *partial* Σ-algebras mentioned above.

The initial object in the category of algebras of an arbitrary endofunctor Σ, ie the **initial Σ-algebra**, does not always exists, but if it does, then its structure is an isomorphism:

> **Initial algebras are isomorphisms.** (*Lambek's Lemma.*) Let $\langle \overline{\Sigma}, \psi \rangle$ be the initial algebra of an arbitrary endofunctor Σ. Then the algebra structure $\psi : \Sigma\overline{\Sigma} \to \overline{\Sigma}$ is always an isomorphism
>
> $$\psi : \Sigma\overline{\Sigma} \cong \overline{\Sigma} \qquad \text{(initial } \Sigma\text{-algebra)}$$
>
> (To prove this notice that the initial algebra structure ψ is also a Σ -algebra arrow from $\langle \Sigma\overline{\Sigma}, \Sigma\psi \rangle$ to $\langle \overline{\Sigma}, \psi \rangle$.)

As mentioned in the introduction, initial algebras give a very useful *induction principle*. Indeed, every algebra structure $h : \Sigma X \to X$ of an arbitrary endofunctor Σ with initial algebra $\Sigma\overline{\Sigma} \cong \overline{\Sigma}$ can be inductively extended to an arrow $h^{\#} : \overline{\Sigma} \to X$ by taking the unique algebra arrow from the initial algebra to the algebra $\langle X, h \rangle$:

Inductive Extension

$$\begin{tikzcd}
\Sigma\overline{\Sigma} \arrow[r, "\Sigma h^{\#}"] \arrow[d, "\cong"'] & \Sigma X \arrow[d, "h"] \\
\overline{\Sigma} \arrow[r, dashed, "h^{\#}"'] & X
\end{tikzcd}$$

Notice this is a definition which holds in any category of algebras, thus, for instance, also for *partial* Σ-algebras.

Next, consider the construction of initial algebras. In the general setting where the endofunctor Σ might not stem from a signature, the initial Σ-algebra does not always arise from an inductive construction and might even fail to exist. But for the so-called $\boldsymbol{\omega}$-cocontinuous endofunctors, like those corresponding to signatures, the construction of the initial algebra is inductive indeed. Here $\boldsymbol{\omega}$ is the category having natural numbers as objects and arrows $n \to m$ iff $n \leq m$; that is, $\boldsymbol{\omega} = \{0 \to 1 \to 2 \to \cdots\}$. An **$\boldsymbol{\omega}$-cocontinuous functor** $F : \mathbf{C} \to \mathbf{D}$ is then a functor which preserves the colimits of functors $J : \boldsymbol{\omega} \to \mathbf{C}$, that is, $F\mathrm{Colim}J \cong \mathrm{Colim}FJ$. (The categories $\mathbf{C}$ and $\mathbf{D}$ are thus supposed to have these colimits.) Notice that a functor $J : \boldsymbol{\omega} \to \mathbf{C}$ is a diagram in $\mathbf{C}$ of the form $\{C_0 \overset{f_0}{\to} C_1 \overset{f_1}{\to} C_2 \overset{f_2}{\to} \cdots\}$.

The construction of the initial algebra of an $\boldsymbol{\omega}$-cocontinuous endofunctor is the functorial generalization of the least fixed point construction of an endofunction f in a partial order, namely as the least upper bound $\bigsqcup_{n<\omega} f^n\bot$. (This works if the partial order has a least element $\bot$ and the desired lub, and the function preserves that lub.) A partial order is a category with at most one arrow from one object to another. For such a category, the initial object is the least element, an endofunctor is a monotone endofunction, and $\boldsymbol{\omega}$-cocompleteness amounts to chain-completeness, ie, to the existence of least upper bounds of ω-chains. An $\boldsymbol{\omega}$-cocontinuous functor is thus a monotone function which preserves lubs of ω-chains. Finally, an algebra is a pre-fixed point $fx \leq x$ and the initial algebra is the least (pre-)fixed point.

Let Σ be an $\boldsymbol{\omega}$-cocontinuous endofunctor on $\mathbf{Set}$. Consider the unique function, say $0_{\Sigma 0}$, from the initial object in $\mathbf{Set}$ (the empty set – denoted by 0) to the set $\Sigma 0$. Next, consider the diagram D obtained by the iterative application of the endofunctor Σ to the initial function $0_{\Sigma 0}$; that is, for every n in ω, map the arrow $n \to n+1$ of $\boldsymbol{\omega}$ into $\Sigma^n 0_{\Sigma 0}$:

$$0 \xrightarrow{0_{\Sigma 0}} \Sigma 0 \xrightarrow{\Sigma 0_{\Sigma 0}} \Sigma^2 0 \xrightarrow{\Sigma^2 0_{\Sigma 0}} \cdots$$

Let Σ^ω be the colimit of this diagram D. Then, since the endofunctor Σ is $\boldsymbol{\omega}$-cocontinuous, $\Sigma\Sigma^\omega$ is the colimit of the diagram ΣD (which is simply D without the first arrow). Next, consider the colimiting cocone $\nu : D \Rightarrow \Sigma^\omega$:

$$\begin{tikzcd}
0 \arrow[r, "0_{\Sigma 0}"] \arrow[rrrr, bend right=50, "\nu_0"'] & \Sigma 0 \arrow[r, "\Sigma 0_{\Sigma 0}"] \arrow[rrr, bend right=40, "\nu_1"'] & \Sigma^2 0 \arrow[r, "\Sigma^2 0_{\Sigma 0}"] \arrow[rr, bend right=30, "\nu_2"'] & \cdots & \Sigma^\omega
\end{tikzcd}$$

Without the first component ν_0 this is also a cocone from ΣD to Σ^ω. Then:

> In the above construction, the mediating arrow from the colimit $\Sigma\Sigma^\omega$ of ΣD into Σ^ω gives the initial Σ-algebra structure. This can be proved by noticing that, for any algebra Σ-algebra $\langle X, h\rangle$, a cocone from D to X can be obtained as illustrated in the diagram below and then the inductive extension of the algebra structure $h : \Sigma X \to X$ is given by the

corresponding mediating arrow.

$$
\begin{tikzcd}
0 \arrow[r, "0_{\Sigma 0}"] \arrow[d, "0_X"'] & \Sigma 0 \arrow[r, "\Sigma 0_{\Sigma 0}"] \arrow[d, "\Sigma 0_X"'] & \Sigma^2 0 \arrow[r, "\Sigma^2 0_{\Sigma 0}"] \arrow[d, "\Sigma^2 0_X"'] & \cdots \\
X & \Sigma X \arrow[l, "h"] & \Sigma^2 X \arrow[l, "\Sigma h"] & \cdots \arrow[l, "\Sigma^2 h"]
\end{tikzcd}
$$

(This is the "Basic Lemma" from [SP82].)

Notice that the above construction applies to *any* category with initial object and $\boldsymbol{\omega}$-colimits. Thus, for instance, it can be applied also to $\boldsymbol{\omega}$-cocontinuous endofunctors on **pSet**.

Evident $\boldsymbol{\omega}$-cocontinuous endofunctors are identity and constant functors, as well as colimit functors (because of the standard "interchange of colimits") like coproducts. In **Set**, also finite products are $\boldsymbol{\omega}$-cocontinuous (see, eg, [Mac71, Theorem IX.2.1]), hence, since $\boldsymbol{\omega}$-cocontinuousness is preserved by composition, the endofunctors corresponding to signatures are $\boldsymbol{\omega}$-cocontinuous. Similarly, for every signature Σ and every set X, the endofunctor

$$(X+\Sigma) : \mathbf{Set} \to \mathbf{Set}$$

with action $Y \mapsto X + \Sigma Y$, is $\boldsymbol{\omega}$-cocontinuous, hence its initial algebra exists: it is the **algebra freely generated by** Σ **on** X, with as carrier TX, the set of terms with variables $x \in X$. Since initial algebras are isomorphisms

$$X + \Sigma TX \cong TX$$

the set TX is a coproduct and its algebra structure is the copair of the injections

$$\mathsf{inl}_X : X \to TX \qquad \mathsf{inr}_X : \Sigma TX \to TX$$

The left injection is the usual insertion of variables $x \in X$ into the terms $t \in TX$, which is usually left implicit. Formally, x is simply an element of the set X and it is only after applying inl_X to it that one obtains a variable. This variable-making function is usually written as

$$\eta_X = \mathsf{inl}_X : X \to TX$$

The other injection $\mathsf{inr}_X : \Sigma TX \to TX$ is the operation which permits to construct a new term given any n-ary operator σ and terms $t_1, \dots, t_n$; also the right injection is usually left implicitly and one writes simply $\sigma(t_1, \dots, t_n)$ for the resulting term.

Like $T0$, also TX, being an initial algebra, comes with an induction principle. and, since it is a coproduct, one can rephrase the principle as follows. For every

Σ-algebra structure $h : \Sigma Z \to Z$ and every '*valuation*' function $f : X \to Z$ of the variables in X as elements of the algebra $\langle Z, h\rangle$, there exists a unique function $f^\sharp : TX \to Z$ making

$$\begin{tikzcd} X \arrow[r, "\eta_X = \mathsf{inl}_X"] \arrow[dr, "f"'] & TX \arrow[d, dashed, "f^\sharp"] & \Sigma TX \arrow[l, "\mathsf{inr}_X"'] \arrow[d, "\Sigma f^\sharp"] \\ & Z & \Sigma Z \arrow[l, "h"] \end{tikzcd}$$

commute. Omitting the injections,

$$f^\sharp(x) = f(x) \quad \text{and} \quad f^\sharp(\sigma(t_1, \ldots, t_n)) = h(\sigma(f^\sharp(t_1), \ldots, f^\sharp(t_n)))$$

This **inductive extension of** h **along the** valuation **function** f is, formally, the inductive extension $[f, h]^\#$ of the $(X+\Sigma)$-algebra structure on Z given by the copair

$$\begin{tikzcd} X \arrow[r, tail] \arrow[dr, "f"'] & X+\Sigma Z \arrow[d, dashed, "{[f,h]}"] & \Sigma Z \arrow[l, tail] \arrow[dl, "h"] \\ & Z & \end{tikzcd}$$

For instance, this induction principle can be used to show that the operator T inductively extends to a functor $T : \mathbf{Set} \to \mathbf{Set}$. Indeed, to define its action Tf on a function $f : X \to Y$, take the inductive extension of $\mathsf{inr}_Y : \Sigma TY \to TY$ along the composite $\mathsf{inl}_Y \circ f$:

$$\begin{tikzcd} X \arrow[r, "\eta_X = \mathsf{inl}_X"] \arrow[d, "f"'] & TX \arrow[d, dashed, "Tf = (\eta_Y \circ f)^\sharp"] & \Sigma TX \arrow[l, "\mathsf{inr}_X"'] \arrow[d, "\Sigma Tf"] \\ Y \arrow[r, "\eta_Y = \mathsf{inl}_Y"'] & TY & \Sigma TY \arrow[l, "\mathsf{inr}_Y"] \end{tikzcd}$$

To prove that this definition is functorial, ie $T(\mathsf{id}_X) = \mathsf{id}_{TX}$ and $T(g \circ f) = Tg \circ Tf$, for $g : Y \to Z$, one exploits the uniqueness of inductive extensions: the function id_{TX} fits as $(\eta_X \circ \mathsf{id}_X)^\sharp = (\eta_X)^\sharp$ and $Tg \circ Tf$ fits as $(\eta_Z \circ g \circ f)^\sharp$.

Notice that a function $f : X \to Y$ can be seen as a '*renaming*' of variables and then the function $Tf : TX \to TY$ is the inductive extension of such a renaming from simple variables to complex terms with variables.

Another example is the definition of the operation $\mu_X : TTX \to TX$ inductively extending $\mathsf{inr}_X : \Sigma TX \to TX$ along the identity on TX:

$$
\begin{tikzcd}
TX \arrow[r, "\eta_{TX}"] \arrow[dr, equal] & T^2X \arrow[d, dashed, "\mu_X = (\mathsf{id}_{TX})^\sharp"] & \Sigma T^2 X \arrow[l, "\mathsf{inr}_{TX}"'] \arrow[d, "\Sigma\mu_X"] \\
 & TX & \Sigma TX \arrow[l, "\mathsf{inr}_X"]
\end{tikzcd}
$$

This permits to form terms from *any* operator derivable from the signature. For instance, for the above sample language, consider the derived (unary) operator '$a \,;\, (\_)$': given any term $t \in TX$, one can form the term $a \,;\, t$ by first applying $a \,;\, (\_)$ to t and then μ_X:

$$a \,;\, t = \mu_X(a \,;\, (t))$$

Derived operators can also be seen as *contexts* and then the operation μ_X is formally needed to remove brackets after plugging terms in the holes of a context.

Notes. For a comprehensive survey on the use of Σ-algebras in semantics see [MT92].

2 Terms, Algebras and Monads

The inductive definition of the syntax of a language as a free algebra on a signature Σ defines a '*syntactical monad*' T. In general, every algebraic theory $\langle \Sigma, E \rangle$ defines a monad T and, 'conversely', every monad is defined by its algebras in a categorical, more abstract sense.

Let I be the identity functor. By definition, the insertion-of-variables function $\eta_X = \mathsf{inl}_X : X \to TX$ introduced in the previous section is natural in X:

$$\eta : I \Rightarrow T$$

Similarly, the brackets-removing function $\mu_X : T^2X \to TX$ is natural in X, because it is the inductive extension of a natural transformation (the right injection $\mathsf{inr} : \Sigma T \Rightarrow T$) along the identity. The triple

$$T = < T, \eta, \mu >$$

is a 'monad' on **Set**.

> A **monad** in a category $\mathbf{C}$ is like a monoid in $\mathbf{C}^{\mathbf{C}}$ – the category having as objects endofunctors on $\mathbf{C}$ and as arrows natural transformations between them: it is a triple $< T, \eta, \mu >$ consisting of an object $T : \mathbf{C} \to \mathbf{C}$, an associative **multiplication** $\mu : T^2 \Rightarrow T$, and a **unit** $\eta : I \Rightarrow T$ for this multiplication. Notice that $T^2 = T \circ T$, thus the composition of functors is used in this definition rather than their product. Diagrammatically, the associativity and the (left and right) unit laws are expressed as follows.

Monad Laws

$$\begin{tikzcd}
T^3 \arrow[r, Rightarrow, "T\mu"] \arrow[d, Rightarrow, "\mu T"'] & T^2 \arrow[d, Rightarrow, "\mu"] \\
T^2 \arrow[r, Rightarrow, "\mu"'] & T
\end{tikzcd}
\qquad
\begin{tikzcd}
IT \arrow[r, Rightarrow, "\eta T"] \arrow[dr, equal] & T^2 \arrow[d, Rightarrow, "\mu"] & TI \arrow[l, Rightarrow, "T\eta"'] \arrow[dl, equal] \\
& T &
\end{tikzcd}$$

In order to prove that the free Σ-algebra functor T, together with the left injection $\eta = \mathsf{inl} : I \Rightarrow T$ as unit and the inductive extension of the right injection $\mathsf{inr} : \Sigma T \Rightarrow T$ along the identity as multiplication $\mu : T^2 \Rightarrow T$, is a monad on **Set**, recall the definition of μ:

$$
\begin{tikzcd}
TX \arrow[r, "\eta_{TX}"] \arrow[dr, equal] & T^2X \arrow[d, "\mu_X"] & \Sigma T^2X \arrow[l, "\mathsf{inr}_{TX}"'] \arrow[d, "\Sigma\mu_X"] \\
 & TX & \Sigma TX \arrow[l, "\mathsf{inr}_X"]
\end{tikzcd}
$$

The commutativity of the triangle on the left shows that η and μ satisfy the left unit law. As for the right unit law, exploit the uniqueness of inductive extensions, noticing that both the identity on TX and the composite $\mu_X \circ T\eta_X$ fit as the (unique!) inductive extension ${\eta_X}^\sharp$ of inr_X along η_X:

$$
\begin{tikzcd}
X \arrow[rr, "\eta_X"] \arrow[dr, "\eta_X"] \arrow[dd, equal] & & TX \arrow[d, "T\eta_X"] & \Sigma TX \arrow[l, "\mathsf{inr}_X"'] \arrow[d, "\Sigma T\eta_X"] \\
 & TX \arrow[r, "\eta_{TX}"] \arrow[dr, equal] & T^2X \arrow[d, "\mu_X"] & \Sigma T^2X \arrow[l, "\mathsf{inr}_{TX}"'] \arrow[d, "\Sigma\mu_X"] \\
X \arrow[rr, "\eta_X"'] & & TX & \Sigma TX \arrow[l, "\mathsf{inr}_X"]
\end{tikzcd}
$$

Indeed, everything in sight in the above diagram commutes, either by definition or by naturality (of η and inr). Similarly, one can prove the associativity law by noticing that both composites $\mu_{TX} \circ \mu_X$ and $T\mu_X \circ \mu_X$ fit as the inductive extension ${\mu_X}^\sharp$ of inr_X along μ_X.

From adjunctions to monads. A source of monads is to be found in adjunctions:

Every adjunction from a category $\mathbf{C}$ *to a category* $\mathbf{D}$

$$\begin{tikzcd} \mathbf{D} \arrow[d, bend left, "G"] & \text{counit} = \varepsilon : FG \Rightarrow I \\ \mathbf{C} \arrow[u, bend left, "F"] \arrow[u, phantom, "\dashv"] & \text{unit} = \eta : I \Rightarrow GF \end{tikzcd}$$

gives rise to a monad $T =< GF, \eta, G\varepsilon F >$ *on* $\mathbf{C}$.

For a proof of this fact see, eg, [Mac71, §VI.1]; here, as an example, consider again the above term monad. Firstly, notice that the property that every Σ-algebra structure $h : \Sigma Z \to Z$ can be inductively extended along any function $f : X \to Z$ to a function $f^\sharp : TX \to Z$ amounts to the fact that the forgetful functor $U^\Sigma : \mathbf{Set}^\Sigma \to \mathbf{Set}$, mapping Σ-algebras to their carriers, has a left adjoint, namely the functor

$$F^\Sigma : \mathbf{Set} \to \mathbf{Set}^\Sigma \qquad\qquad X \mapsto (\mathsf{inr}_X : \Sigma TX \to TX)$$

Indeed, the diagram defining $f^\sharp$

$$\begin{tikzcd} X \arrow[r, "\eta_X"] \arrow[dr, "f"'] & TX \arrow[d, dashed, "f^\sharp"] & \Sigma TX \arrow[l, "\mathsf{inr}_X"'] \arrow[d, "\Sigma f^\sharp"] \\ & Z & \Sigma Z \arrow[l, "h"] \end{tikzcd}$$

can be decomposed into

$$\begin{tikzcd} X \arrow[r, "\eta_X"] \arrow[dr, "f"'] & TX = U^\Sigma\langle TX, \mathsf{inr}_X\rangle \arrow[d, "f^\sharp = U^\Sigma f^\sharp"] \\ & Z = U^\Sigma\langle Z, h\rangle \end{tikzcd} \qquad \begin{tikzcd} TX \arrow[d, dashed, "f^\sharp"] & \Sigma TX \arrow[l, "\mathsf{inr}_X"'] \arrow[d, "\Sigma f^\sharp"] \\ Z & \Sigma Z \arrow[l, "h"] \end{tikzcd}$$

which shows that F^Σ is the left adjoint of U^Σ and, moreover, that η is the unit of the adjunction.

Next, the counit ε of the adjunction is $\mathsf{id}^\sharp$, ie, for every Σ-algebra structure $h : \Sigma X \to X$,

$$\varepsilon_h : TX \to X$$

is the inductive extension of h along the identity on X. Then, indeed, from $F^\Sigma \dashv U^\Sigma$, one gets the above monad as follows.

$$T = U^\Sigma F^\Sigma \qquad \eta = \eta \qquad \mu = \varepsilon_{\mathsf{inr}} = \varepsilon_{F^\Sigma} = U^\Sigma \varepsilon_{F^\Sigma}$$

From monads to adjunctions

Not only every adjunction gives rise to a monad, but also, conversely, every monad splits into an adjunction. In general, there are many categories $\mathbf{D}$ such that a monad in $\mathbf{C}$ splits into an adjunction from $\mathbf{C}$ to $\mathbf{D}$, but there are two canonical ones, namely the initial and the final ones in a suitable sense. Consider the final one; it is defined by adding some extra conditions on the objects of the category of algebras of an endofunctor:

> Let $T =< T, \eta, \mu >$ be a monad in a category $\mathbf{C}$. The **category of T-algebras**, denoted by $\mathbf{C}^T$, has as objects pairs $\langle X, h \rangle$, with X an object of $\mathbf{C}$ and $h : TX \to X$ an arrow of $\mathbf{C}$ such that the following diagrams commute.
>
> **T-Algebra Laws**
>
> $$\begin{tikzcd} T^2X \arrow[r, "Th"] \arrow[d, "\mu_X"'] & TX \arrow[d, "h"] \\ TX \arrow[r, "h"'] & X \end{tikzcd} \qquad \begin{tikzcd} X \arrow[r, "\eta_X"] \arrow[dr, equal] & TX \arrow[d, "h"] \\ & X \end{tikzcd}$$
>
> The arrows of the category are those arrows of the category $\mathbf{C}$ which preserve the algebra structure, that is, making the following diagram commute.
>
> $$\begin{tikzcd} TX \arrow[r, "Tf"] \arrow[d, "h"'] & TY \arrow[d, "k"] \\ X \arrow[r, "f"'] & Y \end{tikzcd}$$
>
> (This category is also called the *Eilenberg-Moore category* of the monad.)

Notice that, in particular, $\langle TX, \mu_X \rangle$ is a T-algebra for every object X in $\mathbf{C}$. Therefore, also $\langle T^2X, \mu_{TX} \rangle$ is a T-algebra and μ_X is an algebra arrow between them.

Another example of a T-algebra structure is given by the above inductive extension $\varepsilon_h : TX \to X$ of a Σ-algebra structure $h : \Sigma X \to X$ along the identity on X. Indeed, the law $\varepsilon_h \circ \eta_X = \mathsf{id}_X$ holds by definition, while the other law holds because both composites $\varepsilon_h \circ \mu_X$ and $\varepsilon_h \circ T\varepsilon_h$ fit as the inductive extension ${\varepsilon_h}^\sharp$ of h along ε_h

$$\begin{tikzcd}
TX \arrow[r, "\eta_{TX}"] \arrow[d, "\varepsilon_h"'] & T^2X \arrow[dd, dashed, "{\varepsilon_h}^\sharp"] & \Sigma T^2X \arrow[l, "\mathsf{inr}_{TX}"'] \arrow[dd, "\Sigma{\varepsilon_h}^\sharp"] \\
X \arrow[dr, equal] & & \\
 & X & \Sigma X \arrow[l, "h"]
\end{tikzcd}$$

as shown by the commutativity of the following two diagrams.

$$\begin{tikzcd}
TX \arrow[r, "\eta_{TX}"] \arrow[d, "\varepsilon_h"'] & T^2X \arrow[d, "T\varepsilon_h"] & \Sigma T^2X \arrow[l, "\mathsf{inr}_{TX}"'] \arrow[d, "\Sigma T\varepsilon_h"] \\
X \arrow[r, "\eta_X"] \arrow[dr, equal] & TX \arrow[d, "\varepsilon_h"] & \Sigma TX \arrow[l, "\mathsf{inr}_X"'] \arrow[d, "\Sigma\varepsilon_h"] \\
 & X & \Sigma X \arrow[l, "h"]
\end{tikzcd}
\qquad
\begin{tikzcd}
TX \arrow[r, "\eta_{TX}"] \arrow[d, "\varepsilon_h"'] \arrow[dr, equal] & T^2X \arrow[d, "\mu_X"] & \Sigma T^2X \arrow[l, "\mathsf{inr}_{TX}"'] \arrow[d, "\Sigma\mu_X"] \\
X \arrow[dr, equal] & TX \arrow[d, "\varepsilon_h"] & \Sigma TX \arrow[l, "\mathsf{inr}_X"'] \arrow[d, "\Sigma\varepsilon_h"] \\
 & X & \Sigma X \arrow[l, "h"]
\end{tikzcd}$$

Σ-algebras are T-algebras. The above mapping

$$(h : \Sigma X \to X) \mapsto (\varepsilon_h : TX \to X)$$

taking a Σ-algebra structure on X into its coinductive extension along the identity on X

$$\begin{tikzcd}
X \arrow[r, "\eta_X"] \arrow[dr, equal] & TX \arrow[d, dashed, "\varepsilon_h"] & \Sigma TX \arrow[l, "\mathsf{inr}_X"'] \arrow[d, "\Sigma\varepsilon_h"] \\
 & X & \Sigma X \arrow[l, "h"]
\end{tikzcd}$$

is an isomorphism between the category of Σ-algebras and the algebras of its corresponding monad T.

For the inverse of this mapping from Σ- to T-algebras, precompose each T-algebra $\langle X, h\rangle$ first with the right injection $\mathsf{inr}_X : \Sigma TX \to TX$ and then with $\Sigma\eta_X : \Sigma X \to \Sigma TX$

$$\langle X, h\rangle \mapsto \langle X, h \circ \mathsf{inr}_X \circ \Sigma\eta_X\rangle$$

One half of this isomorphism is illustrated by the following diagram, which commutes 'almost' by definition.

$$\begin{tikzcd}
X \arrow[r, "\eta_X"] \arrow[dr, equal] & TX \arrow[d, "\varepsilon_h"] & \Sigma TX \arrow[l, "\mathsf{inr}_X"'] \arrow[d, "\Sigma\varepsilon_h"'] & \Sigma X \arrow[l, "\Sigma\eta_X"'] \arrow[dl, equal] \\
 & X & \Sigma X \arrow[l, "h"] &
\end{tikzcd}$$

The other half of the isomorphism, namely the commutativity of

$$\begin{tikzcd}
X \arrow[r, "\eta_X"] \arrow[dr, equal] & TX \arrow[d, "h"] & & & \Sigma TX \arrow[llll, "\mathsf{inr}_X"'] \arrow[d, "\Sigma h"] \\
 & X & TX \arrow[l, "h"] & \Sigma TX \arrow[l, "\mathsf{inr}_X"] & \Sigma X \arrow[l, "\Sigma\eta_X"]
\end{tikzcd}$$

is more complex. To prove it, fill the above diagram with subdiagrams which commute either by the T-algebras laws (for the algebra $\langle X, k\rangle$) or by naturality (of the right injection inr and of the unit η), or by the 'identity law' for the monad T:

$$\begin{tikzcd}
X \arrow[r, "\eta_X"] \arrow[ddr, equal] & TX \arrow[dd, "h"] & & & & \Sigma TX \arrow[llll, "\mathsf{inr}_X"'] \arrow[dl, "\Sigma\eta_{TX}"] \arrow[dd, "\Sigma h"] \\
 & & & \Sigma TX \arrow[ull, "\mathsf{inr}_X"'] \arrow[r, phantom, "="] & \Sigma T^2X \arrow[l, "\Sigma\mu_X"'] \arrow[dl, "\mathsf{inr}_{TX}"'] \arrow[dd, "\Sigma Th"] & \\
 & & T^2X \arrow[uul, "\mu_X"] \arrow[d, "Th"] & & & \\
 & X & TX \arrow[l, "h"] & & \Sigma TX \arrow[ll, "\mathsf{inr}_X"] & \Sigma X \arrow[l, "\Sigma\eta_X"]
\end{tikzcd}$$

This concludes the proof of the isomorphism between Σ- and T-algebras.

Under the above isomorphism, the free Σ-algebra structure $\mathsf{inr}_X : \Sigma TX \to TX$ over X corresponds to the T-algebra structure $\mu_X = \varepsilon_{\mathsf{inr}} : T^2X \to TX$. (See the concrete description of these two operations given in the previous section.) Recall that the forgetful functor $U^\Sigma : \mathbf{Set}^\Sigma \to \mathbf{Set}$ from the Σ-algebras has a left adjoint $F^\Sigma X = \langle X, \mathsf{inr}_X : \Sigma TX \cong TX\rangle$. Correspondingly, also the evident forgetful functor

$U^T : \mathbf{Set}^T \to \mathbf{Set}$ from the T-algebras has a left adjoint namely $F^T X = \langle X, \mu_X : T^2 X \to TX \rangle$ and the following two diagrams commute.

$$\begin{tikzcd}
\mathbf{Set}^\Sigma \arrow[r, phantom, "\cong"] \arrow[dr, "U^\Sigma"'] & \mathbf{Set}^T \arrow[d, "U^T"] \\
& \mathbf{Set}
\end{tikzcd}
\qquad\qquad
\begin{tikzcd}
\mathbf{Set}^\Sigma \arrow[r, phantom, "\cong"] & \mathbf{Set}^T \\
& \mathbf{Set} \arrow[ul, "F^\Sigma"] \arrow[u, "F^T"']
\end{tikzcd}$$

In general, the above adjunction $F^T \dashv U^T$ holds for algebras of monads on *any* category $\mathbf{C}$:

The adjunction $F^T \dashv U^T$ splitting the monad T. The functor

$$F^T : \mathbf{C} \to \mathbf{C}^T \qquad\qquad X \mapsto \langle X, \mu_X : T^2 X \to TX \rangle$$

is the left adjoint of the forgetful functor $U^T : \mathbf{C}^T \to \mathbf{C}$ mapping T-algebras to their carriers. The unit of this adjunction is the unit η of the monad. As for the counit $\varepsilon : F^T U^T \Rightarrow I$, this is simply

$$\varepsilon_{\langle X,h \rangle} = h : F^T U^T \langle X, h \rangle = \langle TX, \mu_X \rangle \to \langle X, h \rangle$$

which is a T-algebra arrow from $\langle TX, \mu_X \rangle$ to $\langle X, h \rangle$ because of the very definition of T-algebra structure. The right unit law of the monad and the T-algebra law for the unit are then the two triangular equalities which prove the adjunction $F^T \dashv U^T$.

The monad arising from this adjunction is the original monad T:

$$T \ = < T, \eta, \mu > = < U^T F^T,\ \eta\ ,\ U^T F^T \varepsilon >$$

Therefore:

Every monad is defined by its algebras.

Moreover, the adjunction $F^T \dashv U^T$ is the 'final' one defining the monad T; that is, from any adjunction

$$\begin{tikzcd}
\mathbf{C} \arrow[r, shift left, "F"] \arrow[r, phantom, "\perp"] & \mathbf{D} \arrow[l, shift left, "U"]
\end{tikzcd}$$

giving rise to the monad T there exists a unique 'comparison' functor $K : \mathbf{D} \to \mathbf{C}^T$ such that the following two diagrams commute.

$$\begin{tikzcd}
\mathbf{D} \arrow[r, dashed, "K"] \arrow[dr, "U"'] & \mathbf{C}^T \arrow[d, "U^T"] \\
& \mathbf{C}
\end{tikzcd}
\qquad\qquad
\begin{tikzcd}
\mathbf{D} \arrow[r, dashed, "K"] & \mathbf{C}^T \\
& \mathbf{C} \arrow[ul, "F"] \arrow[u, "F^T"']
\end{tikzcd}$$

If $\varepsilon : FU \Rightarrow I$ is the counit of the adjunction $F \dashv U$, then, for every object D of $\mathbf{D}$,

$$KD = \langle UD, U\varepsilon_D : UFUD = TUD \to UD \rangle$$

When this comparison functor K is an isomorphism, then the functor $U : \mathbf{D} \to \mathbf{C}$ is called **monadic**. Thus, for instance, the forgetful functor $U^\Sigma : \mathbf{Set}^\Sigma \to \mathbf{Set}$ is monadic.

In general, to prove that a functor is monadic, one can use *Beck's theorem* (see, eg, [Mac71]) stating that a functor is monadic if and only if it 'creates' suitable coequalizers. In particular, this can be used to prove the following generalization of the above correspondence between Σ- and T-algebras.

> **Algebras are T-algebras.** Given a signature Σ and a set E of equations on the (derived) operators of the signature, consider the corresponding category $\mathbf{Set}^{\langle \Sigma, E\rangle}$ of Σ-algebras validating the equations in E and having as arrows functions which preserve the operators. Then, the evident forgetful functor from $\mathbf{Set}^{\langle \Sigma, E\rangle}$ to $\mathbf{Set}$ has a left adjoint and, moreover, it is monadic. Therefore, the category of algebras of the monad T corresponding to this adjunction is isomorphic to the category $\mathbf{Set}^{\langle \Sigma, E\rangle}$.

This shows that the notion of algebras of monads encompasses the standard notion of algebras as *varieties*, that is, as sets with operations from a signature Σ which validate a set of equations E. (Eg, monoids, groups, semi-lattices, etc.)

Notice that one might want to describe the programs of a language as a free $\langle \Sigma, E\rangle$-algebra rather than a free Σ-algebra. For instance, the behaviour of the sequential composition operator is intended to be associative thus one can axiomatize this directly in the syntax by adding the equation

$$x \; ; \; (y \; ; \; z) = (x \; ; \; y) \; ; \; z$$

Then, there will be no distinction in the syntax anymore between the program $t \; ; \; (u \; ; \; v)$ and the program $(t \; ; \; u) \; ; \; v$, ie they will represent the same program. (Another example is in Section 10, where the semi-lattice laws are imposed on the 'non-deterministic choice' operator 'or'.)

Equations can also be used to describe the behaviour of new operators algebraically. For instance, one can define a 'replication' operator '$!$' in terms of sequential composition by means of the equation

$$!x = x \; ; \; (!x)$$

Thus, in general, the programs of a language might be terms of a signature Σ quotiented by (the smallest congruence generated by) a set of equations E. In the sequel, monads T corresponding to $\langle \Sigma, E\rangle$-algebras describing the programs of a language will be called **syntactical monads**.

Finally, notice that the fact that Σ-algebras are T-algebras holds also for arbitrary endofunctors $\Sigma : \mathbf{C} \to \mathbf{C}$ which have an initial $(X+\Sigma)$-algebra $TX \cong X+\Sigma TX$ for every object X in the category $\mathbf{C}$. That is, the forgetful functor $U^\Sigma : \mathbf{C}^\Sigma \to \mathbf{C}$ has a left adjoint $X \mapsto \langle TX, \mathsf{inr}_X : \Sigma TX \to TX\rangle$ and, moreover, it is monadic. Thus the isomorphism of categories

$$\mathbf{Set}^\Sigma \cong \mathbf{Set}^T$$

is not only an instance of

$$\mathbf{Set}^{\langle\Sigma,E\rangle} \cong \mathbf{Set}^T$$

but also of

$$\mathbf{C}^\Sigma \cong \mathbf{C}^T$$

3 Operational Semantics, Transition Systems and Coalgebras

Operational models like transition systems can be seen as '*coalgebras*' of suitable '*behaviour*' endofunctors.

The **operational semantics** of a language defines *how* programs are to be executed and what their observable effect is. More specifically, the operational semantics considered here aims at specifying the actions that programs can perform, like changing a state, and their subsequent *transitions* into new programs, usually the part of the code still remaining to be executed. The result is thus a relation of type

$$\mathsf{Programs} \times \mathsf{Actions} \times \mathsf{Programs}$$

usually denoted element-wise as a labelled arrow of type

$$\mathsf{program} \xrightarrow{\mathsf{act}} \mathsf{program}$$

Relations of this kind are called 'labelled transition systems' as they specify the (labelled) transitions between programs.

In general, a **transition system** with labels $a \in A$ is given by a set X of *states* and a family $\{\xrightarrow{a}\}_A$ of *transition relations* labelled by $a \in A$:

$$\langle X, \{\xrightarrow{a}\}_A \rangle$$

One reads

$$x \xrightarrow{a} x'$$

as 'from the state x the system can perform an action a and reach the state x''. Equivalently, a labelled transition system is a labelled directed graph: nodes = states, labelled arcs = transitions.

The **inert** states of a transition system are those from which no action can be performed. It is convenient to introduce an explicit predicate '$\downarrow *$' on states to express that one can *observe* that a state is inert:

$$x \downarrow * \iff x \text{ is inert}$$

Thus a transition system is a triple

$$\langle X, \{\xrightarrow{a}\}_A\,, \downarrow * \rangle$$

In general, given an operational semantics, it might not be easy to prove things about the behaviour of programs, like, for instance, to see whether a program is deterministic. In order to facilitate reasoning about programs, it is convenient that the operational semantics be **structured**, that is, the transition system should be defined by induction on the program constructs (**structural induction**). For example, the intended operational semantics for the simple imperative language

$$t ::= x \mid a \mid \mathsf{nil} \mid (t\ ;\ t)$$

could be specified by induction on the program constructs as follows.

Consider first the constant nil: its intended meaning is that it is the basic *inert* program, that is, a program which cannot perform any action. The only rule for it is then

$$\mathsf{nil} \downarrow *$$

Next, every constant a in Act is an atomic program which can perform the corresponding action a and then become inert:

$$a \xrightarrow{a} \mathsf{nil}$$

Finally, for the sequential composition operator there are three cases to be considered: (*i*) the first component can perform a transition; (*ii*) the first component is inert but the second component can perform a transition; (*iii*) both components are inert. That is, using also the meta-variables u, v, etc, to range over the programs of the language,

$$\frac{u \xrightarrow{a} u'}{u\ ;\ v \xrightarrow{a} u'\ ;\ v} \qquad \frac{u \downarrow * \quad v \xrightarrow{a} v'}{u\ ;\ v \xrightarrow{a} v'} \qquad \frac{u \downarrow * \quad v \downarrow *}{u\ ;\ v \downarrow *}$$

Let us denote the above set of rules by $\mathcal{R}$. All rules of $\mathcal{R}$ are *well-founded*, hence the least transition system closed under $\mathcal{R}$ does exist: this is the *intended model* for $\mathcal{R}$. Moreover, the rules of $\mathcal{R}$ are *finitary*, hence every transition in the intended model can be proved in a finite number of steps.

By structural induction, one can prove that the set of states of the intended model is the set $T0$ of closed programs. Indeed, there are axioms for all constants and if two programs u and v belong to the states of the model then also $u\ ;\ v$ does. Thus the intended model is of the form

$$\langle T0, \{\xrightarrow{a}\}_{\mathsf{Act}}, \downarrow * \rangle$$

Another property of the above transition system which can be proved by structural induction is that it is **deterministic**: there is only one rule for each constant and the three rules for sequential composition have, by induction, disjoint hypotheses; thus every program can perform at most one action.

A similar argument shows that every program can either perform an action or being inert; that is, for every closed program t, either there exists a unique action

a and a unique program t' such that $t \xrightarrow{a} t'$ or, otherwise, $t \downarrow *$. Therefore, this transition system (ie, the transition *relation* together with the predicate $\downarrow *$) can then be regarded as a single total *function*

$$[\![\text{-}]\!]_{\mathcal{R}} : T0 \to 1 + \mathsf{Act} \times T0$$

For this, put

$$[\![t]\!]_{\mathcal{R}} = * \iff t \downarrow * \qquad \text{and} \qquad [\![t]\!]_{\mathcal{R}} = <a, t'> \iff t \xrightarrow{a} t'$$

where, recall, '$*$' denotes the unique element of the final object 1 in **Set**. In general, this defines a one-to-one correspondence between deterministic transition systems and '*co*-algebras' of the endofunctor $BX = 1 + \mathsf{Act} \times X$ on **Set**.

> Given an endofunctor $B : \mathbf{C} \to \mathbf{C}$ on a category $\mathbf{C}$, the **category of B-coalgebras**, denoted by $\mathbf{C}_B$, has as objects pairs $\langle X, k\rangle$, with X an object of $\mathbf{C}$ and $k : X \to BX$ an arrow of $\mathbf{C}$. The arrows $f : \langle X, k\rangle \to \langle X', k'\rangle$ of $\mathbf{C}_B$ are the arrows $f : X \to X'$ of $\mathbf{C}$ which preserve the coalgebra structure:
>
> $$\begin{tikzcd} X \arrow[r, "f"] \arrow[d, "k"'] & X' \arrow[d, "k'"] \\ BX \arrow[r, "Bf"'] & BX' \end{tikzcd}$$
>
> (Cf Σ-algebras in Section 1.)

Thus a coalgebra of the endofunctor $BX = 1 + \mathsf{Act} \times X$ is a pair $\langle X, k\rangle$, with X a set and k a function

$$k : X \to 1 + \mathsf{Act} \times X$$

This can be seen as a deterministic transition system

$$\langle X, \{\xrightarrow{a}\}_{\mathsf{Act}}, \downarrow *\rangle$$

because of the correspondence

$$x \downarrow * \iff k(x) = * \qquad \text{and} \qquad x \xrightarrow{a} x' \iff k(x) = <a, x>$$

Notions of behaviour and endofunctors. The above correspondence between deterministic transition systems and coalgebras of the 'behaviour' endofunctor $BX = 1{+}\mathsf{Act}{\times}X$ generalizes to several forms of *non-deterministic* transition systems. More generally, the claim is that *coalgebras* are suitable to modelling the *operational behaviour* of the programs of a language. The corresponding endofunctors are called **behaviour endofunctors**.

Consider transition systems without the inert predicate $\downarrow\!*$. Take the endofunctor

$$BX = \mathcal{P}(\mathsf{Act} \times X)$$

where $\mathcal{P} : \mathbf{Set} \to \mathbf{Set}$ is the (covariant) power-set endofunctor: for every set X and function $f : X \to Y$

$$\mathcal{P}X = \{X' \mid X' \subseteq X\} \qquad (\mathcal{P}f)(X') = \{fx \mid x \in X'\}$$

Then, a one-to-one correspondence between coalgebras

$$k : X \to \mathcal{P}(\mathsf{Act} \times X)$$

and transition systems

$$\langle X, \{\xrightarrow{a}\}_{\mathsf{Act}}\rangle$$

is obtained by putting

$$<a, x'> \in k(x) \iff x \xrightarrow{a} x'$$

Another example is obtained by restricting the above behaviour to

$$BX = \mathcal{P}_{\mathrm{fi}}(\mathsf{Act} \times X)$$

where $\mathcal{P}_{\mathrm{fi}} : \mathbf{Set} \to \mathbf{Set}$ is the *finite power-set endofunctor*. Its coalgebras correspond to '*finitely branching* transition systems', that is transition systems which can, at each state, choose among a *finite* set of transitions rather than among an arbitrarily large one.

Notice in the two examples above that a state x is mapped by the coalgebra structure k to the empty set 0 if and only if the corresponding transition system cannot perform any transition from x. Alternatively, one can use the isomorphism

$$\mathcal{P}_{\mathrm{fi}}(\mathsf{Act} \times X) \cong 1 + \check{\mathcal{P}}(\mathsf{Act} \times X)$$

where $\check{\mathcal{P}}$ is the 'relevant' part of the (finite) power-set functor, mapping a set to the set of its (finite) and *non-empty* subsets. The coalgebras of the behaviour $BX = 1{+}\check{\mathcal{P}}(\mathsf{Act}{\times}X)$ are then finitely branching transition system with the explicit inert predicate $\downarrow\!*$. Omitting the injections into the coproduct $1 + \check{\mathcal{P}}(\mathsf{Act} \times X)$, the correspondence is as follows.

$$k(x) = * \iff x \downarrow * \qquad \text{and} \qquad <a, x'> \in k(x) \iff x \xrightarrow{a} x'$$

Here the transition relation and the inert predicate are disjoint: if a state can become inert then it cannot choose to perform an action. If, instead, one wants to consider transition systems with states in which both choices are allowed the following behaviour is to be used.

$$BX = \check{\mathcal{P}}(1 + \mathsf{Act} \times X)$$

Omitting the injections, one has the following correspondence.

$$* \in k(x) \iff x \downarrow * \qquad \text{and} \qquad <a, x'> \in k(x) \iff x \xrightarrow{a} x'$$

One step further is to consider the same behaviours as above but taken in **pSets** – the category of sets and partial functions – rather than in **Set**. This corresponds to considering *partial transition systems*, ie transition systems with states whose behaviour might be undefined.

> It should be stressed that the coalgebras of the above behaviours correspond only as *objects* to transition systems: the *arrows* are quite different. Consider the case of transition systems without the predicate $\downarrow *$. Then, following the definition of transition systems as relations (or as graphs) the natural definition of an arrow
>
> $$f : \langle X, \{\xrightarrow{a}_X\}_{\mathsf{Act}}\rangle \to \langle Y, \{\xrightarrow{a}_Y\}_{\mathsf{Act}}\rangle$$
>
> between transition systems with the same labels is as a function $f : X \to Y$ between their states such that if $x \xrightarrow{a}_X x'$ then $f(x) \xrightarrow{a}_Y f(x')$. Instead, regarding a transition system as a coalgebra, one has the extra condition that the function f must be such that if $f(x) \xrightarrow{a}_Y y$ for some state $y \in Y$, then there exists a state $x' \in X$ such that $x \xrightarrow{a}_X x'$.
>
> Therefore, a category of transition systems is different from the category of coalgebras of the corresponding behaviour. In particular, the universals in the two categories will be different. For instance, while the product of two transition system always exists, the product of two coalgebras does not necessarily exist. Also, the final transition system is different from the final coalgebra. (The latter is an object which enjoys very important semantical properties – cf Section 5.)

The above behaviours, whose coalgebras correspond to various forms of labelled transition systems, are suitable for modelling imperative and concurrent languages. Instead, for modelling applicative languages, one needs behaviours involving some form of function space functor. An example is the endofunctor

$$BX = 1 + X^Y$$

The 'exponent' X^Y is the set of functions from Y to X. In order to avoid the usual 'mixed variance' problems, Y is here treated as a parameter. By putting $Y = X$ one obtains that the corresponding coalgebras are the *quasi-applicative transition systems* defined in [Abr90]. The 'exception' 1 in the above behaviour can be used to encode non-termination.

For example, for X and Y both equal to the set Λ of closed λ-*terms*, one can define a coalgebra structure

$$ev : \Lambda \to 1 + \Lambda^\Lambda$$

by putting, for every λ-term $M \in \Lambda$,

$$ev(M) = P \mapsto N[P/x]$$

if M converges to 'principal weak head normal form' $\lambda x\,.N$, and

$$ev(M) = *$$

otherwise.

Back now to deterministic transition systems and the corresponding behaviour $BX = 1 + \mathsf{Act} \times X$. Recall that the rules $\mathcal{R}$ for the above sample language induce a coalgebra

$$[\![\text{-}]\!]_{\mathcal{R}} : T0 \to BT0$$

This can be seen as a special case of a general construction which, starting from a coalgebra (ie deterministic transition system) structure $k : X \to BX$, yields a new coalgebra structure

$$[\![\text{-}]\!]^k_{\mathcal{R}} : TX \to BTX$$

with the set of terms TX as carrier and which 'conservatively extends' the original structure k.

Indeed, one can add, for every $x \in X$, the value of $k(x)$ as an axiom to the rules in $\mathcal{R}$ that is, if $k(x) = < a, x >$ then add $x \xrightarrow{a} x'$ to $\mathcal{R}$ and if $k(x) = *$ then add $x \downarrow *$. The least transition system induced by these extended rules will have then TX as set of states and be deterministic, hence it can be regarded as a coalgebra with structure $[\![\text{-}]\!]^k_{\mathcal{R}} : TX \to 1 + \mathsf{Act} \times TX$. By structural induction, one can prove that this induced transition system/coalgebra **conservatively extends** the coalgebra/transition system $\langle X, k\rangle$ in the sense that, for every $x \in X$,

$$k(x) = [\![x]\!]^k_{\mathcal{R}}$$

Formally, recalling that $\eta_X : X \to TX$ is the insertion-of-variables function which permits to see the elements $x \in X$ as variable terms in TX, the above conservative extension property amounts to the commutativity of the following diagram.

$$\begin{tikzcd}
X \arrow[r, "\eta_X"] \arrow[d, "k"'] & TX \arrow[d, "{[\![\text{-}]\!]^k_{\mathcal{R}}}"] \\
BX \arrow[r, "B\eta_X"'] & BTX
\end{tikzcd}$$

That is, the function $\eta_X : X \to TX$ 'lifts' to a coalgebra arrow

$$\eta_X : \langle X, k\rangle \to \langle TX, [\![\text{-}]\!]^k_{\mathcal{R}}\rangle$$

for every coalgebras structure k on X.

Notes. The importance of the correspondence between labelled transition systems and coalgebras of the behaviour $BX = \mathcal{P}(\mathsf{Act} \times X)$ has been stressed by Peter Aczel in [Acz88]. (But see also [Ken87] and [Hes88].) For a comprehensive categorical (but not coalgebraic!) treatment of labelled transition systems see [WN95].

As mentioned in the introduction, it would be interesting to sort out the relationship between the present notion of *behaviour* as an *endofunctor* whose coalgebras are operational models and Eugenio Moggi's notion of *computation* as a *monad* [Mog91]. The examples of *computational monads* given in [Mog91] (partiality, non-determinism, side-effects, exceptions, etc) all qualify as behaviours, and the corresponding monadic operations could play an important rôle in further developments. (The operations of the (finite) non-determinism monad $\mathcal{P}_{\mathrm{fi}}$ are already used in Sections 10 and 11.)

4 Functorial Operational Semantics

In this section, a new approach to operational semantics, based on categorical notions of *syntax* and *behaviour*, is introduced: an operational semantics is *functorial* when it is a 'lifting' of the syntactical monad T to the coalgebras of the behaviour endofunctor B.

Inductively, this can be obtained by defining an 'action' of the program constructs on the composite functor BT; as an instance, the operational rules of a simple deterministic language are shown to define such an action. More generally, a functorial operational semantics can be obtained by defining a 'distributive law' of the syntactical monad T over the behaviour functor B.

Given a syntactical monad T and a behaviour endofunctor B on the same category, a **functorial operational semantics** wrt T and B is a 'lifting' of the monad T to the B- coalgebras.

In general, let $U : \mathbf{C}_B \to \mathbf{C}$ be the forgetful functor mapping coalgebras $\langle X, k\rangle$ to their carriers X. Then, a **lifting** of a monad $T = <T, \eta, \mu>$ to the coalgebras of an endofunctor B on the same category $\mathbf{C}$ is a monad Φ such that the diagram

$$\begin{tikzcd} \mathbf{C}_B \arrow[r, "\Phi"] \arrow[d, "U"'] & \mathbf{C}_B \arrow[d, "U"] \\ \mathbf{C} \arrow[r, "T"'] & \mathbf{C} \end{tikzcd}$$

commutes, making $U : \mathbf{C}_B \to \mathbf{C}$ a 'map of monads'. That is, Φ is a triple $<\Phi, \tilde{\eta}, \tilde{\mu}>$ such that

$$\begin{aligned} U\Phi &= TU : \mathbf{C}_B \to \mathbf{C} \\ U\tilde{\eta} &= \eta_U : U \Rightarrow TU \\ U\tilde{\mu} &= \mu_U : T^2U \Rightarrow TU \end{aligned}$$

The second and third equation imply that the unit $\tilde{\eta}$ and multiplication $\tilde{\mu}$ of Φ are the same as the unit η and multiplication μ of $T = <T, \eta, \mu>$, because of the very definition of coalgebra arrows. Therefore:

$$\Phi = <\Phi, \eta, \mu>$$

One can check that the three equations and the fact that the triple $T = < T, \eta, \mu >$ is a monad imply that also the triple $\Phi = < \Phi, \eta, \mu >$ is a monad.

Let us now look at the endofunctor Φ. The equation $U\Phi = TU$ implies that Φ is completely determined by its action on the structure of coalgebras, that is, on the arrow $k : X \to BX$ in a coalgebra $\langle X, k \rangle$:

$$\frac{X \xrightarrow{\ k\ } BX}{TX \xrightarrow[\ \Phi k\ ]{} BTX}$$

Indeed, by the definition of coalgebra arrows, the action of Φ on arrows is the same as the one of T:

$$\begin{tikzcd}
X \arrow[r, "f"] \arrow[d, "k"'] & X' \arrow[d, "k'"] \\
BX \arrow[r, "Bf"'] & BX'
\end{tikzcd}
\quad \overset{\Phi}{\longmapsto} \quad
\begin{tikzcd}
TX \arrow[r, "Tf"] \arrow[d, "\Phi k"'] & TX' \arrow[d, "\Phi k'"] \\
BTX \arrow[r, "BTf"'] & BTX'
\end{tikzcd}$$

Rewriting the above action as

$$\langle X, k \rangle \xrightarrow{\ f\ } \langle X', k' \rangle$$

$$\begin{tikzcd}
TU\langle X, k \rangle \arrow[r, "TUf"] \arrow[d, "\Phi\langle X, k \rangle"'] & TU\langle X', k' \rangle \arrow[d, "\Phi\langle X', k' \rangle"] \\
BTU\langle X, k \rangle \arrow[r, "BTUf"'] & BTU\langle X', k' \rangle
\end{tikzcd}$$

shows the following correspondence.

> **Liftings as Coactions.** A lifting of an *endofunctor* T to the B-coalgebras, that is, an *endofunctor* Φ such that $U\Phi = TU$, is the same as a **coaction** of B on the composite functor $TU : \mathbf{C}_B \to \mathbf{C}$, that is, a natural transformation
>
> $$TU \Rightarrow BTU$$

Finally, the conditions $U\widetilde{\eta} = \eta_U$ and $U\widetilde{\mu} = \mu_U$ amount to say that η and μ lift to natural transformations in the B-coalgebras. That is, for every coalgebra $\langle X, k \rangle$, the two squares in the following diagram commute.

$$\begin{tikzcd}
X \arrow[r, "\eta_X"] \arrow[d, "k"'] & TX \arrow[d, "\Phi k"'] & T^2X \arrow[l, "\mu_X"'] \arrow[d, "\Phi^2 k"] \\
BX \arrow[r, "B\eta_X"'] & BTX & BT^2X \arrow[l, "B\mu_X"]
\end{tikzcd}$$

Inductive Functorial Operational Semantics

An *inductive* way of defining a functorial operational semantics is by specifying the **action** of the program constructs Σ on the 'observables' BT of the language, that is, by giving a natural transformation

$$\phi : \Sigma BT \Rightarrow BT$$

Indeed, for every B-coalgebra $\langle X, k\rangle$, the Σ-algebra structure $\phi_X = \phi_{U\langle X,h\rangle} : \Sigma(BTX) \to BTX$ on BTX can be inductively extended along the composite $B\eta_X \circ k$ to a coalgebra structure $\widehat{\phi}(k) : TX \to BTX$

$$\begin{tikzcd}
X \arrow[r, "\eta_X"] \arrow[d, "k"'] & TX \arrow[d, dashed, "\widehat{\phi}(k)\ =\ "', "(B\eta_X \circ k)^\sharp"] & \Sigma TX \arrow[l, "\mathsf{inr}_X"'] \arrow[d, "\Sigma\widehat{\phi}(k)"] \\
BX \arrow[r, "B\eta_X"'] & BTX & \Sigma BTX \arrow[l, "\phi_X"]
\end{tikzcd}$$

By the naturality of ϕ, this definition is natural in $\langle X, k\rangle$, that is,

$$\widehat{\phi} : TU \Rightarrow BTU$$

thus $\widehat{\phi}$ can be seen as an endofunctor (with the same name) on the B-coalgebras.

Moreover, the triple $<\widehat{\phi}, \eta, \mu>$ – where, recall, η and μ are the unit and multiplication of the term monad T – is a monad on the B-coalgebras, that is, the two squares in

$$\begin{tikzcd}
X \arrow[r, "\eta_X"] \arrow[d, "k"'] & TX \arrow[d, "\widehat{\phi}(k)"'] & T^2X \arrow[l, "\mu_X"'] \arrow[d, "\widehat{\phi}^2(k)"] \\
BX \arrow[r, "B\eta_X"'] & BTX & BT^2X \arrow[l, "B\mu_X"]
\end{tikzcd}$$

commute. Indeed, the square corresponding to the unit η commutes by definition, while the one corresponding to the multiplication μ commutes because both composites $\widehat{\phi}(k) \circ \mu_X$ and $B\mu_X \circ \widehat{\phi}^2(k)$ fit as the (unique!) inductive extension of ϕ_X along $\widehat{\phi}(k)$

$$
\begin{tikzcd}
TX \arrow[r, "\eta_{TX}"] \arrow[d, "\widehat{\phi}(k)"'] \arrow[d] & T^2X \arrow[dd, dashed, "\widehat{\phi}(k)^\sharp"] & \Sigma T^2X \arrow[l, "\mathsf{inr}_{TX}"'] \arrow[dd, "\Sigma\widehat{\phi}(k)^\sharp"] \\
BTX \arrow[dr, equal] & & \\
& BTX & \Sigma BTX \arrow[l, "\phi_X"]
\end{tikzcd}
$$

because

$$
\begin{tikzcd}
TX \arrow[r, "\eta_{TX}"] \arrow[d, "\widehat{\phi}(k)"'] & T^2X \arrow[d, "\widehat{\phi}^2(k)"] & \Sigma T^2X \arrow[l, "\mathsf{inr}_{TX}"'] \arrow[d, "\Sigma\widehat{\phi}^2(k)"] \\
BTX \arrow[r, "B\eta_{TX}"] \arrow[dr, equal] & BT^2X \arrow[d, "B\mu_X"] & \Sigma BT^2X \arrow[l, "B\mathsf{inr}_{TX}"'] \arrow[d, "\Sigma B\mu_X"] \\
& BTX & \Sigma BTX \arrow[l, "\phi_X"]
\end{tikzcd}
\qquad
\begin{tikzcd}
TX \arrow[r, "\eta_{TX}"] \arrow[d, "\widehat{\phi}(k)"'] \arrow[dr, equal] & T^2X \arrow[d, "\mu_X"] & \Sigma T^2X \arrow[l, "\mathsf{inr}_{TX}"'] \arrow[d, "\Sigma\mu_X"] \\
BTX \arrow[dr, equal] & TX \arrow[d, "\widehat{\phi}(k)"] & \Sigma TX \arrow[l, "\mathsf{inr}_X"'] \arrow[d, "\Sigma\widehat{\phi}(k)"] \\
& BTX & \Sigma BTX \arrow[l, "\phi_X"]
\end{tikzcd}
$$

Some terminology: in the sequel, a functorial operational semantics Φ is also called the **operational monad** Φ and the natural transformation $\phi : \Sigma BT \Rightarrow BT$ inducing the operational monad $\widehat{\phi}$ is called the **germ** of $\widehat{\phi}$.

Operational Rules and Inductive Functorial Operational Semantics

Now the claim is that the operational rules $\mathcal{R}$ given in the previous section for the simple deterministic language $t ::= x \mid a \mid \mathsf{nil} \mid (t\ ;\ t)$ can be regarded as a natural transformation

$$\lceil \mathcal{R} \rceil : \Sigma B \Rightarrow BT$$

Moreover, by taking the composite $B\mu \circ \lceil \mathcal{R} \rceil_T : \Sigma BT \Rightarrow BT$ one obtains the germ of an inductive functorial operational semantics which is 'observationally equivalent' to the operational semantics induced by the rules $\mathcal{R}$. (This result is generalized in Section 11 to the large class of 'GSOS' operational rules, which are suitable to model most of imperative and concurrent programming languages.)

Recall that the algebras of the signature $\Sigma = \mathsf{Act} \cup \{\mathsf{nil},\ ;\ \}$ for the above language are the same as the algebras of the endofunctor

$$\Sigma X = 1 + \mathsf{Act} + X \times X$$

on **Set** and that the programs t are the elements of TX, the carrier of the free Σ-algebra on X. Also, recall that the operational semantics induced by the rules $\mathcal{R}$ of the language is a deterministic transition system and that there is a one-to-one correspondence between deterministic transition systems and coalgebras of the endofunctor

$$BX = 1 + \mathsf{Act} \times X$$

This correspondence says that a transition $x \xrightarrow{a} x'$ of a deterministic transition system can be seen as the action $x \mapsto <a, x'>$ of a coalgebra structure $X \to BX$; similarly, the action $x \mapsto *$ corresponds to the fact that $x \downarrow *$ holds. Thus the operational rules $\mathcal{R}$ given in the previous section can be written as follows.

$$\mathsf{nil} \mapsto * \qquad\qquad a \mapsto <a, \mathsf{nil}>$$

$$\frac{u \mapsto <a, u'>}{u\ ;\ v \mapsto <a, u'\ ;\ v>} \qquad \frac{u \mapsto * \quad v \mapsto <a, v'>}{u\ ;\ v \mapsto <a, v'>} \qquad \frac{u \mapsto * \quad v \mapsto *}{u\ ;\ v \mapsto *}$$

Next, let us define the natural transformation $\lceil \mathcal{R} \rceil : \Sigma B \Rightarrow BT$. Let r and s be meta-variables ranging over elements of $BX = 1 + \mathsf{Act} \times X$, for arbitrary sets of variables X. One has to define the value of $\lceil \mathcal{R} \rceil_X$ at nil, at a, and at $r\ ;\ s$, for all r, s. Omitting the subscript X, put

$$\lceil \mathcal{R} \rceil(\mathsf{nil}) = * \qquad \text{and} \qquad \lceil \mathcal{R} \rceil(a) = <a, \mathsf{nil}>$$

For sequential composition there are three cases to be considered, namely

1. $r = <a, x>$
2. $r = *$ and $s = <a, y>$

3. $r = *$ and $s = *$

In the second and third case one can follow the definition of $\mathcal{R}$ and put $<a, y>$ and $*$, respectively, for the value of $\lceil\mathcal{R}\rceil$ at $r\,;\,s$. Instead, in the first case, one cannot put simply $<a, x\,;\,s>$ because $x\,;\,s$ is not of type T. The problem is that s is of type B rather than T. But notice that B can be *embedded* in T:

The embedding γ of the behaviour into the syntax. The action

$$* \mapsto \mathsf{nil} \qquad <a, x> \mapsto a\,;\,x$$

defines an *injective* function from BX to TX, for every set X. It is manifestly natural in X; call it

$$\gamma : B \Rightarrow T$$

One can then put

$$\lceil\mathcal{R}\rceil(r\,;\,s) \;=\; \begin{cases} <a, x\,;\,\gamma s> & \text{if } r = <a, x> \\ <a, y> & \text{if } r = * \text{ and } s = <a, y> \\ * & \text{if } r = * = s \end{cases}$$

Altogether, in a more suggestive notation:

$$\frac{\mathsf{nil} \overset{\lceil\mathcal{R}\rceil}{\longmapsto} * \qquad r = <a, x>}{r\,;\,s \overset{\lceil\mathcal{R}\rceil}{\longmapsto} <a, x\,;\,\gamma s>} \qquad \frac{a \overset{\lceil\mathcal{R}\rceil}{\longmapsto} <a, \mathsf{nil}> \qquad r = * \quad s = <a, y>}{r\,;\,s \overset{\lceil\mathcal{R}\rceil}{\longmapsto} <a, y>} \qquad \frac{r = * \quad s = *}{r\,;\,s \overset{\lceil\mathcal{R}\rceil}{\longmapsto} *}$$

This definition yields a natural transformation

$$\lceil\mathcal{R}\rceil : \Sigma B \Rightarrow BT$$

Indeed, the only problematic clause for the naturality of $\lceil\mathcal{R}\rceil$ is $\lceil\mathcal{R}\rceil(r\,;\,s)$ for $r =<a, x>$. One has to show that, for every 'renaming' $f : X \to Y$, the following holds.

$$\begin{tikzcd}
<a,x>\,;\,s \arrow[r, mapsto, "\Sigma Bf"] \arrow[d, mapsto, "\lceil\mathcal{R}\rceil_X"'] & <a, fx>\,;\,(Bf)(s) \arrow[d, mapsto, "\lceil\mathcal{R}\rceil_Y"] \\
<a, x\,;\,\gamma_X s> \arrow[r, mapsto, "BTf"'] & <a, (Tf)(x\,;\,\gamma_X s)> = <a, fx\,;\,\gamma_Y(Bf)(s)>
\end{tikzcd}$$

That is,

$$(Tf)(\gamma_X s) = \gamma_Y(Bf)(s)$$

But this is immediate from the fact that γ is a natural transformation from B to T.

(As shown in Section 11, the argument in the above proof generalizes to any (possibly non-deterministic) rule in the 'GSOS-format'.)

Next, consider the germ of the functorial operational semantics corresponding to $\mathcal{R}$. It is essentially the same as $\lceil\mathcal{R}\rceil$, only it is applied to terms, hence the multiplication μ of the syntactical monad T is needed in order to remove brackets from the resulting terms of terms to yield simple terms. Thus $\phi^{\mathcal{R}} = B\mu \circ \lceil\mathcal{R}\rceil_T : \Sigma BT \Rightarrow BT$, that is,

$$\begin{tikzcd} \Sigma BT \arrow[r, Rightarrow, "\lceil\mathcal{R}\rceil_T"] \arrow[dr, Rightarrow, "\phi^{\mathcal{R}}"'] & BT^2 \arrow[d, Rightarrow, "B\mu"] \\ & BT \end{tikzcd}$$

Therefore:

$$\mathsf{nil} \overset{\phi^{\mathcal{R}}}{\longmapsto} * \qquad a \overset{\phi^{\mathcal{R}}}{\longmapsto} <a, \mathsf{nil}>$$

$$\frac{r = <a, t>}{r \,;\, s \overset{\phi^{\mathcal{R}}}{\longmapsto} <a, t \,;\, \gamma s>} \qquad \frac{r = * \quad s = <a, t>}{r \,;\, s \overset{\phi^{\mathcal{R}}}{\longmapsto} <a, t>} \qquad \frac{r = * \quad s = *}{r \,;\, s \overset{\phi^{\mathcal{R}}}{\longmapsto} *}$$

The resulting $\phi^{\mathcal{R}} : \Sigma BT \Rightarrow BT$ is the germ of a functorial operational semantics. In particular, consider the case of closed terms $T0$ and write

$$[\![\text{-}]\!]_{\lceil\mathcal{R}\rceil} : T0 \to BT0$$

for the operational model obtained by taking the inductive extension of $\phi_0^{\mathcal{R}} : \Sigma BT0 \to BT0$

$$\begin{tikzcd} \Sigma T0 \arrow[r, "\Sigma\, [\![\text{-}]\!]_{\lceil\mathcal{R}\rceil}"] \arrow[d, "\cong"'] & \Sigma BT0 \arrow[d, "\phi_0^{\mathcal{R}}"] \\ T0 \arrow[r, dashed, "{[\![\text{-}]\!]_{\lceil\mathcal{R}\rceil}}"'] & BT0 \end{tikzcd}$$

Then, by definition,

$$[\![u \,;\, v]\!]_{\lceil\mathcal{R}\rceil} = \begin{cases} <a, u' \,;\, \gamma[\![v]\!]_{\lceil\mathcal{R}\rceil}> & \text{if } [\![u]\!]_{\lceil\mathcal{R}\rceil} = <a, u'> \\ <a, v'> & \text{if } [\![u]\!]_{\lceil\mathcal{R}\rceil} = * \text{ and } [\![v]\!]_{\lceil\mathcal{R}\rceil} = <a, v'> \\ * & \text{if } [\![u]\!]_{\lceil\mathcal{R}\rceil} = * = [\![v]\!]_{\lceil\mathcal{R}\rceil} \end{cases}$$

Contrast this with the operational model

$$[\![\text{-}]\!]_{\mathcal{R}} : T0 \to BT0$$

'directly' induced by the rules $\mathcal{R}$ on the closed terms: they are the same, except for

$$[\![u\ ;\ v]\!]_{\mathcal{R}} = <a, u'\ ;\ v> \qquad \text{if } [\![u]\!]_{\mathcal{R}} = <a, u'>$$

In Section 6 it is shown that for every term v, the term $\gamma[\![v]\!]_{\lceil\mathcal{R}\rceil}$ exhibits the same 'observable behaviour' as v, under any context. Therefore, the two models $[\![\text{-}]\!]_{\mathcal{R}}$ and $[\![\text{-}]\!]_{\lceil\mathcal{R}\rceil}$ are 'observationally equivalent'. This is based on the fact that the above natural transformation $\gamma : B \Rightarrow T$ is a 'retraction' for the operational semantics induced by $\mathcal{R}$. More precisely, for every coalgebra structure $k : X \to BX$, the composite arrow $\mu_X \circ \gamma_{TX} : BTX \to TX$ is a right inverse for the operational model $[\![\text{-}]\!]^k_{\mathcal{R}} : TX \to BTX$ induced by $\mathcal{R}$. Indeed, omitting, as usual, the multiplication μ,

$$\gamma_{TX}(*) \;=\; \mathsf{nil} \stackrel{\mathcal{R}}{\longmapsto} * \qquad \text{and} \qquad \gamma_{TX}(<a, t>) \;=\; a\ ;\ t \stackrel{\mathcal{R}}{\longmapsto} <a, t>$$

hence

$$[\![\gamma_{TX}(*)]\!]^k_{\mathcal{R}} = * \qquad \text{and} \qquad [\![\gamma_{TX}(<a, t>)]\!]^k_{\mathcal{R}} = <a, t>$$

Semantics as a Distributive Law

The germ $\phi : \Sigma BT \Rightarrow BT$ of an inductive functorial operational semantics $\widehat{\phi}$ defines a 'distributive law' $\phi^{\#} : TB \Rightarrow BT$ of the syntactical monad T over the behaviour B. The operational monad $\widehat{\phi}$ can be then decomposed in terms of this distributive law and of T itself. In turn, every distributive law $\lambda : TB \Rightarrow BT$ defines a lifting of the monad T to the B-coalgebras.

In general, a **distributive law** of a monad $T = <T, \eta, \mu>$ over an endofunctor B (on the same category) is a natural transformation

$$\lambda : TB \Rightarrow BT$$

such that the following two diagrams commute.

$$\begin{tikzcd} & B \arrow[dl, Rightarrow, "\eta_B"'] \arrow[dr, Rightarrow, "B\eta"] & \\ TB \arrow[rr, Rightarrow, "\lambda"'] & & BT \end{tikzcd} \qquad \begin{tikzcd} T^2B \arrow[r, Rightarrow, "T\lambda"] \arrow[d, Rightarrow, "\mu_B"'] & TBT \arrow[r, Rightarrow, "\lambda_T"] & BT^2 \arrow[d, Rightarrow, "B\mu"] \\ TB \arrow[rr, Rightarrow, "\lambda"'] & & BT \end{tikzcd}$$

Every distributive law $\lambda : TB \Rightarrow BT$ defines an endofunctor lifting T to the B-coalgebras by mapping a coalgebra $\langle X, k\rangle$ to the coalgebra $\langle TX, \lambda \circ Tk\rangle$:

$$\frac{X \xrightarrow{k} BX}{TX \xrightarrow[Tk]{} TBX \xrightarrow[\lambda_X]{} BTX}$$

Moreover, this is a lifting of the whole monad $T = <T, \eta, \mu>$ to the B-coalgebras, because everything in sight in the following diagram commutes (either by the naturality of η and μ or by distributivity).

$$\begin{tikzcd} X \arrow[r, "\eta_X"] \arrow[ddddd, "k"'] & TX \arrow[dd, "Tk"'] & T^2X \arrow[l, "\mu_X"'] \arrow[d, "T^2k"] \\ & & T^2BX \arrow[dl, "\mu_{BX}"'] \arrow[dd, "T\lambda_X"] \\ & TBX \arrow[ddd, "\lambda_X"'] & \\ & & TBTX \arrow[dd, "\lambda_{TX}"] \\ & & \\ BX \arrow[r, "B\eta_X"'] \arrow[uuur, "\eta_{BX}"] & BTX & BT^2X \arrow[l, "B\mu_X"] \end{tikzcd}$$

A distributive laws can be defined from a germ $\phi : \Sigma BT \Rightarrow BT$ by taking the inductive extension

$$\phi^{\#} = (B\eta)^{\sharp} : TB \Rightarrow BT$$

of the germ ϕ along the natural transformation $B\eta : B \Rightarrow BT$.

$$\begin{tikzcd}
B \arrow[r, Rightarrow, "\eta_B"] \arrow[dr, Rightarrow, "B\eta"'] & TB \arrow[d, Rightarrow, dashed, "\phi^{\#} = (B\eta)^{\sharp}"] & \Sigma TB \arrow[l, Rightarrow, "\mathsf{inr}_B"'] \arrow[d, Rightarrow, "\Sigma\phi^{\#}"] \\
& BT & \Sigma BT \arrow[l, Rightarrow, "\phi"]
\end{tikzcd}$$

Indeed, the left triangle shows that $\phi^{\#}$ satisfies the first of the two conditions for being a distributive law. To prove the second, one can show that both composites $\phi^{\#} \circ \mu_B$ and $B\mu \circ \phi^{\#}_T \circ T\phi^{\#}$ fit as the unique inductive extension of ϕ along $\phi^{\#}$. (This is very much the same as the above proof that μ lifts to a multiplication for the inductive functorial operational semantics $\widehat{\phi}$.)

Notice that then the action of the inductive operational monad $\widehat{\phi}$ on a coalgebra $\langle X, k\rangle$ can be decomposed into the action of the syntactical monad T on the structure k, followed by the distributive law $\phi^{\#}$ at the carrier X:

$$\widehat{\phi}\,\langle X, k\rangle = \phi^{\#}_X \circ Tk$$

Notes. The notion of a distributive law of a monad over an endofunctor is derived from the more familiar notion of a distributive law of a monad T_1 over another monad T_2 introduced in [Bec69]. In that paper, the equivalence is proved between distributive laws of the monad T_1 over the monad T_2, liftings of the monad T_2 to the T_1-algebras, and actions of the monad T_1 over the functor $T_2U^{T_1}$. (See also [BW85], Chapter 9.) Here monads are lifted to coalgebras (of a functor) rather than to algebras (of a monad) and this gives a slightly different situation, with a monad distributing over a functor (and with distributive laws implying liftings but not vice versa). More symmetry is gained in Section 7 by considering the *comonad* D cofreely generated by the behaviour B.

5 Recursive Behaviours, Final Coalgebras and Coinduction

The rôle of final coalgebras is dual to the one played by initial algebras, and dual are their properties and constructions. For instance, as initial algebras account for induction, final coalgebras account for the dual notion of 'coinduction', which is useful to deal with the behaviour of recursive programs. Also, as the programs of a language may be described as the initial algebra of a signature Σ, the abstract global behaviours – the 'processes' – of a language may be described as the final coalgebra of a behaviour B.

Let B be an endofunctor which has a **final coalgebra** (ie the final object in the corresponding category of coalgebras) and let $\widehat{B}$ denote the carrier of this final coalgebra. The structure of a final coalgebra is, like that of an initial algebra, an isomorphism, because the notion of isomorphism is 'self-dual'. Thus:

$$\widehat{B} \cong B\widehat{B} \qquad\qquad \text{(final } B\text{-coalgebra)}$$

Any coalgebra structure $k : X \to BX$ can be 'coinductively' extended to an arrow $k^{@} : X \to \widehat{B}$ by taking the unique coalgebra arrow from the coalgebra $\langle X, k\rangle$ to the final coalgebra:

Coinductive Extension

$$\begin{tikzcd}
X \arrow[r, dashed, "k^{@}"] \arrow[d, "k"'] & \widehat{B} \arrow[d, "\cong"] \\
BX \arrow[r, "Bk^{@}"'] & B\widehat{B}
\end{tikzcd}$$

Of particular interest are the coinductive extensions of operational models. In order to illustrate this, let us consider languages, like the one in Section 3, which have an operational semantics yielding deterministic transition systems, that is, coalgebras of the behaviour endofunctor

$$BX = 1 + \mathsf{Act} \times X$$

on **Set**. Thus, if T is the syntactical monad for the language, an operational model is a coalgebra with structure

$$[\![\text{-}]\!] : TX \to BTX$$

where X is the set of variables of the language. Then, under the assumption that the final B-coalgebra exists, the coinductive extension of this coalgebra structure yields the

Final Coalgebra Semantics

$$\begin{tikzcd} TX \arrow[r, dashed, "[\![-]\!]^{@}"] \arrow[d, "[\![-]\!]"'] & \widehat{B} \arrow[d, "\cong"] \\ BTX \arrow[r, "B[\![-]\!]^{@}"'] & B\widehat{B} \end{tikzcd}$$

of the language. Since $BTX = 1+\mathsf{Act}\times TX$, this yields, for any term t, the following definition.

$$[\![t]\!]^{@} \;=\; \begin{cases} \qquad * & \text{if } \;[\![t]\!] = * \\ <a, [\![t']\!]^{@}> & \text{if } \;[\![t]\!] = <a, t'> \end{cases}$$

(Notice that the isomorphism $\widehat{B} \cong B\widehat{B}$ has been treated as an equality in order to simplify the notation.) Thus, for instance, wrt the operational model $[\![-]\!]$ given in Section 3, the programs $a\ ;\ b$ and $a\ ;\ \mathsf{nil}\ ;\ b$ have the same final coalgebra semantics:

$$[\![a\ ;\ b]\!]^{@} = <a, b, *> = [\![a\ ;\ \mathsf{nil}\ ;\ b]\!]^{@}$$

In general, under this final coalgebra semantics, a program is mapped into the stream of actions that it can perform.

Next, consider the construction of the final coalgebra for the above endofunctor $BX = 1 + \mathsf{Act} \times X$. This is an $\boldsymbol{\omega}^{\mathrm{op}}$-continuous endofunctor, that is, it preserves limits of functors from $\boldsymbol{\omega}^{\mathrm{op}} = \{0 \leftarrow 1 \leftarrow 2 \leftarrow \cdots\}$. Indeed, it is made of constants, a product, and a coproduct: constants and products (like all limit functors) are $\boldsymbol{\omega}^{\mathrm{op}}$-continuous in every category; finite coproducts are $\boldsymbol{\omega}^{\mathrm{op}}$-continuous in **Set**, by the dual of a theorem [Mac71, Theorem IX.2.1] mentioned in Section 1. By further dual considerations, the carrier of the final coalgebra of an $\boldsymbol{\omega}^{\mathrm{op}}$-continuous endofunctor B is the limit $\widehat{B}$ of the following diagram obtained by iterative applications of B to the unique function from $B1$ to the singleton set 1, the final object in **Set**.

$$1 \xleftarrow{1_{B1}} B1 \xleftarrow{B1_{B1}} B^2 1 \xleftarrow{B^2 1_{B1}} \cdots$$

The isomorphism $\varphi : \widehat{B} \cong B\widehat{B}$ giving the coalgebra structure is obtained as a mediating arrow just like in the initial algebra construction.

This general construction of final coalgebras of $\boldsymbol{\omega}^{\mathrm{op}}$-continuous endofunctors yields, in the particular case considered here, the final coalgebra with carrier the set

$$\mathsf{Act}^{\infty} = \coprod_{\alpha \leq \omega} \mathsf{Act}^{\alpha}$$

of finite ($\alpha = n$) and infinite ($\alpha = \omega$) streams of actions generated by Act, and with structure the isomorphism

$$\varphi : \mathsf{Act}^{\infty} \cong 1 + \mathsf{Act} \times \mathsf{Act}^{\infty}$$

This isomorphism is an operation which allows one to explore the streams $w \in \mathsf{Act}^{\infty}$: if $w = \epsilon$, the empty stream, then $\varphi(w) = *$, otherwise $w = a \cdot w'$ and $\varphi(w) = <a, w'>$, that is, φ applied to a non-empty stream returns the first element of the stream plus its continuation. Also notice that its inverse $\varphi^{-1} : 1 + \mathsf{Act} \times \mathsf{Act}^{\infty} \cong \mathsf{Act}^{\infty}$ is a *B-algebra* structure; it gives the empty stream constant $\epsilon = \varphi^{-1}(*)$ and the prefixing operators $a \cdot \text{-} = \varphi^{-1}(a, \text{-})$, and the identity $a \cdot \epsilon = a$ follows from the fact that $\mathsf{Act} \times 1 \cong \mathsf{Act}$.

Next, the unique coalgebra arrow from a B-coalgebra $\langle X, k \rangle$ to $\langle \mathsf{Act}^{\infty}, \varphi \rangle$ is defined as follows. Let $\langle X, \{\xrightarrow{a}\}_{Act} \rangle$ be the deterministic transition system corresponding to the coalgebra $\langle X, k \rangle$. (Cf Section 3.) Then, for every $x \in X$, consider the **global behaviour** of the state x in the transition system: there are three possibilities, namely either (i) the state x is inert, or (ii) the system performs a finite sequence

$$x \xrightarrow{a_1} x_1 \xrightarrow{a_2} \cdots \xrightarrow{a_n} x_n$$

of transitions starting from the state x and then reaches an inert state x_n, or (iii) the system performs an infinite sequence

$$x \xrightarrow{a_1} x_1 \xrightarrow{a_2} \cdots \xrightarrow{a_n} x_n \xrightarrow{a_{n+1}} \cdots$$

of transitions, never reaching an inert state. Correspondingly, define the function $k^{@} : X \to \mathsf{Act}^{\infty}$ by putting, for every $x \in X$,

$$k^{@}(x) \;=\; \begin{cases} * & \text{if } \ (i) \\ <a_1, a_2, \ldots, a_n, *> & \text{if } \ (ii) \\ <a_1, a_2, \ldots, a_n, a_{n+1}, \ldots> & \text{if } \ (iii) \end{cases}$$

One can check this is the desired unique coalgebra arrow from $\langle X, k \rangle$ to $\langle \mathsf{Act}^{\infty}, \varphi \rangle$.

Thus the coinductive extension of a coalgebra structure is defined in terms of the global behaviours in the corresponding transition system. The carrier of the final coalgebra itself is the set of all possible '*abstract global behaviours*' wrt B, in which the name of the states is irrelevant. In other words, streams are global behaviours of deterministic transition systems with a single state.

Notice that, taking the behaviour $BX = 1 + \mathsf{Act} \times X$ in the category **pSet** of sets and partial functions rather than in **Set**, the (carrier of the) final B-coalgebra in **pSet** does not contain infinite steams but only the finite ones. Indeed, using partial functions, the coinductive extension of a state having an infinite global behaviour can be left undefined.

Now, the set of finite streams is the carrier of the *initial B-algebra*, both in **Set** and **pSet**. Similarly, the set of natural numbers $N \cong 1 + N$ is both the carrier of the initial algebra and of the final coalgebra of the endofunctor $X \mapsto 1 + X$ on **pSet**, while in **Set** the final coalgebra needs an extra infinity point ∞. This fact generalizes to all functors $X \mapsto \coprod_{\sigma \in \Sigma} X^{\mathrm{ar}(\sigma)}$ corresponding to signatures Σ.

Guarded Recursion. So far, the operational interpretation of the sample language

$$t ::= x \mid a \mid \mathsf{nil} \mid (t \mathbin{;} t)$$

yields global behaviours which are always *finite*. In order to obtain *infinite* global behaviours, let us use the variables $x \in X$ of the language and define *recursive programs* as solutions of 'term-equations' like

$$x = a \mathbin{;} x$$

Intuitively, the solution of the above equation should be a program having as abstract global behaviour the infinite stream a^ω.

In general, not all term-equations have solutions which can be interpreted as streams. For instance, the equation

$$x = x \mathbin{;} x$$

should have as solution a program which keeps on unfolding itself

$$x \longrightarrow x \mathbin{;} x \longrightarrow x \mathbin{;} x \mathbin{;} x \mathbin{;} x \longrightarrow \dots$$

never performing any action. In order to rule out this kind of equation one usually considers only recursive definitions which are 'guarded', that is, equations $x = t$ in which t is of the form $a \mathbin{;} t'$.

Operationally, the above presentation of recursive programs can be made formal by introducing a *fixed point* binding operator fix: given a variable x and a 'guarded' term $t = a \mathbin{;} t'$, the expression $\mathsf{fix} x.t$ is then a term with operational behaviour described by the rule

$$\frac{t[\mathsf{fix} x.t/x] \xrightarrow{a} u}{\mathsf{fix} x.t \xrightarrow{a} u}$$

in which the expression $t[\mathsf{fix} x.t/x]$ stands for the term obtained by substituting the term $\mathsf{fix} x.t$ for every occurrence of x in t.

One of the advantages and novelties of the present functorial approach to operational semantics is that it allows for an elegant operational description of recursive programs which, quite surprisingly, does not require the introduction of a binding operator like the above fix (at least for 'top-level' recursive definitions). Moreover, it allows for a general formal description of guarded recursion, independent of the use of actions and transitions.

Firstly, every *system* of term-equations

$$\begin{array}{rcl} x_1 & = & t_1 \\ x_2 & = & t_2 \\ & \vdots & \end{array}$$

with $x_i \in X$ and $t_i \in TX$, can be seen as a *coalgebra* of the syntax T having as carrier the set $X = \{x_1, x_2, \ldots\}$ of variables appearing in the system and as structure the function

$$k : X \to TX \qquad\qquad x_i \mapsto t_i$$

The generalization of allowing for systems of equations, rather than single equations amounts to allowing for *mutually recursive definitions* like, eg,

$$\begin{array}{rcl} x & = & a \,;\, y \\ y & = & b \,;\, c \,;\, y \end{array}$$

Next, recall the embedding $\gamma : B \Rightarrow T$ of the behaviour into the syntax, mapping $*$ to nil and $< a, x >$ to $a \,;\, x$. Then, a system of (mutually) recursive definitions $k : X \to TX$ is **guarded** if it factorizes through a coalgebra

$$g : X \to BTX$$

of the composite endofunctor BT in the sense that

$$\begin{tikzcd} X \arrow[rr, "k"] \arrow[d, "g"'] & & TX \\ BTX \arrow[r, "\gamma_{TX}"'] & T^2X \arrow[ur, "\mu_X"'] & \end{tikzcd}$$

commutes, that is, $k = \mu_X \circ \gamma_{TX} \circ g : X \to TX$, where $\mu : T^2 \Rightarrow T$ is the multiplication of the syntactical monad T (cf Section 2). For instance, the above system is guarded because the corresponding T-coalgebra factorizes through

$$g(x) = < a, y > \qquad g(y) = < b,\, c \,;\, y >$$

Next, given the germ

$$\phi : \Sigma BT \Rightarrow BT$$

of an inductive functorial operational semantics, write

$$[\![\text{-}]\!]_g : TX \to BTX$$

for the *inductive* extension $g^\sharp$ of the Σ-algebra structure $\phi_X : \Sigma BTX \to BTX$ along a system

$$g : X \to BTX$$

of guarded recursive definitions:

$$
\begin{tikzcd}
X \arrow[r, "\eta_X"] \arrow[dr, "g"'] & TX \arrow[d, dashed, "\llbracket\text{-}\rrbracket_g = g^\sharp"] & \Sigma TX \arrow[l, "\mathsf{inr}_X"'] \arrow[d, "\Sigma\llbracket\text{-}\rrbracket_g"] \\
 & BTX & \Sigma BTX \arrow[l, "\phi_X"]
\end{tikzcd}
$$

Notice the left triangle tells that, up to the insertion-of-variables η_X,

$$\llbracket x \rrbracket_g = g(x)$$

for every x in X. In this way the variables $x \in X$ can be seen as states of a transition systems whose behaviour is described by the semantics ϕ. For instance, in the above example, $x \xrightarrow{a} y$ and $y \xrightarrow{b} c\ ;\ y$.

Then, the desired interpretation of g as a recursive process is obtained by taking the corresponding final coalgebra semantics $(g^\sharp)^@ = \llbracket\text{-}\rrbracket_g^@ : TX \to \widehat{B}$ precomposed with the insertion-of-variables $\eta_X : X \to TX$. Write, abusing the notation, $g^@$ for this function:

$$
\begin{tikzcd}
X \arrow[r, "\eta_X"] \arrow[dr, "g"'] \arrow[rr, bend left, "g^@"] & TX \arrow[d, "\llbracket\text{-}\rrbracket_g = g^\sharp"] \arrow[r, dashed, "\llbracket\text{-}\rrbracket_g^@"] & \widehat{B} \arrow[d, "\cong"] \\
 & BTX \arrow[r, "B\llbracket\text{-}\rrbracket_g^@"'] & B\widehat{B}
\end{tikzcd}
$$

Thus, for the above example, one has, omitting, as usual, both the insertion-of-variables η_X and the final coalgebra isomorphism φ,

$$
\begin{array}{rcl}
g^@(x) & = & < a, g^@(y) > \\
g^@(y) & = & < b,\ \llbracket c\ ;\ y \rrbracket_g^@ > \quad = \quad < b, c, g^@(y) >
\end{array}
$$

that is, $g^@(x) = a(bc)^\omega$ and $g^@(y) = (bc)^\omega$. (Cf the above final coalgebra semantics.)

To be formal, the functorial operational semantics of the previous section gives

$$\llbracket c\ ;\ y \rrbracket_g = < c, \gamma_{TX} \llbracket y \rrbracket_g >$$

hence $g^@(y) = < b, c, \llbracket \gamma_{TX} \llbracket y \rrbracket_g \rrbracket_g^@ >$. However, by 'unfolding' $\llbracket \gamma_{TX} \llbracket y \rrbracket_g \rrbracket_g^@$ by one step

$$\llbracket \gamma_{TX} \llbracket y \rrbracket_g \rrbracket_g^@ = \varphi^{-1} \circ B\llbracket\text{-}\rrbracket_g^@ \circ \llbracket \gamma_{TX} \llbracket y \rrbracket_g \rrbracket_g$$

and by using the fact that γ_{TX} is a retraction for the operational model

$$\llbracket \gamma_{TX} \llbracket y \rrbracket_g \rrbracket_g = \llbracket y \rrbracket_g$$

one obtains

$$g^{@}(y) = \llbracket \gamma_{TX} \llbracket y \rrbracket_g \rrbracket_g^{@}$$

Therefore, the equation

$$g^{@}(y) = <b, c, g^{@}(y)>$$

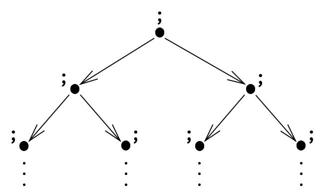
does hold.

Notice that the above recursive definition is automatically well-defined because of the coinduction principle given by finality. In general, final coalgebras allow recursive constructs to be interpreted also in categories where there is no structure to ensure the existence of (canonical) fixed points of functions. In other words, the above interpretation of recursion by final coalgebras encompasses the traditional methods using least fixed points in complete partial orders, or unique fixed points in complete metric spaces, or, more recently, the solution lemma in non-well-founded sets (see Part V), and it permits to interpret recursion in any category, including the ordinary category of (standard) sets.

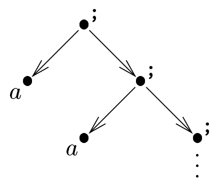
Unguarded Recursion. An alternative approach to recursive programs is obtained by regarding them as (possibly) infinite terms. Representing a term as a tree whose root is labelled by the outermost constructor of the term, one has, for instance, that the solution of the equation

$$x = x \; ; \; x$$

is an infinite tree with no leaf and all nodes labelled by ‘ ; ’:



Similarly, the solution of $x = a \; ; \; x$ is the infinite term represented by the following tree



The advantage of this approach is that it can be applied also to unguarded definitions, but, in order for an infinite term to be given an operational meaning, one needs to shift from the category of ordinary sets to categories with more structured objects like cpos or complete metric spaces.

Coalgebraically, the idea is that, while the initial Σ-algebra is the set of finite terms in Σ, the final Σ-coalgebra contains also the infinite terms. The argument is similar to the one above showing that the final coalgebra of the behaviour $X \mapsto 1+\mathsf{Act}\times X$ contains both finite and infinite streams, while its initial algebra only the finite ones. Now, apart from 'meaningless' equations like $x = x$ (or, more generally, $x = y$, $y = x$) every (possibly unguarded) system of term-equations can be seen as a coalgebra of the composite endofunctor ΣT, that is, as a function

$$k : X \to \Sigma TX$$

This can be made into a Σ-coalgebra with carrier TX by 'copairing' k with the identity on ΣTX using the fact that TX, since it is the carrier of the initial $(X{+}\Sigma)$-algebra, is a coproduct $TX \cong X + \Sigma TX$

$$\begin{tikzcd}
X \arrow[r, tail] \arrow[dr, "k"'] & TX \arrow[d, dashed, "{[k, \mathsf{id}_{\Sigma TX}]}"] & \Sigma TX \arrow[l, tail] \arrow[dl, equal] \\
& \Sigma TX &
\end{tikzcd}$$

(By definition, the value of this coalgebra structure at a variable x is the same as the value of k at x.) Abusing the notation, write

$$k^{@} : X \to \widehat{\Sigma}$$

for the composition of the insertion-of-variables $\eta_X : X \to TX$ with the coinductive extension of the copair $[k, \mathsf{id}_{\Sigma TX}] : TX \to \Sigma TX$

$$\begin{tikzcd}
X \arrow[r, "\eta_X"] \arrow[rr, bend left, "k^{@}"] \arrow[dr, "k"'] & TX \arrow[r, dashed, "{[k, \mathsf{id}_{\Sigma TX}]^{@}}"] \arrow[d, "{[k, \mathsf{id}_{\Sigma TX}]}"] & \widehat{\Sigma} \arrow[d, "\cong"] \\
& \Sigma TX \arrow[r, "{\Sigma[k, \mathsf{id}_{\Sigma TX}]^{@}}"'] & \Sigma\widehat{\Sigma}
\end{tikzcd}$$

Thus, for the coalgebra structure k corresponding to the equation $x = x\ ;\ x$ one has, omitting, as usual, the final coalgebra isomorphism,

$$k^{@}x = (k^{@}x)\ ;\ (k^{@}x)$$

which is the desired infinite term.

Once infinite terms are introduced in the syntax, the problem arises of how to interpret them operationally. One possible solution is to consider categories in which initial algebras and final coalgebras coincide. Indeed, if the inverse of the initial Σ-*algebra* isomorphism $\Sigma\overline{\Sigma} \cong \overline{\Sigma}$ is the final Σ-*coalgebra* isomorphism $\widehat{\Sigma} \cong \Sigma\widehat{\Sigma}$ and, hence,

$$\overline{\Sigma} = T0 = \widehat{\Sigma}$$

then the interpretation of a recursive definition k is the composition of the above $k^{@}$: $X \to T0 = \widehat{\Sigma}$ with the coinductive extension $[\![\text{-}]\!]^{@} : T0 = \overline{\Sigma} \to \widehat{B}$ of the operational model $[\![\text{-}]\!] : T0 \to BT0$.

$$\begin{tikzcd}
 & X \arrow[rr, "k^{@}"] \arrow[dl, "k"'] & & \widehat{\Sigma} = T0 \arrow[rr, "{[\![\text{-}]\!]^{@}}"] \arrow[dl, "\cong"] \arrow[dd, "{[\![\text{-}]\!]}"'] & & \widehat{B} \arrow[dd, "\cong"] \\
\Sigma T X \arrow[rr, "{\Sigma[k, \mathsf{id}_{\Sigma T X}]^{@}}"'] & & \Sigma\widehat{\Sigma} & & & \\
 & & & BT0 \arrow[rr, "{B[\![\text{-}]\!]^{@}}"'] & & B\widehat{B}
\end{tikzcd}$$

As mentioned in Section 1, a category where the initial Σ-algebra is also the final Σ-coalgebra is **pSet**, the category of sets and partial functions. However, like in **Set**, also in **pSet** the object $\overline{\Sigma} = T0$ is the set of finite terms only: the arrow $k^{@} : X \to T0$ is thus a partial function mapping to 'undefined' *every* variable whose intended solution is an infinite term. Thus, in particular, both $x = x\,;x$ and $x = a\,;x$ would be interpreted as undefined, which is not what one expects.

To obtain both infinite terms as elements of an initial algebra and the coincidence of initial algebra and final coalgebras one can move from **pSets** to **pCpo**, the category having as objects complete partial orders (possibly without a bottom element) and as arrows *partial Scott-continuous* functions. The signature $\Sigma X = \coprod_{\sigma} X^{\mathsf{ar}(\sigma)}$ extends to **pCpo** but its initial algebra is the same as the one in **pSets**. In order to obtain infinite terms, one needs to modify Σ by applying to every element of the coproduct $\coprod_{\sigma\in\Sigma} X^{\mathrm{ar}(\sigma)}$ the *lifting monad* $X \mapsto X_{\perp}$ adding a new bottom element to a cpo. That is, take

$$\Sigma X = \coprod_{\sigma\in\Sigma} (X^{\mathrm{ar}(\sigma)})_{\perp}$$

In this way, the syntax will contain both *partial terms* of the form $\perp\,;(a\,;\perp)$ and infinite terms obtained as limit of finite terms. (Cf [Plo81a]: "Syntax considered as a cpo".)

Notice that the behaviour $BX = 1 + \mathsf{Act} \times X$ also extends to **pCpo** but, in general, the problem remains of how to extend a functorial operational semantics from sets to cpos. This is not treated in the present study and left to future work. It shows anyway the importance of the generality of the formulation of functorial operational semantics, where the base category **C** is not necessarily **Set**.

Notes. The standard solution of *domain equations* in **pCpo** [SP82, Plo85] has long been known to be a final coalgebra, but this was obscured by the fact that initial algebras and final coalgebras of ('locally continuous') endofunctors on **pCpo** do coincide in the sense that they are 'canonically isomorphic'. (And the same holds for 'locally contracting' endofunctors on complete metric spaces – cf [AR89, RT93].) Correspondingly, the availability of a coinduction principle was obscured by the use of induction and by 'internal' properties, like the existence of least (respectively unique) fixed points of continuous (respectively contracting) functions.

It has been Peter Aczel's work on 'non-well-founded sets' [Acz88] which has brought to light the main semantic properties of final coalgebras. (But see also [Ole82] for an early example of coinductive definitions by means of final coalgebras.) In [RT93], a first attempt is made towards systematizing these properties and the term 'final (coalgebra) semantics' is introduced. Examples of final coalgebra semantics appear in [RT94] (both with ordinary sets and with semi-lattices), [Acz94, Bal94, HL95, Har96] (with non-well-founded sets), [Fio93] (with complete partial orders), and [TJ93] (both with complete partial orders and with semi-lattices).

The above coalgebraic/functorial approach to the operational semantics of recursive programs deals neatly with top-level, mutually recursive definitions, but it ignores some aspects of the expressivity of the 'fix' operation, like the ability of dealing with local definitions and parameterized definitions: this is left to future work.

II

6 The Functorial Operational Semantics is Compositional

The semantics of a programming language is called **compositional** when the meaning of compound programs can be derived from the meaning of their subcomponents. A typical compositional semantics is obtained by defining the meaning of a program by induction, starting from a '**denotation**' $\langle\!|\, \sigma \,|\!\rangle$ for each n-ary program construct σ:

$$\langle\!|\, \sigma(t_1, \ldots, t_n) \,|\!\rangle^{\#} = \langle\!|\, \sigma \,|\!\rangle(\langle\!|\, t_1 \,|\!\rangle^{\#}, \ldots, \langle\!|\, t_n \,|\!\rangle^{\#})$$

This is called a **denotational semantics**.

A complete account of the meaning of a programming language requires both an operational and a denotational semantics. The former explains how a machine should execute programs, specifying their executable behaviours. The latter, because of its modularity, is better suited for reasoning about programs. The two meanings should be related in such a way that one should be able to infer from the denotational semantics the operational behaviour of the programs – up to a suitable abstraction. In other words, the denotational semantics of a language should be **adequate** wrt the operational semantics.

In general, much work is needed to prove the adequacy of a denotational semantics wrt an operational one. However, from operational semantics of transition systems defined by operational rules satisfying suitable *syntactic restrictions* (eg, the rules are in the *GSOS* format – see Section 11), it is possible to derive adequate denotational semantics systematically. (Cf notes below.)

Now, the novelty of the present functorial approach to operational semantics is that *every* functorial operational semantics coinduces a denotational semantics and, moreover, this denotational semantics is adequate wrt the operational one; as a corollary, every functorial operational semantics is compositional. Being formulated in terms of *abstract* notions of syntax and behaviour, this gives a general notion of 'well-behaved' operational semantics, based on purely mathematical properties. This encompasses and explains the 'syntactic' arguments otherwise used in the literature. (Cf Section 11.)

Assume, as usual, the (closed) programs of the language to be interpreted are the elements of the initial algebra of the endofunctor corresponding to some program constructs Σ. That is, let T be the syntactical monad of the language and $\Sigma T0 \cong T0$ the corresponding initial algebra of closed programs. Then, the problem of defining a denotational semantics can be reduced to the problem of finding a *suitable* Σ-algebra

$\langle D, \llparenthesis\text{-}\rrparenthesis\rangle$, whose carrier D is the **semantic domain** and whose structure

$$\llparenthesis\text{-}\rrparenthesis : \Sigma D \to D$$

is the set of denotations. The desired denotational interpretation of the programs is then the inductive extension of this algebra of denotations, that is, the unique algebra arrow $\llparenthesis\text{-}\rrparenthesis^{\#}$ from the initial algebra of programs to $\llparenthesis\text{-}\rrparenthesis : \Sigma D \to D$. Diagrammatically:

Initial Algebra Semantics

$$\begin{tikzcd}
\Sigma T0 \arrow[r, "\Sigma\llparenthesis\text{-}\rrparenthesis^{\#}"] \arrow[d, "\cong", "\text{initial algebra}"'] & \Sigma D \arrow[d, "\llparenthesis\text{-}\rrparenthesis\ =\ \text{denotations}"] \\
T0 \arrow[r, dashed, "\llparenthesis\text{-}\rrparenthesis^{\#}"'] & D
\end{tikzcd}$$

The restriction to closed programs is adopted only to simplify the presentation. In general, the interpretation of programs with variables $x \in X$, that is, for the elements of $TX \cong X + \Sigma TX$, is parametric in a 'valuation' function $\rho : X \to D$ mapping each variable to an element of the semantic domain D. Indeed, the inductive extension $\llparenthesis\text{-}\rrparenthesis^{\#}_{\rho} : TX \to D$ of the denotations $\llparenthesis\text{-}\rrparenthesis : \Sigma D \to D$ along the valuation $\rho : X \to D$ has the familiar clauses

$$\llparenthesis\, x \,\rrparenthesis^{\#}_{\rho} = \rho(x)$$

$$\llparenthesis\, \sigma(t_1, \ldots, t_n) \,\rrparenthesis^{\#}_{\rho} = \llparenthesis\, \sigma \,\rrparenthesis(\llparenthesis\, t_1 \,\rrparenthesis^{\#}_{\rho}, \ldots, \llparenthesis\, t_n \,\rrparenthesis^{\#}_{\rho})$$

The denotational model of a language is adequate wrt the operational one when it contains enough information to infer the *abstract behaviour* of the programs. Now, recall (from the previous section) that when the operational model of the (closed) programs can be expressed as a coalgebra structure $[\![\text{-}]\!] : T0 \to BT0$ of a behaviour B, then the abstract (global) behaviour of the programs is given by its coinductive extension, that is, by the corresponding final coalgebra semantics:

$$\begin{tikzcd}
T0 \arrow[r, dashed, "\text{final coalgebra semantics}", "{[\![\text{-}]\!]^{@}}"'] \arrow[d, "{[\![\text{-}]\!]}"'] & \widehat{B} \arrow[d, "\cong"', "\text{final coalgebra}"] \\
BT0 \arrow[r, "{B[\![\text{-}]\!]^{@}}"'] & B\widehat{B}
\end{tikzcd}$$

Then, in this setting, a denotational model is adequate wrt an operational one when its initial algebra semantics $\llparenthesis\text{-}\rrparenthesis^{\#} : T0 \to D$ is equal to the final coalgebra semantics $[\![\text{-}]\!]^{@} : T0 \to \widehat{B}$ corresponding to the operational model. Thus, in particular, the semantic domain D should be the carrier $\widehat{B}$ of the final coalgebra of the behaviour. Diagrammatically:

$$
\begin{tikzcd}[column sep=huge, row sep=large]
\Sigma T0 \arrow[r, "\Sigma\llparenthesis\text{-}\rrparenthesis^{\#}"] \arrow[d, "\text{initial algebra}"'] \arrow[dr, phantom, "\text{initial algebra semantics}"] & \Sigma\widehat{B} \arrow[d, "\llparenthesis\text{-}\rrparenthesis"] \\
T0 \arrow[r, shift left, "\llparenthesis\text{-}\rrparenthesis^{\#}"] \arrow[r, shift right, "\llbracket\text{-}\rrbracket^{@}"'] \arrow[r, phantom, "||"] \arrow[d, "\llbracket\text{-}\rrbracket"'] \arrow[dr, phantom, "\text{final coalgebra semantics}"] & \widehat{B} \arrow[d, "\text{final coalgebra}"] \\
BT0 \arrow[r, "B\llbracket\text{-}\rrbracket^{@}"'] & B\widehat{B}
\end{tikzcd}
$$

That is, for all programs $t \in T0$,

$$\llparenthesis\, t\, \rrparenthesis^{\#} = \llbracket t \rrbracket^{@}$$

As a corollary, the equivalence relation corresponding to the final coalgebra semantics is a *congruence*, that is, if, for $i = 1, \ldots, n$,

$$\llbracket t_i \rrbracket^{@} = \llbracket t'_i \rrbracket^{@}$$

then

$$\llbracket \sigma(t_1, \ldots, t_n) \rrbracket^{@} = \llbracket \sigma(t'_1, \ldots, t'_n) \rrbracket^{@}$$

for every n-ary operator $\sigma \in \Sigma$.

The above equivalence relation

$$t \sim t' \iff \llbracket t \rrbracket^{@} = \llbracket t' \rrbracket^{@}$$

is the **observational equivalence** corresponding to the operational semantics of the language, as it is determined by the abstract global behaviour of the programs, which is their intended *observable behaviour*. Now, if observational equivalence of a language is a congruence, one can systematically derive a denotational model adequate wrt the operational semantics. In turn, to ensure that the observational equivalence is a congruence one can impose *suitable* syntactic restrictions on the format of the operational rules. (Eg, *GSOS* – see Section 11.) This gives a satisfactory method to derive adequate denotational models from operational semantics, but it strongly relies on the assumption that the operational semantics is given in terms of structural rules for transition systems.

The novelty of the present functorial approach to operational semantics is that it gives a general notion of 'well-behaved' operational semantics formulated in terms of *abstract* notions of syntax and behaviour: every functorial operational semantics coinduces a denotational model adequate wrt it. As shown in Section 11, this purely mathematical approach encompasses – and explains – the above 'syntactic' method.

The denotational model coinduced by Φ. Let us now look at the actual construction of the denotations corresponding to a functorial operational semantics Φ. Recall that the operational monad $\Phi = <\Phi, \eta, \mu>$ is a lifting of the syntactical monad $T = <T, \eta, \mu>$ freely generated by the program constructs Σ. It is convenient to use the isomorphism, illustrated in Section 2, between the categories of Σ-algebras and T-algebras, and define the desired denotational model as a T-algebra rather than as a Σ-algebra. That is, let us look for an arrow

$$\llparenthesis\text{-}\rrparenthesis : T\widehat{B} \to \widehat{B}$$

such that the following diagrams commute.

$$\begin{tikzcd} T^2\widehat{B} \arrow[r, "T\llparenthesis\text{-}\rrparenthesis"] \arrow[d, "\mu_{\widehat{B}}"'] & T\widehat{B} \arrow[d, "\llparenthesis\text{-}\rrparenthesis"] \\ T\widehat{B} \arrow[r, "\llparenthesis\text{-}\rrparenthesis"'] & \widehat{B} \end{tikzcd} \qquad \begin{tikzcd} \widehat{B} \arrow[r, "\eta_{\widehat{B}}"] \arrow[dr, equal] & T\widehat{B} \arrow[d, "\llparenthesis\text{-}\rrparenthesis"] \\ & \widehat{B} \end{tikzcd}$$

The idea is to exploit the fact that $\widehat{B} \cong B\widehat{B}$ is a final coalgebra and that the operational monad Φ maps a coalgebra structure $k : X \to BX$ to a coalgebra structure $\Phi k : TX \to BTX$. Thus, by applying Φ to the final coalgebra isomorphism $\varphi : \widehat{B} \cong B\widehat{B}$, one obtains a coalgebra structure on $T\widehat{B}$:

$$\Phi\varphi : T\widehat{B} \to BT\widehat{B}$$

Its coinductive extension $(\Phi\varphi)^{@} : T\widehat{B} \to \widehat{B}$

$$\begin{tikzcd} T\widehat{B} \arrow[r, dashed, "(\Phi\varphi)^{@}"] \arrow[d, "\Phi\varphi"'] & \widehat{B} \arrow[d, "\varphi"] \\ BT\widehat{B} \arrow[r, "B(\Phi\varphi)^{@}"'] & B\widehat{B} \end{tikzcd} \tag{1}$$

is then the natural candidate for the desired denotational model $\llparenthesis\text{-}\rrparenthesis : T\widehat{B} \to \widehat{B}$.

Let us prove, using finality, that this arrow is a T-algebra indeed. Consider first the multiplication law:

$$\begin{tikzcd} T^2\widehat{B} \arrow[r, "T(\Phi\varphi)^{@}"] \arrow[d, "\mu_{\widehat{B}}"'] & T\widehat{B} \arrow[d, "(\Phi\varphi)^{@}"] \\ T\widehat{B} \arrow[r, "(\Phi\varphi)^{@}"'] & \widehat{B} \end{tikzcd} \tag{2}$$

This is the upper side of the cube

$$
\begin{tikzcd}
T^2\widehat{B} \arrow[rr, "T(\Phi\varphi)^{@}"] \arrow[dr, "\mu_{\widehat{B}}"] \arrow[dd, "\Phi^2\varphi"'] & & T\widehat{B} \arrow[dr, "(\Phi\varphi)^{@}"] & \\
& T\widehat{B} \arrow[rr, "(\Phi\varphi)^{@}"] \arrow[dd, "\Phi\varphi"'] & & \widehat{B} \arrow[dd, "\varphi"] \\
BT^2\widehat{B} \arrow[dr, "B\mu_{\widehat{B}}"'] & & & \\
& BT\widehat{B} \arrow[rr, "B(\Phi\varphi)^{@}"'] & & B\widehat{B}
\end{tikzcd}
$$

whose vertical sides all commute:

> The front side and the other (not visible) side underlying the arrow $(\Phi\varphi)^{@}$: $T\widehat{B} \to \widehat{B}$ are two copies of the definition of $(\Phi\varphi)^{@}$, hence commute. The back (not visible) side is the square
>
> $$
\begin{tikzcd}
T^2\widehat{B} \arrow[r, "T(\Phi\varphi)^{@}"] \arrow[d, "\Phi^2\varphi"'] & T\widehat{B} \arrow[d, "\Phi\varphi"] \\
BT^2\widehat{B} \arrow[r, "BT(\Phi\varphi)^{@}"'] & BT\widehat{B}
\end{tikzcd}
$$
>
> which is nothing but the image under the functor Φ of, once more, the square defining $(\Phi\varphi)^{@} : T\widehat{B} \to \widehat{B}$, hence, by *functoriality*, it commutes. Finally, the last vertical side is a square which commutes by the fact that, by definition of lifting of a monad, multiplication $\mu : T^2 \Rightarrow T$ of the syntactical monad T lifts to the multiplication $\mu : \Phi^2 \Rightarrow \Phi$ of the operational monad Φ.

Therefore, both composites $(\Phi\varphi)^{@} \circ \mu_{\widehat{B}}$ and $(\Phi\varphi)^{@} \circ T(\Phi\varphi)^{@}$ fit as the (unique!) coinductive extension of the coalgebra structure $\Phi^2\varphi : T^2\widehat{B} \to BT^2\widehat{B}$, hence they must be the same.

The proof of the other T-algebra law

$$
\begin{tikzcd}
\widehat{B} \arrow[r, "\eta_{\widehat{B}}"] \arrow[dr, equal] & T\widehat{B} \arrow[d, "(\Phi\varphi)^{@}"] \\
& \widehat{B}
\end{tikzcd}
$$

is similar and follows from the fact that $\eta : I \Rightarrow T$ lifts to the unit of the monad Φ:

$$\begin{tikzcd}
\widehat{B} \arrow[r, "\eta_{\widehat{B}}"] \arrow[d, "\varphi"'] & T\widehat{B} \arrow[d, "\Phi\varphi"] \\
B\widehat{B} \arrow[r, "B\eta_{\widehat{B}}"'] & BT\widehat{B}
\end{tikzcd}$$

(Notice this last commuting diagram tells us that, using the terminology of Section 3, the coalgebra $\langle T\widehat{B}, \Phi\varphi\rangle$ conservatively extends the final coalgebra $\langle \widehat{B}, \varphi\rangle$.)

Adequacy. Now, the claim is that the initial algebra semantics induced by the above denotational model

$$(\!|\text{-}|\!) = (\Phi\varphi)^{@} : T\widehat{B} \to \widehat{B}$$

is the same as the final coalgebra semantics coinduced by the operational model

$$[\![\text{-}]\!] = \Phi 0 : T0 \to BT0$$

That is,

$$\begin{tikzcd}
T^2 0 \arrow[rr, "T(\!|\text{-}|\!)^{\#}"] \arrow[d, "\text{initial algebra}\quad \mu_0"'] \arrow[drr, phantom, "\text{initial algebra semantics}"] & & T\widehat{B} \arrow[d, "(\!|\text{-}|\!) = (\Phi\varphi)^{@}"] \\
T0 \arrow[rr, shift left, "(\!|\text{-}|\!)^{\#}"] \arrow[rr, shift right, "{[\![\text{-}]\!]^{@}}"'] \arrow[rr, phantom, "||"] \arrow[d, "\Phi 0 = {[\![\text{-}]\!]}"'] \arrow[drr, phantom, "\text{final coalgebra semantics}"] & & \widehat{B} \arrow[d, "\varphi\quad \text{final coalgebra}"] \\
BT0 \arrow[rr, "{B[\![\text{-}]\!]^{@}}"'] & & B\widehat{B}
\end{tikzcd}$$

(Formally, the initial algebra semantics $(\!|\text{-}|\!)^{\#} : T0 \to \widehat{B}$ is the unique T-algebra arrow from the initial T-algebra $\langle T0, \mu_0\rangle$ to the denotational model $\langle \widehat{B}, (\!|\text{-}|\!)\rangle$. By the isomorphism between T- and Σ-algebras, it is the same as the initial algebra semantics of the Σ-algebra corresponding to the T-algebra $\langle \widehat{B}, (\Phi\varphi)^{@}\rangle$.)

This follows from the fact that everything in sight in the diagram

$$\begin{tikzcd}
T^2 0 \arrow[r, "T^2 0^@"] \arrow[d, "\mu_0", "\text{initial algebra}"'] & T^2\widehat{B} \arrow[r, "T(\Phi\varphi)^@"] \arrow[d, "\mu_{\widehat{B}}"] & T\widehat{B} \arrow[d, "(\Phi\varphi)^@ = \langle\!| \text{-} |\!\rangle"] \\
T0 \arrow[r, "T0^@"] \arrow[d, "\Phi 0 = [\![\text{-}]\!]"'] & T\widehat{B} \arrow[r, "(\Phi\varphi)^@"] \arrow[d, "\Phi\varphi"] & \widehat{B} \arrow[d, "\varphi", "\text{final coalgebra}"'] \\
BT0 \arrow[r, "BT0^@"'] & BT\widehat{B} \arrow[r, "B(\Phi\varphi)^@"'] & B\widehat{B}
\end{tikzcd}$$

commutes:

> The upper right square is the multiplication law (2) for the T-algebra structure $(\Phi\varphi)^@ : T\widehat{B} \to \widehat{B}$ and the lower right square is the defining square (1) of the arrow $(\Phi\varphi)^@ : T\widehat{B} \to \widehat{B}$. For the left squares first recall that 0 is the initial object in the base category $\mathbf{C}$. (Eg, in $\mathbf{C} = \mathbf{Set}$, 0 is the empty set.) and also recall the convention of writing $0 : 0 \to B0$ for the unique arrow from 0 to $B0$, which, by the way, is the structure of the initial B-coalgebra. The corresponding coinductive extension $0^@ : 0 \to \widehat{B}$ makes the diagram
>
> $$\begin{tikzcd}
0 \arrow[r, "0^@"] \arrow[d, "0"'] & \widehat{B} \arrow[d, "\varphi"] \\
B0 \arrow[r, "B0^@"'] & B\widehat{B}
\end{tikzcd}$$
>
> commute. (It is also the unique arrow from the initial object 0 to $\widehat{B}$.) Then the lower left square commutes because it is the image under the functor Φ of the above commuting square, and the upper left square commutes by the naturality of $\mu : T^2 \Rightarrow T$.

That is,

$$\langle\!| \text{-} |\!\rangle^{\#} = (\Phi\varphi)^@ \circ T0^@ = [\![\text{-}]\!]^@ : T0 \to \widehat{B}$$

Indeed, the composite arrow $(\Phi\varphi)^@ \circ T0^@ : T0 \to \widehat{B}$ is both a coalgebra arrow – hence the coinductive extension $[\![\text{-}]\!]^@$ – and an algebra arrow – hence the inductive extension $\langle\!| \text{-} |\!\rangle^{\#}$.

Equivalently,

$$((\Phi\varphi)^@)^{\#} = (\Phi\varphi)^@ \circ T0^@ = (\Phi 0)^@ : T0 \to \widehat{B}$$

Again, the restriction to closed programs is not essential. Given a set X of variables with a coalgebra structure $k : X \to BX$ on it, one has that the composite $(\Phi\varphi)^@ \circ Tk^@ : TX \to \widehat{B}$ is both the coinductive extension $(\Phi k)^@$ of the operational model $\Phi k : TX \to$

BTX and the inductive extension $(\!|\text{-}|\!)^{\#}_{\rho}$ of the denotational model along the valuation function

$$\rho = (\Phi k)^{@} \circ \eta_X : X \to \widehat{B}$$

That is,

$$\rho^{\sharp} = ((\Phi k)^{@} \circ \eta_X)^{\sharp} = (\Phi \varphi)^{@} \circ Tk^{@} = (\Phi k)^{@} : TX \to \widehat{B}$$

Example. Consider the functorial operational semantics corresponding to the rules $\mathcal{R}$ for the language

$$t ::= x \mid \mathsf{nil} \mid a \mid (t \ ; t)$$

The base category is **Set**. The syntactical monad $T = < T, \eta, \mu >$ is the one freely generated by the endofunctor

$$\Sigma X = 1 + \mathsf{Act} + X \times X$$

on **Set**. (Cf Section 2.) The behaviour is

$$BX = 1 + \mathsf{Act} \times X$$

whose coalgebras are the deterministic transition systems. (Cf Section 3.) Its final coalgebra $\langle \widehat{B}, \varphi \rangle$ has as carrier $\widehat{B}$ the set Act^{∞} of finite and infinite streams of actions in Act and the isomorphism $\varphi : \mathsf{Act}^{\infty} \cong 1 + \mathsf{Act} \times \mathsf{Act}^{\infty}$ applied to a non-empty stream $p = a \cdot p'$ in Act^{∞} returns a pair with first component a and second component the continuation p', while φ applied to the empty stream returns $*$. (Cf Section 5.) Equivalently, the final coalgebra $\langle \mathsf{Act}^{\infty}, \varphi \rangle$ can be seen as a deterministic transition system with finite and infinite streams as states and with transitions $p = a \cdot p' \xrightarrow{a} p'$.

Next, the set $T\widehat{B}$ is the set of terms over the constructs in Σ and with streams in Act^{∞} as variables. Thus, for instance, the term $a \ ; (a \cdot b)$ is in this set. (Notice the distinction between the first a which is a constant of the language and the second a which is the first element of the stream $a \cdot b$, which is a variable.) Also, all closed terms of the language belong to the set $T\widehat{B}$ and the function $T0^{@} : T0 \to T\widehat{B}$ is nothing but this inclusion.

The operational rules $\mathcal{R}$ for the language are the axioms

$$\mathsf{nil} \downarrow * \qquad \text{and} \qquad a \xrightarrow{a} \mathsf{nil}$$

and the three rules for sequential composition:

$$\frac{u \xrightarrow{a} u'}{u \ ; v \xrightarrow{a} u' \ ; v} \qquad \frac{u \downarrow * \quad v \xrightarrow{a} v'}{u \ ; v \xrightarrow{a} v'} \qquad \frac{u \downarrow * \quad v \downarrow *}{u \ ; v \downarrow *}$$

(Cf Section 3.) These rules induce an operational model denoted by

$$[\![\text{-}]\!]_{\mathcal{R}} : T0 \to BT0$$

such that $[\![\mathsf{nil}]\!]_{\mathcal{R}} = *$, $[\![a]\!]_{\mathcal{R}} = <a, \mathsf{nil}>$, and for all terms u, v,

$$[\![u \; ; \; v]\!]_{\mathcal{R}} \;=\; \begin{cases} <a, u' \; ; \; v> & \text{if } \; [\![u]\!]_{\mathcal{R}} = <a, u'> \\ <a, v'> & \text{if } \; [\![u]\!]_{\mathcal{R}} = * \; \text{ and } \; [\![v]\!]_{\mathcal{R}} = <a, v'> \\ * & \text{if } \; [\![u]\!]_{\mathcal{R}} = * = [\![v]\!]_{\mathcal{R}} \end{cases}$$

More generally, recall that every coalgebra structure $k : X \to BX$ can be seen as a set of axioms for the variables $x \in X$ by putting

$$x \xrightarrow{a} x'$$

if $k(x) = <a, x>$. Then, for every such k, the above rules $\mathcal{R}$ induce an operational model

$$[\![\text{-}]\!]^{k}_{\mathcal{R}} : TX \to BTX$$

which adds to the above the behaviours

$$[\![x]\!]^{k}_{\mathcal{R}} = k(x)$$

for every $x \in X$.

Consider now the inductive functorial operational semantics $\Phi = \widehat{\phi^{\mathcal{R}}}$ which the method in Section 4 assigns to the rules $\mathcal{R}$. It yields operational models

$$\Phi k = [\![\text{-}]\!]^{k}_{\lceil \mathcal{R} \rceil} : TX \to BTX$$

which differ from the above $[\![\text{-}]\!]^{k}_{\mathcal{R}}$ only in the treatment of the first case of sequential composition:

$$[\![u \; ; \; v]\!]^{k}_{\lceil \mathcal{R} \rceil} \;=\; <a, u' \; ; \; \gamma_{TX} [\![v]\!]^{k}_{\lceil \mathcal{R} \rceil}> \qquad \text{if } \; [\![u]\!]^{k}_{\lceil \mathcal{R} \rceil} = <a, u'>$$

where, recall, the transformation $\gamma : B \Rightarrow T$ is the embedding of the behaviour into the syntax mapping $*$ to nil and $< a, x >$ to $a \; ; \; x$. It is a retraction for the operational semantics in the sense that, in particular,

$$[\![\gamma_{TX} [\![v]\!]^{k}_{\lceil \mathcal{R} \rceil}]\!]^{k}_{\lceil \mathcal{R} \rceil} = [\![v]\!]^{k}_{\lceil \mathcal{R} \rceil}$$

This equation allows one to use the compositionality of functorial operational semantics to prove that the coinductive extension of $[\![\text{-}]\!]^{k}_{\lceil \mathcal{R} \rceil}$ is equal to the coinductive extension of $[\![\text{-}]\!]^{k}_{\mathcal{R}}$, that is,

$$([\![\text{-}]\!]^{k}_{\lceil \mathcal{R} \rceil})^{@} = ([\![\text{-}]\!]^{k}_{\mathcal{R}})^{@} : TX \to \widehat{B}$$

which implies that the abstract global behaviours corresponding to the former are the same as those corresponding to the latter, so that the two models are 'observationally equivalent' as claimed in Section 4. The proof is as follows.

$\mathcal{R}$ is observationally equivalent to $\lceil\mathcal{R}\rceil$. Recall that the definition of coinductive extension gives, for $[\![u \,;\, v]\!]^k_{\lceil\mathcal{R}\rceil} = <a, u' \,;\, \gamma_{TX}[\![v]\!]^k_{\lceil\mathcal{R}\rceil}>$,

$$([\![u \,;\, v]\!]^k_{\lceil\mathcal{R}\rceil})^@ = <a, ([\![u' \,;\, \gamma_{TX}[\![v]\!]^k_{\lceil\mathcal{R}\rceil}]\!]^k_{\lceil\mathcal{R}\rceil})^@>$$

Then, it is enough to show that

$$([\![u' \,;\, \gamma_{TX}[\![v]\!]^k_{\lceil\mathcal{R}\rceil}]\!]^k_{\lceil\mathcal{R}\rceil})^@ = ([\![u' \,;\, v]\!]^k_{\lceil\mathcal{R}\rceil})^@$$

If $\langle\!|\text{-}|\!\rangle : T\widehat{B} \to \widehat{B}$ is the denotational model coinduced by the operational monad $\Phi = \widehat{\phi^{\mathcal{R}}}$, one has, by the above adequacy result,

$$\begin{array}{rcll} ([\![u' \,;\, \gamma_{TX}[\![v]\!]^k_{\lceil\mathcal{R}\rceil}]\!]^k_{\lceil\mathcal{R}\rceil})^@ & = & \langle\!|\, ([\![u']\!]^k_{\lceil\mathcal{R}\rceil})^@ \,;\, ([\![\gamma_{TX}[\![v]\!]^k_{\lceil\mathcal{R}\rceil}]\!]^k_{\lceil\mathcal{R}\rceil})^@ \,|\!\rangle & \text{(adequacy)} \\ & = & \langle\!|\, ([\![u']\!]^k_{\lceil\mathcal{R}\rceil})^@ \,;\, ([\![v]\!]^k_{\lceil\mathcal{R}\rceil})^@ \,|\!\rangle & \text{(retraction)} \\ & = & ([\![u' \,;\, v]\!]^k_{\lceil\mathcal{R}\rceil})^@ & \end{array}$$

This concludes the proof.

The denotational model $\langle\!|\text{-}|\!\rangle : T\widehat{B} \to \widehat{B}$ coinduced by the operational monad $\Phi = \widehat{\phi^{\mathcal{R}}}$ is the coinductive extension of the operational model $[\![\text{-}]\!]^\varphi_{\lceil\mathcal{R}\rceil} : T\widehat{B} \to BT\widehat{B}$ which, from the above result that $\mathcal{R}$ is observationally equivalent to $\lceil\mathcal{R}\rceil$, is the same as the coinductive extension of $[\![\text{-}]\!]^\varphi_{\mathcal{R}} : \widehat{B} \to BT\widehat{B}$, that is,

$$\langle\!|\text{-}|\!\rangle = ([\![\text{-}]\!]^\varphi_{\mathcal{R}})^@ : T\widehat{B} \to \widehat{B}$$

By definition of coinductive extension, this gives, for every term $t \in T\widehat{B}$,

$$\langle\!|\, t \,|\!\rangle = \begin{cases} * & \text{if } [\![t]\!]^\varphi_{\mathcal{R}} = * \\ <a, \langle\!|\, t' \,|\!\rangle> & \text{if } [\![t]\!]^\varphi_{\mathcal{R}} = <a, t'> \end{cases}$$

Thus, in particular, the nil constant is denoted by $*$,

$$\langle\!|\, \mathsf{nil} \,|\!\rangle = *$$

every action a is denoted by the pair $<a, *>$,

$$\langle\!|\, a \,|\!\rangle = <a, \langle\!|\, \mathsf{nil} \,|\!\rangle> = <a, *>$$

and the denotation of the sequential composition of two streams p and q is

$$\langle\!|\, p \,;\, q \,|\!\rangle = \begin{cases} <a, \langle\!|\, p' \,;\, q \,|\!\rangle> & \text{if } p = a \cdot p' \\ <a, q'> & \text{if } p = \epsilon \text{ and } q = a \cdot q' \\ * & \text{if } p = \epsilon = q \end{cases}$$

The adequacy of this denotational model wrt the operational semantics induced by the rules $\mathcal{R}$, that is, the commutativity of

$$\begin{tikzcd} T^2 0 \arrow[r, "T[\![\text{-}]\!]^{@}_{\mathcal{R}}"] \arrow[d, "\mu_0"'] & T\widehat{B} \arrow[d, "(\!|\text{-}|\!) = ([\![\text{-}]\!]^{\varphi}_{\mathcal{R}})^{@}"] \\ T0 \arrow[r, "{[\![\text{-}]\!]^{@}_{\mathcal{R}}}"'] & \widehat{B} \end{tikzcd}$$

tells then that

$$(\!|\, C[\, [\![t]\!]^{@}_{\mathcal{R}} \,] \,|\!) \;=\; [\![\mu_0(C[\, t \,])]\!]^{@}_{\mathcal{R}}$$

for every context $C[\text{-}]$ and term t. (The multiplication $\mu_0 : T^2 0 \to T0$ is needed in order to make of the context $C[\text{-}]$ and of the term t a term in $T0$.) In particular, for the context with two 'holes' $(\text{-}\ ;\ \text{-})$ one has, omitting the multiplication μ_0, the equation

$$(\!|\, [\![u]\!]^{@}_{\mathcal{R}} \,;\, [\![v]\!]^{@}_{\mathcal{R}} \,|\!) \;=\; [\![u \,;\, v]\!]^{@}_{\mathcal{R}}$$

used in the above proof of the equivalence between $\mathcal{R}$ and $\lceil \mathcal{R} \rceil$.

Another consequence of the above adequacy is that programs with the same abstract global behaviour can be interchanged in any context. That is, if $[\![u]\!]^{@}_{\mathcal{R}} = [\![v]\!]^{@}_{\mathcal{R}}$, then $[\![C[\, u \,]]\!]^{@}_{\mathcal{R}} = [\![C[\, v \,]]\!]^{@}_{\mathcal{R}}$, or, equivalently, in terms of the observational equivalence $\sim$ introduced earlier in this section,

$$u \sim v \quad \text{implies} \quad C[\, u \,] \sim C[\, v \,].$$

Finally, notice that a denotational model is adequate also if the final coalgebra semantics is not equal to but only 'included' in the initial algebra semantics; that is, one can be more liberal and define a denotational model $\langle D, (\!|\text{-}|\!) \rangle$ to be adequate if it contains $\widehat{B}$ as a subalgebra and the inclusion sends the final coalgebra semantics to the initial algebra semantics:

$$\begin{tikzcd} & & D \\ T0 \arrow[urr, "(\!|\text{-}|\!)^{\#}"] \arrow[r, "{[\![\text{-}]\!]^{@}}"'] & \widehat{B} \arrow[ur, tail] & \end{tikzcd}$$

Notes. The relevance of initial algebras for semantics, type theory, and algebraic specification was recognized by the 'ADJ' group in the mid-seventies. (Some references on initial algebra semantics are [GTW78, MG85, Mos90, MT92].)

The idea of coupling initial algebra with final coalgebra semantics was first used in [RT94] to give a categorical account of the method described in [Rut92] for systematically deriving denotational models from structural operational semantics. (For precursors of this method see [Bad87, Rut90].) This method is based on results like those in [dS85, BIM88, GV92, Gro93] which show that the above notion of observational equivalence ('strong bisimulation') is a congruence if suitable restrictions are imposed on the syntactic format of the rules. (Cf Section 11.) This kind of results, although of great practical relevance, is very much dependent on the use of labelled transition systems and hard to export to other notions of operational model. Instead here the idea is that the structural rules correspond to the germ of an inductive functorial semantics, that is, they can be seen as an *action* of the syntax on the composite functor BT, for *abstract* notions of syntax T and behaviour B.

Like in the present approach, in [RT94] the denotational model is *coinduced* by the operational rules and the equivalence between initial algebra and final coalgebra semantics is proved by means of a four-squares diagram

$$\begin{tikzcd}
T^2 0 \arrow[r, "T^2 0^{@}"] \arrow[d, "\text{initial algebra}\ \mu_0"'] & T^2\widehat{B} \arrow[r, "T\llparenthesis - \rrparenthesis"] \arrow[d, "\mu_{\widehat{B}}"] & T\widehat{B} \arrow[d, "\llparenthesis - \rrparenthesis"] \\
T0 \arrow[r, "T0^{@}"] \arrow[d, "\llbracket - \rrbracket_{\mathcal{R}}"'] & T\widehat{B} \arrow[r, dashed, "\llparenthesis - \rrparenthesis"] \arrow[d, "\llbracket - \rrbracket^{\varphi}_{\mathcal{R}}"] & \widehat{B} \arrow[d, "\varphi\ \text{final coalgebra}"] \\
BT0 \arrow[r, "BT0^{@}"'] & BT\widehat{B} \arrow[r, "B\llparenthesis - \rrparenthesis"'] & B\widehat{B}
\end{tikzcd}$$

The difference is that, in order to ensure the commutativity of the upper right square, it is *assumed* in [RT94] that the observational equivalence coinduced by the operational semantics is a congruence, which fact, instead, becomes here a trivial consequence of functoriality. In fact, the *functorial* description of 'well-behaved' operational rules is the essence of the present approach.

7 A Dual Lifting: Functorial Denotational Semantics

A functorial operational semantics is a *monad* lifting the syntactical monad (freely generated by the signature) to the coalgebras of the behaviour. As shown in the previous section, this operational monad coinduces a denotational model. In fact, this denotational model is just one particular action of a '*comonad*' coinduced by the operational monad. This 'denotational comonad' is a lifting (to the algebras of the syntax) of another comonad, namely the 'observational comonad' cofreely generated by the behaviour.

The property that every operational monad coinduces a denotational comonad is the **basic property** of the functorial approach to operational semantics. Its dual also holds, namely every denotational comonad induces an operational monad; this gives a useful method to derive an operational semantics from a denotational one.

The notion of comonad is dual to the one of monad: a **comonad** on a category $\mathbf{C}$ is a triple

$$D = < D, \varepsilon, \delta >$$

with D an endofunctor on $\mathbf{C}$

$$D : \mathbf{C} \to \mathbf{C}$$

and with the **counit** ε and the **comultiplication** δ natural transformations

$$\varepsilon : D \Rightarrow I \qquad \delta : D \Rightarrow D^2$$

which satisfy the following laws.

Comonad Laws

$$\begin{tikzcd}
D \arrow[r, Rightarrow, "\delta"] \arrow[d, Rightarrow, "\delta"'] & D^2 \arrow[d, Rightarrow, "\delta_D"] \\
D^2 \arrow[r, Rightarrow, "D\delta"'] & D^3
\end{tikzcd}
\qquad\qquad
\begin{tikzcd}
 & D \arrow[dl, equal] \arrow[d, Rightarrow, "\delta"] \arrow[dr, equal] & \\
ID & D^2 \arrow[l, Rightarrow, "\varepsilon_D"] \arrow[r, Rightarrow, "D\varepsilon"'] & DI
\end{tikzcd}$$

A first example of a comonad is given by the **observational comonad** $D = < D, \varepsilon, \delta >$ cofreely generated by the behaviour endofunctor $BX = 1 + \mathsf{Act} \times X$ on **Set**. For every set X, the value of D at X is the carrier DX of the final coalgebra

$$DX \cong X \times B(DX)$$

of the endofunctor $(X \times B) : \mathbf{Set} \to \mathbf{Set}$. (Cf definition of TX in Section 2.) In particular, the value of D at singleton 1 – the final object of $\mathbf{Set}$ – is the carrier $\widehat{B} = \mathsf{Act}^\infty$ of the final B-coalgebra, because $1 \times X = X$. Thus $D1$ is the set of *abstract* global behaviours corresponding to B, that is, the finite and infinite streams generated by Act. (See Section 5.)

Now, a stream $(a_1 a_2 \cdots)$ can be seen as a sequence of transitions

$$\bullet \xrightarrow{a_1} \bullet \xrightarrow{a_2} \bullet \cdots$$

in which the states have no name or, equivalently, have all the same name $* \in \{*\} = 1$. Therefore, $D1$ is the set of global behaviours with a single state.

In general, the set DX is the set of global behaviours observable with states $x \in X$, that is, the finite and infinite sequences of transitions

$$x \xrightarrow{a_1} x_1 \xrightarrow{a_2} x_2 \cdots$$

with states $x \in X$ and actions $a \in \mathsf{Act}$. Formally, one can check that

$$DX = X + \coprod_{1 \leq \alpha \leq \omega} (X \times \mathsf{Act})^\alpha$$

The final coalgebra isomorphism $DX \cong X \times BDX$ splits into two projections:

$$X \xleftarrow{\mathsf{fst}_X} DX \cong X \times BDX \xrightarrow{\mathsf{snd}_X} BDX$$

These are the operations which allow one to *observe* these global behaviours: the first projection extracts the root of a global behaviour, the second projection gives its continuation. For instance:

$$x \quad \xleftarrow{\mathsf{fst}_X} \quad x \xrightarrow{a_1} x_1 \xrightarrow{a_2} x_2 \cdots \quad \xmapsto{\mathsf{snd}_X} \quad \xrightarrow{a_1} x_1 \xrightarrow{a_2} x_2 \cdots$$

The first projection $\mathsf{fst}_X : X \times BDX$ is the natural candidate for the value of the counit $\varepsilon : D \Rightarrow I$ at X:

$$\varepsilon_X = \mathsf{fst}_X : DX \to X$$

while the second projection can be coinductively extended to yield the comultiplication $\delta : D \Rightarrow D^2$. Indeed, by finality, the coalgebra $DX \cong X \times BDX$ comes with a *coinduction principle* which can be used to extend the operator D to an endofunctor and to define its comultiplication:

> Every $(X \times B)$-coalgebra structure $Y \to X \times BY$ is a pair $<f, k>$, with $f : Y \to X$ and $k : Y \to BY$. The first function can be seen as a 'covaluation' function, while the second is a B-coalgebra structure. By duality with the definition of inductive extensions along valuation

functions, call the corresponding coinductive extension $f^\flat = < f, k >^{@}$: $Y \to DX$

$$
\begin{tikzcd}
 & Y \arrow[r, "k"] \arrow[dl, "f"'] \arrow[d, dashed, "f^\flat"] & BY \arrow[d, "Bf^\flat"] \\
X & DX \arrow[l, "\varepsilon_X = \mathsf{fst}_X"] \arrow[r, "\mathsf{snd}_X"'] & BDX
\end{tikzcd}
$$

the **coinductive extension of** k **along the** covaluation **function** f.

Then, extend D to a functor by putting, for every function $f : X \to Y$,

$$Df = (f \circ \varepsilon_X)^\flat : DX \to DY$$

This function Df applied to a global behaviour $d_x = (x \xrightarrow{a_1} x_1 \xrightarrow{a_2} x_2 \cdots)$ substitutes every state in $d_x \in DX$ by its image under the 'renaming' f:

$$(Df)(d_x) = f(x) \xrightarrow{a_1} f(x_1) \xrightarrow{a_2} f(x_2) \cdots$$

Similarly, the value of the comultiplication $\delta : D \Rightarrow D^2$ at X is given by the coinductive extension of the second projection $\mathsf{snd}_X : DX \to BDX$ along the identity on DX:

$$
\begin{tikzcd}
 & DX \arrow[r, "\mathsf{snd}_X"] \arrow[dl, equal] \arrow[d, dashed, "\delta_X = (\mathsf{id}_{DX})^\flat"] & BDX \arrow[d, "B\delta_X"] \\
DX & D^2X \arrow[l, "\varepsilon_{DX}"] \arrow[r, "\mathsf{snd}_{DX}"'] & BD^2X
\end{tikzcd}
$$

The left triangle tells that ε is a left counit for δ. The proof that it is also a right counit and that δ is a comultiplication is dual to the proof in Section 2 for the unit η and the multiplication μ of the syntactical monad T.

Concretely, the comultiplication $\delta_X : DX \to D^2X$ maps a global behaviour $d_x = (x \xrightarrow{a_1} x_1 \xrightarrow{a_2} x_2 \cdots)$ to a global behaviour with the same transitions but with every state x_i replaced by its whole global behaviour d_{x_i}:

$$\delta_X(d_x) = (d_x \xrightarrow{a_1} d_{x_1} \xrightarrow{a_2} d_{x_2} \cdots)$$

In general, the coinductive extension of a coalgebra structure $k : Y \to BY$ along a function $f : Y \to X$ can be interpreted in terms of (deterministic) transition systems as follows. The B-coalgebra $\langle Y, k \rangle$ is a transition system with Y as set of states; the covaluation function $f : Y \to X$ maps every state $y \in Y$ to a state

$f(y) \in X$. Then, if the global behaviour of a state y in the transition system corresponding to $\langle Y, k\rangle$ is the (possibly infinite) sequence

$$y \xrightarrow{a_1} y_1 \xrightarrow{a_2} y_2 \cdots$$

the coinductive extension $f^\flat : Y \to DX$ maps y to this global behaviour, but replacing every state y_i by $f(y_i)$:

$$f^\flat(y_i) \;=\; f(y) \xrightarrow{a_1} f(y_1) \xrightarrow{a_2} f(y_2) \cdots$$

As an example, let the set Y of states be the set $\mathbb{Z}$ of integers and let the set Act of actions be trivial, that is, let Act be made of only one action a:

$$Y = \mathbb{Z} \qquad \text{and} \qquad \mathsf{Act} = \{a\}$$

Next, let the deterministic transition system corresponding to the coalgebra structure $k : \mathbb{Z} \to B(\mathbb{Z})$ be such that 0 is inert, a positive integer n performs a transition to its predecessor $n-1$, and a negative integer $-n$ performs a transition to its successor $-n+1$:

$$0 \downarrow * \qquad n \xrightarrow{a} n-1 \qquad -n \xrightarrow{a} -n+1$$

Now, if X is the three-elements set $\{0, \clubsuit, \diamondsuit\}$ and $f : \mathbb{Z} \to \{0, \clubsuit, \diamondsuit\}$ is the function mapping 0 to 0, positive numbers to $\diamondsuit$, and negative numbers to $\clubsuit$, then the coinductive extension

$$f^\flat : \mathbb{Z} \to D\{0, \clubsuit, \diamondsuit\}$$

of the transition system along this covaluation function f maps every integer z to a sequence of a-transitions of length $|z|$ having 0 as last state and $\diamondsuit$ (resp., $\clubsuit$) as all other states if z is positive (resp., negative). Thus, for instance,

$$f^\flat(3) \;=\; \diamondsuit \xrightarrow{a} \diamondsuit \xrightarrow{a} \diamondsuit \xrightarrow{a} 0 \qquad f^\flat(-3) \;=\; \clubsuit \xrightarrow{a} \clubsuit \xrightarrow{a} \clubsuit \xrightarrow{a} 0$$

Notice that the same set $X = \{0, \clubsuit, \diamondsuit\}$ can be used to observe the global behaviours of the above transition system in quite a different way. Consider the function $g : \mathbb{Z} \to \{0, \clubsuit, \diamondsuit\}$ mapping odd numbers to $\clubsuit$ and even numbers to $\diamondsuit$. Then the coinductive extension $g^\flat : \mathbb{Z} \to \{0, \clubsuit, \diamondsuit\}$ of the transition system along g identifies n and $-n$. For instance:

$$g^\flat(3) \;=\; \clubsuit \xrightarrow{a} \diamondsuit \xrightarrow{a} \clubsuit \xrightarrow{a} 0 \;=\; g^\flat(-3)$$

The same identification can be obtained by setting $X = 1$ and thus forcing the covaluation function to map everything to the same state $\bullet \in \{\bullet\} = 1$. Then, the coinductive extension of k along this trivial function $Y \to 1$ is nothing but the simple coinductive extension

$$k^@ : Y \to \widehat{B} = D1 \cong 1 \times BD1 \cong BD1 = B\widehat{B}$$

of k (see Section 5). In particular,

$$k^@(3) \;=\; \bullet \xrightarrow{a} \bullet \xrightarrow{a} \bullet \xrightarrow{a} \bullet \;=\; k^@(-3)$$

Consider now, for an arbitrary comonad $D =< D, \varepsilon, \delta >$ in a category $\mathbf{C}$, the **category $\mathbf{C}_D$ of D-coalgebras**. It is the category of coalgebras of the endofunctor D which 'respect' the counit ε and the comultiplication δ of the comonad D; that is, its objects are pairs $\langle X, k\rangle$, with X an object of $\mathbf{C}$ and $k : X \to DX$ an arrow of $\mathbf{C}$ satisfying the laws

$$
\begin{tikzcd}
X \arrow[r, "k"] \arrow[d, "k"'] & DX \arrow[d, "\delta_X"] \\
DX \arrow[r, "Dk"'] & D^2X
\end{tikzcd}
\qquad\qquad
\begin{tikzcd}
 & X \arrow[d, "k"] \arrow[dl, equal] \\
X & DX \arrow[l, "\varepsilon_X"]
\end{tikzcd}
$$

and its arrows $f : \langle X, k\rangle \to \langle Y, h\rangle$ are arrows $f : X \to Y$ of $\mathbf{C}$ such that $Df \circ k = h \circ f$.

> **B-coalgebras are D-coalgebras.** There is an isomorphism between the category of coalgebras of an endofunctor B and the coalgebras of its cofreely generated comonad D. This isomorphism maps every B-coalgebra $\langle X, k\rangle$ to the D-coalgebra with same carrier X and with structure the coinductive extension of k along the identity on X:
>
> $$
\begin{tikzcd}
 & X \arrow[r, "k"] \arrow[d, dashed, "(\mathsf{id}_X)^\flat"] \arrow[dl, equal] & BX \arrow[d, "B(\mathsf{id}_X)^\flat"] \\
X & DX \arrow[l, "\varepsilon_X"] \arrow[r, "\mathsf{snd}_X"'] & BDX
\end{tikzcd}
$$
>
> The inverse of this isomorphism is obtained by composing each D-coalgebra structure $k : X \to DX$ first with the second projection $\mathsf{snd}_X : X \times BDX \to BDX$ and then with $B\varepsilon_X : BDX \to BX$. That is:
>
> $$\langle X, k\rangle \mapsto \langle X, B\varepsilon_X \circ \mathsf{snd}_X \circ k\rangle$$
>
> The proof is simply the dual of the proof that *Σ-algebras are T-algebras* given in Section 2.

Notice that, under the above isomorphism of categories, the final B-coalgebra $\widehat{B} \cong B\widehat{B}$ corresponds to the cofree D-coalgebra over the final object, namely $\langle P1, \delta_1\rangle$, just like the initial Σ-algebra corresponds to $\langle T0, \mu_0\rangle$, the free T-algebra over the initial object.

The dualities between signature and syntactical monad on the one side and behaviour and observational comonad on the other side can be summarized as follows.

Signature $\Sigma : \mathbf{C} \to \mathbf{C}$	Behaviour $B : \mathbf{C} \to \mathbf{C}$
Algebras	Coalgebras
$X{+}\Sigma TX \cong TX$ = initial $(X{+}\Sigma)$-algebra	$DX \cong X{\times}BDX$ = final $(X{\times}B)$-coalgebra
Induction $(\text{-})^{\#}$	Coinduction $(\text{-})^{@}$
$\eta = \mathsf{inl} : I \Rightarrow T$	$\varepsilon = \mathsf{fst} : P \Rightarrow I$
$\mu = [\mathsf{id}, \mathsf{inr}]^{\#} = \mathsf{id}^{\sharp} : T^2 \Rightarrow T$	$\delta = <\mathsf{id}, \mathsf{snd}>^{@} = \mathsf{id}^{\flat} : P \Rightarrow P^2$
Syntactical Monad $T{=}{<}T, \eta, \mu{>}$	Observational Comonad $D{=}{<}D, \varepsilon, \delta{>}$
TX = Programs	DX = Global Behaviours
$\mathbf{C}^{\Sigma} \cong \mathbf{C}^{T}$	$\mathbf{C}_B \cong \mathbf{C}_D$
$\langle T0, \mu_0 \rangle$ = Initial Algebra	$\langle D1, \delta_1 \rangle$ = Final Coalgebra

Next, notice that the isomorphism between B- and D-coalgebras implies that every operational monad $\Phi = <\Phi, \eta, \mu>$ can be seen as a lifting of the syntactical monad $T{=}{<}T, \eta, \mu{>}$ to the coalgebras of the observational comonad D rather than to the coalgebras of the behaviour B (and vice versa). Thus, writing $U_D : \mathbf{C}_D \to \mathbf{C}$ for the forgetful functor mapping a D-coalgebra $\langle X, k \rangle$ to its carrier X, one has

Operational Monad

$$
\begin{tikzcd}
\mathbf{C}_D \arrow[r, "\Phi"] \arrow[d, "U_D"'] & \mathbf{C}_D \arrow[d, "U_D"] \\
\mathbf{C} \arrow[r, "T"'] & \mathbf{C}
\end{tikzcd}
$$

That is, for every D-coalgebra structure $k : X \to DX$, one has that $\Phi k : TX \to DTX$ is also a D-coalgebra structure and, moreover, the two squares in the diagram

$$
\begin{tikzcd}
X \arrow[r, "\eta_X"] \arrow[d, "k"'] & TX \arrow[d, "\Phi k"'] & T^2X \arrow[l, "\mu_X"'] \arrow[d, "\Phi^2 k"] \\
DX \arrow[r, "D\eta_X"'] & DTX & DT^2X \arrow[l, "D\mu_X"]
\end{tikzcd}
$$

commute.

In the above form, the definition of functorial operational semantics can be easily dualized to yield the definition of **functorial denotational semantics**, namely as a comonad Ψ lifting the observational comonad $D\!=<D, \varepsilon, \delta>$ to the T-algebras:

Denotational Comonad

$$\begin{tikzcd}
\mathbf{C}^T \arrow[r, "\Psi"] \arrow[d, "U^T"'] & \mathbf{C}^T \arrow[d, "U^T"] \\
\mathbf{C} \arrow[r, "D"'] & \mathbf{C}
\end{tikzcd}$$

That is, Ψ is a comonad with counit and comultiplication inherited from the observational comonad $D\!=<D, \varepsilon, \delta>$

$$\Psi\!=<\Psi, \varepsilon, \delta>$$

and with $\Psi : \mathbf{C}^T \to \mathbf{C}^T$ such that

$$U^T\Psi = DU^T : \mathbf{C}^T \to \mathbf{C}$$

Equivalently, Ψ is an **action** of the monad T on the composite functor $DU^T : \mathbf{C}^T \to \mathbf{C}$, ie a natural transformation

$$\Psi : TDU^T \Rightarrow DU^T$$

such that, for every T-algebra $h : TX \to X$, $\Psi h : TDX \to DX$ is also a T-algebra. Therefore, the fact that the counit and comultiplication of the observational comonad D lift to those of the denotational comonad Ψ is equivalent to the commutativity of the two squares in the following diagram.

$$\begin{tikzcd}
TX \arrow[d, "h"'] & TDX \arrow[l, "T\varepsilon_X"'] \arrow[r, "T\delta_X"] \arrow[d, "\Psi h"'] & TD^2X \arrow[d, "\Psi^2 h"] \\
X & DX \arrow[l, "\varepsilon_X"] \arrow[r, "\delta_X"'] & D^2X
\end{tikzcd}$$

(Cf Section 4.)

The basic property of the functorial approach to operational semantics can now be stated.

The denotational comonad $\Phi^@$ coinduced by an operational monad Φ.

Every operational monad $\Phi = <\Phi, \eta, \mu>$ lifting a syntactical monad $T = <T, \eta, \mu>$ to the coalgebras of an observational comonad $D = <D, \varepsilon, \delta>$ coinduces an endofunctor

$$\Phi^@ : \mathbf{C}^T \to \mathbf{C}^T$$

such that $\Phi^@ = <\Phi^@, \varepsilon, \delta>$ is a denotational comonad lifting D to the T-algebras:

$$\begin{tikzcd} \mathbf{C}_D \arrow[r, "\Phi"] \arrow[d, "U_D"'] & \mathbf{C}_D \arrow[d, "U_D"] \\ \mathbf{C} \arrow[r, "T"'] & \mathbf{C} \end{tikzcd} \quad \longmapsto \quad \begin{tikzcd} \mathbf{C}^T \arrow[r, "\Phi^@"] \arrow[d, "U^T"'] & \mathbf{C}^T \arrow[d, "U^T"] \\ \mathbf{C} \arrow[r, "D"'] & \mathbf{C} \end{tikzcd}$$

The endofunctor $\Phi^@$ on the T-algebras is defined by coinduction as follows. For simplicity, recalling the isomorphism $\mathbf{C}_D \cong \mathbf{C}_B$ between the categories of D- and B-coalgebras, consider the operational monad Φ to be on the B-coalgebras rather than on the D-coalgebras. Now, one needs, for every T-algebra structure $h : TX \to X$, a T-algebra structure $\Phi^@ h : TDX \to DX$. Therefore, first apply the given operational monad Φ to the B-coalgebra structure

$$\mathsf{snd}_X : DX \to BDX$$

obtaining the B-coalgebra structure

$$\Phi(\mathsf{snd}_X) : TDX \to BTDX$$

and then take the coinductive extension of this coalgebra structure $\Phi(\mathsf{snd}_X)$ along the composite arrow $h \circ T\varepsilon_X : TDX \to X$

$$\begin{tikzcd} TX \arrow[d, "h"'] & TDX \arrow[l, "T\varepsilon_X"'] \arrow[r, "\Phi\,\mathsf{snd}_X"] \arrow[d, dashed, "\Phi^@ h = (h \circ T\varepsilon_X)^\flat"] & BTDX \arrow[d, "B\Phi^@ h"] \\ X & DX \arrow[l, "\varepsilon_X"] \arrow[r, "\mathsf{snd}_X"'] & BDX \end{tikzcd}$$

That is,

$$\Phi^@ h = <h \circ T\varepsilon_X, \Phi(\mathsf{snd}_X)>^@ = (h \circ T\varepsilon_X)^\flat : TDX \to DX$$

The claim is threefold: (i) $\Phi^@ h : TDX \to DX$ is a T-algebra structure, (ii) the operation $\Phi^@$ is functorial, and (iii) the counit and comultiplication of the observational comonad $D = < D, \varepsilon, \delta >$ lift to counit and comultiplication for $\Phi^@$. The proofs are all by coinduction.

Let us start from (iii), that is, from the claim that the two squares in the diagram

$$\begin{tikzcd}
TX \arrow[d, "h"'] & TDX \arrow[l, "T\varepsilon_X"'] \arrow[r, "T\delta_X"] \arrow[d, "\Phi^{@}h"'] & TD^2X \arrow[d, "\Phi^{@^2}h"] \\
X & DX \arrow[l, "\varepsilon_X"] \arrow[r, "\delta_X"'] & D^2X
\end{tikzcd}$$

commute:

> The left square of the above diagram commutes by definition. As for the right square, it commutes because both composite arrows $\Phi^{@^2}h \circ T\delta_X$ and $\delta_X \circ \Phi^{@}h$ from TDX to X fit as the (unique!) coinductive extension
>
> $$\begin{tikzcd}
& TDX \arrow[r, "\Phi\,\mathsf{snd}_X"] \arrow[dd, dashed, "!"] & BTDX \arrow[dd, "B!"] \\
TDX \arrow[d, "\Phi^{@}h"'] \arrow[ur, phantom, "/\!\!/"] & & \\
DX & D^2X \arrow[l, "\varepsilon_{DX}"] \arrow[r, "\mathsf{snd}_{DX}"'] & BD^2X
\end{tikzcd}$$
>
> of the coalgebra structure $\Phi(\mathsf{snd}_X) : TDX \to BTDX$ along the arrow $\Phi^{@}h : TDX \to DX$. Indeed:
>
> $$\begin{tikzcd}
& TDX \arrow[r, "\Phi\,\mathsf{snd}_X"] \arrow[d, "T\delta_X"] & BTDX \arrow[d, "BT\delta_X"] \\
TDX \arrow[d, "\Phi^{@}h"'] \arrow[ur, phantom, "/\!\!/"] & TD^2X \arrow[l, "T\varepsilon_{DX}"] \arrow[r, "\Phi\,\mathsf{snd}_{DX}"'] \arrow[d, "\Phi^{@^2}h"] & BTD^2X \arrow[d, "B\Phi^{@^2}h"] \\
DX & D^2X \arrow[l, "\varepsilon_{DX}"] \arrow[r, "\mathsf{snd}_{DX}"'] & BD^2X
\end{tikzcd}
\qquad
\begin{tikzcd}
& TDX \arrow[r, "\Phi\,\mathsf{snd}_X"] \arrow[d, "\Phi^{@}h"] & BTDX \arrow[d, "B\Phi^{@}h"] \\
TDX \arrow[d, "\Phi^{@}h"'] \arrow[ur, phantom, "/\!\!/"] \arrow[dr, phantom, "/\!\!/"] & DX \arrow[r, "\mathsf{snd}_X"'] \arrow[d, "\delta_X"] & BDX \arrow[d, "B\delta_X"] \\
DX & D^2X \arrow[l, "\varepsilon_{DX}"] \arrow[r, "\mathsf{snd}_{DX}"'] & BD^2X
\end{tikzcd}$$

Next, consider the claim (i) that the arrow $\Phi^{@}h : TDX \to DX$ is a T-algebra structure, that is,

$$\Phi^{@}h \circ T\Phi^{@}h = \Phi^{@}h \circ \mu_{DX} \qquad \text{and} \qquad \Phi^{@}h \circ \eta_{DX} = \mathsf{id}_{DX}$$

The first equation holds because both $\Phi^@h \circ T\Phi^@h$ and $\Phi^@h \circ \mu_X$ fit as the coinductive extension of the coalgebra structure $\Phi^2(\mathsf{snd}_X) : T^2DX \to BT^2DX$ along the arrow $(h \circ Th) \circ T^2\varepsilon_X = (h \circ \mu_X) \circ T^2\varepsilon_X : T^2DX \to X$

$$\begin{tikzcd}
 & T^2X \arrow[dl, "Th"'] \arrow[dr, "\mu_X"] & & T^2DX \arrow[ll, "T^2\varepsilon_X"'] \arrow[r, "\Phi^2\mathsf{snd}_X"] \arrow[dd, dashed, "!"] & BT^2DX \arrow[dd, "B!"] \\
TX \arrow[dr, "h"'] & & TX \arrow[dl, "h"] & & \\
 & X & & DX \arrow[ll, "\varepsilon_X"] \arrow[r, "\mathsf{snd}_X"'] & BDX
\end{tikzcd}$$

$$\begin{tikzcd}
T^2DX \arrow[r, "\Phi^2\mathsf{snd}_X"] \arrow[d, "T\Phi^@h"'] & BT^2DX \arrow[d, "BT\Phi^@h"] \\
TDX \arrow[r, "\Phi\,\mathsf{snd}_X"'] & BTDX
\end{tikzcd}
\qquad
\begin{tikzcd}
T^2DX \arrow[r, "\Phi^2\mathsf{snd}_X"] \arrow[d, "\mu_{DX}"'] & BT^2DX \arrow[d, "B\mu_{DX}"] \\
TDX \arrow[r, "\Phi\,\mathsf{snd}_X"'] & BTDX
\end{tikzcd}$$

Similarly, the second equation, namely $\Phi^@h \circ \eta_{DX} = \mathsf{id}_{DX}$, holds because both $\Phi^@h \circ \eta_{DX}$ and the identity on DX fit as the coinductive extension of the coalgebra structure $\Phi(\mathsf{snd}_X) : TDX \to BTDX$ along $\varepsilon_X = \mathsf{id}_X \circ \varepsilon_X = (h \circ \eta_X) \circ \varepsilon_X : DX \to X$

$$\begin{tikzcd}
 & X \arrow[dl, "\eta_X"'] \arrow[dd, phantom, "\|"] & DX \arrow[l, "\varepsilon_X"'] \arrow[r, "\Phi\,\mathsf{snd}_X"] \arrow[dd, dashed, "!"] & BTDX \arrow[dd, "B!"] \\
TX \arrow[dr, "h"'] & & & \\
 & X & DX \arrow[l, "\varepsilon_X"] \arrow[r, "\mathsf{snd}_X"'] & BD^2X
\end{tikzcd}$$

Finally, the claim (ii) that, for every T-algebra arrow $f : \langle X, h\rangle \to \langle Y, k\rangle$, the operation

$$\begin{tikzcd}
TX \arrow[r, "Tf"] \arrow[d, "h"'] & TY \arrow[d, "k"] \\
X \arrow[r, "f"'] & Y
\end{tikzcd}
\quad \longmapsto \quad
\begin{tikzcd}
TDX \arrow[r, "TDf"] \arrow[d, "\Phi^@h"'] & TDY \arrow[d, "\Phi^@k"] \\
DX \arrow[r, "\Phi^@f = Df"'] & DY
\end{tikzcd}$$

is functorial amounts to

$$\Phi^@g \circ \Phi^@f = \Phi^@(g \circ f) \qquad \text{and} \qquad \Phi^@\mathsf{id}_X = \mathsf{id}_{DX}$$

for every T-algebra arrow $g : \langle Y, k\rangle \to \langle Z, l\rangle$. Its proof is similar to the one of (i) and left to the reader.

Notice that the above construction applies to any lifting of a (not necessarily freely generated) monad to the coalgebras of a cofreely generated comonad on any category.

As an example, consider the denotational comonad coinduced by the operational monad $\Phi = \widehat{\phi^{\mathcal{R}}}$ corresponding to the rule $\mathcal{R}$ for the sample language

$$t ::= x \mid \mathsf{nil} \mid a \mid (t \,;\, t)$$

Thus, using the notation of Section 4, $\Phi\,\mathsf{snd}_X = [\![\text{-}]\!]^{\mathsf{snd}_X}_{\lceil\mathcal{R}\rceil} : TDX \to BTDX$. As a short-hand, write

$$\Phi\,\mathsf{snd}_X = [\![\text{-}]\!]_X : TDX \to BTDX$$

Next, recall the set DX is the set of global behaviours

$$d_x \;=\; x \xrightarrow{a_1} x_1 \xrightarrow{a_2} x_2 \cdots$$

with $x_i \in X$; the counit $\varepsilon_X : DX \to X$ is the operation returning the root x of a global behaviour d_x and the second projection $\mathsf{snd}_X : DX \to BDX$ returns its continuation.

Then, the value of the corresponding coinduced denotational comonad $\Phi^{@}$ at a T-algebra structure $h : TX \to X$ is

$$\begin{tikzcd}
TX \arrow[d, "h"'] & TDX \arrow[l, "T\varepsilon_X"'] \arrow[r, "{[\![\text{-}]\!]_X}"] \arrow[d, dashed, "{\Phi^{@}h = <h \circ T\varepsilon_X, [\![\text{-}]\!]_X>^{@}}"'] & BTDX \arrow[d, "B\Phi^{@}X"] \\
X & DX \arrow[l, "\varepsilon_X"] \arrow[r, "\mathsf{snd}_X"'] & BDX
\end{tikzcd}$$

which gives, for all global behaviours $d_x, d_y \in DX$,

$$(\Phi^{@}h)(d_x \,;\, d_y) \;=\; \langle h(x \,;\, y), (\Phi^{@}h)(\mathsf{snd}_X(d_x) \,;\, d_y)\rangle$$

if $\mathsf{snd}_X(d_x)$ is different from $*$. Thus, for instance, the term

$$d_x \,;\, d_y \;=\; (x \xrightarrow{a_1} x_1 \xrightarrow{a_2} x_2) \,;\, (y \xrightarrow{b_1} y_1 \cdots)$$

is mapped to the global behaviour

$$h(x \,;\, y) \xrightarrow{a_1} h(x_1 \,;\, y) \xrightarrow{a_2} h(x_2 \,;\, y) \xrightarrow{b_1} y_1 \cdots$$

That is, the meaning of the sequential composition of two global behaviours d_x and d_y is obtained by first concatenating d_y to d_x and then replacing the states x_i of d_x by $h(x_i \,;\, y)$, where y is the root of d_y, while apart from y which is removed, all states of d_y are left the same.

In particular, consider X equal to the singleton 1, the final object in **Set**. There exists only one function from $T1$ to 1, namely the trivial function $1 : T1 \to 1$ mapping every term of $T1$ to the state $\bullet \in \{\bullet\} = 1$. Next, the set $D1 \cong 1 \times BD1 \cong BD1$ is the carrier of the final coalgebra $\widehat{B} \cong B\widehat{B}$ and, moreover, the structure $\mathsf{snd}_1 : D1 \to BD1$ is isomorphic to the final coalgebra isomorphism $\varphi : \widehat{B} \cong B\widehat{B}$. That is,

$$\langle D1, \mathsf{snd}_1 \rangle \cong \langle \widehat{B}, \varphi \rangle$$

Then, the T-algebra structure $\Phi^{@}1 : TD1 \to D1$ is isomorphic to the canonical denotational model

$$\begin{tikzcd} T\widehat{B} \arrow[r, dashed, "(\Phi\varphi)^{@}"] \arrow[d, "\Phi\varphi"'] & \widehat{B} \arrow[d, "\varphi"] \\ BT\widehat{B} \arrow[r, "B(\Phi\varphi)^{@}"'] & B\widehat{B} \end{tikzcd}$$

given in the previous section.

Finally, consider the dual of the above construction, namely

The operational monad $\Psi^{\#}$ induced by a denotational comonad Ψ.

Every denotational comonad $\Psi = <\Psi, \varepsilon, \delta>$ lifting an observational comonad $D = < D, \varepsilon, \delta>$ to the algebras of a syntactical monad $T = <T, \eta, \mu>$ induces an endofunctor

$$\Psi^{\#} : \mathbf{C}_D \to \mathbf{C}_D$$

such that $\Psi^{\#} = <\Psi^{\#}, \eta, \mu>$ is an operational monad lifting T to the D-coalgebras:

$$\begin{tikzcd} \mathbf{C}^T \arrow[r, "\Psi"] \arrow[d, "U^T"'] & \mathbf{C}^T \arrow[d, "U^T"] & & \mathbf{C}_D \arrow[r, "\Psi^{\#}"] \arrow[d, "U_D"'] & \mathbf{C}_D \arrow[d, "U_D"] \\ \mathbf{C} \arrow[r, "D"'] & \mathbf{C} \arrow[r, phantom, "\longmapsto"] & {} & \mathbf{C} \arrow[r, "T"'] & \mathbf{C} \end{tikzcd}$$

The endofunctor $\Psi^{\#}$ on the D-coalgebras is defined by induction as follows.

$$\begin{tikzcd} X \arrow[r, "\eta_X"] \arrow[d, "k"'] & TX \arrow[d, dashed, "\Psi^{\#}k \,=\, (D\eta_X \circ k)^{\sharp}"] & \Sigma TX \arrow[l, "\mathsf{inr}_X"'] \arrow[d, "\Sigma\Psi^{\#}h"] \\ DX \arrow[r, "D\eta_X"'] & DTX & \Sigma DTX \arrow[l, "\Psi\,\mathsf{inr}_X"] \end{tikzcd}$$

That is, for every coalgebra structure $k : X \to DX$,

$$\Psi^{\#}k = [D\eta_X \circ k, \Psi(\mathsf{inr}_X)]^{\#} = (D\eta_X \circ k)^{\sharp} : TX \to DTX$$

(Again, for simplicity, the denotational monad Ψ is assumed to be on the Σ-algebras rather than on the isomorphic category of T-algebras.)

Notes. Comonads in semantics appear in Brookes and Geva's work [BG92], which bears resemblance with Moggi's work on computational monads [Mog91]. The *computational comonads* defined in [BG92] are comonads $D{=}{<}D, \varepsilon, \delta{>}$ with an extra operation $\gamma : I \Rightarrow D$ such that

$$\varepsilon \circ \gamma = \mathsf{id} \qquad \text{and} \qquad \delta \circ \gamma = \gamma_T \circ \gamma$$

The type D is the type of computations and the operation γ embeds data into computations. For instance, the observational comonad D cofreely generated by the endofunctor $X \mapsto 1 + X$ is a computational comonad as well: the set DX is the set X^∞ of finite and infinite sequences of $x \in X$ and the operation $\gamma : I \Rightarrow D$ 'saturates' every $x \in X$ by mapping it to the infinite sequence x^ω.

Brookes and Geva's work focuses on the ('co-Kleisli') subcategory of cofree coalgebras of a computational comonad rather than on the full category of coalgebras as in the present work. It would be interesting to understand whether there is a closer relationship between the two notions "computational comonad" and "observational comonad".

As pointed out to this author by Axel Poigné, liftings of functors to algebras of monads were studied in [Joh75]. In particular, Lemma 1 of [Joh75] shows that such liftings are in one-to-one correspondence with distributive laws (cf Section 4); in particular, every lifting of an *endofunctor* (thus without comonad operations!) D to the T-algebras is equivalent to a distributive law of the monad T over the endofunctor D.

The systematic method introduced in this section for deriving operational models from denotational ones is simply the dual to the already known method for deriving denotational models from operational ones. The existence of such a method had been forecasted in Section 5.3 of [RT94] (thanks to the mixed algebraic/coalgebraic approach used there which already allowed for a dualization), yet it had never been described before. (In general, one of the advantages of bringing to light the categorical structure underlying a given phenomenon is that then the mighty *duality principle* can be applied.) A concrete example of an operational monad $\Psi^\#$ induced by a denotational comonad Ψ is given in Section 10, where it is used to prove that 'basic process algebra' is functorial.

8 Operational is Denotational

The coinductive construction $\Phi \mapsto \Phi^{@}$ is a bijection between operational monads and denotational comonads whose inverse is the inductive construction $\Psi \mapsto \Psi^{\#}$. The proof of this fact is given in terms of adjunctions.

Let us rephrase the inductive construction at the end of the previous section of an operational monad $\Psi^{\#}$ from a denotational comonad Ψ in terms of adjunctions.

Recall, for every D-coalgebra structure $k : X \to DX$, the structure $\Psi^{\#}k : TX \to DTX$ is defined as the inductive extension of k along the composite $D\eta_X \circ k : X \to DTX$.

$$\begin{tikzcd}
X \arrow[r, "\eta_X"] \arrow[d, "k"'] & TX \arrow[d, dashed, "\Psi^{\#}k \;=\; (D\eta_X \circ k)^{\sharp}" description] & \Sigma TX \arrow[l, "\mathsf{inr}_X"'] \arrow[d, "\Sigma\Psi^{\#}h"] \\
DX \arrow[r, "D\eta_X"'] & DTX & \Sigma DTX \arrow[l, "\Psi\,\mathsf{inr}_X"]
\end{tikzcd}$$

But this is the same as saying that $\Psi^{\#}k$ is obtained by taking the left adjunct of the function

$$D\eta_X \circ k : X \to DTX = U^{\Sigma}\langle DTX, \Psi\,\mathsf{inr}_X\rangle$$

wrt the adjunction $F^{\Sigma} \dashv U^{\Sigma}$, where $U^{\Sigma} : \mathbf{Set}^{\Sigma} \to \mathbf{Set}$ is the forgetful functor mapping Σ-algebras to their carriers and $F^{\Sigma} : \mathbf{Set} \to \mathbf{Set}^{\Sigma}$ is its left adjoint mapping a set X to the free Σ-algebra $\langle TX, \mathsf{inr}_X\rangle$ over X. (Cf Section 2.)

$$\begin{tikzcd}
X \arrow[r, "\eta_X"] \arrow[d, "k"'] & TX \arrow[d, "\Psi^{\#}k"'] \\
DX \arrow[r, "D\eta_X"'] & DTX
\end{tikzcd}
\qquad
\begin{tikzcd}
TX \arrow[d, dashed, "\Psi^{\#}k \;=\; (D\eta_X \circ k)^{\sharp}" description] & \Sigma TX \arrow[l, "\mathsf{inr}_X"'] \arrow[d, "\Sigma\Psi^{\#}k"] \\
DTX & \Sigma DTX \arrow[l, "\Psi\,\mathsf{inr}_X"]
\end{tikzcd}$$

This is for an operational monad Ψ on the Σ-algebras. If, instead, the monad Ψ is on the isomorphic categories of T-algebras, one can use the similar adjunction $F^{T} \dashv U^{T}$ regarding DTX as carrying the T-algebra structure $\Psi\mu_X : TDTX \to DTX$ and thus obtaining $\Phi^{@}k$ as the left adjunct of

$$D\eta_X \circ k : X \to DTX = U^{T}\langle DTX, \Psi\mu_X\rangle$$

wrt this latter adjunction. That is:

$$
\begin{tikzcd}
X \arrow[r, "\eta_X"] \arrow[d, "k"'] & TX \arrow[d, "\Psi^{\#}k"'] \\
DX \arrow[r, "D\eta_X"'] & DTX
\end{tikzcd}
\qquad
\begin{tikzcd}
TX \arrow[d, dashed, "\Psi^{\#}k = (D\eta_X \circ k)^{\sharp}"'] & T^2X \arrow[l, "\mu_X"'] \arrow[d, "T\Psi^{\#}k"] \\
DTX & TDTX \arrow[l, "\Psi\mu_X"]
\end{tikzcd}
$$

Next, recall that, while the syntactical monad T is freely generated by the signature Σ, the observational comonad is cofreely generated by the behaviour B. Then, by duality, the forgetful functor $U_B : \mathbf{Set}_B \to \mathbf{Set}$ mapping coalgebras to their carriers has a *right adjoint*, namely the functor

$$G_B : \mathbf{Set} \to \mathbf{Set}_B \qquad\qquad X \mapsto \langle DX, \mathsf{snd}_X \rangle$$

mapping a set X to the cofree coalgebra over it. (This holds for arbitrary endofunctors $B : \mathbf{C} \to \mathbf{C}$, provided that the endofunctor $(X \times B) : \mathbf{C} \to \mathbf{C}$ has a final coalgebra for every object X in $\mathbf{C}$.) Similarly, the forgetful functor $U_D : \mathbf{C}_D \to \mathbf{C}$ mapping the coalgebras of a comonad $D = < D, \varepsilon, \delta >$ to their carriers has a right adjoint

$$G_D : \mathbf{C} \to \mathbf{C}_D \qquad\qquad X \mapsto \langle DX, \delta_X \rangle$$

and the counit $\varepsilon : U_D G_D = D \Rightarrow I$ of this adjunction $U_D \dashv G_D$ is simply the counit of the comonad D. Therefore, the coinductive construction of the denotational monad $\Phi^{@}$ from an operational monad Φ on the B-coalgebras

$$
\begin{tikzcd}
TX \arrow[d, "h"'] & TDX \arrow[l, "T\varepsilon_X"'] \arrow[r, "\Phi\,\mathsf{snd}_X"] \arrow[d, dashed, "\Phi^{@}h = (h \circ T\varepsilon_X)^{\flat}"'] & BTDX \arrow[d, "B\Phi^{@}h"] \\
X & DX \arrow[l, "\varepsilon_X"] \arrow[r, "\mathsf{snd}_X"'] & BDX
\end{tikzcd}
$$

can be rephrased in terms of operational monads Φ on the D-coalgebras as the right adjunct wrt the adjunction $U_D \dashv G_D$ of the arrow

$$h \circ T\varepsilon_X : U_D \langle TDX, \Psi\delta_X \rangle = TDX \to X$$

That is:

$$
\begin{tikzcd}
TX \arrow[d, "h"'] & TDX \arrow[l, "T\varepsilon_X"'] \arrow[d, "\Phi^{@}h"'] \\
X & DX \arrow[l, "\varepsilon_X"]
\end{tikzcd}
\qquad
\begin{tikzcd}
TDX \arrow[r, "\Phi\delta_X"] \arrow[d, dashed, "\Phi^{@}h = (h \circ T\varepsilon_X)^{\flat}"'] & DTDX \arrow[d, "D\Phi^{@}h"] \\
DX \arrow[r, "\delta_X"'] & D^2X
\end{tikzcd}
$$

In order to calculate the value of this right adjunct $\Phi^@ h = (h \circ T\varepsilon_X)^\flat$, one can use the standard formula

$$f^\flat = Gf \circ \eta_X$$

valid for every adjunction $F \dashv G$ (with unit $\eta : I \Rightarrow GF$), which, pictorially, amounts to the following bijection.

$$\frac{FX \xrightarrow{f} Y}{X \xrightarrow[\eta_X]{} GFX \xrightarrow[Gf]{} GY}$$

In particular, the unit itself is the right adjunct of the identity. For the adjunction $U_D \dashv G_D$ this gives that the unit at a coalgebra $\langle X, k\rangle$ is the structure $k : X \to DX$ of the coalgebra itself, since, by the D-coalgebra laws,

$$\begin{tikzcd} & X \arrow[d, "k"] \arrow[dl, phantom, "//"] \\ X & DX \arrow[l, "\varepsilon_X"] \end{tikzcd} \qquad \begin{tikzcd} X \arrow[r, "k"] \arrow[d, "k"'] & DX \arrow[d, "\delta_X"] \\ DX \arrow[r, "Dk"'] & D^2X \end{tikzcd}$$

Therefore

$$\frac{TDX \xrightarrow{h \circ T\varepsilon_X} X}{TDX \xrightarrow[\Phi\delta_X]{} DTDX \xrightarrow[D(h \circ T\varepsilon_X)]{} DX}$$

and thus

$$\Phi^@ h = (h \circ T\varepsilon)^\flat = Dh \circ DT\varepsilon_X \circ \Phi\delta_X$$

Finally, notice that, by using the adjunction, the comonad D needs not to be cofreely generated by an endofunctor, the coinduction principle being replaced by the more general adjunction principle. Dually, also the induction principle can be replaced by the adjunction principle, which holds for every monad T.

To summarize:

Monad $T {=} {<} T, \eta, \mu {>}$	Comonad $D {=} {<} D, \varepsilon, \delta {>}$
$F^T X = \langle TX, \mu_X \rangle$	$G_D = \langle DX, \delta_X \rangle$
$F^T \dashv U^T$	$U_D \dashv G_D$
$\begin{tikzcd} \mathbf{C}_D \arrow[r, "\Phi"] \arrow[d, "U_D"'] & \mathbf{C}_D \arrow[d, "U_D"] \\ \mathbf{C} \arrow[r, "T"'] & \mathbf{C} \end{tikzcd}$	$\begin{tikzcd} \mathbf{C}^T \arrow[r, "\Phi^{@}"] \arrow[d, "U^T"'] & \mathbf{C}^T \arrow[d, "U^T"] \\ \mathbf{C} \arrow[r, "D"'] & \mathbf{C} \end{tikzcd}$
$\begin{aligned} \Psi^{\#} &= (D\eta \circ \_)^{\sharp} \\ &= \Psi\mu_X \circ TD\eta_X \circ T(\_) \end{aligned}$	$\begin{aligned} \Phi^{@} &= (\_ \circ T\varepsilon)^{\flat} \\ &= D(\_) \circ DT\varepsilon_X \circ \Phi\delta_X \end{aligned}$
$\begin{tikzcd} \mathbf{C}_D \arrow[r, "\Psi^{\#}"] \arrow[d, "U_D"'] & \mathbf{C}_D \arrow[d, "U_D"] \\ \mathbf{C} \arrow[r, "T"'] & \mathbf{C} \end{tikzcd}$	$\begin{tikzcd} \mathbf{C}^T \arrow[r, "\Psi"] \arrow[d, "U^T"'] & \mathbf{C}^T \arrow[d, "U^T"] \\ \mathbf{C} \arrow[r, "D"'] & \mathbf{C} \end{tikzcd}$

Operational is Denotational. The mapping $\Phi \mapsto \Phi^{@}$ is a bijection between operational monads and denotational comonads with $\Psi \mapsto \Psi^{\#}$ as inverse:

$$
\begin{tikzcd}
\Phi \arrow[dd, mapsto] & \mathbf{C}_D \arrow[r, "\Phi"] \arrow[d, "U_D"'] & \mathbf{C}_D \arrow[d, "U_D"] & \Psi^{\#} \\
 & \mathbf{C} \arrow[r, "T"'] & \mathbf{C} & \\
\Phi^{@} & \mathbf{C}^T \arrow[r, "\Psi"] \arrow[d, "U^T"'] & \mathbf{C}^T \arrow[d, "U^T"] & \Psi \arrow[uu, mapsto] \\
 & \mathbf{C} \arrow[r, "D"'] & \mathbf{C} &
\end{tikzcd}
$$

In order to prove that, for every D-coalgebra structure $k : X \to DX$, one has $\Phi k = (\Phi^{@})^{\#}k$, let us first rewrite $(\Phi^{@})^{\#}k$ in terms of Φ: because $\Phi^{@}h = Dh \circ DT\varepsilon_X \circ \Phi\delta_X$ for every T-algebra structure h and hence

$$\Phi^{@}\mu_X = D\mu_X \circ DT\varepsilon_{TX} \circ \Phi\delta_{TX}$$

and because

$$\Psi^{\#}k = \Psi\mu_X \circ TD\eta_X \circ Tk$$

one has

$$\begin{aligned}(\Phi^{@})^{\#}k &= \Phi^{@}\mu_X \circ TD\eta_X \circ Tk \\ &= D\mu_X \circ DT\varepsilon_{TX} \circ \Phi\delta_{TX} \circ TD\eta_X \circ Tk\end{aligned}$$

But then everything in sight in the following diagram commutes.

$$\begin{tikzcd}
TX \arrow[rrr, "\Phi k"] & & & DTX \\
TX \arrow[u, equal] \arrow[r, "T\eta_X"] \arrow[d, "Tk"'] \arrow[rrru, bend right=90, looseness=1.5, "(\Phi^{@})^{\#}k"'] & TDX \arrow[ul, "\mu_X"'] \arrow[r, "\Phi^2 k"] \arrow[d, "T\Phi k"'] & DT^2X \arrow[ur, "D\mu_X"] \arrow[d, "DT\Phi k"'] \arrow[dr, phantom, "="] & \\
TDX \arrow[r, "TD\eta_X"'] & TDTX \arrow[r, "\Phi\delta_{TX}"'] & DTDTX \arrow[r, "DT\varepsilon_{TX}"'] & DT^2X \arrow[uu, "D\mu_X"']
\end{tikzcd}$$

> The only non-trivial fact is the commutativity of the sub-diagram in the middle, but this follows from the fact that it is the image under the functor Φ of one of the two D-coalgebra laws for the structure $\Phi k : TX \to DTX$. That is,
>
> $$\begin{tikzcd}
TX \arrow[r, "\Phi k"] \arrow[d, "\Phi k"'] & DTX \arrow[d, "\delta_{TX}"] & & T^2X \arrow[r, "T\Phi k"] \arrow[d, "\Phi^2 k"'] & TDTX \arrow[d, "\Phi\delta_{TX}"] \\
DTX \arrow[r, "D\Phi k"'] & D^2TX \arrow[r, phantom, "\overset{\Phi}{\longmapsto}" near end, shift left=30] & & DT^2X \arrow[r, "DT\Phi k"'] & DTDTX
\end{tikzcd}$$

This proves that $\Phi k = (\Phi^{@})^{\#}k$ and, by duality, $\Psi h = (\Psi^{\#})^{@}h$.

Notes. The original proof of "operational is denotational" was more complex: the above simplified proof is due to Bart Jacobs.

9 A Category of Models

The algebras (ie the denotational models) of an operational monad Φ are the same as the coalgebras (ie the operational models) of its coinduced denotational comonad $\Phi^{@}$. Therefore, one can define a general category of Φ-*models* (ie Φ-algebras or, equivalently, $\Phi^{@}$-coalgebras) where both operational and denotational aspects are displayed: this is the proper setting for understanding the *adequacy* results of functorial semantics. In particular, the unique arrow from the initial to the final Φ-model is both the initial algebra and the final coalgebra semantics corresponding to Φ.

By instantiating the general definition of algebras of a monad to a monad $\Phi = < \Phi, \eta, \mu >$ on the D-coalgebras one has that a Φ-algebra has as carrier a D-coalgebra $\langle X, k \rangle$ and as structure a D-coalgebra arrow $h : \Phi\langle X, k \rangle \to \langle X, k \rangle$ such that

$$
\begin{tikzcd}
\Phi^2\langle X,k\rangle \arrow[r, "\Phi h"] \arrow[d, "\mu_{\langle X,k\rangle}"'] & \Phi\langle X,k\rangle \arrow[d, "h"] \\
\Phi\langle X,k\rangle \arrow[r, "h"'] & \langle X,k\rangle
\end{tikzcd}
\qquad
\begin{tikzcd}
\langle X,k\rangle \arrow[r, "\eta_{\langle X,k\rangle}"] \arrow[dr, equal] & \Phi\langle X,k\rangle \arrow[d, "h"] \\
 & \langle X,k\rangle
\end{tikzcd}
$$

If, like in functorial operational semantics, the monad Φ is a lifting of a monad $T = < T, \eta, \mu >$ to the D-coalgebras, then the structure $h : \Phi\langle X, k \rangle \to \langle X, k \rangle$ is of the form

$$
\begin{tikzcd}
TX \arrow[r, "h"] \arrow[d, "\Phi k"'] & X \arrow[d, "k"] \\
DTX \arrow[r, "Dh"'] & DX
\end{tikzcd}
$$

Moreover, $h : \Phi\langle X, k \rangle \to \langle X, k \rangle$ is a Φ-algebra structure if and only if the underlying $h : TX \to X$ is a T-algebra structure. Indeed, for instance, the first Φ-algebra law for h amounts to the commutativity of the following cube

$$\begin{tikzcd}
T^2X \arrow[rr, "Th"] \arrow[dr, "\mu_X"] \arrow[dd, "\Phi^2 k"'] & & TX \arrow[dr, "h"] & \\
& TX \arrow[rr, "h"] \arrow[dd, "\Phi k"'] & & X \arrow[dd, "k"] \\
DT^2X \arrow[dr, "D\mu_X"'] & & & \\
& DTX \arrow[rr, "Dh"'] & & DX
\end{tikzcd}$$

> The front side and the other (not visible) side underlying h are two copies of the definition of h, hence commute. The back (not visible) side is the image of the front side under the functor Φ, hence it commutes. The remaining vertical side commutes because the multiplication μ of T lifts to the multiplication of Φ. The bottom (not visible) side is the image under the functor D of the top side, hence to prove the commutativity of the whole cube it suffices to prove that the top side commutes. But this is nothing but the first T-algebra law for h.

Therefore a **Φ-algebra** is a triple $\langle X, k, h\rangle$ with $k : X \to DX$ a D-coalgebra and $h : TX \to X$ a T-algebra structure such that

$$\begin{tikzcd}
TX \arrow[r, "\Phi k"] \arrow[d, "h"'] & DTX \arrow[d, "Dh"] \\
X \arrow[r, "k"'] & DX
\end{tikzcd}$$

commutes.

Similarly, a Φ-algebra arrow $f : \langle X, k, h\rangle \to \langle Y, m, l\rangle$ is an arrow $f : X \to Y$ such that everything in sight in the diagram

$$\begin{tikzcd}
& TY \arrow[rr, "\Phi m"] \arrow[dd, "l"'] & & DTY \arrow[dd, "Dl"] \\
TX \arrow[ur, "Tf"] \arrow[rr, "\Phi k"] \arrow[dd, "h"'] & & DTX \arrow[ur, "DTf"] \arrow[dd, "Dh"] & \\
& Y \arrow[rr, "m"] & & DY \\
X \arrow[ur, "f"] \arrow[rr, "k"'] & & DX \arrow[ur, "Df"'] &
\end{tikzcd}$$

commutes, but for this it suffices that

$$\begin{tikzcd}
 & & TY \arrow[d, "l"] & \\
TX \arrow[urr, "Tf"] \arrow[d, "h"'] & & Y \arrow[r, "m"] & DY \\
X \arrow[urr, "f"] \arrow[r, "k"'] & DX \arrow[urr, "Df"'] & &
\end{tikzcd}$$

commutes, that is, f is both a D-coalgebra arrow $f : \langle X, k\rangle \to \langle Y, m\rangle$ and a T-algebra arrow $f : \langle X, h\rangle \to \langle Y, l\rangle$.

Dually, given a lifting Ψ of a comonad D on a category of T-algebras, a **Ψ-coalgebra** is a triple $\langle X, h, k\rangle$ with $h : TX \to X$ a T-algebra and $k : X \to DX$ a D-coalgebra structure such that

$$\begin{tikzcd}
TX \arrow[r, "Tk"] \arrow[d, "h"'] & TDX \arrow[d, "\Psi h"] \\
X \arrow[r, "k"'] & DX
\end{tikzcd}$$

commutes. The arrows $f : \langle X, h, k\rangle \to \langle Y, l, m\rangle$ of the corresponding category $\mathbf{C}^{T}{}_{\Psi}$ are again arrows $f : X \to Y$ which preserve both the T-algebra and the D-coalgebra structure.

The claim now is that a triple $\langle X, k, h\rangle$ is a Φ-algebra if and only if $\langle X, h, k\rangle$ is a $\Phi^{@}$-coalgebra, that is,

Φ-algebras are $\Phi^{@}$-coalgebras

$$\begin{tikzcd}
TX \arrow[r, "\Phi k"] \arrow[d, "h"'] & DTX \arrow[d, "Dh"] \\
X \arrow[r, "k"'] & DX
\end{tikzcd}
\quad\Longleftrightarrow\quad
\begin{tikzcd}
TX \arrow[r, "Tk"] \arrow[d, "h"'] & TDX \arrow[d, "\Phi^{@} h"] \\
X \arrow[r, "k"'] & DX
\end{tikzcd}$$

Equivalently, the claim is that the diagram

$$\begin{tikzcd} TX \arrow[r, "\Phi k"] \arrow[d, "Tk"'] & DTX \arrow[d, "Dh"] \\ TDX \arrow[r, "\Phi^{@} h"'] & DX \end{tikzcd}$$

commutes. But then fill this last diagram as follows and notice that all sub-diagrams commute.

$$\begin{tikzcd} TX \arrow[rr, "\Phi k"] \arrow[dd, "Tk"'] & & DTX \arrow[dl, "DTk"'] \arrow[d, equal] \\ & DTDX \arrow[r, "DT\varepsilon_X"'] & DTX \arrow[d, "Dh"] \\ TDX \arrow[ur, "\Phi\delta_X"'] \arrow[rr, "\Phi^{@} h"'] & & DX \end{tikzcd}$$

The only non-trivial sub-diagram is the one corresponding to the upper left corner but this is the image under the functor Φ of one of the two D-coalgebra laws for the structure $k : X \to DX$. That is,

$$\begin{tikzcd} X \arrow[r, "k"] \arrow[d, "k"'] & DX \arrow[d, "\delta_X"] \\ DX \arrow[r, "Dk"'] & D^2X \end{tikzcd} \quad \overset{\Phi}{\longmapsto} \quad \begin{tikzcd} TX \arrow[r, "Tk"] \arrow[d, "\Phi k"'] & TDX \arrow[d, "\Phi\delta_X"] \\ DTX \arrow[r, "DTk"'] & DTDX \end{tikzcd}$$

Thus, up to the permutation $\langle X, k, h\rangle \mapsto \langle X, h, k\rangle$, for any monad Φ lifting a monad T to the coalgebras of a comonad D, the two categories of Φ-algebras and $\Phi^{@}$-coalgebras are the same:

$$\mathbf{C}_D{}^{\Phi} = \mathbf{C}^T{}_{\Phi^{@}}$$

Dually,

$$\mathbf{C}^T{}_{\Psi} = \mathbf{C}_D{}^{\Psi^{\#}}$$

that is, **Ψ-coalgebras are $\Psi^{\#}$-algebras**.

Notice that, since every monad is defined by its algebras and, dually, every comonad is defined by its coalgebras, this gives an alternative proof that the mapping $\Phi \mapsto \Phi^{@}$ is a bijection with $\Psi \mapsto \Psi^{\#}$ as inverse.

Φ-Models. If Φ is an operational monad, then the category $\mathbf{C}_D{}^{\Phi} = \mathbf{C}^T{}_{\Phi^@}$ can be seen as the category of **models of** Φ:

$$\Phi\text{-}\mathbf{Mod} = \mathbf{C}_D{}^{\Phi} = \mathbf{C}^T{}_{\Phi^@}$$

This category has both an initial and a final object which are 'lifted' from the initial T-algebra and the final D-coalgebra, respectively.

The claim is that the *initial* Φ*-model* is the Φ-algebra

$$T^2 0 \xrightarrow{\mu_0} T0 \xrightarrow{\Phi 0} DT0$$

where, recall, the set $T0$ is the set of closed programs, the structure μ_0 is the initial T-algebra structure, and the structure

$$\Phi 0 = [\![\text{-}]\!] : T0 \to DT0$$

is the initial operational model corresponding to Φ. Dually, the *final* Φ*-model* is the $\Phi^@$-coalgebra

$$TD1 \xrightarrow{\delta_1} D1 \xrightarrow{\Phi^@ 1} D^2 1$$

where, recall, the set $D1$ is the set of abstract global behaviours, the structure δ_1 is the final D-coalgebra structure, and the structure

$$\Phi^@ 1 = (\!|\text{-}|\!) : TD1 \to D1$$

is the denotational model coinduced by Φ on the final coalgebra.

If the above holds, then one has, by the very definition of Φ-algebra and $\Phi^@$-coalgebra arrows, that the unique arrow from the initial to the final Φ-model is both the initial algebra and the final coalgebra semantics corresponding to Φ

$$\begin{tikzcd}
T^2 0 \arrow[r, "T!"] \arrow[dd, "\mu_0"'] & TD1 \arrow[d, "\Phi^@ 1 = (\!|\text{-}|\!)"] & \\
& D1 \arrow[r, "\delta_1"] & D^2 1 \\
T0 \arrow[ur, dashed, "(\!|\text{-}|\!)^{\#} = \,!\, = [\![\text{-}]\!]^@"] \arrow[r, "\Phi 0 = [\![\text{-}]\!]"'] & DT0 \arrow[ur, "D!"'] &
\end{tikzcd}$$

That is,

$$[\![\text{-}]\!]^@ = (\!|\text{-}|\!)^{\#} : T0 \to D1$$

The fact that the triple $\langle T0, \mu_0, \Phi 0\rangle$ is the initial Φ-model can be proved directly, but it is more informative to obtain it by means of an adjunction as follows. First notice that the Φ-model $\langle T0, \mu_0, \Phi 0\rangle$ can be obtained by applying the functor

$$\widetilde{F^T} : \mathbf{C}_D \to \Phi\text{-}\mathbf{Mod} \qquad\qquad \langle X, k\rangle \mapsto \langle TX, \mu_X, \Phi k\rangle$$

to the initial D-coalgebra:

$$(0 \xrightarrow{0} D0) \overset{\widetilde{F^T}}{\longmapsto} (T^2 0 \xrightarrow{\mu_0} T0 \xrightarrow{\Phi 0} DT0)$$

Next, if a functor has a right adjoint, then it 'preserves colimits' (see, eg, §V.5 of [Mac71]), thus, in particular, if the functor $\widetilde{F^T}$ has a right adjoint then it maps the initial D-coalgebra to the initial Φ-model. Now, the claim is that this right adjoint exists and it is the functor

$$\widetilde{U^T} : \Phi\text{-}\mathbf{Mod} \to \mathbf{C}_D \qquad\qquad \langle X, h, k\rangle \mapsto \langle X, k\rangle$$

which forgets the T-algebra structure in a Φ-model. Moreover, this adjunction

$$\widetilde{F^T} \dashv \widetilde{U^T}$$

is a 'lifting' of the adjunction $F^T \dashv U^T$ corresponding to the algebras of the monad T (see Section 2).

Let us prove this claim in its dual form, namely that the adjunction $U_D \dashv G_D$, corresponding to the coalgebras of the comonad D (see previous section), lifts to an adjunction

$$\widetilde{U_D} \dashv \widetilde{G_D}$$

between the forgetful functor

$$\widetilde{U_D} : \Phi\text{-}\mathbf{Mod} \to \mathbf{C}^T \qquad\qquad \langle X, k, h\rangle \mapsto \langle X, h\rangle$$

and the functor

$$\widetilde{G_D} : \mathbf{C}^T \to \Phi\text{-}\mathbf{Mod} \qquad\qquad \langle X, h\rangle \mapsto \langle DX, \delta_X, \Phi^{@} h\rangle$$

The adjunction $\widetilde{U_D}\dashv\widetilde{G_D}$ splitting the comonad $\Phi^@$. Given a monad Φ lifting a monad T to the coalgebras of a comonad D, the composition $\widetilde{U_D}\widetilde{G_D} : \mathbf{C}^T \to \mathbf{C}^T$ of the above functors $\widetilde{G_D} : \mathbf{C}^T \to {\mathbf{C}_D}^{\Phi} = \Phi\text{-}\mathbf{Mod}$ and $\widetilde{U_D} : \Phi\text{-}\mathbf{Mod} = {\mathbf{C}_D}^{\Phi} \to \mathbf{C}^T$ is equal to the endofunctor $\Phi^@ : \mathbf{C}^T \to \mathbf{C}^T$. Indeed,

$$(TX \xrightarrow{h} X) \overset{\widetilde{G_D}}{\longmapsto} (TDX \xrightarrow{\Phi^@ h} DX \xrightarrow{\delta_X} D^2X) \overset{\widetilde{U_D}}{\longmapsto} (TDX \xrightarrow{\Phi^@ h} DX)$$

for every T-algebra structure $h : TX \to X$. The claim is that $\widetilde{G_D}$ is right adjoint to $\widetilde{U_D}$ and the whole comonad $\Phi^@ =< \Phi^@, \varepsilon, \delta >$ arises from the adjunction $\widetilde{U_D}\dashv\widetilde{G_D}$. Moreover, the adjunction $\widetilde{U_D}\dashv\widetilde{G_D}$ 'lifts' the adjunction $U_D\dashv G_D$ (which splits the comonad D):

$$\begin{tikzcd}
\mathbf{C}^T \arrow[r, shift right, "\widetilde{G_D}"'] \arrow[d, "U^T"'] \arrow[r, phantom, "\perp"] & {\mathbf{C}_D}^{\Phi} \arrow[l, shift right, "\widetilde{U_D}"'] \arrow[d, "U^{\Phi}"] \\
\mathbf{C} \arrow[r, shift right, "G_D"'] \arrow[r, phantom, "\perp"] & \mathbf{C}_D \arrow[l, shift right, "U_D"']
\end{tikzcd}$$

That is,

$$\begin{array}{lcl}
U^T\widetilde{U_D} & = & U_D U^{\Phi} : {\mathbf{C}_D}^{\Phi} \to \mathbf{C} \\
G_D U^T & = & U^{\Phi}\widetilde{G_D} : \mathbf{C}^T \to \mathbf{C}_D \\
\varepsilon_{U^T} & = & U^{\Phi}\widetilde{\varepsilon}_{U^{\Phi}} : DU^T \Rightarrow U^T
\end{array}$$

The first and second equation are immediate, while the third is to be checked: by definition of T-algebra arrows, it tells that the counit of the upper adjunction is the same as the counit $\varepsilon : U_D G_D = D \Rightarrow I$ of the lower one. That is, the claim is that, for every T-algebra arrow $f : \widetilde{U_D}\langle Y, k, l\rangle = \langle Y, l\rangle \to \langle X, h\rangle$, the right adjunct $U^T f^\flat = f^\flat : \langle Y, k\rangle \to \langle DX, \delta_X\rangle$ of $U^T f = f : U_D\langle Y, k\rangle = Y \to X$ wrt the adjunction $U_D\dashv G_D$ is the unique Φ-algebra arrow from $\langle Y, k, l\rangle$ to $\widetilde{G_D}\langle X, h\rangle = \langle DX, \delta_X, \Phi^@ h\rangle$ factorizing f through ε_X. Diagrammatically:

$$\begin{tikzcd}
 & & TY \arrow[d, "l"] \arrow[dl, "Tf^\flat"'] \arrow[dll, bend right, "Tf"'] \\
 & & Y \arrow[dll, bend right, "f"'] \arrow[dl, "f^\flat"] \\
TX \arrow[d, "h"'] & TDX \arrow[l, "T\varepsilon_X"'] \arrow[d, "\Phi^@ h"] & \\
X & DX \arrow[l, "\varepsilon_X"] &
\end{tikzcd}
\qquad
\begin{tikzcd}
 & TY \arrow[r, "\Phi k"] \arrow[d, "l"] \arrow[dl, "Tf^\flat"'] & & DTY \arrow[d, "Dl"] \arrow[dl, "DTf^\flat"'] \\
 & Y \arrow[rr, "k"] \arrow[dl, dashed, "f^\flat"] & & DY \arrow[dl, "Df^\flat"] \\
TDX \arrow[rr, "\Phi\delta_X"] \arrow[d, "\Phi^@ h"'] & & DTDX \arrow[d, "D\Phi^@ h"'] & \\
DX \arrow[rr, "\delta_X"'] & & D^2X &
\end{tikzcd}$$

All sub-diagrams commute either by definition or because they are obtained by applying a functor to a commuting diagram, except for

$$\begin{tikzcd}
TY \arrow[r, "Tf^\flat"] \arrow[d, "l"'] & TDX \arrow[d, "\Phi^@ h"] \\
Y \arrow[r, "f^\flat"'] & DX
\end{tikzcd}$$

(and its image under D). But the commutativity of the latter follows from the fact that both composite arrows $f^\flat \circ l$ and $\Phi^@ h \circ T f^\flat$ fit as the (unique!) arrow $(h \circ Tf)^\flat : \langle TY, \Phi k \rangle \to \langle DX, \delta_X \rangle$. (If the comonad D is cofreely generated, then this arrow is the unique coinductive extension of Φk along the composite $h \circ Tf : TY \to X$.)

This shows that $\widetilde{G_D}$ is right adjoint to $\widetilde{U_D}$ and that ε is the counit of the adjunction. The unit of the adjunction is obtained by taking the right adjunct of the identity and, by the D-coalgebra laws, its value at a Φ-algebra $\langle X, k, h \rangle$ is the coalgebraic component of the Φ-algebra, namely $k : X \to DX$.

$$\begin{tikzcd}
 & & TX \arrow[d, "h"] \arrow[ddl, "Tk"'] \\
 & & X \arrow[ddl, "k"] \\
TX \arrow[d, "h"'] & TDX \arrow[l, "T\varepsilon_X"'] \arrow[d, "\Phi^@ h"] & \\
X & DX \arrow[l, "\varepsilon_X"] &
\end{tikzcd}
\qquad
\begin{tikzcd}
 & TX \arrow[rr, "\Phi k"] \arrow[d, "h"] \arrow[ddl, "Tk"'] & & DTX \arrow[d, "Dh"] \arrow[ddl, "DTf^\flat"'] \\
 & X \arrow[rr, "k"'] \arrow[ddl, "k"] & & DX \arrow[ddl, "Dk"] \\
TDX \arrow[rr, "\Phi\delta_X"'] \arrow[d, "\Phi^@ h"'] & & DTDX \arrow[d, "D\Phi^@ h"'] & \\
DX \arrow[rr, "\delta_X"'] & & D^2X &
\end{tikzcd}$$

Finally, notice that also the comultiplication of the comonad $\Phi^@ =< \Phi^@, \varepsilon, \delta >$ arises from this adjunction by first taking the unit at $\widetilde{G_D}\langle X, h \rangle$ and then applying the functor $\widetilde{U_D}$ to it. In general, every adjunction $F \dashv G$ defines a comonad $< FG, \varepsilon, F\eta_G >$, where ε and η are the counit and the unit of the adjunction respectively. (Cf Section 2 for the dual 'every adjunction defines a monad'.)

To summarize, there are two adjunctions for the category of Φ-models, namely

$$\begin{tikzcd}[column sep=large]
\mathbf{C}^T \arrow[r, shift right=1ex, "\widetilde{G_D}"'] \arrow[r, phantom, "\perp"] & {\mathbf{C}_D}^{\Phi} = \Phi\text{-}\mathbf{Mod} = {\mathbf{C}^T}_{\Phi^@} \arrow[l, shift right=1ex, "\widetilde{U_D}"'] \arrow[r, shift right=1ex, "\widetilde{U^T}"'] \arrow[r, phantom, "\perp"] & \mathbf{C}_D \arrow[l, shift right=1ex, "\widetilde{F^T}"']
\end{tikzcd}$$

and the unique arrow from the initial Φ-model $\widetilde{F^T}(0) = \langle T0, \mu_0, \Phi 0\rangle$ to the final Φ-model $\widetilde{G_D}(1) = \langle D1, \delta_1, \Phi^@ 1\rangle$ is both the *initial algebra semantics* induced by the denotational model $\Phi^@ 1 = \llparenthesis\text{-}\rrparenthesis$ and the *final coalgebra semantics* coinduced by the operational model $\Phi 0 = \llbracket\text{-}\rrbracket$. Diagrammatically:

$$\begin{tikzcd}
 & & TD1 \arrow[d, "\Phi^@ 1 = \llparenthesis\text{-}\rrparenthesis"] & & & T1 \arrow[dd, dashed, "1"] \\
T^2 0 \arrow[urr, "T!"] \arrow[dd, "\mu_0"'] & & & {} & {} \arrow[l, mapsto, "\widetilde{F^T}"'] & \\
 & & D1 \arrow[r, "\delta_1"] & D^2 1 & & 1 \\
T0 \arrow[urr, dashed, "\llparenthesis\text{-}\rrparenthesis^{\#} \,=\, ! \,=\, \llbracket\text{-}\rrbracket^@"] \arrow[r, "\Phi 0 = \llbracket\text{-}\rrbracket"'] & DT0 \arrow[urr, "D!"'] & & & & \\
 & {} & & & & \\
 & {} \arrow[u, mapsto, "\widetilde{G_D}"] & & & & \\
0 \arrow[r, dashed, "0"'] & D0 & & & &
\end{tikzcd}$$

This is a more compact and symmetric formulation of the adequacy result given in Section 6.

Notes. The idea that adequacy results 'live' in categories of "algebras over coalgebras" is due to Gordon Plotkin and it has been fundamental for the development of the present functorial approach to operational semantics.

Liftings of adjunctions are treated in [Joh75]. In particular, the adjunction splitting the comonad $\Phi^@$ can be obtained by applying Theorem 4 of [Joh75] (see also, eg, [HJ95a] for a 2-categorical account of this theorem).

III

10 Semi-Lattices, Non-Determinism and Basic Process Algebra

The 'non-deterministic choice' construct is understood as the union of a power-set endofunctor, which, categorically, is a monad whose algebras are *semi-lattices*. This leads to a non-deterministic behaviour endofunctor $BX = \check{\mathcal{P}}(1 + \mathsf{Act} \times X)$ whose coalgebras are non-deterministic transition systems. A functorial *denotational* semantics is 'naturally' associated to this behaviour and its induced functorial *operational* semantics turns out to be '*basic process algebra*'.

Let us consider programs with a *non-deterministic*' behaviour. For this, let us introduce the new construct 'or' of **non-deterministic choice**. The intended meaning of a program $u \,\mathsf{or}\, v$ is that it can choose whether to behave either as the subprogram u or as the subprogram v. The following equations should then hold in the operational model $[\![\text{-}]\!]$. For all programs t, u, v,

$$\begin{array}{rcll} [\![(t \,\mathsf{or}\, u) \,\mathsf{or}\, v]\!] & = & [\![t \,\mathsf{or}\, (u \,\mathsf{or}\, v)]\!] & \text{(associativity)} \\ [\![u \,\mathsf{or}\, v]\!] & = & [\![v \,\mathsf{or}\, u]\!] & \text{(commutativity)} \\ [\![t \,\mathsf{or}\, t]\!] & = & [\![t]\!] & \text{(absorption)} \end{array}$$

Algebraically, a set Y with a binary operator $\vee : Y \times Y \to Y$ which is associative, commutative, and absorptive, that is, such that for all x, y, z in Y,

$$\begin{array}{rcl} (x \vee y) \vee z & = & x \vee (y \vee z) \\ x \vee y & = & y \vee x \\ x \vee x & = & x \end{array}$$

forms a **semi-lattice**; the operator $\vee$ is called the **join** of the semi-lattice. The program construct or should then behave as the join of a semi-lattice:

$$[\![u \,\mathsf{or}\, v]\!] = [\![u]\!] \vee [\![v]\!]$$

As an example of a semi-lattice, consider the set $\mathcal{P}X$ of the subsets of a set X: the binary union $\cup : \mathcal{P}X \times \mathcal{P}X \to \mathcal{P}X$ is associative, commutative, and absorptive, hence $\langle \mathcal{P}X, \cup \rangle$ is a semi-lattice. A similar semi-lattice is the one obtained by considering the set

$$\mathcal{P}_{\mathrm{fi}}X \;=\; \{X' \subseteq X \mid X' \text{ finite}\}$$

of finite subsets of a set X, as well as its '*relevant*' part

$$\check{\mathcal{P}}X \;=\; \{X' \subseteq X \mid X' \text{ finite, } X' \neq \emptyset\}$$

obtained by omitting the empty set. This latter semi-lattice (the binary union $\cup : \check{\mathcal{P}}X \times \check{\mathcal{P}}X \to \check{\mathcal{P}}X$ is its join) is of particular importance because it is the free semi-lattice over X; that is, the functor

$$X \mapsto \langle \check{\mathcal{P}}X, \cup \rangle$$

is left adjoint to the forgetful functor

$$\langle Y, \vee \rangle \mapsto Y$$

from the category of semi-lattices and join-preserving functions to sets.

> Write $SL(\mathbf{Set})$ for **the category of semi-lattices** with arrows $f : \langle X, \vee \rangle \to \langle Y, \sqcup \rangle$, the join-preserving functions $f : X \to Y$ between the underlying sets:
>
> $$\begin{tikzcd} X \times X \arrow[r, "f \times f"] \arrow[d, "\vee"'] & Y \times Y \arrow[d, "\sqcup"] \\ X \arrow[r, "f"'] & Y \end{tikzcd}$$
>
> Equationally, for every pair (x, x') in $X \times X$,
>
> $$f(x \vee x') = fx \sqcup fx'$$

Free semi-lattices. Recall that a functor $U : \mathbf{D} \to \mathbf{C}$ has a left adjoint $F : \mathbf{C} \to \mathbf{D}$ if and only if there exists a natural transformation $\eta : I \Rightarrow UF$ such that each η_X is universal from X to U. That is, for every X in $\mathbf{C}$, Z in $\mathbf{D}$, and $f : X \to UZ$ there exists a unique arrow $f^\sharp : FX \to Z$ in $\mathbf{D}$, such that $f = Uf^\sharp \circ \eta_X$:

$$\begin{tikzcd} X \arrow[r, "\eta_X"] \arrow[dr, "f"'] & UFX \arrow[d, "Uf^\sharp"] & & FX \arrow[d, dashed, "f^\sharp"] \\ & UZ & & Z \end{tikzcd}$$

Let now $U : SL(\mathbf{Set}) \to \mathbf{Set}$ be the above forgetful functor mapping semi-lattices to their carriers and let

$$FX = \langle \check{\mathcal{P}}X, \cup \rangle$$

Then, for every set X, the function

$$\{\text{-}\}_X : X \to \check{\mathcal{P}}X = UFX \qquad x \mapsto \{x\}$$

mapping every element x of X to the corresponding singleton set $\{x\}$ gives the unit $\eta : I \Rightarrow UF$ of the adjunction:

$$\begin{tikzcd}
X \arrow[r, "\{\text{-}\}_X"] \arrow[dr, "f"'] & UFX = \check{\mathcal{P}}X \arrow[d, "Uf^\sharp = f^\sharp"] \\
& U\langle Y, \vee\rangle = Y
\end{tikzcd}
\qquad
\begin{tikzcd}
\check{\mathcal{P}}X \arrow[d, dashed, "f^\sharp"'] & \check{\mathcal{P}}X \times \check{\mathcal{P}}X \arrow[l, "\cup"'] \arrow[d, "f^\sharp \times f^\sharp"] \\
Y & Y \times Y \arrow[l, "\vee"]
\end{tikzcd}$$

Indeed, for every finite subset $\{x_1, \ldots, x_n\} = \{x_1\} \cup \ldots \cup \{x_n\}$ of X

$$f^\sharp\{x_1, \ldots, x_n\} = fx_1 \vee \ldots \vee fx_n$$

is the required unique join-preserving function. (The properties of the join make bracketing irrelevant.) This shows that, for every set X, the pair $\langle\check{\mathcal{P}}X, \cup\rangle$ is the free semi-lattice on X.

As usual, the counit $\varepsilon : FU \Rightarrow I$ of the above adjunction can be obtained by taking for f the identity on $Y = U\langle Y, \vee\rangle$. This gives a 'big join'

$$\bigvee : \check{\mathcal{P}}Y \to Y$$

mapping every finite subset of Y to the join of its elements:

$$\bigvee\{y_1, \ldots, y_n\} = y_1 \vee \ldots \vee y_n$$

In particular, the value of the counit at a free semi-lattice $\langle Y, \vee\rangle = \langle\check{\mathcal{P}}X, \cup\rangle$ is the 'big union'

$$\bigcup : \check{\mathcal{P}}^2 \Rightarrow \check{\mathcal{P}}$$

sending each set of sets into its union. Since every adjunction $F \dashv G$ (with unit η and counit ε) gives rise to a monad $< GF, \eta, G\varepsilon F >$, (cf Section 2), the triple

$$\check{\mathcal{P}} = < \check{\mathcal{P}}, \{\text{-}\}, \bigcup >$$

is a monad. The isomorphism of categories

$$\mathit{SL}(\mathbf{Set}) \cong \mathbf{Set}^{\check{\mathcal{P}}}$$

gives then an alternative description of semi-lattices as algebras of the monad $\check{\mathcal{P}}$. (See "Algebras are T-algebras" in Section 2 or check directly.) Similarly, one can check that the algebras of the the *unrestricted* power-set monad $\mathcal{P} = < \mathcal{P}, \{\text{-}\}, \bigcup >$ are *complete semi-lattices* – semi-lattices with joins of arbitrary cardinality:

$$\mathit{CSL}(\mathbf{Set}) \cong \mathbf{Set}^{\mathcal{P}}$$

Formally, a complete semi-lattice is a partial order $\langle Y, \leq\rangle$ in which *every* subset $Y' \subseteq Y$ has a least upper bound $\bigvee Y'$, while a semi-lattice can be seen as a partial order with least upper bounds only of finite and non empty subsets. (Conversely, every semi-lattice defines a partial order $x \leq y \iff x \vee y = y$.)

In general, 'κ-complete' semi-lattices can be used to define power-set monads of any (*regular*) cardinality κ. Semantically, the cardinality to be used depends on the kind of non-determinism one is interested in. Here only finite determinism is studied, hence (finite) semi-lattices are used.

Even more in general, semi-lattices give an axiomatic description of various 'powerdomains' used in semantics. This holds because semi-lattices can be defined 'internally' in any category $\mathbf{C}$ with binary products:

> A semi-lattice in $\mathbf{C}$ is a pair $\langle Y, \vee\rangle$ with Y an object of $\mathbf{C}$ and $\vee : Y \times Y \to Y$ an arrow of $\mathbf{C}$ which is associative, commutative, and absorptive in a diagrammatic sense. For instance, the commutativity of the join can be described diagrammatically using the canonical 'swap' arrow
>
> $$\begin{tikzcd} & Y \times Y \arrow[dl, "\mathsf{snd}"'] \arrow[d, dashed, "\mathsf{swap}"] \arrow[dr, "\mathsf{fst}"] & \\ Y & Y \times Y \arrow[l, "\mathsf{fst}"] \arrow[r, "\mathsf{snd}"'] & Y \end{tikzcd}$$
>
> as follows.
>
> $$\begin{tikzcd} Y \times Y \arrow[rr, "\mathsf{swap}"] \arrow[dr, "\vee"'] & & Y \times Y \arrow[dl, "\vee"] \\ & Y & \end{tikzcd}$$
>
> Write then $SL(\mathbf{C})$ for the corresponding category with as arrows the join-preserving arrows of $\mathbf{C}$.

For instance, the Plotkin powerdomain monad can be shown to arise from the semi-lattices in a category of complete partial orders and continuous functions. (Notice, the order induced by the semi-lattice structure has nothing to do with the one of the underlying category of complete partial orders.) Similarly, the semi-lattices in a category of complete metric spaces and non-distance-increasing functions give rise to the compact metric powerdomain.

In order to deal with non-deterministic behaviours as introduced by the binary choice construct 'or' consider the new behaviour endofunctor

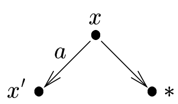
$$B : \mathbf{Set} \to \mathbf{Set} \qquad\qquad X \mapsto \check{\mathcal{P}}(1 + \mathsf{Act} \times X)$$

obtained by composing the (deterministic) behaviour endofunctor $X \mapsto 1 + \mathsf{Act} \times X$ with the semi-lattice monad $\check{\mathcal{P}}$. Its coalgebras are the **finitely branching transition systems**, that is, transition systems which in every state can choose among a *finite* set of transitions. This finite non-determinism reflects the finiteness of the choice construct; this restriction simplifies the presentation, but, in general, one can consider semi-lattices (and corresponding monads) with joins of larger cardinality.

Formally, the correspondence between coalgebras $\langle X, k\rangle$ of the above behaviour and finitely branching transition systems $\langle X, \{\xrightarrow{a}\}_{Act}, \downarrow *\rangle$ is as follows. Omitting, as usual, the injections into the coproduct $1 + \mathsf{Act} \times X$,

$$x \xrightarrow{a} x' \iff k(x) \ni <a, x'> \qquad\qquad x \downarrow * \iff kx \ni *$$

for every $x \in X$. (Cf Section 3.) Notice that a state might both perform an action or become inert; for instance, $k(x) = \{<a, x'>, *\}$ corresponds to the transitions



Notice that above, and whenever convenient, the fact that $x \downarrow *$ holds is treated as a special transition $x \longrightarrow *$:

$$x \downarrow * \iff x \longrightarrow *$$

Next, consider the following 'minimal' language for producing behaviours of type B.

Basic Process Algebra. The basic language for the behaviour $BX = \check{\mathcal{P}}(1+\mathsf{Act}\times X)$ should contain a basic inert program nil, an 'action prefixing' unary operator for every $a \in \mathsf{Act}$, and the binary choice or. Formally, the language is defined by the grammar

$$t ::= x \mid \mathsf{nil} \mid a\,.\,t \mid (t \,\mathsf{or}\, t)$$

and its operational model $[\![\text{-}]\!]$ (a B-coalgebra structure on the above terms) is defined by induction on the structure of the terms as follows.

$$[\![\mathsf{nil}]\!] = \{*\} \qquad [\![a\,.\,t]\!] = \{<a, t>\} \qquad [\![u \,\mathsf{or}\, v]\!] = [\![u]\!] \cup [\![v]\!]$$

(For the treatment of the variables x see the next section.) In terms of transition systems, this corresponds to the following set $\mathcal{R}$ of operational rules.

$$\mathsf{nil} \longrightarrow * \qquad a\,.\,t \xrightarrow{a} t \qquad \frac{u \xrightarrow{a} u'}{u \,\mathsf{or}\, v \xrightarrow{a} u'} \qquad \frac{u \xrightarrow{a} u'}{u \,\mathsf{or}\, v \xrightarrow{a} u'}$$

(In order to simplify the notation, a transition $u \xrightarrow{a} u'$ is here intended possibly to be of the form $u \longrightarrow *$. Thus in particular if $u \longrightarrow *$ then also $u \,\mathsf{or}\, v \longrightarrow *$.)

Basic process algebra is functorial. Let us prove that basic process algebra is functorial by defining a functorial *denotational* semantics Ψ such that its operational dual $\Psi^{\#}$ is equal to the operational semantics induced by the rules of basic process algebra.

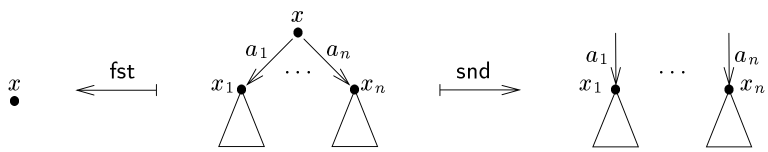
One would like to use the above rules $\mathcal{R}$ for defining directly the functorial operational semantics by induction on a germ $\phi : \Sigma BT \Rightarrow BT$, for Σ and T the signature and the syntactical monad corresponding to basic process algebra, respectively. This is easily done for or and nil using the union $\cup$ and the termination state $*$ available in $B = \check{\mathcal{P}}(1 + \mathsf{Act} \times \text{-})$, but action prefixing causes troubles. Indeed, for any object r of type BT, $a\,.\,r$ should be mapped by ϕ to $\{<a, r>\}$, but this is of type B^2T rather than BT. Instead, the definition of a functorial denotational semantics Ψ lifting the observational comonad $D{=}<D, \varepsilon, \delta>$ to the Σ- (or, equivalently, to the T-) algebras using the rules of basic process algebra causes no problem.

Recall that, for every X, the set DX is the carrier of the final $(X{\times}B)$-coalgebra:

$$X \xleftarrow{\varepsilon_X \,=\, \mathsf{fst}_X} DX \cong X{\times}BDX \xrightarrow{\mathsf{snd}_X} BDX = \check{\mathcal{P}}(1 + \mathsf{Act} \times DX)$$

As shown in Section 13, although the endofunctor $\check{\mathcal{P}} : \mathbf{Set} \to \mathbf{Set}$ is not $\boldsymbol{\omega}^{\mathrm{op}}$-continuous, the final $(X{\times}B)$-coalgebra exists, hence the observational comonad D cofreely generated by B can be defined.

The set DX is the set of global behaviours of states $x \in X$ wrt B. These can be seen as *trees* which are finitely branching, whose nodes are labelled by $x \in X$, and whose arcs are labelled by $a \in \mathsf{Act}$. The counit $\varepsilon_X = \mathsf{fst}_X : DX \to X$ of the comonad gives the label of the root node for each tree in DX and the other projection $\mathsf{snd}_X : DX \to BDX$ gives the remaining part of the tree (and it coinductively extends to give the comultiplication $\delta : D \Rightarrow D^2$ of the comonad D):

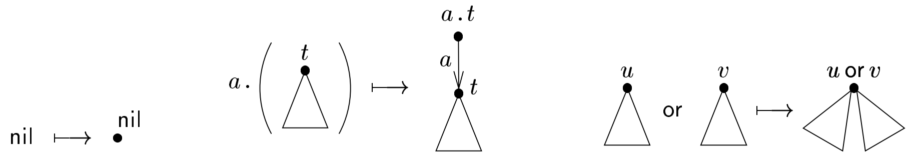


Now, let us first lift the *endofunctor* D to an endofunctor Ψ on the Σ-algebras and then check that also the operations of the comonad D lift. By the equivalence between liftings and actions illustrated in Section 7, the desired endofunctor Ψ is the same as the action

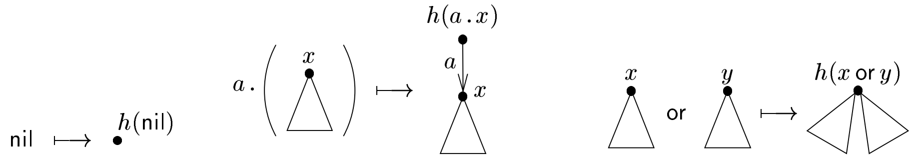
$$\Psi : \Sigma DU^{\Sigma} \Rightarrow DU^{\Sigma}$$

of the constructs Σ on the composite functor $DU^{\Sigma} : \mathbf{Set}^{\Sigma} \to \mathbf{Set}$, where $U^{\Sigma} : \mathbf{Set}^{\Sigma} \to \mathbf{Set}$ is the forgetful functor mapping Σ-algebras to their carriers.

The action Ψ. Let us consider first the case of free Σ-algebras, that is, the action of the program constructs nil, $a\,.$, and or on DT, where, notice, an object of type DT is a tree whose nodes are labelled by terms t of basic process algebra.



Then, in general, for every Σ-algebra $\langle X, h\rangle$, the action of Ψ on $DU^{\Sigma}\langle X, h\rangle = DX$ is



Formally, using the meta-variables p and q to range over objects of type D, ie global behaviours wrt B, the value of the natural transformation $\Psi : \Sigma DU^{\Sigma} \Rightarrow DU^{\Sigma}$ at a Σ-algebra $\langle X, h\rangle$ is defined as follows.

$$
\begin{array}{lll}
\mathsf{nil} & \mapsto & <h(\mathsf{nil}), \{*\}> \\
a\,.\,p & \mapsto & <h(a\,.\,(\mathsf{fst}_X p)), \{<a, p>\}> \\
p \,\mathsf{or}\, q & \mapsto & <h((\mathsf{fst}_X p)\,\mathsf{or}\,(\mathsf{fst}_X q)), (\mathsf{snd}_X p) \cup (\mathsf{snd}_X q)>
\end{array}
$$

Naturality follows from the fact that no assumption is made on the form of the Σ-algebra $\langle X, h\rangle$.

Therefore, for every Σ-algebra structure $h : \Sigma X \to X$, the structure $\Psi h : \Sigma DX \to DX$ is a pair, whose first component is simply the composite function $h \circ \Sigma\mathsf{fst}_X = h \circ \Sigma\varepsilon_X : \Sigma DX \to X$. Writing $\Psi' h : \Sigma DX \to BDX$ for the second component of Ψh, one has the following commuting diagram.

$$
\begin{tikzcd}
\Sigma X \arrow[d, "h"'] & \Sigma DX \arrow[l, "\Sigma\varepsilon_X"'] \arrow[d, "\Psi h"'] \arrow[dr, "\Psi' h = \mathsf{snd}_X \circ \Psi h"] & \\
X & DX \arrow[l, "\varepsilon_X = \mathsf{fst}_X"] \arrow[r, "\mathsf{snd}_X"'] & BDX
\end{tikzcd}
$$

The left square is one of the two diagrams which have to commute in order for $\Psi =< \Psi, \varepsilon, \delta >$ to lift the whole comonad $D =< D, \varepsilon, \delta >$. The other diagram, namely

$$\begin{tikzcd}
\Sigma DX \arrow[r, "\Sigma\delta_X"] \arrow[d, "\Psi h"'] & \Sigma D^2X \arrow[d, "\Psi^2 h"] \\
DX \arrow[r, "\delta_X"'] & D^2X
\end{tikzcd}$$

also commutes, because:

> The composite functions $\delta_X \circ \Psi h$ and $\Psi^2 h \circ \Sigma\delta_X$ both fit as the (unique!) pair $<\Psi h, B\delta_X \circ \Psi' h> : \Sigma DX \to D^2X$
>
> $$\begin{tikzcd}
& \Sigma DX \arrow[dl, phantom, "/\!\!/"] \arrow[dd, dashed, "!"] \arrow[dr, "\Psi' h"] & \\
\Sigma DX \arrow[d, "\Psi h"'] & & BDX \arrow[d, "B\delta_X"] \\
DX & D^2X \arrow[l, "\varepsilon_{DX} = \mathsf{fst}_{DX}"] \arrow[r, "\mathsf{snd}_{DX}"'] & BD^2X
\end{tikzcd}$$
>
> Indeed, noticing that Ψ' is natural, everything in sight in the following two diagrams commutes.
>
> $$\begin{tikzcd}
& \Sigma DX \arrow[dl, phantom, "/\!\!/"] \arrow[d, "\Psi h"'] \arrow[dr, "\Psi' h"] & \\
\Sigma DX \arrow[d, "\Psi h"'] \arrow[dr, phantom, "/\!\!/"] & DX \arrow[r, "\mathsf{snd}_X"] \arrow[d, "\delta_X"] & BDX \arrow[d, "B\delta_X"] \\
DX & D^2X \arrow[l, "\varepsilon_{DX} = \mathsf{fst}_{DX}"] \arrow[r, "\mathsf{snd}_{DX}"'] & BD^2X
\end{tikzcd}
\qquad
\begin{tikzcd}
& \Sigma DX \arrow[dl, phantom, "/\!\!/"] \arrow[d, "\Sigma\delta_X"] \arrow[dr, "\Psi' h"] & \\
\Sigma DX \arrow[d, "\Psi h"'] & \Sigma D^2X \arrow[l, "\Sigma\varepsilon_{DX}"'] \arrow[d, "\Psi^2 h"'] \arrow[dr, "\Psi'^2 h"] & BDX \arrow[d, "B\delta_X"] \\
DX & D^2X \arrow[l, "\varepsilon_{DX} = \mathsf{fst}_{DX}"] \arrow[r, "\mathsf{snd}_{DX}"'] & BD^2X
\end{tikzcd}$$

The above shows thus that $\Psi =< \Psi, \varepsilon, \delta >$ is a functorial denotational semantics lifting $D=< D, \varepsilon, \delta >$ to the Σ-algebras. It induces a functorial operational semantics $\Psi^\#$ as follows. For every D-coalgebra structure $k : X \to DX$, the structure $\Psi^\# k : TX \to DTX$ is the inductive extension of $\Psi\mathsf{inr}_X$ along the composite $D\eta_X \circ k$:

$$\begin{tikzcd}
X \arrow[r, "\eta_X = \mathsf{inl}_X"] \arrow[d, "k"'] & TX \arrow[d, dashed, "\Psi^\# k"'] & \Sigma TX \arrow[l, "\mathsf{inr}_X"'] \arrow[d, "\Sigma\Psi^\# h"] \\
DX \arrow[r, "D\eta_X"'] & DTX & \Sigma DTX \arrow[l, "\Psi\,\mathsf{inr}_X"]
\end{tikzcd}$$

As shown in Section 7, the triple $\Psi^{\#} =< \Psi^{\#}, \eta, \mu >$ is a lifting of the syntactical monad $T{=}{<}T, \eta, \mu{>}$ to the coalgebras of the comonad D. For comparing it with the operational semantics induced by basic process algebra, one has then to translate it to a lifting to the coalgebras of the endofunctor B. For this, since B cofreely generates D, one can use the isomorphism of categories

$$\vartheta : \mathbf{Set}_D \cong \mathbf{Set}_B \qquad\qquad \langle X, k\rangle \mapsto \langle X, B\mathsf{fst}_X \circ \mathsf{snd}_X \circ k\rangle$$

illustrated in Section 7. Thus the composite

$$\vartheta\Psi^{\#}\vartheta^{-1} : TU_B \Rightarrow BTU_B$$

is of the desired form; let us check that also its 'content' is the right one:

> Consider, without loss of generality, the case $k = 0 : 0 \to B0$, that is, let k be the initial B-coalgebra structure. The isomorphism ϑ^{-1} maps it to the initial D-coalgebra structure $0 : 0 \to D0$. Write, for simplicity,
>
> $$[\![\text{-}]\!]_\Psi = \Psi^{\#}(0) : T0 \to DT0$$
>
> The claim is that, for all terms t,
>
> $$\vartheta[\![t]\!]_\Psi = [\![t]\!]$$
>
> where $[\![t]\!]$ is the operational semantics induced by the rules of basic process algebra. Indeed, omitting the subscript 0,
>
> $$\begin{array}{rcl} \vartheta[\![a\,.\,t]\!]_\Psi & = & (B\mathsf{fst} \circ \mathsf{snd})[\![a\,.\,t]\!]_\Psi \\ & = & B\mathsf{fst}(\mathsf{snd} < a\,.\,t, \{<a, [\![t]\!]_\Psi>\}>) \\ & = & B\mathsf{fst}\{<a, [\![t]\!]_\Psi>\} \\ & = & \{<a, \mathsf{fst}[\![t]\!]_\Psi>\} \\ & = & \{<a, t>\} \\ & = & [\![a\,.\,t]\!] \end{array}$$
>
> Similarly, one can see that also
>
> $$\vartheta[\![u \mathbin{\mathsf{or}} v]\!]_\Psi = [\![u \mathbin{\mathsf{or}} v]\!] \qquad \text{and} \qquad \vartheta[\![\mathsf{nil}]\!]_\Psi = [\![\mathsf{nil}]\!]$$

This concludes the proof that basic process algebra is functorial.

The syntax as a semi-lattice. Having established that the choice construct or of basic process algebra really behaves as the join of a semi-lattice, let us treat it as a join also in the syntax. That is, let us consider the algebras of the signature $\Sigma = \{\mathsf{nil}, a\,.\,(\text{-}),\ \mathsf{or}\,\}$ which validate the equations

$$E = \begin{cases} (x \,\mathsf{or}\, y) \,\mathsf{or}\, z &=& x \,\mathsf{or}\, (y \,\mathsf{or}\, z) \\ x \,\mathsf{or}\, y &=& y \,\mathsf{or}\, x \\ x \,\mathsf{or}\, x &=& x \end{cases}$$

and take for the syntactical monad for basic process algebra the monad

$$\mathbf{T}_E = < T_E, \eta, \mu >$$

corresponding to the $\langle \Sigma, E\rangle$-algebras, rather than simply to the Σ-algebras. In other words, the monad $\mathbf{T}_E$ is the one arising from the standard adjunction between $\langle \Sigma, E\rangle$-algebras and sets. (See "algebras are T-algebras" in Section 2.)

For every set X, the set $T_E X$ is nothing but the quotient wrt (the congruence relation generated by) E of the free algebra of terms over X; thus one cannot distinguish in this syntax between, for instance, the terms $u \,\mathsf{or}\, v$ and $v \,\mathsf{or}\, u$. Keeping this quotient in mind, one can still regard the elements of $T_E X$ as terms, that is, one can use representatives rather than equivalence classes. The unit $\eta_X : X \to T_E X$ and the multiplication $\mu_X : T_E T_E X \to T_E X$ are the usual operations on variables and terms: the former is the insertion of the variables $x \in X$ into terms; the latter maps every term $t \in T_E T_E X$ containing a sub-term $u \in T_E X$ as a variable to the 'same' term $t \in T_E X$ by removing the distinction between terms and terms as variables. For instance,

$$\mu((a\,.\,t) \,\mathsf{or}\, \eta_{T_E}(u \,\mathsf{or}\, v)) \;=\; (a\,.\,t) \,\mathsf{or}\, u \,\mathsf{or}\, v$$

Now, by definition, the above denotational semantics Ψ for basic process algebra is not only a Σ-action but also a $\langle \Sigma, E\rangle$-action; that is, for every $h : \Sigma X \to X$ which validates the equations E, also $\Psi h : \Sigma D X \to D X$ validates E. In other words, Ψ is a lifting of the observational comonad D to the $\langle \Sigma, E\rangle$-algebras.

$$\begin{tikzcd} \mathbf{Set}^{\langle \Sigma, E\rangle} \arrow[r, "\Psi"] \arrow[d] & \mathbf{Set}^{\langle \Sigma, E\rangle} \arrow[d] \\ \mathbf{Set} \arrow[r, "D"'] & \mathbf{Set} \end{tikzcd}$$

Correspondingly, its operational dual $\Psi^{\#}$ can be seen as a lifting of the monad $\mathbf{T}_E$ to the D-coalgebras.

Next, write Φ for the operational monad on the B-coalgebras obtained by applying the isomorphism $\vartheta : \mathbf{Set}_D \cong \mathbf{Set}_B$ between D- and B-coalgebras; that is,

$$\Phi = \vartheta \Psi^{\#} \vartheta^{-1} : T_E U_B \Rightarrow B T_E U_B$$

This coaction, because of the equation $[\![u \mathsf{\ or\ } v]\!]_\Psi = [\![u]\!]_\Psi \cup [\![v]\!]_\Psi$, is join-preserving, that is, the following diagram commutes.

$$\begin{tikzcd}
T_E U_B \arrow[d, Rightarrow, "\Phi"'] & & T_E U_B \times T_E U_B \arrow[ll, Leftarrow, "\mathsf{or}"'] \arrow[d, Rightarrow, "\Phi \times \Phi"] \\
\check{\mathcal{P}}(1+\mathsf{Act}\times T_E U_B) & & \check{\mathcal{P}}(1+\mathsf{Act}\times T_E U_B) \times \check{\mathcal{P}}(1+\mathsf{Act}\times T_E U_B) \arrow[ll, Leftarrow, "\cup"]
\end{tikzcd}$$

In other words, the operational semantics of basic process algebra

$$\begin{tikzcd}
& \mathbf{Set}_B \arrow[dl, bend right, "U_B"'] \arrow[dr, bend left, "U_B"] & \\
\mathbf{Set} \arrow[rr, phantom, "\overset{\Phi}{\Longrightarrow}"] \arrow[dr, bend right, "T_E"'] & & \mathbf{Set} \arrow[dl, bend left, "BT_E"] \\
& \mathbf{Set} &
\end{tikzcd}$$

takes place in the category of semi-lattices:

$$\begin{tikzcd}
& \mathbf{Set}_B \arrow[dl, bend right, "U_B"'] \arrow[dr, bend left, "U_B"] & \\
\mathbf{Set} \arrow[rr, phantom, "\overset{\Phi}{\Longrightarrow}"] \arrow[dr, bend right, "{\langle T_E(\text{-}), \mathsf{or} \rangle}"'] & & \mathbf{Set} \arrow[dl, bend left, "{\langle BT_E(\text{-}), \cup \rangle}"] \\
& SL(\mathbf{Set}) &
\end{tikzcd}$$

That is,

$$\Phi : \langle T_E U_B, \mathsf{or} \rangle \Rightarrow \langle B T_E U_B, \cup \rangle$$

The retraction for basic process algebra. One of the advantages of working with the syntax as a $\langle\Sigma, E\rangle$-algebra is that it gives a simple construction of a retraction for basic process algebra. This retraction is used in the next section to show that a certain class of operational rules (the 'GSOS' rules) is functorial.

Recall, from Section 4, the embedding of the (deterministic) behaviour $X \mapsto 1 + \mathsf{Act} \times X$ into the syntax T of the language with atomic actions and sequential composition:

$$\gamma : 1 + \mathsf{Act} \times (\text{-}) \Rightarrow T \qquad\qquad * \mapsto \mathsf{nil} \qquad\qquad <a, x> \mapsto a\ ;\ x$$

The term $a\ ;\ x$ behaves like the term $a\,.\,x$ of the above syntax T_E, hence one can write equivalently

$$\gamma : 1 + \mathsf{Act} \times (\text{-}) \Rightarrow T_E \qquad\qquad * \mapsto \mathsf{nil} \qquad\qquad <a, x> \mapsto a\,.\,x$$

Notice that $\check{\mathcal{P}}(1{+}\mathsf{Act}{\times}X)$ is the carrier of the free semi-lattice over the set $1{+}\mathsf{Act}{\times}X$ and that the syntax $\langle T_E X,\ \mathsf{or}\rangle$ is itself a semi-lattice. Then, by taking the left adjunct of γ wrt the standard adjunction from sets to semi-lattices

$$\begin{tikzcd}
1{+}\mathsf{Act}{\times}(\text{-}) \arrow[r, Rightarrow, "\{\text{-}\}"] \arrow[dr, Rightarrow, "\gamma"'] & B = \check{\mathcal{P}}(1{+}\mathsf{Act}{\times}\text{-}) \arrow[d, Rightarrow, "\gamma^\sharp"] \\
& T_E
\end{tikzcd}
\qquad
\begin{tikzcd}
B \arrow[d, dashed, Rightarrow, "\gamma^\sharp"'] & B \times B \arrow[l, Rightarrow, "\cup"'] \arrow[d, Rightarrow, "\gamma^\sharp \times \gamma^\sharp"] \\
T_E & T_E \times T_E \arrow[l, Rightarrow, "\mathsf{or}"]
\end{tikzcd}$$

one obtains a natural transformation

$$\gamma^\sharp : B = \check{\mathcal{P}}(1 + \mathsf{Act} \times\ \text{-}\,) \Rightarrow T_E$$

which embeds the behaviour $BX = \check{\mathcal{P}}(1 + \mathsf{Act} \times X)$ into the above syntax T_E. That is, using the meta-variables r and s to range over objects of type B,

$$\begin{array}{lclcl}
\gamma^\sharp\{*\} & = & \gamma(*) & = & \mathsf{nil} \\
\gamma^\sharp\{<a, x>\} & = & \gamma(<a, x>) & = & a\,.\,x \\
\gamma^\sharp(r \cup s) & = & (\gamma^\sharp r)\ \mathsf{or}\ (\gamma^\sharp s) & &
\end{array}$$

Now the claim is this embedding $\gamma^\sharp$ is a retraction for basic process algebra. That is, for $\Phi = <\Phi, \eta, \mu>$ the above operational monad corresponding to basic process algebra, one has that the composite $\Phi \circ \mu_{U_B} \circ \gamma^\sharp_{T_E U_B}$ is the identity natural transformation on the functor $BT_E U_B$:

$$\Phi \circ \mu_{U_B} \circ \gamma^\sharp_{T_E U_B} = I$$

(Cf Section 4.) In order to prove this, notice that each $\gamma^\sharp_X$ is an arrow in $SL(\mathbf{Set})$, that is,

$$\gamma^\sharp : \langle B, \cup\rangle \Rightarrow \langle T_E,\ \mathsf{or}\rangle$$

Therefore, the composite

$$\langle BT_E U_B, \cup\rangle \overset{\gamma^\sharp_{T_E U_B}}{\Longrightarrow} \langle T^2_E U_B, \mathsf{or}\rangle \overset{\mu_{U_B}}{\Longrightarrow} \langle T_E U_B, \mathsf{or}\rangle \overset{\Phi}{\Longrightarrow} \langle BT_E U_B, \cup\rangle$$

is necessarily the identity on the functor $\langle BT_E U_B, \cup\rangle : \mathbf{Set}_B \to SL(\mathbf{Set})$ because, for every set X, there exists a unique join-preserving arrow from the free semi-lattice $\langle BT_E X, \cup\rangle$ to itself which respects the unit of the monad $\check{\mathcal{P}}$. This proves that the composite $\mu_{U_B} \circ \gamma^\sharp_{T_E U_B}$ is a retraction for the operational semantics Φ induced by basic process algebra.

The above retraction can be used to give an alternative (more direct) proof of the functoriality of basic process algebra. For this, define the germ

$$\phi^{\mathcal{R}} : \Sigma BT_E \Rightarrow BT_E$$

of the operational semantics corresponding to the rules $\mathcal{R}$ of basic process algebra as follows. For r and s meta-variables ranging over objects of type BT_E,

$$\phi^{\mathcal{R}} = \begin{cases} \mathsf{nil} & \mapsto \quad \{*\} \\ a\,.\,r & \mapsto \quad \{<a, \gamma^\sharp r>\} \\ r \;\mathsf{or}\; s & \mapsto \quad r \cup s \end{cases}$$

Formally the operational monad $\widehat{\phi^{\mathcal{R}}}$ induced by this germ $\phi^{\mathcal{R}}$ is not equal to the above operational monad Φ for basic process algebra. However, the two operational semantics are equivalent in a suitable sense

$$\Phi \sim \widehat{\phi^{\mathcal{R}}}$$

as it is shown in the next section. Here already notice that

$$\mu_{U_B} \circ \gamma^\sharp_{T_E U_B} \text{ is a retraction also for } \widehat{\phi^{\mathcal{R}}}$$

Notes. The interpretation of the non-deterministic choice as a semi-lattics join dates back at least to [HP79], where the Plotkin powerdomain is treated as the semi-lattice monad on a category of complete partial orders.

For a textbook on various non-deterministic languages for concurrency, including basic process algebra, see [BW90].

The above idea of quotienting of the terms (of basic process algebra) by an algebraic congruence for defining the programs of a language is not new: it is used, for instance, in the 'Chemical Abstract Machine' approach to operational semantics [BB92] and in some presentations of the 'π-calculus' [Mil90].

11 GSOS is Functorial

One of the largest classes of 'well-behaved' structural operational rules for transition systems is the class of '*GSOS rules*'. These are rules satisfying suitable syntactic restrictions which ensure the compositionality of the corresponding operational models. Almost all transition systems in the literature are defined by means of GSOS rules. For instance, languages like basic process algebra, *CCS*, and *CSP* have GSOS rules.

It is proved here that the operational semantics induced by a set of GSOS rules is always functorial (under the mild assumption that it embeds basic process algebra). This result shows the generality of the functorial approach to operational semantics motivating the claim that it is a first step towards a mathematical theory of 'well-behaved' operational semantics.

A GSOS rule specifies one possible transition for terms of the form $\sigma(u_1, \ldots u_l)$, for σ a given program construct of arity l:

GSOS Rule

$$\frac{\{u_i \xrightarrow{a_{ij}} v_{ij}\}^{1 \leq i \leq l}_{1 \leq j \leq m_i} \qquad \{u_i \stackrel{b_{ij}}{\not\longrightarrow}\}^{1 \leq i \leq l}_{1 \leq j \leq n_i}}{\sigma(u_1, \ldots, u_l) \xrightarrow{a} C[\overrightarrow{u}, \overrightarrow{v}]}$$

The a_{ij}'s and b_{ij}'s are actions in Act; the u_i's and v_{ij}'s are all distinct (meta) variables ranging over terms, the expression $C[\overrightarrow{u}, \overrightarrow{v}]$ is a term formed by the context $C[\overrightarrow{-}]$ and some (meta) variables contained in the set of u_i's and v_{ij}'s. The expression

$$u_i \stackrel{b_{ij}}{\not\longrightarrow}$$

stands for 'u_i cannot perform a transition with action b_{ij}'.

For instance, the rule

$$\frac{u_1 \xrightarrow{a} v_1}{u_1 \,;\, u_2 \xrightarrow{a} v_1 \,;\, u_2}$$

is in GSOS, as well as the rule

$$\frac{u_1 \longrightarrow * \qquad u_2 \xrightarrow{a} v_2}{u_1 \,;\, u_2 \xrightarrow{a} v_2}$$

by considering that a state becomes inert $u \longrightarrow *$ (ie $u \downarrow *$) as a special case of transition $u \xrightarrow{a} v$. In this way, all rules considered so-far are GSOS.

Before setting out to prove the functoriality of GSOS, let us introduce an intermediate notation between transitions and actions of coalgebras $\langle X, k\rangle$ of the behaviour endofunctor

$$BX = \check{\mathcal{P}}(1 + \mathsf{Act} \times X)$$

Write $x \overset{k}{\leadsto} {<}a, y{>}$ for $k(x) \ni {<}a, y{>}$ and $x \overset{k}{\not\leadsto} {<}a, \text{-}{>}$ for 'there exists no y such that ${<}a, y{>}$ is in $k(x)$'. That is,

$$x \overset{k}{\leadsto} {<}a, y{>} \iff x \xrightarrow{a} y \qquad\qquad x \overset{k}{\not\leadsto} {<}a, \text{-}{>} \iff x \overset{a}{\not\longrightarrow}$$

A GSOS rule is then of the form

$$\frac{\{u_i \leadsto {<}a_{ij}, v_{ij}{>}\}^{1\le i\le l}_{1\le j\le m_i} \qquad \{u_i \not\leadsto {<}b_{ij}, \text{-}{>}\}^{1\le i\le l}_{1\le j\le n_i}}{\sigma(u_1, \ldots, u_l) \leadsto {<}a, C[\overrightarrow{u}, \overrightarrow{v}]{>}}$$

Again, one has that $u \leadsto *$ is a special case of $u \leadsto {<}a, u'{>}$.

Now, the proof of the functoriality of GSOS given here is based on the assumption that every set $\mathcal{R}$ of GSOS rules embeds the basic process algebra of the previous section. This does not seem to be a serious restriction, because most of the languages defined by means of GSOS rules do have programs behaving like nil, $a.t$, and $u \,\mathsf{or}\, v$. Therefore, let us assume that the signature Σ of the language contains the basic inert program nil, a unary action-prefixing operator for every action in Act and the binary non-deterministic choice 'or':

$$t ::= x \mid \mathsf{nil} \mid a \mid (t \,\mathsf{or}\, t) \mid \sigma(t, \ldots, t)$$

Moreover, assume that the semi-lattice laws

$$E = \left\{ \begin{array}{rcl} (x \,\mathsf{or}\, y) \,\mathsf{or}\, z & = & x \,\mathsf{or}\, (y \,\mathsf{or}\, z) \\ x \,\mathsf{or}\, y & = & y \,\mathsf{or}\, x \\ x \,\mathsf{or}\, x & = & x \end{array} \right.$$

for the choice construct hold. Thus, the corresponding syntactical monad

$$T = {<}T, \eta, \mu{>}$$

is the free ${<}\Sigma, E{>}$-algebra monad. (Cf Sections 2 and 10.) As a consequence, the embedding $\gamma^\sharp : B \Rightarrow T_E$ of the above behaviour into the syntax of basic process algebra extends to an embedding

$$\gamma^\sharp : B \Rightarrow T$$

into this syntax T. Since the rules $\mathcal{R}$ extend the rules of basic process algebra one also has that this embedding is a retraction for (the operational semantics induced by) $\mathcal{R}$. (Cf previous section.)

GSOS is natural. The claim is that every set $\mathcal{R}$ of GSOS rules over T containing basic process algebra can be seen as a natural transformation

$$\lceil\mathcal{R}\rceil : \Sigma B \Rightarrow BT$$

Moreover, the operational models induced by $\mathcal{R}$ and by $\lceil\mathcal{R}\rceil$ are 'observationally equivalent' in the sense that their coinductive extensions are equal.

The definition of the transformation $\lceil\mathcal{R}\rceil : \Sigma B \Rightarrow BT$ is based on the rules $\mathcal{R}$ as follows. Let the meta-variables r and s range over objects of type $B = \check{\mathcal{P}}(1+\mathsf{Act}\times\text{-})$. For the rules corresponding to basic process algebra, put

$$\lceil\mathcal{R}\rceil(\mathsf{nil}) = \{*\} \qquad \lceil\mathcal{R}\rceil(a\,.\,r) = \{<a, \gamma^\sharp r>\} \qquad \lceil\mathcal{R}\rceil(r \;\mathsf{or}\; s) = r \cup s$$

and, in general, for every rule

$$\frac{\{u_i \rightsquigarrow <a_{ij}, v_{ij}>\}^{1\leq i\leq l}_{1\leq j\leq m_i} \quad \{u_i \not\rightsquigarrow <b_{ij}, \text{-}>\}^{1\leq i\leq l}_{1\leq j\leq n_i}}{\sigma(u_1, \ldots, u_l) \rightsquigarrow <a, C[\overrightarrow{u}, \overrightarrow{v}]>}$$

in $\mathcal{R}$, put

$$<a, C[\overrightarrow{\gamma_X r}, \overrightarrow{x}]> \in \lceil\mathcal{R}\rceil_X(\sigma(r_1, \ldots, r_l))$$

if $<a_{ij}, x_{ij}> \in r_i$ for $1 \leq i \leq l$ and $1 \leq j \leq m_i$, and, for every $x \in X$, $<b_{ij}, x> \notin r_i$ for $1 \leq i \leq l$ and $1 \leq j \leq n_i$. The only difference between $\mathcal{R}$ and $\lceil\mathcal{R}\rceil$ is in the use in the latter of the embedding $\gamma^\sharp : B \Rightarrow T$, which is necessary in order to plug objects of type B into the context $C[\overrightarrow{\text{-}}, \overrightarrow{x}]$. The fact that this embedding is a retraction wrt the operational semantics will ensure that this difference is observationally irrelevant.

To prove that the above definition of the arrow $\lceil\mathcal{R}\rceil_X : \Sigma BX \to BTX$ is natural in X, let us first use the following more suggestive notation.

$$\frac{\{r_i \ni <a_{ij}, x_{ij}>\}^{1\leq i\leq l}_{1\leq j\leq m_i} \quad \{r_i \not\ni <b_{ij}, \text{-}>\}^{1\leq i\leq l}_{1\leq j\leq n_i}}{\sigma(r_1, \ldots, r_l) \overset{\lceil\mathcal{R}\rceil_X}{\rightsquigarrow} <a, C[\overrightarrow{\gamma^\sharp_X r}, \overrightarrow{x}]>}$$

The proof that this definition is natural is a simple generalization of the one given in Section 4 corresponding to the rules for the simple deterministic language used there:

> **Naturality.** The claim is that, for every 'renaming' $f : X \to Y$, the diagram
>
> $$\begin{tikzcd} \Sigma BX \arrow[r, "\Sigma Bf"] \arrow[d, "\lceil\mathcal{R}\rceil_X"'] & \Sigma BY \arrow[d, "\lceil\mathcal{R}\rceil_Y"] \\ BTX \arrow[r, "BTf"'] & BTY \end{tikzcd}$$
>
> commutes. Consider the case of negative premises: if there is no pair $<a, x>$ in $r \in BX$ for a given action a and arbitrary $x \in X$ then

there is also no pair $< a, y >$ in $(Bf)(r) \in BY$ for arbitrary $y \in Y$. Therefore, the problem of proving the naturality of $\lceil \mathcal{R} \rceil$ can be reduced to the problem of proving that the following holds.

$$\begin{tikzcd}
\sigma(r_1,\ldots,r_l) \arrow[rr, mapsto, "\Sigma Bf"] \arrow[d, "\lceil\mathcal{R}\rceil_X"'] & & \sigma((Bf)(r_1),\ldots,(Bf)(r_l)) \arrow[d, "\lceil\mathcal{R}\rceil_Y"] \\
<a, C[\overrightarrow{\gamma^\sharp_X r}, \vec{x}]> \arrow[r, mapsto, "BTf"'] & <a, (Tf)(C[\overrightarrow{\gamma^\sharp_X r}, \vec{x}])> \arrow[r, phantom, "="] & <a, C[\overrightarrow{\gamma^\sharp_Y (Bf)(r)}, \overrightarrow{f\vec{x}}]>
\end{tikzcd}$$

But, again, like in Section 4, the equation

$$(Tf)(C[\overrightarrow{\gamma^\sharp_X r}, \overrightarrow{x}]) = C[\overrightarrow{\gamma^\sharp_Y (Bf)(r)}, \overrightarrow{f\vec{x}}]$$

is an immediate consequence of the naturality of the retraction $\gamma^\sharp$ from B to T and of the GSOS condition that all variables in $C[\overrightarrow{u}, \overrightarrow{v}]$ are of the form u_i or v_{ij} (hence $(Tf)C[\ldots] = C[(Tf)\ldots]$).

Notice that it is very easy to violate the naturality of $\lceil \mathcal{R} \rceil$ by relaxing the assumptions on $\mathcal{R}$. For instance, one cannot drop the assumption that all meta-variables v_{ij} on the right hand side of the premises $u_i \xrightarrow{a_{ij}} v_{ij}$ are different. Indeed, one would then permit rules like

$$\frac{u_1 \xrightarrow{a} v \qquad u_2 \xrightarrow{b} v}{\sigma(u_1, u_2) \xrightarrow{c} \mathsf{nil}}$$

which fails to be natural: under the above translation $\mathcal{R} \mapsto \lceil \mathcal{R} \rceil$ and in absence of other rules for the operator σ, one has that $\sigma(<a, x_1>, <b, x_2>)$ cannot perform any transition while, by using the renaming

$$f(x_1) = y = f(x_2)$$

one has that

$$\sigma(<a, f(x_1)>, <b, f(x_2)>) \rightsquigarrow <c, \mathsf{nil}>$$

There exists however a useful extension of GSOS which is 'well-behaved' in the sense that it induces operational models which are always compositional. It is the so-called '*ntyft*'-format (see notes below) which is obtained by allowing for whole contexts C_i rather than for simple (meta) variables to appear in the left hand side of the premises of the rules:

$$\frac{\{C_i \xrightarrow{a_{ij}} v_{ij}\}^{1\le i\le l}_{1\le j\le m_i} \qquad \{C_i \overset{b_{ij}}{\not\longrightarrow}\}^{1\le i\le l}_{1\le j\le n_i}}{\sigma(u_1, \ldots, u_l) \xrightarrow{a} C}$$

The u_i's and v_{ij}'s are still all distinct meta-variables, but there might now appear some extra meta-variables in the contexts C and C_i. (The induction on these rules is made more problematic by the appearance of contexts also in the premises, hence some restriction (eg, 'stratification') on the use of negative premises is needed.) It is not yet clear whether these rules fit in the present functorial approach.

$\mathcal{R}$ is observationally equivalent to $\lceil\mathcal{R}\rceil$. Like in the example in Section 4, the transformation $\lceil\mathcal{R}\rceil : \Sigma B \Rightarrow BT$ can be made into the germ $\phi^{\mathcal{R}} : \Sigma BT \Rightarrow BT$ of a functorial operational semantics by composing $\lceil\mathcal{R}\rceil$ at the syntax T with the behaviour B applied to the multiplication μ of the syntax:

$$\begin{tikzcd} \Sigma BT \arrow[r, Rightarrow, "\lceil\mathcal{R}\rceil_T"] \arrow[dr, Rightarrow, "\phi^{\mathcal{R}}"'] & BT^2 \arrow[d, Rightarrow, "B\mu"] \\ & BT \end{tikzcd}$$

Spelling out the details, this germ $\phi^{\mathcal{R}} : \Sigma BT \Rightarrow BT$ is defined by 'rules'

$$\frac{\{r_i \ni <a_{ij}, t_{ij}>\}^{1\leq i\leq l}_{1\leq j\leq m_i} \qquad \{r_i \not\ni <b_{ij}, \text{-}>\}^{1\leq i\leq l}_{1\leq j\leq n_i}}{\sigma(r_1,\ldots,r_l) \overset{\phi^{\mathcal{R}}}{\rightsquigarrow} <a, \mu(C[\overrightarrow{\gamma^{\sharp}_T r}, \overrightarrow{t}])>}$$

corresponding to the rules in $\mathcal{R}$. The multiplication $\mu : T^2 \Rightarrow T$ is formally needed in order to remove bracketing and make of the term

$$C[\overrightarrow{\gamma^{\sharp}_T r}, \overrightarrow{t}]$$

(with as *variables* the *terms* $\gamma^{\sharp}_T r_i$ and t_{ij}) a simpler term with the variables of $\gamma^{\sharp}_T r_i$ and t_{ij} as variables. In the sequel, for simplicity, μ is omitted.

For every set X, the function $\phi^{\mathcal{R}}_X : \Sigma BTX \to BTX$ is not only a Σ- but also a $\langle\Sigma, E\rangle$-algebra structure for $\langle BTX, \cup\rangle$. That is, $\phi^{\mathcal{R}}$ is join-preserving. Therefore, by the isomorphism between $\langle\Sigma, E\rangle$- and T-algebras (cf Section 2) it can be seen as an action of the monad T on the composite functor BT:

$$\phi^{\mathcal{R}} : TBT \Rightarrow BT$$

Then, for every coalgebra structure $k : X \to BX$, the germ $\phi^{\mathcal{R}}$ induces an operational model

$$[\![\text{-}]\!]^k_{\lceil\mathcal{R}\rceil} : TX \to BTX$$

by taking the left adjunct of the composite arrow $B\eta_X \circ k : X \to BTX = U^T\langle BTX, \phi^{\mathcal{R}}_X\rangle$ wrt the standard adjunction $F^T \dashv U^T$ from the T-algebras to their carriers

$$\begin{tikzcd} X \arrow[r, "\eta_X"] \arrow[d, "k"'] & TX \arrow[d, "{[\![\text{-}]\!]^k_{\lceil\mathcal{R}\rceil}}"'] \\ BX \arrow[r, "B\eta_X"'] & BTX \end{tikzcd} \qquad \begin{tikzcd} TX \arrow[d, dashed, "{[\![\text{-}]\!]^k_{\lceil\mathcal{R}\rceil} = (B\eta_X \circ k)^{\sharp}}"'] & T^2X \arrow[l, "\mu_X"'] \arrow[d, "{T[\![\text{-}]\!]^k_{\lceil\mathcal{R}\rceil}}"] \\ BTX & TBTX \arrow[l, "\phi^{\mathcal{R}}_X"] \end{tikzcd}$$

(See Section 2.)

Regarding the coalgebra structure $k : X \to BX$ as a set of transitions $x \xrightarrow{a} x'$, with $x, x' \in X$, one can also take the least transition system induced by these transitions and by the rules in $\mathcal{R}$ and obtain another operational model

$$[\![\text{-}]\!]^k_{\mathcal{R}} : TX \to BTX$$

> The claim is that these two operational models are *observationally equivalent* in the sense that their coinductive extensions are the same; in other words, they have the same final coalgebra semantics.

Without loss of generality, let us prove this claim taking for k the 'empty' coalgebra structure $0 : 0 \to B0$ as the base of the induction. Correspondingly, one has the models

$$[\![\text{-}]\!]_{\mathcal{R}} : T0 \to BT0 \qquad \text{and} \qquad [\![\text{-}]\!]_{\lceil\mathcal{R}\rceil} : T0 \to BT0$$

with the set $T0$ of closed terms as carrier. The claim is that, for all closed terms t,

$$[\![t]\!]^{@}_{\lceil\mathcal{R}\rceil} = [\![t]\!]^{@}_{\mathcal{R}}$$

Diagrammatically:

$$\begin{tikzcd}
 & \widehat{B} \arrow[d, "\varphi\ \text{final coalgebra}", "\cong"'] \\
T0 \arrow[ur, shift left, "{[\![\text{-}]\!]^{@}_{\mathcal{R}}}"] \arrow[ur, shift right, "{[\![\text{-}]\!]^{@}_{\lceil\mathcal{R}\rceil}}"'] \arrow[ur, phantom, "="] \arrow[d, shift right, "{[\![\text{-}]\!]_{\mathcal{R}}}"'] \arrow[d, shift left, "{[\![\text{-}]\!]_{\lceil\mathcal{R}\rceil}}"] & B\widehat{B} \\
BT0 \arrow[ur, shift left, "{B[\![\text{-}]\!]^{@}_{\mathcal{R}}}"] \arrow[ur, shift right, "{B[\![\text{-}]\!]^{@}_{\lceil\mathcal{R}\rceil}}"'] &
\end{tikzcd}$$

where, recall, $BX = \check{\mathcal{P}}(1 + \mathsf{Act} \times X)$ and the final B-coalgebra $\langle \widehat{B}, \varphi \rangle$ is described in Section 13.

First notice that

$$[\![t]\!]_{\mathcal{R}} \rightsquigarrow * \iff [\![t]\!]_{\lceil\mathcal{R}\rceil} \rightsquigarrow *$$

Thus, consider, without loss of generality, only the case when t might not become inert. Then, the functions $[\![\text{-}]\!]^{@}_{\mathcal{R}}$ and $[\![\text{-}]\!]^{@}_{\lceil\mathcal{R}\rceil}$ are the *unique* functions which, for all t, satisfy the coinductive definitions

$$[\![t]\!]^{@}_{\mathcal{R}} \;=\; \varphi^{-1}\{<a, [\![t']\!]^{@}_{\mathcal{R}}> \mid [\![t]\!]_{\mathcal{R}} \rightsquigarrow <a, t'>\}$$

and

$$[\![t]\!]^{@}_{\lceil\mathcal{R}\rceil} \;=\; \varphi^{-1}\{<a, [\![t']\!]^{@}_{\lceil\mathcal{R}\rceil}> \mid [\![t]\!]_{\lceil\mathcal{R}\rceil} \rightsquigarrow <a, t'>\}$$

respectively, for φ^{-1} the inverse of the final coalgebra isomorphism $\varphi : \widehat{B} \cong B\widehat{B}$. If one can show that, for all terms t, the identity

$$\{<a, [\![t']\!]^{@}_{\lceil\mathcal{R}\rceil}> \mid [\![t]\!]_{\lceil\mathcal{R}\rceil} \rightsquigarrow <a, t'>\} \;=\; \{<a, [\![t']\!]^{@}_{\lceil\mathcal{R}\rceil}> \mid [\![t]\!]_{\mathcal{R}} \rightsquigarrow <a, t'>\} \qquad (3)$$

holds, then one has that, for all t, both

$$[\![t]\!]^{@}_{\lceil\mathcal{R}\rceil} = \varphi^{-1}\{<a, [\![t']\!]^{@}_{\lceil\mathcal{R}\rceil}> \mid [\![t]\!]_{\mathcal{R}} \rightsquigarrow <a, t'>\}$$

and

$$[\![t]\!]^{@}_{\mathcal{R}} = \varphi^{-1}\{<a, [\![t']\!]^{@}_{\mathcal{R}}> \mid [\![t]\!]_{\mathcal{R}} \rightsquigarrow <a, t'>\}$$

which, by the uniqueness of coinductive extensions, implies that they are the same.

Now, the identity (3) can be proved as follows. Notice that, by definition of $\lceil\mathcal{R}\rceil$,

$$[\![t]\!]_{\lceil\mathcal{R}\rceil} \rightsquigarrow <a, t'> \iff [\![t]\!]_{\mathcal{R}} \rightsquigarrow <a, C[\overrightarrow{u}, \overrightarrow{v}]> \text{ and } t' = C[\overrightarrow{\gamma^{\sharp}_{T0}[\![u]\!]_{\lceil\mathcal{R}\rceil}}, \overrightarrow{v}]$$

It suffices then to show that

$$[\![C[\overrightarrow{\gamma^{\sharp}_{T0}[\![u]\!]_{\lceil\mathcal{R}\rceil}}, \overrightarrow{v}]]\!]^{@}_{\lceil\mathcal{R}\rceil} = [\![C[\overrightarrow{u}, \overrightarrow{v}]]\!]^{@}_{\lceil\mathcal{R}\rceil}$$

For this, one can use the compositionality of functorial operational semantics (the abstract semantics of a term is invariant under substitution of sub-terms with the same abstract semantics) and reduce it to the problem of proving that

$$[\![\gamma^{\sharp}_{T0}[\![u_i]\!]_{\lceil\mathcal{R}\rceil}]\!]^{@}_{\lceil\mathcal{R}\rceil} = [\![u_i]\!]^{@}_{\lceil\mathcal{R}\rceil}$$

holds. This, in turn, is a consequence of the fact that $\gamma^{\sharp}$ is a retraction for $\lceil\mathcal{R}\rceil$, ie $[\![\gamma^{\sharp}_{T0} r]\!]_{\lceil\mathcal{R}\rceil} = r$ (see previous section):

$$\begin{array}{rcll} [\![u_i]\!]^{@}_{\lceil\mathcal{R}\rceil} & = & \varphi^{-1} \circ B[\![\text{-}]\!]^{@}_{\lceil\mathcal{R}\rceil} \circ [\![u_i]\!]_{\lceil\mathcal{R}\rceil} & \text{(unfold)} \\ & = & \varphi^{-1} \circ B[\![\text{-}]\!]^{@}_{\lceil\mathcal{R}\rceil} \circ [\![\gamma^{\sharp}[\![u_i]\!]_{\lceil\mathcal{R}\rceil}]\!]_{\lceil\mathcal{R}\rceil} & \text{(retraction)} \\ & = & [\![\gamma^{\sharp}_{T0}[\![u_i]\!]_{\lceil\mathcal{R}\rceil}]\!]^{@}_{\lceil\mathcal{R}\rceil} & \text{(fold)} \end{array}$$

This concludes the proof.

Structural Coinduction. A more direct way of proving that the set $[\![t]\!]^{@}_{\mathcal{R}}$ is equal to the set $[\![t]\!]^{@}_{\lceil\mathcal{R}\rceil}$ would be to prove that the two sets are equal under the *coinductive* hypothesis that the $[\![t']\!]^{@}_{\mathcal{R}}$'s are equal to the $[\![t']\!]^{@}_{\lceil\mathcal{R}\rceil}$'s. Intuitively, this principle holds by duality wrt the structural induction principle, the algebraic structure of the program constructs being here replaced by the coalgebraic structure of the behaviour observations. However, a formal foundation for this particular 'structural coinduction principle' is still to be investigated.

Guarded Recursion in GSOS. Recall from Section 5 that every set of terms (mutually) recursively defined by means of equations in some variables $x_i \in X$

$$x_1 = t_1[X],\ x_2 = t_2[X], \dots$$

where $t_i[X]$ are elements of TX (hence might contain variables from X), can be seen as a T-coalgebra $k : X \to TX$ by putting $k(x_i) = t_i[X]$. (And vice versa.) Also recall that a system of (mutually) recursive definitions $k : X \to TX$ is *guarded* if it factorizes through a coalgebra

$$g : X \to BTX = \check{\mathcal{P}}(1 + \mathsf{Act} \times TX)$$

of the composite endofunctor BT in the sense that

$$\begin{tikzcd} X \arrow[rr, "k"] \arrow[d, "g"'] & & TX \\ BTX \arrow[rr, "\gamma^\sharp_{TX}"'] & & T^2X \arrow[u, "\mu_X"'] \end{tikzcd}$$

commutes, that is, $k = \mu_X \circ \gamma^\sharp_{TX} \circ g : X \to TX$, where $\mu : T^2 \Rightarrow T$ is the multiplication of the syntactical monad T (cf Section 2) and $\gamma^\sharp : B \Rightarrow T$ is the retraction for basic process algebra. Clearly:

$$g(x_i) = \{<a_{i_1}, t_{i_1}>, \dots, <a_{i_n}, t_{i_n}>\}$$

that is, the equations $x_i = t_i$ are guarded if they are of the form

$$x_i = (a_{i_1} . t_{i_1})\ \mathsf{or}\ \dots\ \mathsf{or}\ (a_{i_n} . t_{i_n})$$

Conversely, every BT-coalgebra can be seen as a set of mutually recursive definitions.

Now, for every set $\mathcal{R}$ of GSOS rules, one can take the left adjunct of every $g : X \to BTX = U^T\langle BTX, \phi^{\mathcal{R}}_X\rangle$ wrt the adjunction $F^T \dashv U^T$ from the T-algebras to their carriers:

$$\begin{tikzcd} X \arrow[r, "\eta_X"] \arrow[dr, "g"'] & TX \arrow[d, "\llbracket\text{-}\rrbracket^g_{\lceil\mathcal{R}\rceil}"] \\ & BTX \end{tikzcd} \qquad \begin{tikzcd} TX \arrow[d, dashed, "\llbracket\text{-}\rrbracket^g_{\lceil\mathcal{R}\rceil} = g^\sharp"'] & T^2X \arrow[l, "\mu_X"'] \arrow[d, "T\llbracket\text{-}\rrbracket^g_{\lceil\mathcal{R}\rceil}"] \\ BTX & TBTX \arrow[l, "\phi^{\mathcal{R}}_X"] \end{tikzcd}$$

Then, the desired interpretation of g as a recursive process is obtained by taking the corresponding final coalgebra semantics $(\llbracket\text{-}\rrbracket^g_{\lceil\mathcal{R}\rceil})^@ = (\llbracket\text{-}\rrbracket^g_{\mathcal{R}})^@ : TX \to \widehat{B}$ precomposed with the insertion-of-variables $\eta_X : X \to TX$. With the usual abuse of notation, write $g^{@} : X \to \widehat{B}$ for this composite arrow:

$$\begin{tikzcd}
X \arrow[r, "\eta_X"] \arrow[dr, "g"'] \arrow[rr, bend left, "g^{@}"] & TX \arrow[r, dashed, "(\llbracket\text{-}\rrbracket^g_{\mathcal{R}})^{@}"] \arrow[d, "\llbracket\text{-}\rrbracket^g_{\lceil\mathcal{R}\rceil}"] & \widehat{B} \arrow[d, "\cong"] \\
& BTX \arrow[r, "B(\llbracket\text{-}\rrbracket^g_{\mathcal{R}})^{@}"'] & B\widehat{B}
\end{tikzcd}$$

Notice that no variable binding operator (like, eg, the operator "fix" in the original definition of GSOS) is needed here to deal with recursion.

As an example, let $\mathcal{R}$ be basic process algebra together with the rules for (simple) *interleaving*

$$\frac{u_1 \xrightarrow{a} v_1}{u_1 \parallel u_2 \xrightarrow{a} v_1 \parallel u_2} \qquad \frac{u_2 \xrightarrow{a} v_2}{u_1 \parallel u_2 \xrightarrow{a} u_1 \parallel v_2}$$

and let g be the BT-coalgebra corresponding to the guarded recursive definition

$$x = a\,.\,x \qquad y = (a\,.\,y)\ \textsf{or}\ (b\,.\,x) \qquad z = (a\,.\,z)\ \textsf{or}\ (b\,.\,(x \parallel y))$$

in $X = \{x, y, z\}$. Write, for simplicity,

$$\llbracket\text{-}\rrbracket_g = \llbracket\text{-}\rrbracket^g_{\lceil\mathcal{R}\rceil} : TX \to BTX$$

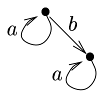
and, correspondingly, let

$$\llbracket\text{-}\rrbracket_g^{@} = (\llbracket\text{-}\rrbracket^g_{\lceil\mathcal{R}\rceil})^{@} = (\llbracket\text{-}\rrbracket^g_{\mathcal{R}})^{@} : TX \to \widehat{B}$$

be its coinductive extension. Then, omitting, as usual, the insertion-of-variables $\eta_X : X \to TX$ and the final coalgebra isomorphism $\widehat{B} \cong B\widehat{B}$,

$$\begin{aligned}
\llbracket a\,.\,t\rrbracket_g^{@} &= \{<a, \llbracket\gamma^{\sharp}_{TX}\llbracket t\rrbracket_g\rrbracket_g^{@}>\} \ = \ \{<a, \llbracket t\rrbracket_g^{@}>\} \\
\llbracket t_1\ \textsf{or}\ t_2\rrbracket_g^{@} &= \llbracket t_1\rrbracket_g^{@} \cup \llbracket t_2\rrbracket_g^{@} \\
\llbracket t_1 \parallel t_2\rrbracket_g^{@} &= \{<a, \llbracket t_1' \parallel t_2\rrbracket_g^{@}> \mid t_1 \xrightarrow{a} t_1'\} \cup \{<a, \llbracket t_1 \parallel t_2'\rrbracket_g^{@}> \mid t_2 \xrightarrow{a} t_2'\}
\end{aligned}$$

$$g^{@}(x) = \{<a, g^{@}(x)>\} =$$ 

$$g^{@}(y) = \{<a, g^{@}(y)>\} \cup \{<b, g^{@}(x)>\} =$$ 

$$g^{@}(z) = \{<a, g^{@}(z)>\} \cup \{<b, \llbracket x \parallel y\rrbracket_g^{@}>\}$$

GSOS models are Φ-models. The functoriality of GSOS gives a systematic method for deriving an adequate denotational model from any set $\mathcal{R}$ of GSOS rules. Another systematic method proposed in the literature (see notes below) permits to derive a *proof system* from any set $\mathcal{R}$ of GSOS rules. This proof system can be used for proving that the programs of the language of $\mathcal{R}$ satisfy assertions in *Hennessy-Milner* logic.

The main result on this proof system is that it is complete wrt a certain class of '*models*' of $\mathcal{R}$. The problem arises then of finding an independent motivation for the definition of GSOS models. It is here shown that the models of a set $\mathcal{R}$ of GSOS rules are exactly the algebras of the operational monad Φ induced by the rules $\mathcal{R}$. This supports the choice of that class of models as the 'natural' one.

A **model for a set of GSOS rules** $\mathcal{R}$ is a triple $\langle X, h, k\rangle$ with $h : TX \to X$ an algebra of the syntactical monad $T = < T, \eta, \mu >$ corresponding to $\mathcal{R}$ and $k : X \to BX = \check{\mathcal{P}}(1 + \mathsf{Act} \times X)$ a B-coalgebra structure such that

$$\sigma(x_1, \ldots, x_l) \overset{h}{\longmapsto} x \overset{k}{\rightsquigarrow} < a, x' >$$

holds if and only if there exists a rule

$$\frac{\{u_i \rightsquigarrow < a_{ij}, v_{ij} >\}^{1\le i\le l}_{1\le j\le m_i} \qquad \{u_i \not\rightsquigarrow < b_{ij}, \text{-} >\}^{1\le i\le l}_{1\le j\le n_i}}{\sigma(u_1, \ldots, u_l) \rightsquigarrow < a, C[\overrightarrow{u}, \overrightarrow{v}] >}$$

in $\mathcal{R}$ such that

$$x_i \overset{k}{\rightsquigarrow} < a_{ij}, y_{ij} > \qquad x_i \overset{k}{\not\rightsquigarrow} < b_{ij}, \text{-} > \qquad C[\overrightarrow{x}, \overrightarrow{y}] \overset{h}{\longmapsto} x'$$

(Formally, this definition is obtained from the original definition of GSOS models by using the one-to-one correspondences between $\langle \Sigma, E\rangle$- and T-algebras and (finitely branching) transition systems and B-coalgebras.)

Next, let $\Phi = < \Phi, \eta, \mu >$ be the operational monad induced by a set of GSOS rules $\mathcal{R}$. That is,

$$\begin{tikzcd}
X \arrow[r, "\eta_X"] \arrow[d, "k"'] & TX \arrow[d, "\Phi k"] \\
BX \arrow[r, "B\eta_X"'] & BTX
\end{tikzcd}
\qquad
\begin{tikzcd}
TX \arrow[d, dashed, "\Phi k"'] & T^2X \arrow[l, "\mu_X"'] \arrow[d, "T\Phi k"] \\
BTX & TBTX \arrow[l, "\phi^{\mathcal{R}}_X"]
\end{tikzcd}$$

Recall, from Section 9, that an algebra of the monad Φ is a triple $< X, h, k >$, with

h a T-algebra and k a B-coalgebra structure over the set X such that the diagram

$$\begin{tikzcd}
TX \arrow[r, "\Phi k"] \arrow[d, "h"'] & BTX \arrow[d, "Bh"] \\
X \arrow[r, "k"'] & BX
\end{tikzcd}$$

commutes. But this means that

$$h(\sigma(x_1, \dots, x_l)) \overset{k}{\rightsquigarrow} < a, x' >$$

holds if and only if

$$\sigma(x_1, \dots, x_l) \overset{\Phi k}{\rightsquigarrow} < a, t > \qquad \text{and} \qquad t \overset{h}{\longmapsto} x'$$

In turn, by definition of Φ, the latter holds if and only if there exists a rule

$$\frac{\{u_i \rightsquigarrow < a_{ij}, v_{ij} >\}^{1 \le i \le l}_{1 \le j \le m_i} \qquad \{u_i \not\rightsquigarrow < b_{ij}, - >\}^{1 \le i \le l}_{1 \le j \le n_i}}{\sigma(u_1, \dots, u_l) \rightsquigarrow < a, C[\overrightarrow{u}, \overrightarrow{v}] >}$$

in $\mathcal{R}$ such that

$$x_i \overset{k}{\rightsquigarrow} < a_{ij}, y_{ij} > \qquad x_i \overset{k}{\not\rightsquigarrow} < b_{ij}, - > \qquad t = C[\overrightarrow{x}, \overrightarrow{y}]$$

Since $ht = x'$ this proves that every GSOS model is a Φ-algebra, and vice versa. In Section 9, Φ-algebras are also called Φ-models, hence this result can be rephrased formally as *GSOS models are* Φ*-models.*

Notes. The notion of a GSOS model has been introduced in [Sim95]. The GSOS rules have been defined in [BIM88], considerably extending a previous definition of 'well-behaved' rules from [dS85]. More recent proposals are the *tyft* format [GV92] extending GSOS without negative premises and its subsequent *ntyft* format [Gro93] mentioned above. It would be interesting to understand whether the functorial approach can deal also with these latter formats.

12 Coalgebraic Bisimulations

There are several notions of *observational equivalence* for a transition system; the most general one corresponds to a relation on its states called *(strong) bisimulation*. The final coalgebra of the behaviour corresponding to transition systems 'classifies' bisimilar states in the sense that two states are bisimilar if and only if they have the same final coalgebra semantics, ie the same abstract global behaviour. In other words, coinduction can be 'pulled back' to bisimulation. As a corollary, the final coalgebra is 'internally fully abstract'.

Categorically, this can be generalized to every behaviour functor B preserving 'weak pullbacks'.

Recall from Section 10 the correspondence between (finitely) non-deterministic transition systems and coalgebras of the behaviour endofunctor

$$BX = \check{\mathcal{P}}(1 + \mathsf{Act} \times X)$$

Recall also the notation $x \overset{k}{\leadsto} < a, x' >$ introduced in Section 11 to express that $< a, x' > \in k(x)$ in a coalgebra structure $k : X \to \check{\mathcal{P}}(1 + \mathsf{Act} \times X)$; in other words, the transition system corresponding to the coalgebra $\langle X, k\rangle$ can perform the transition $x \xrightarrow{a} x'$.

A relation R between the carriers X and Y of two coalgebras $\langle X, k\rangle$ and $\langle Y, \ell\rangle$ **lifts to a (strong) bisimulation** between the two coalgebras when, for all x in X and y in Y such that xRy (ie $< x, y > \in R$), the following three conditions are satisfied.

1. $x \overset{k}{\leadsto} *$ if and only if $y \overset{\ell}{\leadsto} *$
2. if $x \overset{k}{\leadsto} < a, x' >$ then $y \overset{\ell}{\leadsto} < a, y' >$ for some y' such that $x' R y'$
3. and, conversely, if $y \overset{\ell}{\leadsto} < a, y' >$ then $x \overset{k}{\leadsto} < a, x' >$ for some x' such that $x' R y'$

Notice that bisimulations are themselves coalgebras. Indeed, from the above conditions, one can define a coalgebra structure

$$\widetilde{R} : R \to \check{\mathcal{P}}(1 + \mathsf{Act} \times R)$$

on the relation R by putting

$$xRy \overset{\widetilde{R}}{\leadsto} * \iff x \overset{k}{\leadsto} * \;(\iff y \overset{\ell}{\leadsto} *\,)$$

and

$$xRy \overset{\widetilde{R}}{\rightsquigarrow} <a,<x',y'>> \iff x \overset{k}{\rightsquigarrow} <a,x'> \qquad y \overset{\ell}{\rightsquigarrow} <a,y'> \qquad x'Ry'$$

In the sequel, the above notion of bisimulation is also called **ordinary bisimulations**, in order to distinguish it from the following more general notion of 'coalgebraic bisimulation'.

Bisimulations are coalgebras; now the question is: *Is there a coalgebraic description of bisimulation?* For this, consider the two '**legs**' $r_1 : R \to X$ and $r_2 : R \to Y$ obtained by composing the insertion $R \hookrightarrow X \times Y$ of the relation R into the cartesian product $X \times Y$ with the first and second projection, respectively. Now, if the relation R lifts to an ordinary bisimulation, then its legs r_1 and r_2 lift to coalgebra arrows; that is, the two squares in

$$\begin{tikzcd}
 & R \arrow[dl, "r_1"'] \arrow[dr, "r_2"] \arrow[dd, "\widetilde{R}"] & \\
X \arrow[dd, "k"'] & & Y \arrow[dd, "\ell"] \\
 & BR \arrow[dl, "Br_1"'] \arrow[dr, "Br_2"] & \\
BX & & BY
\end{tikzcd}$$

commute. The converse is also true; namely, if a relation lifts to a coalgebra of the above behaviour endofunctor B in a way that its legs also lift to corresponding coalgebra arrows as in the above diagram, then this relation is a bisimulation. Indeed, the first condition is obvious, while the second and the third follow from the commutativity of the left and the right diagram, respectively. Notice that there might be more structures $\widetilde{R}$ making the above diagram commute, corresponding to the several ways in which, in general, a relation can lift to a bisimulation.

The above diagram can be defined wrt any endofunctor B. Call the corresponding notion **coalgebraic bisimulation**. It applies also to endofunctors on categories other than **Set**, by taking a **relation** between two objects X and Y **in a category C** to be a span

$$\begin{tikzcd}
 & R \arrow[dl, "r_1"'] \arrow[dr, "r_2"] & \\
X & & Y
\end{tikzcd}$$

which is monic in the sense that the two legs are **jointly monic** in **C**; that is, if f and g are two 'parallel' arrows such that

$$r_1 \circ f = r_1 \circ g \quad \text{and} \quad r_2 \circ f = r_2 \circ g$$

then f is equal to g. (To be precise, a monic span is not a relation, but just one representative of an equivalence class (of monic spans) which forms the actual relation – more details below.) Then, a relation R between the carriers X and Y

of two coalgebras $\langle X, k\rangle$ and $\langle Y, \ell\rangle$ of an endofunctor B on $\mathbf{C}$ lifts to a coalgebraic bisimulation if there exists a B-coalgebra structure $\widetilde{R} : R \to BR$ which makes

$$\begin{tikzcd}
 & R \arrow[dl, "r_1"'] \arrow[dr, "r_2"] \arrow[dd, "\widetilde{R}"] & \\
X \arrow[dd, "k"'] & & Y \arrow[dd, "\ell"] \\
 & BR \arrow[dl, "Br_1"] \arrow[dr, "Br_2"'] & \\
BX & & BY
\end{tikzcd}$$

commute. Notice the stress is put on the fact that the legs of the relation R *lift* to coalgebra arrows, rather than on the actual (possibly not unique) coalgebraic structure of R. Therefore, let us *forget* about the coalgebraic structure of R and write

$$\begin{tikzcd}
 & R \arrow[dl, "r_1"'] \arrow[dr, "r_2"] \arrow[d, phantom, "\sim", very near start] & \\
\langle X, k\rangle & & \langle Y, \ell\rangle
\end{tikzcd}$$

to express that R is a relation between the carriers X and Y which lifts to a bisimulation between the coalgebras $\langle X, k\rangle$ and $\langle Y, \ell\rangle$.

A canonical way of defining relations is by *pullbacks*: for any diagram

$$\begin{tikzcd}
X \arrow[dr, "f"'] & & Y \arrow[dl, "g"] \\
 & Z &
\end{tikzcd}$$

the two legs of the corresponding pullback

$$\begin{tikzcd}
 & R \arrow[dl, "r_1"'] \arrow[dr, "r_2"] \arrow[dd, phantom, "\lrcorner" rotate=-45, very near start] & \\
X \arrow[dr, "f"'] & & Y \arrow[dl, "g"] \\
 & Z &
\end{tikzcd}$$

are jointly monic (by the universal property of the pullback). For instance, in $\mathbf{Set}$, the pullback of two functions f and g is the relation $\{<x, y> \mid fx = gy\}$. Another example is given by the equality relation: the pullback

$$\begin{tikzcd}
 & \mathrm{EQ}(X) \arrow[dl, "e_1"'] \arrow[dr, "e_2"] \arrow[dd, phantom, "\lrcorner" rotate=-45, very near start] & \\
X \arrow[dr, equal] & & X \arrow[dl, equal] \\
 & X &
\end{tikzcd}$$

is the **equality relation on the object** X in a category $\mathbf{C}$ with pullbacks.

> **The equality relation always lifts to a coalgebraic bisimulation.** Firstly, notice that the two legs e_1 and e_2 of the equality are the same. Next, consider the 'diagonal' $\mathsf{d}_X : X \to \mathrm{EQ}(X)$
>
> $$\begin{tikzcd} & X \arrow[d, dashed, "\mathsf{d}_X"] \arrow[ddl, bend right, "\mathsf{id}_X"'] \arrow[ddr, bend left, "\mathsf{id}_X"] & \\ & \mathrm{EQ}(X) \arrow[dl, "e_1"'] \arrow[dr, "e_2"] & \\ X \arrow[dr, equal] & & X \arrow[dl, equal] \\ & X & \end{tikzcd}$$
>
> given by the universal property of $\mathrm{EQ}(X)$. (In **Set**, the value of the diagonal d_X at an element x of X is the pair $< x, x >$.) For any endofunctor B and any B-coalgebra $\langle X, k\rangle$ since the composite $e_i \circ \mathsf{d}_X$ is the identity on X, the diagram
>
> $$\begin{tikzcd} \mathrm{EQ}(X) \arrow[r, "e_i"] \arrow[d, "e_i"'] \arrow[dr, phantom, "//"] & X \arrow[r, "k"] & BX \arrow[r, "B\mathsf{d}_X"] \arrow[dr, phantom, "\backslash\backslash"] & B(\mathrm{EQ}(X)) \arrow[d, "Be_i"] \\ X \arrow[rrr, "k"'] & & & BX \end{tikzcd}$$
>
> commutes; hence, the composite $B\mathsf{d}_X \circ k \circ e_i$ lifts the equality $\mathrm{EQ}(X)$ to a bisimulation on the coalgebra $\langle X, k\rangle$:
>
> $$\begin{tikzcd} & \mathrm{EQ}(X) \arrow[dl, "e_1"'] \arrow[dr, "e_2"] \arrow[d, phantom, "\sim"] & \\ \langle X, k\rangle & {} & \langle X, k\rangle \end{tikzcd}$$

Next, let B be an endofunctor on a category $\mathbf{C}$ with pullbacks. Recall that pullbacks, like all universals, are determined by two conditions: *uniqueness* and *existence*. When only the existence part is known to hold one speaks of a **weak pullback** (and of a **weak universal** in general). Now, not all pullbacks lift to B-bisimulations, but a sufficient condition is that the functor B **preserves weak pullbacks**. That is, if the image under B of a weak pullback is still a weak pullback, then every pullback in $\mathbf{C}$ of arrows which are coalgebra homomorphisms lifts to a B-bisimulation. Indeed, since pullbacks are also weak pullbacks, for all $f : \langle X, k\rangle \to \langle Z, j\rangle$ and $g : \langle Y, \ell\rangle \to$

$\langle Z, j \rangle$ in $\mathbf{C}_B$, the *existence* of a (possibly not unique) suitable coalgebra structure $\widetilde{R} : R \to BR$ for the pullback R of f and g in $\mathbf{C}$ is ensured by the weakly universal property of the weak pullback BR:

$$\begin{tikzcd}
& R \arrow[dl, "r_1"'] \arrow[dr, "r_2"] \arrow[rrrr, dotted, "\widetilde{R}"] & & & & BR \arrow[dl, "Br_1"'] \arrow[dr, "Br_2"] & \\
X \arrow[dr, "f"'] \arrow[rrrr, bend left, "k"] & & Y \arrow[dl, "g"] \arrow[rrrr, bend right, "\ell"] & & BX \arrow[dr, "Bf"] & & BY \arrow[dl, "Bg"] \\
& Z \arrow[rrrr, "j"'] & & & & BZ &
\end{tikzcd}$$

(The coalgebra structures k and ℓ turn the legs of R into a cone over the diagram for which BR is a weak pullback.)

Pullbacks lift to ordinary bisimulations. Let us check that the behaviour functor $BX = \check{\mathcal{P}}(1 + \mathsf{Act} \times X)$ preserves weak pullbacks and hence, by the above argument, pullbacks lift to (ordinary) bisimulations.

Let us consider the functor $BX = \check{\mathcal{P}}(\mathsf{Act} \times X)$; the proof carries over trivially to the case $BX = \check{\mathcal{P}}(1 + \mathsf{Act} \times X)$. The problem of showing that the functor B preserves weak pullbacks can be reduced to the problem of showing that B maps (ordinary) pullbacks to weak pullbacks. Indeed, the following holds.

In Set, weak pullbacks embed pullbacks. That is, the diagram

$$\begin{tikzcd}
W \arrow[r, "w_2"] \arrow[d, "w_1"'] & Y \arrow[d, "g"] \\
X \arrow[r, "f"'] & Z
\end{tikzcd}$$

is a weak pullback diagram if and only if there exists an injection $m : R \rightarrowtail W$ of the pullback $R = \{<x, y>|\ fx = gy\}$ of f and g into W such that

$$\begin{tikzcd}
R \arrow[dr, tail, "m"] \arrow[drr, bend left, "r_2"] \arrow[ddr, bend right, "r_1"'] & & \\
& W \arrow[r, "w_2"] \arrow[d, "w_1"] & Y \arrow[d, "g"] \\
& X \arrow[r, "f"'] & Z
\end{tikzcd}$$

commutes.

Therefore, if

$$BR = \{<a, x, y>|\ a \in \mathsf{Act},\ fx = gy\}$$

is a weak pullback for $Bf : BX \to BZ$ and $Bg : BY \to BZ$, the set BW inherits the weak universality of BR by means of the mediating arrow $Bm : BR \to BW$.

In turn, in order to prove that BR is a weak pullback for Bf and Bg it suffices to prove that the (ordinary) pullback R' of Bf and Bg factorizes through it in the sense that there exists a function $h : R' \to BR$ such that $r'_i = Br_i \circ h$:

$$\begin{tikzcd}
R' \arrow[dr, dotted, "h"] \arrow[drr, bend left, "r'_2"] \arrow[ddr, bend right, "r'_1"'] \arrow[dr, phantom, very near start, "\lrcorner"] & & \\
& BR \arrow[r, "Br_2"] \arrow[d, "Br_1"] & BY \arrow[d, "Bg"] \\
& BX \arrow[r, "Bf"'] & BZ
\end{tikzcd}$$

Indeed, then every other cone (f', g') over the co-span $\langle Bf, Bg\rangle$ factorizes through the pullback as follows.

$$\begin{tikzcd}
\cdot \arrow[dr, dashed] \arrow[ddrrr, bend left, "g'"] \arrow[dddrr, bend right, "f'"'] & & & \\
& R' \arrow[dr, dotted, "h"] \arrow[drr, bend left, "r'_2"] \arrow[ddr, bend right, "r'_1"'] \arrow[dr, phantom, very near start, "\lrcorner"] & & \\
& & BR \arrow[r, "Br_2"] \arrow[d, "Br_1"] & BY \arrow[d, "Bg"] \\
& & BX \arrow[r, "Bf"'] & BZ
\end{tikzcd}$$

Let us now try and define such a function $h : R' \to BR$ from the pullback R' of Bf and Bg to the image under B of the pullback R of f and g. By definition of pullbacks in **Set**, the set R' is made of those pairs

$$<\{<a_i, x_i>\}_{i\in I}, \{<a_j, y_j>\}_{j\in J}>$$

such that the index sets I and J are finite and

$$Bf\{<a_i, x_i>\}_{i\in I} = Bg\{<a_j, y_j>\}_{j\in J}$$

The latter holds if and only if for every $i \in I$ there exists a $j \in J$ such that

$$<a_i, fx_i> = <a_j, gy_j> \qquad \text{(ie } a_i = a_j,\ fx_i = gy_j)$$

and, conversely, for every $j \in J$ there exists an $i \in I$ such that $<a_i, fx_i> = <a_j, gy_j>$. But then one can define $h : R' \to BR$ as mapping every pair

$$\{<a_i, x_i>\}_{i\in I}\ R'\ \{<a_j, y_j>\}_{j\in J}$$

to the set

$$\{<a_i, x_i, y_j> \mid a_i = a_j,\ fx_i = gy_j\} \in BR$$

This gives the desired factorization. Notice that the mediating function h is not unique and that this construction also applies to the simpler behaviour $BX = 1 + \mathsf{Act} \times X$.

The semantic import of coalgebraic bisimulation is shown by a list of properties which relate it to final coalgebras. One property is that coinductive extensions identify bisimilar elements; in particular, if two programs are bisimilar, then they have the same final coalgebra semantics. Another way of expressing this fact is to say that the equality on the final coalgebra lifts to the final bisimulation (in a suitable category of relations). As a corollary, final coalgebras are **internally fully-abstract**, in the sense that in a final coalgebra one cannot distinguish between bisimilar elements; this property is also called **strong extensionality**.

Next, if the pullback of two coinductive extensions lifts to a bisimulation, like, eg, when the functor B under consideration preserves weak pullbacks, then this pullback is the greatest relation lifting to a bisimulation. Together with the above property that coinductive extensions identify bisimilar elements, this gives that two programs have the same final coalgebra semantics if and only if they are bisimilar. In other words, coinduction can be 'pulled back' to bisimulation.

Let us look at these properties in detail.

Coinductive extensions identify bisimilar elements. That is, for any relation R lifting to a bisimulation the following diagram commutes.

$$\begin{tikzcd}[column sep=small]
 & R \arrow[dl, "r_1"'] \arrow[dr, "r_2"] \arrow[d, phantom, "\sim"] & \\
\langle X, k\rangle \arrow[dr, "k^@"'] & {} & \langle Y, \ell\rangle \arrow[dl, "\ell^@"] \\
 & \langle \widehat{B}, \varphi\rangle &
\end{tikzcd}$$

This is a trivial consequence of the fact that both composites in the diagram are coalgebra arrows to the final coalgebra, hence they must be the same.

Corollary (Strong Extensionality): Final coalgebras are internally fully-abstract. That is, every relation which lifts to a bisimulation on the final coalgebra has equal legs:

$$\begin{tikzcd}[column sep=small]
 & R \arrow[dl, "r_1"'] \arrow[dr, "r_2"] \arrow[d, phantom, "\sim"] & \\
\langle \widehat{B}, \varphi\rangle \arrow[dr, equal] & {} & \langle \widehat{B}, \varphi\rangle \arrow[dl, equal] \\
 & \langle \widehat{B}, \varphi\rangle &
\end{tikzcd}$$

The equality on the final coalgebra lifts to the final bisimulation. Consider the category having as objects relations lifting to bisimulations of an endofunctor B and as arrows triples of arrows $<r, f, g>$ making everything in sight in

$$
\begin{tikzcd}
 & R \arrow[dl, "r_1"'] \arrow[dr, "r_2"] \arrow[rrr, "r"] & & & R' \arrow[dl, "r'_1"'] \arrow[dr, "r'_2"] & \\
\langle X, k\rangle \arrow[rrr, "f"'] & & \langle Y, \ell\rangle \arrow[rrr, "g"] & \langle X', k'\rangle & & \langle Y', \ell'\rangle
\end{tikzcd}
$$

commute – where f and g are arrows in $\mathbf{C}_B$, while r is an arrow in $\mathbf{C}$. Then the equality $\mathrm{EQ}(\widehat{B})$ on (the carrier of) the final coalgebra is the final object of this category. This is an immediate consequence of the fact that $\mathrm{EQ}(\widehat{B})$ is a pullback (in $\mathbf{C}$):

$$
\begin{tikzcd}
 & R \arrow[dl, "r_1"'] \arrow[dr, "r_2"] \arrow[rrr, dashed] & & & \mathrm{EQ}(\widehat{B}) \arrow[dl, "e_1"'] \arrow[dr, "e_2"] & \\
\langle X, k\rangle \arrow[rrr, "k^{@}"'] & & \langle Y, \ell\rangle \arrow[rrr, "\ell^{@}"] & \langle \widehat{B}, \varphi\rangle & & \langle \widehat{B}, \varphi\rangle
\end{tikzcd}
$$

That is, from any relation R lifting to a bisimulation there is a mediating arrow to the equality $\mathrm{EQ}(\widehat{B})$ on the final coalgebra because the two legs of R can be coinductively prolonged to form a suitable cone on (the carrier of) the final coalgebra.

Greatest bisimulations. So far, we have made no distinction between relations and monic spans (like pullbacks). To be precise, one should first define an equivalence relation among monic spans with a common codomain and then take the corresponding equivalence classes as the actual relations; this equivalence relation is defined as follows.

For any two monic spans with a common codomain

$$
\begin{tikzcd}
 & M \arrow[dl, "m_1"'] \arrow[dr, "m_2"] & \\
X & & Y
\end{tikzcd}
\qquad
\begin{tikzcd}
 & M' \arrow[dl, "m'_1"'] \arrow[dr, "m'_2"] & \\
X & & Y
\end{tikzcd}
$$

write

$$
\begin{tikzcd}
 & M \arrow[dl, "m_1"'] \arrow[dr, "m_2"] & \\
X & & Y
\end{tikzcd}
\leq
\begin{tikzcd}
 & M' \arrow[dl, "m'_1"'] \arrow[dr, "m'_2"] & \\
X & & Y
\end{tikzcd}
$$

if there is an arrow $f : M \to M'$ such that M_i factorizes as $M'_i \circ f$, for both $i = 1$

and $i = 2$:

$$
\begin{tikzcd}
M \arrow[r, "f"] \arrow[d, "m_1"'] \arrow[drr, "m_2"' near end] & M' \arrow[dll, "m_1'" near end] \arrow[dr, "m_2'"] & \\
X & & Y
\end{tikzcd}
$$

The two monic spans are then equivalent (hence represent the same relation) if the converse also holds, that is, if also

$$
\begin{tikzcd}[column sep=small]
& M \arrow[dl, "m_1"'] \arrow[dr, "m_2"] & & & & M' \arrow[dl, "m_1'"'] \arrow[dr, "m_2'"] & \\
X & & Y & \geq & X & & Y
\end{tikzcd}
$$

The above defines a partial order '$\leq$' of relations (and also of relations which lift to bisimulations). If the cartesian product $X \times Y$ of two objects X and Y in a category exists, then its equivalence class is the *greatest relation* between X and Y wrt this partial order. If the category has finite limits, then products are pullbacks wrt the final object; in particular,

$$
\begin{tikzcd}
& X \times Y \arrow[dl, "\mathsf{fst}"'] \arrow[dr, "\mathsf{snd}"] & \\
X \arrow[dr, dashed] & & Y \arrow[dl, dashed] \\
& 1 &
\end{tikzcd}
$$

In semantics, the 'base' category $\mathbf{C}$ should, like $\mathbf{Set}$, have all finite limits. The same cannot be said in general of the category $\mathbf{C}_B$ of coalgebras of the behaviour endofunctor B. What certainly is true is that the behaviour should have a final coalgebra, that is, the category $\mathbf{C}_B$ should have a final object. Now, recall that the coinductive extension $k^{@} : X \to \widehat{B}$ of a coalgebra structure $k : X \to BX$ is the unique coalgebra arrow from the coalgebra $\langle X, k\rangle$ to the final coalgebra $\langle \widehat{B}, \varphi\rangle$; then one can take the pullback (in $\mathbf{C}$) of two coinductive extensions and, *if it lifts to a bisimulation* between the corresponding coalgebras

$$
\begin{tikzcd}
& \sim \arrow[dl] \arrow[dr] & \\
\langle X, k\rangle \arrow[dr, "k^{@}"'] & & \langle Y, \ell\rangle \arrow[dl, "\ell^{@}"] \\
& \langle \widehat{B}, \varphi\rangle &
\end{tikzcd}
$$

then this is the **greatest** (relation lifting to a) **bisimulation** between the coalgebras $\langle X, k\rangle$ and $\langle Y, \ell\rangle$.

Write $\overset{k,\ell}{\sim}$ for the relation obtained above by 'pulling back' the coinductive extensions of the coalgebra structures k and ℓ. Then, in **Set**, if the relation $\overset{k,\ell}{\sim}$ lifts to a bisimulation,

$$x \overset{k,\ell}{\sim} y \iff k^{@}(x) = \ell^{@}(y)$$

for any two elements $x \in X$ and $y \in Y$. (The implication from left to right follows the property that coinductive extensions always identify bisimilar elements.) Semantically, for an operational model $[\![\text{-}]\!] : TX \to BTX$ with syntax T and behaviour $BX = \check{\mathcal{P}}(1 + \mathsf{Act} \times X)$, two programs $t, t' \in TX$ are bisimilar if and only if they have the same final coalgebra semantics:

$$t \overset{[\![\text{-}]\!]}{\sim} t' \iff [\![t]\!]^{@} = [\![t']\!]^{@}$$

Notice the underlying assumption that the pullback $\overset{[\![\text{-}]\!]}{\sim}$ lifts to a bisimulation on the operational model $[\![\text{-}]\!]$:

$$\begin{tikzcd}
 & \overset{\cdot}{\sim} \arrow[dl] \arrow[dr] \arrow[dd, phantom, very near start, "\lrcorner" rotate=-45] & \\
\langle TX, [\![\text{-}]\!] \rangle \arrow[dr, "{[\![\text{-}]\!]^{@}}"'] & & \langle TX, [\![\text{-}]\!] \rangle \arrow[dl, "{[\![\text{-}]\!]^{@}}"] \\
 & \langle \widehat{B}, \varphi \rangle &
\end{tikzcd}$$

As shown above, pullbacks lift to ordinary bisimulations, ie to the bisimulations of the behaviour functor $BX = \check{\mathcal{P}}(1 + \mathsf{Act} \times X)$. As a consequence, one can thus obtain the familiar result that the union of all bisimulations on a transition system is itself a bisimulation.

Bisimulations along arrows. The fact that coinductive extensions can be pulled back to bisimulations can be generalized to coinductive extensions *along arrows*. This leads to a new, more general notion of ordinary bisimulation in which not only the actions but also some (properties of the) states can be observed.

Recall that final coalgebras $\widehat{B} \cong B\widehat{B}$ are a special case of cofree coalgebras $DX \cong X \times BDX$ (namely $\widehat{B} = D1$) and that, correspondingly, the coinduction principle of final coalgebras generalizes to the arbitrary cofree coalgebras: for every coalgebra structure $k : X \to BX$ and arrow $f : X \to Z$ one has a unique coalgebra arrow $f^\flat : \langle X, k\rangle \to \langle DZ, \mathsf{snd}_Z\rangle$, namely the coinductive extension of k along f:

$$\begin{tikzcd} & X \arrow[r, "k"] \arrow[dl, "f"'] \arrow[d, dashed, "f^\flat"] & BX \arrow[d, "Bf^\flat"] \\ Z & DZ \arrow[l, "\varepsilon_Z = \mathsf{fst}_Z"] \arrow[r, "\mathsf{snd}_Z"'] & BDZ \end{tikzcd}$$

(Cf Section 7.)

Next, consider a **relation** R **between two arrows** $f : X \to Z$ **and** $g : Y \to Z$ **over the object** Z, that is, a relation between X and Y such that the diagram

$$\begin{tikzcd} & R \arrow[dl, "r_1"'] \arrow[dr, "r_2"] & \\ X \arrow[dr, "f"'] & & Y \arrow[dl, "g"] \\ & Z & \end{tikzcd}$$

commutes. Then, if X and Y carry coalgebra structures $k : X \to BX$ and $\ell : Y \to BY$ respectively and the relation R lifts to a bisimulation between them

$$\begin{tikzcd} & R \arrow[dl, "r_1"'] \arrow[dr, "r_2"] \arrow[d, phantom, "\sim"] & \\ \langle X, k\rangle & {} & \langle Y, \ell\rangle \end{tikzcd}$$

then also the diagram

$$\begin{tikzcd} & R \arrow[dl, "r_1"'] \arrow[dr, "r_2"] \arrow[d, phantom, "\sim"] & \\ \langle X, k\rangle \arrow[dr, "f^\flat"'] & {} & \langle Y, \ell\rangle \arrow[dl, "g^\flat"] \\ & \langle DZ, \mathsf{snd}_Z\rangle & \end{tikzcd}$$

commutes, because both composites $f^\flat \circ r_1$ and $g^\flat \circ r_2$ fit as the unique coinductive extension of the (no matter which!) coalgebra structure on R along the composite $f \circ r_1 = g \circ r_2 : R \to Z$.

If pullbacks lift to B-bisimulation, then the pullback (in the base category) of the coinductive extensions $f^\flat$ and $g^\flat$ of k and ℓ along f and g is the greatest relation between f and g which lifts to a bisimulation between $\langle X, k\rangle$ and $\langle Y, \ell\rangle$.

As an example, consider the simple behaviour $BX = 1 + \mathsf{Act} \times X$ and, correspondingly, ordinary bisimulation for deterministic transition systems. Let the set Act of actions be trivial, that is, let Act be made of only one action a. Let $\langle X, k\rangle$ and $\langle Y, \ell\rangle$ be the same coalgebra having as carrier the set $\mathbb{Z}$ of integers and as structure $\ell : \mathbb{Z} \to B(\mathbb{Z})$ the one corresponding to the following (deterministic) transition system: 0 is inert, a positive integer n performs a transition to its predecessor $n-1$, and a negative integer $-n$ performs a transition to its successor $-n+1$:

$$0 \downarrow * \qquad n \xrightarrow{a} n-1 \qquad -n \xrightarrow{a} -n+1$$

(Cf Example in Section 7.) Finally, let Z be the three-elements set $\{0, \clubsuit, \diamondsuit\}$. Thus:

$$X = \mathbb{Z} = Y \qquad Z = \{0, \clubsuit, \diamondsuit\} \qquad \mathsf{Act} = \{a\}$$

Now, different bisimulations are possible according to the choice of the functions $f, g : \mathbb{Z} \to \{0, \clubsuit, \diamondsuit\}$. Let us fix the function $g : \mathbb{Z} \to \{0, \clubsuit, \diamondsuit\}$ to be the one mapping odd numbers to $\clubsuit$ and even numbers to $\diamondsuit$. If f is equal to g, then every number is bisimilar to itself and to its opposite. For instance,

$$f^\flat(-3) = \clubsuit \xrightarrow{a} \diamondsuit \xrightarrow{a} \clubsuit \xrightarrow{a} 0 = g^\flat(3)$$

and thus -3 is bisimilar to 3 (wrt g).

The above amounts to assume that one can observe in both transition systems whether a number is odd or even. If, instead, in the first transition system one can observe this only for positive numbers, thus, eg, $f(-n) = 0$ and $f(n) = g(n)$, then one has that a positive number n is bisimilar to both $-n$ and n (wrt f and g) but its opposite $-n$ is not bisimilar to any number in the second transition system.

Finally, if one cannot observe at all in the first transition system whether a number is odd or even (ie $f(z) = 0$ for all $z \in \mathbb{Z}$) then only the two 0's are bisimilar.

(Notice that the arrows f and g can be regarded as *abstract interpretations* of the states.)

Another example is when one has a distinguished subset $Obs(X) \subset X$ of states which are 'observable'. This can be expressed by taking $Z = Obs(X) \cup \{\bot\}$ and $f : X \to Obs(X) \cup \{\bot\}$ to be

$$f(x) = \begin{cases} x & \text{if } x \in Obs(X) \\ \bot & \text{otherwise} \end{cases}$$

Bisimulations vs Congruences

Consider the case in which, like for the above behaviour functor, pullbacks lift to coalgebraic bisimulations. Then, in any situation like in functorial operational semantics

$$
\begin{tikzcd}[column sep=huge, row sep=large]
\Sigma T0 \arrow[r, "\Sigma \llparenthesis\text{-}\rrparenthesis^{\#}"] \arrow[d, "\text{initial algebra}"', "\cong"] & \Sigma\widehat{B} \arrow[d, "\llparenthesis\text{-}\rrparenthesis"] \\
T0 \arrow[r, shift left, "\text{initial algebra semantics } \llparenthesis\text{-}\rrparenthesis^{\#}"] \arrow[r, shift right, "\text{final coalgebra semantics } [\![\text{-}]\!]^{@}"'] \arrow[r, phantom, "\shortparallel"] \arrow[d, "{[\![\text{-}]\!]}"'] & \widehat{B} \arrow[d, "\cong"', "\text{final coalgebra}"] \\
BT0 \arrow[r, "B[\![\text{-}]\!]^{@}"'] & B\widehat{B}
\end{tikzcd}
$$

in which both an operational and a denotational model are given and the denotational model is adequate wrt the operational one in the sense that initial algebra and final coalgebra semantics coincide, one has that 'bisimulation is a congruence'. That is, if

$$u_1 \overset{[\![-]\!]}{\sim} v_1, \ldots, u_n \overset{[\![-]\!]}{\sim} v_n$$

for terms u_i and v_i, then, for every n-ary construct σ in Σ,

$$\sigma(u_1, \ldots, u_n) \overset{[\![-]\!]}{\sim} \sigma(v_1, \ldots, v_n)$$

Indeed, using the hypothesis that pullbacks lift to coalgebraic bisimulations, one has that, for all terms t and t',

$$t \overset{[\![-]\!]}{\sim} t' \iff [\![t]\!]^{@} = [\![t']\!]^{@}$$

hence, for $i = 1, \ldots, n$,

$$[\![u_i]\!]^{@} = [\![v_i]\!]^{@}$$

and thus

$$
\begin{array}{rcl}
[\![\sigma(u_1, \ldots, u_n)]\!]^{@} & = & \llparenthesis\, \sigma(u_1, \ldots, u_n) \,\rrparenthesis^{\#} \\
 & = & \llparenthesis\, \sigma \,\rrparenthesis(\llparenthesis\, u_1 \,\rrparenthesis^{\#}, \ldots, \llparenthesis\, u_n \,\rrparenthesis^{\#}) \\
 & = & \llparenthesis\, \sigma \,\rrparenthesis([\![u_1]\!]^{@}, \ldots, [\![u_n]\!]^{@}) \\
 & = & \llparenthesis\, \sigma \,\rrparenthesis([\![v_1]\!]^{@}, \ldots, [\![v_n]\!]^{@}) \\
 & = & [\![\sigma(v_1, \ldots, v_n)]\!]^{@}
\end{array}
$$

Therefore,

$$\sigma(u_1, \ldots, u_n) \overset{[\![-]\!]}{\sim} \sigma(v_1, \ldots, v_n)$$

which means that the (bisimulation) relation $\overset{[\![-]\!]}{\sim}$ is a *congruence*. In general, a relation R between the carriers X and Y of two Σ-algebras $\langle X, h\rangle$ and $\langle Y, l\rangle$ is a

congruence when, for all $x_1, \ldots, x_n$ in X and $y_1, \ldots, y_n$ in Y and n-ary construct σ in Σ,

$$\text{if } x_1 \, R \, y_1, \ldots, x_n \, R \, y_n \text{ then } h(\sigma(x_1, \ldots, x_n)) \; R \; l(\sigma(y_1, \ldots, y_n))$$

Diagrammatically, this is equivalent to saying that the relation R lifts to the Σ-algebras in the sense that there exists a Σ-algebra structure $\widetilde{R} : \Sigma R \to R$ making the following diagram commute.

$$\begin{tikzcd}
 & \Sigma R \arrow[dl, "\Sigma r_1"'] \arrow[dr, "\Sigma r_2"] \arrow[dd, "\widetilde{R}"] & \\
\Sigma X \arrow[dd, "h"'] & & \Sigma Y \arrow[dd, "l"] \\
 & R \arrow[dl, "r_1"'] \arrow[dr, "r_2"] & \\
X & & Y
\end{tikzcd}$$

This definition generalizes to algebras of (arbitrary) monads T:

$$\begin{tikzcd}
 & TR \arrow[dl, "Tr_1"'] \arrow[dr, "Tr_2"] \arrow[dd, "\widetilde{R}"] & \\
TX \arrow[dd, "h"'] & & TY \arrow[dd, "l"] \\
 & R \arrow[dl, "r_1"'] \arrow[dr, "r_2"] & \\
X & & Y
\end{tikzcd}$$

In particular, if R is a Σ-congruence, then its inductive extension is a congruence of the monad T freely generated by Σ. This amounts to the well-known fact that if R is a (Σ-) congruence then, for every context C[-], if $x \; R \; y$ then $C[x] \; R \; C[y]$.

Notice that for coalgebras one speaks of relations *lifting* to bisimulations while for algebras one speaks of relations *being* congruences. The point is that, while there are many ways of lifting a relation to a bisimulation, it is often the case that there exists a unique way of lifting a relation to a congruence. This is certainly true with pullback relations:

Pullbacks uniquely lift to T-congruences. The lifting $\widetilde{R} : TR \to R$ in

$$\begin{tikzcd}
 & TR \arrow[rrr, dashed, "\widetilde{R}"] \arrow[dl, "Tr_1"'] \arrow[dr, "Tr_2"] & & & R \arrow[dl, "r_1"'] \arrow[dr, "r_2"] \arrow[dd, phantom, "\lrcorner" very near start, rotate=-45] & \\
TX \arrow[rrr, bend left, "h"] \arrow[dr, "Tf"'] & & TY \arrow[rrr, bend right, "l"'] \arrow[dl, "Tg"] & X \arrow[dr, "f"'] & & Y \arrow[dl, "g"] \\
 & TZ \arrow[rrr, "j"'] & & & Z &
\end{tikzcd}$$

is given by the unique mediating arrow from the cone $\langle h \circ TR_1, k \circ TR_2 \rangle$ to the pullback R of f and g. The universality of R can be used to prove that the function $\widetilde{R} : R \to BR$ is a T-algebra structure.

One can check that the above implies that $\langle R, \widetilde{R} \rangle$ is the pullback of f and g in the T-algebras:

$$\begin{tikzcd} & \langle R, \widetilde{R} \rangle \arrow[dl, "r_1"'] \arrow[dr, "r_2"] \arrow[dd, phantom, "\lrcorner" rotate=-45, very near start] & \\ \langle X, h \rangle \arrow[dr, "f"'] & & \langle Y, l \rangle \arrow[dl, "g"] \\ & \langle Z, j \rangle & \end{tikzcd}$$

The fact that pullbacks of functions between carriers of algebras lift uniquely to pullbacks (of the same functions but) in the category of algebras amounts to say that the forgetful functor $U^T : \mathbf{C}^T \to \mathbf{C}$ **creates pullbacks**. In turn, this is a consequence of the more general fact (see, eg, §VI.2 of [Mac71]) that

The forgetful functor $U^T : \mathbf{C}^T \to \mathbf{C}$ creates limits.

In other words, a category of algebras has the same limits as its base category. Colimits are more difficult. Dually, a category of coalgebras has the same colimits as its base category, ie:

The forgetful functor $U_B : \mathbf{C}_B \to \mathbf{C}$ creates colimits.

Instead, in general, the limits (eg, products and pullbacks) of coalgebras are difficult. This explains why there is no systematic way of lifting a pullback relation to a bisimulation, and extra assumptions are needed like the preservation of weak pullbacks.

Notes. Preliminary material presented in this section has appeared in [RT93, RT94]. Bisimulations along arrows appear here for the first time.

The notion of an ordinary bisimulation stems from the work of Park [Par81] and Milner [Mil80] on concurrency. Coalgebraic bisimulations were introduced in [AM89], while their dual algebraic congruences already appear in [Man76, page 167]. An order-enriched form of coalgebraic bisimulations is studied by Marcelo Fiore in [Fio93] (improving a previous definition from [RT93]); Fiore's notion cuts down, for a particular functor, to the notion of an *applicative bisimulation* from [Abr90].

For a categorical definition of relations see, eg, [FS90]. When dealing with categories other than **Set** like, eg, the category **pCpo** as in [Fio93], one might want to use a more subtle definition of relations, considering only a class of *admissible* monic spans, closed under pullbacks.

A drawback of the present definition of coalgebraic bisimulations is that it requires that the relations live in the same category as the coalgebras. In **Set** this is not a problem, but when one is working with more structured objects it might be too strong a requirement. For instance, in categories of complete partial orders one has to consider *chain-closed* relations.

Andy Pitts [Pit94a, Pit93, Pit94c] has introduced a different notion of generalized bisimulations for the functor types most commonly used in semantics which overcomes this problem and, moreover, it is 'compositional': if two composable relations are bisimulations (in the sense of Pitts) wrt two different functors F and G, then their composition is a bisimulation wrt the composite functor FG, which is not the case for coalgebraic bisimulations. These two properties really make the 'pulling back' of coinduction to (generalized) bisimulation a useful method for reasoning about coinductively defined objects. (Notice, however, that the actual construction of bisimulation relations can be quite involved, hence it would be important to generalize to functorial operational semantics the existing methods for constructing ordinary bisimulation like those treated in [San95].)

Pitts' notion is implicitly based on lifting the functors to a category of relations. This idea is formalized by Claudio Hermida and Bart Jacobs [Her93, HJ95a, HJ95b, Jac95] by means of the categorical notion of a 'fibration': a category $\mathbf{R}$ of relations over a given category $\mathbf{C}$ is a certain fibration on $\mathbf{C}$; functors F on $\mathbf{C}$ defined by universal properties lift to functors $\widetilde{F}$ on $\mathbf{R}$; a bisimulation wrt to F is then a $\widetilde{F}$-coalgebra in $\mathbf{R}$. Notice that an object of $\mathbf{R}$ does *not* need to be an object of $\mathbf{C}$ as well.

An alternative categorical approach to generalized bisimulations is pursued in [JNW93]; its relationship with the above approaches is still to be investigated.

13 The Observational Comonad for Bisimulation

The behaviour $BX = \check{\mathcal{P}}(1+\mathsf{Act}\times X)$ is not an $\boldsymbol{\omega}^{\mathsf{op}}$-continuous endofunctor, because the power-set functor $\check{\mathcal{P}}$ is not, hence its final coalgebra cannot be obtained as the limit of the usual $\boldsymbol{\omega}^{\mathsf{op}}$-chain. This section illustrates two alternative methods for establishing the existence of the final coalgebra of the finite power-set functor. The first method, due to Peter Aczel, amounts to quotienting a *weakly final* coalgebra by its greatest bisimulation.

The second method is due to Michael Barr. It amounts to finding a 'generating set' for the coalgebras of the finite power-set functor, that is, a (small) set $\{\langle X_i, k_i\rangle\}_I$ of coalgebras such that every coalgebra is a quotient of a coproduct of $\langle X_i, k_i\rangle$'s. By the *Special Adjoint Functor Theorem (SAFT)*, the final coalgebra is then the greatest quotient of the coproduct of all the $\langle X_i, k_i\rangle$'s.

More generally, SAFT ensures the existence of a right adjoint for the forgetful functor mapping coalgebras to their carriers. This right adjoint maps a set to its cofree coalgebra and the whole adjunction defines the cofree comonad for the finite power-set functor. The same can be done with the composite behaviour functor $BX = \check{\mathcal{P}}(1+\mathsf{Act}\times X)$ thus obtaining the observational comonad D for bisimulation.

For simplicity, let us consider the finite power-set functor

$$\mathcal{P}_{\mathrm{fi}} : \mathbf{Set} \to \mathbf{Set} \qquad\qquad X \mapsto \{X' \subseteq X \mid X' \text{ finite}\}$$

instead of its 'relevant' part only, the functor $\check{\mathcal{P}}$ which does not produce the empty set. The coalgebras of the finite power-set functor $\mathcal{P}_{\mathrm{fi}}$ are in a one-to-one correspondence with the finitely branching, directed graphs. Indeed, a coalgebra structure $k : X \to \mathcal{P}_{\mathrm{fi}}X$ defines a graph with $x \in X$ as nodes and with arcs $x \to x'$ for every $x' \in k(x)$. That is, the children of x in the graph are the elements of the image of x under k. Conversely, every finitely branching and directed graph defines a $\mathcal{P}_{\mathrm{fi}}$-coalgebra:

$$x \to x' \iff x' \in k(x)$$

Next, if the final coalgebra $\widehat{\mathcal{P}_{\mathrm{fi}}} \cong \mathcal{P}_{\mathrm{fi}}\widehat{\mathcal{P}_{\mathrm{fi}}}$ of the finite power-set functor exists, then every coalgebra structure $k : X \to \mathcal{P}_{\mathrm{fi}}(X)$ can be coinductively extended to a function $k^{@} : X \to \widehat{\mathcal{P}_{\mathrm{fi}}}$ such that

$$k^{@}(x) = \{k^{@}(x_i) \mid x_i \in k(x)\}$$

Moreover, this coinductive extension is unique:

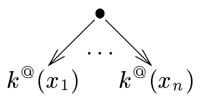
$$\begin{tikzcd}
X \arrow[r, dashed, "k^{@}"] \arrow[d, "k"'] & \widehat{\mathcal{P}_{\mathrm{fi}}} \arrow[d, "\cong"] \\
\mathcal{P}_{\mathrm{fi}}X \arrow[r, "\mathcal{P}_{\mathrm{fi}}(k^{@})"'] & \mathcal{P}_{\mathrm{fi}}\widehat{\mathcal{P}_{\mathrm{fi}}}
\end{tikzcd}$$

The equation

$$k^{@}(x) = \{k^{@}(x_i) \mid x_i \in k(x)\}$$

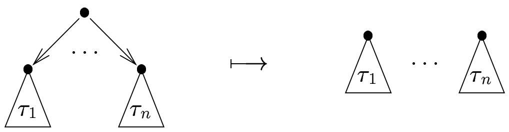
can be seen as the recursive definition of a tree:

$k^{@}(x) =$



(for $k(x) = \{x_1, \ldots, x_n\}$)

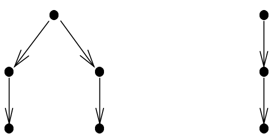
This is a rooted tree, finitely branching, and possibly of infinite depth. Neither nodes nor arcs are labelled. The set $\mathcal{T}$ of these rooted finitely branching trees can be seen as (the carrier of) a coalgebra of the finite power-set functor: every tree $\tau \in \mathcal{T}$ is mapped to the (finite) set $\{\tau_1, \ldots, \tau_n\}$ of children of its root:



This coalgebra is not a final but a *weakly final* coalgebra, that is, it is a coalgebra which ensures the existence but not the uniqueness of coinductive extensions. For example, the coalgebra structure $k : X = \{x, x_1, x_2, x'_1, x'_2\} \to \mathcal{P}_{\mathrm{fi}}(X)$

$$k(x) = \{x_1, x_2\} \quad k(x_1) = \{x'_1\} \quad k(x_2) = \{x'_2\} \quad k(x'_1) = 0 = k(x'_2)$$

can be extended to both the following trees.



Proposition. The final coalgebra of the finite power-set functor is the set of rooted finitely branching trees quotiented by the corresponding (greatest) coalgebraic bisimulation.

More generally, the quotient modulo bisimulation of any weakly final $\mathcal{P}_{\mathrm{fi}}$-coalgebra yields the final $\mathcal{P}_{\mathrm{fi}}$-coalgebra.

Recall, from the previous section, that a relation R between the carriers X and Y of two coalgebras $\langle X, k\rangle$ and $\langle Y, \ell\rangle$ lifts to a coalgebraic bisimulation if there exists a coalgebra structure $\widetilde{R}$ on the relation making its legs coalgebra arrows:

$$\begin{tikzcd}
& R \arrow[dl, "r_1"'] \arrow[dr, "r_2"] \arrow[dd, "\widetilde{R}"] & \\
X \arrow[dd, "k"'] & & Y \arrow[dd, "\ell"] \\
& \mathcal{P}_{\mathrm{fi}} R \arrow[dl, "\mathcal{P}_{\mathrm{fi}}(r_1)"'] \arrow[dr, "\mathcal{P}_{\mathrm{fi}}(r_2)"] & \\
\mathcal{P}_{\mathrm{fi}} X & & \mathcal{P}_{\mathrm{fi}} Y
\end{tikzcd}$$

That is, for all x in X and y in Y such that xRy,

- if $x \to_k x'$ then $y \to_\ell y'$ for some y' such that $x' R\, y'$
- and, conversely, if $y \to_\ell y'$ then $x \to_k x'$ for some x' such that $x' R\, y'$.

(Here the notation $x \to_k x'$ stands for 'there is an arc from x to x' in the graph corresponding to the coalgebra $\langle X, k\rangle$'.)

As shown in the previous section, the finite power-set functor preserves weak pullbacks, hence pullbacks lift to $\mathcal{P}_{\mathrm{fi}}$-bisimulations. As a consequence, for every $\mathcal{P}_{\mathrm{fi}}$-coalgebra $\langle X, k\rangle$, the greatest relation on X lifting to a $\mathcal{P}_{\mathrm{fi}}$-bisimulation exists if the final $\mathcal{P}_{\mathrm{fi}}$-coalgebra exists. The argument is *not* circular because, later in this section, the existence of the final $\mathcal{P}_{\mathrm{fi}}$-coalgebra is proved by means of SAFT and *without* using bisimulations.

Next, consider the quotient of a $\mathcal{P}_{\mathrm{fi}}$-coalgebra $\langle X, k\rangle$ modulo its greatest bisimulation R_k. Categorically, this amounts to taking the *coequalizer* $q : X \to X/R_k$ of the two legs $r_1, r_2 : R_k \to X$ of the relation R_k:

$$\begin{tikzcd}
R_k \arrow[r, shift left, "r_1"] \arrow[r, shift right, "r_2"'] \arrow[d, dotted] & X \arrow[r, "q"] \arrow[d, "k"] & X/R_k \arrow[d, dashed] \\
\mathcal{P}_{\mathrm{fi}}(R_k) \arrow[r, shift left, "\mathcal{P}_{\mathrm{fi}}(r_1)"] \arrow[r, shift right, "\mathcal{P}_{\mathrm{fi}}(r_2)"'] & \mathcal{P}_{\mathrm{fi}} X \arrow[r, "\mathcal{P}_{\mathrm{fi}}(q)"'] & \mathcal{P}_{\mathrm{fi}}(X/R_k)
\end{tikzcd}$$

Notice this lifts to a coequalizer in the category of coalgebras. The coalgebra structure for X/R_k is given by the universal property of the coequalizer. Indeed, since the legs of the relation R_k lift to coalgebra arrows, the composite function $\mathcal{P}_{\mathrm{fi}}(q) \circ k : X \to \mathcal{P}_{\mathrm{fi}}(X/R_k)$ equates the two legs of the relation R_k. The corresponding unique mediating function from X/R_k to $\mathcal{P}_{\mathrm{fi}}(X/R_k)$ is the desired structure. Write $\langle X, k\rangle/R_k$ for this quotient coalgebra.

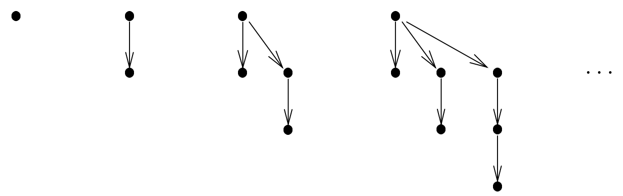
> **Lemma.** From every coalgebra there is *at most one arrow* to the quotient coalgebra $\langle X, k\rangle/R_k$.
>
> Indeed, consider two coalgebra arrows $f, g : \langle Y, \ell\rangle \to \langle X, k\rangle/R_k$. Since, as shown in the previous section, the equality relation always lifts to a coalgebraic bisimulation
>
> $$\begin{tikzcd}[column sep=small] & \mathrm{EQ}(Y) \arrow[dl, "e_1"'] \arrow[dr, "e_2"] \arrow[d, phantom, "\sim" very near start] & \\ \langle Y, \ell\rangle & & \langle Y, \ell\rangle \end{tikzcd}$$
>
> one has that the equality on Y with as legs the composites $f \circ e_1, g \circ e_2 : \mathrm{EQ}(Y) \to X/R_k$ lifts to a bisimulation on the quotient coalgebra $\langle X, k\rangle/R_k$:
>
> $$\begin{tikzcd}[column sep=small] & & \mathrm{EQ}(Y) \arrow[dl, "e_1"'] \arrow[dr, "e_2"] \arrow[d, phantom, "\sim" very near start] & & \\ & \langle Y, \ell\rangle \arrow[dl, "f"'] & & \langle Y, \ell\rangle \arrow[dr, "g"] & \\ \langle X, k\rangle/R_k & & & & \langle X, k\rangle/R_k \end{tikzcd}$$
>
> Therefore, for every $y \in Y$, $f(y)$ is bisimilar to $g(y)$. Since, by construction, the quotient $\langle X, k\rangle/R_k$ is *strongly extensional*, that is, bisimulation is the equality, one has that $f(y)$ is equal to $g(y)$ for every $y \in Y$, hence $f = g$ and the lemma is proved. (Cf [Acz88, Theorem 2.19].)

Therefore, the quotient modulo bisimulation of a weakly final $\mathcal{P}_{\mathrm{fi}}$-coalgebra is necessarily final: the existence of an arrow from every coalgebra is guaranteed by being the quotient of a weakly final coalgebra, the uniqueness is guaranteed by the above property of quotients modulo bisimulation. In particular, the weakly final coalgebra of rooted finitely branching trees can be thus quotiented by bisimulation to yield the final coalgebra of the finite power-set functor. This concludes the proof of the above proposition.

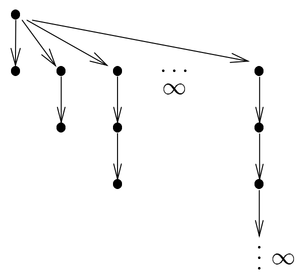
Notice that the finite power-set functor is *not* $\boldsymbol{\omega}^{\mathsf{op}}$-continuous, that is, the limit of the following chain is not a fixed point for the finite power-set functor $\mathcal{P}_{\mathrm{fi}}$. (Cf Section 5.)

$$1 \xleftarrow{\;1\;} \mathcal{P}_{\mathrm{fi}}1 \xleftarrow{\;\mathcal{P}_{\mathrm{fi}}1\;} {\mathcal{P}_{\mathrm{fi}}}^2 1 \xleftarrow{\;{\mathcal{P}_{\mathrm{fi}}}^2 1\;} \cdots$$

Indeed: Each object ${\mathcal{P}_{\mathrm{fi}}}^n 1$ of the chain is the set of finitely branching trees with depth at most n, quotiented by bisimulation. Correspondingly, the following sequence of trees belong to the above chain.



The problem is then that the limit has to contain the following tree with *infinitely* many branches,



while the final coalgebra, as shown above, contains only *finitely* branching trees.

* * *

The coequalizer $q : \langle X, k\rangle \to \langle X, k\rangle / R_k$ of the two legs of the greatest bisimulation on a coalgebra $\langle X, k\rangle$ is the 'greatest quotient' of $\langle X, k\rangle$. Formally, a **quotient** is an equivalence class of epis, just like a relation is an equivalence class of monic spans (see previous section). Coequalizers are always epi, ie they are 'right-cancellable': given a coequalizer $q : X \to Y$ and two parallel arrows $f, g : Y \to Z$, if $f \circ q = g \circ q$ then $f = g$. (This is immediate because of the universal property of coequalizers.)

Given two epis $f : X \twoheadrightarrow Z$ and $g : X \twoheadrightarrow Y$ with a common domain X put

$$f \leq g \iff f = f' \circ g$$

for some (necessarily unique and epi) arrow $f' : Y \twoheadrightarrow Z$. The two epis are equivalent (hence represent the same quotient) if the converse also holds, that is, if also

$$g \leq f$$

It is wrt this partial order on quotients that one can prove that $q : \langle X, k\rangle \twoheadrightarrow \langle X, k\rangle / R_k$ is (a representative of) the *greatest quotient* of the coalgebra $\langle X, k\rangle$. Indeed, since the pullback of $\check{\mathcal{P}}$-coalgebra arrows lifts to coalgebraic bisimulations (see previous section), the pullback $K(f)$ of (two copies of) every other quotient f lifts to a bisimulation and hence it is smaller than the relation R_k:

$$\begin{tikzcd}
 & & \langle X,k\rangle \arrow[dr, two heads, "q"'] \arrow[drr, "f"] & & \\
K(f) \arrow[urr] \arrow[drr] \arrow[r, phantom, "\leq"] & R_k \arrow[ur] \arrow[dr] & & \langle X,k\rangle/R_k \arrow[r, dashed] & \langle Y,\ell\rangle \\
 & & \langle X,k\rangle \arrow[ur, two heads, "q"] \arrow[urr, "f"'] & &
\end{tikzcd}$$

Therefore $f \leq q$, for every quotient $f : \langle X, k\rangle \twoheadrightarrow \langle Y, \ell\rangle$ of $\langle X, k\rangle$.

The greatest quotient of an object can be seen as the least upper bound of all quotients of that object. Dually, and more generally, also the greatest lower bound, ie the **intersection**, of all quotients of a suitable object can be used for finding the final object of a category.

In general, the intersection of a set of quotients of an object is their *pushout*, if this exists, because “pushouts of epis are epi” and “epis are closed under composition”. (See, eg, [Mac71, §V.7].) For instance, the following diagram shows that the pushout of two epis f_1 and f_2 is their least upper bound.

$$\begin{tikzcd}
& \cdot \arrow[dr, two heads, "f_1'"] \arrow[drr, two heads, bend left] & & \\
\cdot \arrow[ur, two heads, "f_1"] \arrow[dr, two heads, "f_2"'] & & \cdot \arrow[r, dashed] & \cdot \\
& \cdot \arrow[ur, two heads, "f_2'"'] \arrow[urr, two heads, bend right] & &
\end{tikzcd}$$

Indeed, by definition of pushout, the composite $f_1' \circ f_1$ is equal to the composite $f_2' \circ f_2$, it is an epi, and it is greater that both f_1 and f_2. Moreover it is smaller than every other upper bound for f_1 and f_2, because of the universal property of pushouts.

Even if pushouts exist, the intersection of *all* quotients of an object in a category might fail to exist: one needs that the category be ‘**co-well-powered**’, that is, the collection of all quotients of a given object should be a (small) set, so that its pushout can be taken. Now, by a standard cardinality argument, for every coalgebra $\langle X, k\rangle$ of an *arbitrary* endofunctor B on **Set**, one can form the set of its quotients, hence

$\mathbf{Set}_B$ is co-well-powered.

for all $B : \mathbf{Set} \to \mathbf{Set}$ (and thus the finite power-set functor in particular).

As for pushouts, these are colimits and coalgebras inherit all colimits from their underlying category, since, as mentioned in the previous section,

The forgetful functor $U_B : \mathbf{C}_B \to \mathbf{C}$ creates colimits.

Therefore, since **Set** is cocomplete (ie it has all colimits), the category of coalgebras of an endofunctor B on **Set** is also cocomplete:

$\mathbf{Set}_B$ is cocomplete.

Now, a more general way of finding a final object in a cocomplete and co-well-powered category is by finding a (small) set of objects $\{X_i\}_I$ such that every object in the category is the quotient of a coproduct of X_i’s. (A set $\{X_i\}_I$ with this property is called a **generating set** for the (cocomplete) category.)

From a generating set to the final object. In a cocomplete category with a generating set $\{X_i\}_I$, if the intersection of all quotients $q : \coprod_I X_i \twoheadrightarrow Q$ of the coproduct $\coprod_I X_i$ exists, then Q is the final object of the category.

Let us first check uniqueness, that is, that from every object Y there exists at most one arrow to Q. Indeed, if there were two distinct arrows one can coequalize them. Let

$$q' : Q \twoheadrightarrow E$$

be this coequalizer. Since coequalizers are epi, the composite

$$\coprod_I X_i \overset{q}{\twoheadrightarrow} Q \overset{q'}{\twoheadrightarrow} E$$

would then be greater than q, which is a contradiction.

For the uniqueness part one only uses the fact that Q is the greatest quotient of an object. It is for the existence part that the generating set $\{X_i\}_I$ is used. Indeed, by definition of generating sets, every object Y is (the codomain of) a quotient

$$q' : \coprod_J X_j \twoheadrightarrow Y$$

of a coproduct $\coprod_J X_j$ of elements of $\{X_i\}_I$. Since every X_j is an element of $\{X_i\}_I$, there is, by the universal property of the coproduct, a function from $\coprod_J X_j$ to $\coprod_I X_i$, mapping each X_j to the corresponding X_i. (Notice that this function is not an embedding, because there might be more copies in $\coprod_J X_j$ of the same X_i.) One can then take the pushout

$$\begin{tikzcd}
\coprod_J X_j \arrow[r, two heads, "q'"] \arrow[d] & Y \arrow[d] \\
\coprod_I X_i \arrow[r, two heads, "q''"'] & Q' \arrow[ul, phantom, "\ulcorner", very near start]
\end{tikzcd}$$

of this function and the quotient $q' : \coprod_J X_j \twoheadrightarrow Y$. Since pushouts of epis are epi, the arrow $q'' : \coprod_I X_i \twoheadrightarrow Q'$ is epi, hence there exists an arrow from Q' to the codomain Q of the intersection of all quotients of $\coprod_I X_i$. One can then form a composite $Y \to Q' \twoheadrightarrow Q$, which proves the existence of an arrow from an arbitrary Y to Q. This concludes the proof. (Cf, eg, [Mac71, Theorem V.8.1].)

A generating set for the $\mathcal{P}_{\text{fi}}$-coalgebras.

The set

$$\mathcal{G} = \{\langle U, k\rangle \mid k : U \to \mathcal{P}_{\text{fi}}U \text{ and } U \subseteq \omega\}$$

of $\mathcal{P}_{\text{fi}}$-coalgebras with ordinals less than or equal to ω as carriers is a (small) generating set for the category of $\mathcal{P}_{\text{fi}}$-coalgebras.

Firstly notice that, if a set U has cardinality $\leq \omega$, then also its set of finite subsets $\mathcal{P}_{\text{fi}}U$ has cardinality $\leq \omega$. Therefore the above collection $\mathcal{G}$ really is a set (and not a proper class).

Next, let us show that the set $\mathcal{G}$ is a generating set for the $\mathcal{P}_{\text{fi}}$-coalgebras. For this, notice that, in a category which (like the one of $\mathcal{P}_{\text{fi}}$-coalgebras) is cocomplete, a set $\mathcal{G}$ of objects is generating if and only if for every two parallel arrows f_1 and f_2 such that $f_1 \neq f_2$ there exists an arrow g from an object in $\mathcal{G}$ such that

$$f_1 \circ g \neq f_2 \circ g$$

This definition is easier to check (and it makes sense also in categories which are not cocomplete). For instance, in **Set**, two functions $f_1, f_2 : X \to Y$ are distinct if and only if there exists an $x \in X$ such that $f_1(x) \neq f_2(x)$, therefore, the singleton set 1 is a generator.

Two coalgebra arrows

$$f_1, f_2 : \langle X, k\rangle \to \langle Y, \ell\rangle$$

are functions, thus also they have a distinct value at some $x \in X$. However, the coalgebras with carrier 1 do not suffice to form a generating set for the $\mathcal{P}_{\text{fi}}$-coalgebras, because the discriminating x will be mapped by $k : X \to \mathcal{P}_{\text{fi}}X$ to a set $\{x_1, \ldots, x_m\}$ in which the x_i's are, in general, different from x.

The idea is that, since every $\mathcal{P}_{\text{fi}}$-coalgebra structure $k : X \to \mathcal{P}_{\text{fi}}X$ maps elements $x \in X$ to *finite* subsets of X, one can start from x and recursively apply ($\mathcal{P}_{\text{fi}}$ of) k to it. Thus at the first step one has $\{x\}$ only, at the second $\{x\} \cup \{x_1, \ldots, x_m\}$, and so on, until a subset $U \subseteq X$ is found such that $x \in U$ and k restricted to U is a $\mathcal{P}_{\text{fi}}$-coalgebra structure on U itself. Because at each step only finitely many x_i's are added, the set U cannot be larger than ω. Therefore, the coalgebra $\langle U, k\rangle$ is isomorphic to a coalgebra in $\mathcal{G}$.

Formally, given a $\mathcal{P}_{\text{fi}}$-coalgebra $\langle X, k\rangle$ and an $x \in X$, let

$$U = \bigcup_{n \in \omega} U_n$$

where

$$U_0 = \{x\} \qquad U_{n+1} = U_n \cup \mathcal{P}_{\text{fi}}(k)(U_n)$$

By definition U is a subset of X of cardinality at most ω. It remains thus only to show that, if $x_i \in U$, then $k(x_i) \subseteq \mathcal{P}_{\text{fi}}U$. But this follows from the fact that $x_i \in U$ implies there exists an n such that $x_i \in U_n$, hence $k(x_i) \subseteq U_{n+1} \subseteq U$.

This concludes the proof that the above $\mathcal{G}$ is a generating set for the $\mathcal{P}_{\mathrm{fi}}$-coalgebras. (Cf [Bar93, Proposition 1.3].)

> **Corollary 1.** The final coalgebra of the finite power-set functor is the intersection of all quotients of the coproduct $\coprod_{\mathcal{G}}\langle U, k\rangle$.

Notice that, by essentially the same argument, one can show that the set

$$\mathcal{G}_B = \{\langle U, k\rangle \mid k : U \to BU \text{ and } U \subseteq \omega\}$$

is a generating set for the behaviour $BX = \check{\mathcal{P}}(1 + \mathsf{Act} \times X)$. Thus:

> **Corollary 2.** The final coalgebra of the behaviour $BX = \check{\mathcal{P}}(1+\mathsf{Act}\times X)$ is the intersection of all quotients of the coproduct $\coprod_{\mathcal{G}_B}\langle U, k\rangle$.

Next, a category is **locally small** if the collection of arrows between every two objects forms a (small) set. For instance, **Set** is locally small. Now, the above proof of the existence of a final coalgebra by means of a generating set is an application of the following general theorem.

> **The Special Adjoint Functor Theorem (SAFT).** If **D** is cocomplete, co-well-powered, and with a (small) generating set, and if **C** is locally small, then every cocontinuous functor $F : \mathbf{D} \to \mathbf{C}$ has a right adjoint.
>
> (For a proof see, eg, [FS90] or [Mac71].)

Indeed, by instantiating the above theorem to the unique functor

$$\mathbf{Set}_B \to \mathbf{1}$$

from the coalgebras of the endofunctor $BX = \check{\mathcal{P}}(1 + \mathsf{Act} \times X)$ (or $BX = \mathcal{P}_{\mathrm{fi}}X$) to the (final) category $\mathbf{1}$ with one object and one arrow (the identity), one obtains the existence of the final B-coalgebra.

A functor which creates colimits also preserves them, that is, it is cocontinuous, hence, for every endofunctor B on **Set**,

The forgetful functor $U_B : \mathbf{Set}_B \to \mathbf{Set}$ is cocontinuous.

Therefore, the Special Adjoint Functor Theorem also shows that, for every endofunctor (like the behaviour $BX = \check{\mathcal{P}}(1 + \mathsf{Act} \times X)$ or the finite power-set functor) whose coalgebras have a generating set,

The forgetful functor $U_B : \mathbf{Set}_B \to \mathbf{Set}$ has a right adjoint.

This adjunction gives rise to a cofree comonad $D{=}{<}D, \varepsilon, \delta{>}$:

Let $G_B : \mathbf{Set} \to \mathbf{Set}_B$ be the right adjoint of the above forgetful functor $U_B : \mathbf{Set}_B \to \mathbf{Set}$. By definition of right adjoint, given a set X, a coalgebra $\langle Y, \ell \rangle$, and a function $f : Y \to X$ there exists a unique coalgebra arrow $f^\flat : \langle Y, \ell \rangle \to G_B X$ such that $f = \varepsilon_X \circ U_B f^\flat$, where $\varepsilon : U_B G_B \Rightarrow I$ is the counit of the adjunction. (Cf Section 8.)

Write DX for the carrier of the coalgebra $G_B X$ and $\lambda_X : DX \to BDX$ for its structure. Then $U_B f^\flat : Y \to DX$ is the unique $(X \times B)$-coalgebra arrow from the coalgebra (structure) $< f, \ell > : Y \to X \times BY$ to the coalgebra (structure) $< \varepsilon_X, \lambda_X > : DX \to X \times BDX$, which means that the latter is the (structure of the) final $(X \times B)$-coalgebra.

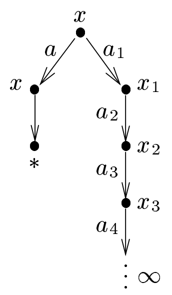
Because final coalgebras are isomorphisms, this implies that $DX \cong X \times BDX$, the counit at X is the first projection $\mathsf{fst}_X : DX \to X$ and the structure $\lambda_X : DX \to BDX$ is the second projection.

As shown in Section 7, the operation $X \mapsto DX$ extends to an endofunctor $D : \mathbf{Set} \to \mathbf{Set}$, and the counit $\varepsilon : D \Rightarrow I$ and the coinductive extension $\delta : D \Rightarrow D^2$ of the second projection along the identity are comonad operations for it.

In particular, the cofree comonad corresponding to the behaviour $BX = \check{\mathcal{P}}(1 + \mathsf{Act} \times X)$ is the **observational comonad for bisimulation**.

Concretely, the value of the observational comonad for bisimulation at a set X can be obtained by means of a quotient construction in terms of trees and bisimulations as follows.

Let $\mathcal{T}_X$ be the set of trees which are coinductively generated by finitely branching transition systems. That is, the set $\mathcal{T}_X$ is the set of trees which are rooted, finitely branching, with nodes labelled by $x \in X$, arcs labelled by $a \in \mathsf{Act}$, and whose leaves are labelled by $*$ (and the arcs to leaves are then unlabelled). These trees are possibly of infinite depth. For instance, the following is a tree in $\mathcal{T}_X$ if $x, x_1, x_2, \ldots$ are in X.



This tree has root labelled by x, one leaf, and one infinite branch.

Just like the set $\mathcal{T}$ given at the beginning of this section can be seen as a coalgebra of the finite power-set functor, this set $\mathcal{T}_X$ can be seen as an $(X \times B)$-coalgebra. The function $\mathcal{T}_X \to X$ is the operation which, given a tree, returns the label $x \in X$ of its root.

One can check that, with this structure, the set $\mathcal{T}_X$ is a weakly final $(X\times B)$-coalgebra. The final $(X\times B)$-coalgebra (with carrier DX!) can be then obtained by taking the quotient modulo the greatest $(X\times B)$-bisimulation. By instantiating the coalgebraic notion of bisimulation to the functor $(X\times B)$, one obtains relations R on $\mathcal{T}_X$ such that two trees $\tau_1, \tau_2 \in \mathcal{T}_X$ by R (ie $\tau_1 R \tau_2$) iff the following four conditions are satisfied.

1. The label $x \in X$ of the root of τ_1 is the same as the label of the root of τ_2;
2. $\tau_1 \longrightarrow *$ if and only if $\tau_2 \longrightarrow *$;
3. if $\tau_1 \xrightarrow{a} \tau_1'$ then $\tau_2 \xrightarrow{a} \tau_2'$ for some τ_2' such that $\tau_1' R\, \tau_2'$,
4. and, conversely, if $\tau_2 \xrightarrow{a} \tau_2'$ then $\tau_1 \xrightarrow{a} \tau_1'$ for some τ_1' such that $\tau_1' R\, \tau_2'$.

The first clause is the one corresponding to the extra information given by the states $x \in X$. By putting $X = 1$ one recovers the ordinary notion of bisimulation between trees (with unlabelled nodes).

Notes. The idea of defining semantics by taking quotients of transition systems (ie coalgebras) by greatest (ordinary) bisimulations dates back at least to [Mil80]. The Final Coalgebra Theorem in [AM89] (based on a previous result in [Acz88]) generalizes that idea: it shows that final coalgebras of endofunctors can be obtained by quotienting weakly final coalgebras by the greatest (coalgebraic) *congruence*. This is stated for 'set-based' endofunctors on the category **SET** of classes (ie large sets – cf Part V): an endofunctor is *set-based* if its value at a class X is determined by its value at the (small) subsets of X [Acz88, Definition 6.1]. An example is the endofunctor $\mathcal{P}_S : \mathbf{SET} \to \mathbf{SET}$ mapping a class to the class of its (small) subsets, which is used in Part V. If an endofunctor preserves weak pullbacks then the notion of a (coalgebraic) conguence cuts down to that of a (coalgebraic) bisimulation [AM89, Proposition 6.2].

In [Bar93], the final coalgebra theorem of [Acz88] is reformulated in **Set** (thus without use of classes) by replacing the set-based condition by that of 'accessibility', modelling with inaccessible cardinals the size distinction between sets and classes. In particular, the endofunctors $\mathcal{P}_{\mathrm{fi}}$ and $BX = \check{\mathcal{P}}(1 + \mathsf{Act} \times X)$ are accessible and the above 'construction' of the corresponding generating sets is a special case of that in [Bar93, Proposition 1.3].

IV

A Summary

In this section a technical summary of the above results is given. It can be read independently from the other sections by a reader familiar with the categorical notions of *adjunction* and *monad*. After some preliminaries recalling the basic definitions and facts about algebras and coalgebras, the notion of functorial denotational semantics is introduced; as an example, *basic process algebra* [BW90] is defined denotationally. Next, every functorial denotational semantics is shown to induce an operational dual (and vice versa). (The *Basic Property*.) Next, several results are proved (*Operational is Denotational*, Φ*-algebras are* $\Phi^{@}$*-coalgebras*, *Adequacy Theorem*) which illustrate the *adequacy* of the denotational semantics $\Phi^{@}$ coinduced by a functorial operational semantics Φ. Finally, it is proved that the operational semantics induced by *GSOS rules* [BIM88] is always functorial.

Algebras. The category of the algebras of a monad $T{=}{<}T,\eta,\mu{>}$ on a category $\mathbf{C}$ is denoted by $\mathbf{C}^T$. Its objects are the arrows $h : TX \to X$ of $\mathbf{C}$ such that $h \circ \eta_X = \mathsf{id}_X$ and $h \circ \mu_X = h \circ Th$; its arrows $f : (TX \xrightarrow{h} X) \to (TX' \xrightarrow{h'} X')$ are the arrows $f : X \to X'$ in $\mathbf{C}$ such that $f \circ h = h' \circ Tf$.

The evident forgetful functor $U^T : \mathbf{C}^T \to \mathbf{C}$ has a left adjoint $X \mapsto (T^2X \xrightarrow{\mu_X} TX)$. This adjoint situation is here denoted as follows.

$$
\begin{tikzcd}
X \arrow[r, "\eta_X"] \arrow[dr, "f"'] & TX \arrow[d, "f^\sharp"] \\
& Y
\end{tikzcd}
\qquad
\begin{tikzcd}
TX \arrow[d, dashed, "f^\sharp"'] & T^2X \arrow[l, "\mu_X"'] \arrow[d, "Tf^\sharp"] \\
Y & TY \arrow[l, "h"]
\end{tikzcd}
\tag{4}
$$

(In the sequel, $f^\sharp$ does always denote the above left adjunct of f wrt the adjunction. The uniqueness of $f^\sharp$ is exploited here to prove several equalities between arrows.) The monad defined by this adjunction is trivially equal to the original monad T, hence *every monad is defined by its algebras*.

Given a *signature* Σ and a cocomplete category $\mathbf{C}$ with finite products, one can define an endofunctor (with the same name) on $\mathbf{C}$ as the coproduct $\Sigma X = \coprod_\sigma X^{\mathrm{arity}(\sigma)}$ indexed by the operators σ of the signature. Then the Σ-algebras $h : \Sigma X \to X$ form a category $\mathbf{C}^\Sigma$ with as arrows $f : (\Sigma X \xrightarrow{h} X) \to (\Sigma X' \xrightarrow{h'} X')$ the arrows $f : X \to X'$ in $\mathbf{C}$ such that $f \circ h = h' \circ \Sigma f$.

Also the forgetful functor $U^\Sigma : \mathbf{C}^\Sigma \to \mathbf{C}$ has a left adjoint and, moreover, it is *monadic*, ie, if T is the monad arising from this adjunction, then there is an isomorphism of categories $\mathbf{C}^\Sigma \cong \mathbf{C}^T$ making the following diagram commute.

$$\begin{tikzcd} \mathbf{C}^\Sigma \arrow[r, phantom, "\cong"] \arrow[dr, "U^\Sigma"'] & \mathbf{C}^T \arrow[d, "U^T"] \\ & \mathbf{C} \end{tikzcd} \tag{5}$$

For $\mathbf{C} = \mathbf{Set}$, TX is the usual set of terms inductively defined by the operators in Σ and the variables $x \in X$. In particular, T at the empty set 0 is the set of closed terms. In other words, $T0$ is the carrier of the initial Σ-algebra $\Sigma T0 \cong T0$ (*Lambek's lemma*: initial algebras are always isomorphisms [SP82]) and, in general, TX is the carrier of the initial $(X+\Sigma)$-algebra $X+\Sigma TX \cong TX$. The unit $\eta_X = \mathsf{inl}_X : X \to TX$ at X is the formal insertion of the variables $x \in X$ in the terms $t \in TX$ and the multiplication $\mu_X : T^2X \Rightarrow TX$ is the 'inductive extension' of the right injection inr_X. Thus, for instance, $\mu_X(\sigma(\eta_X t_1, \ldots, \eta_X t_n)) = \sigma(t_1, \ldots, t_n)$. To ease the notation, η and μ is often omitted from the terms.

In the sequel, also $\langle \Sigma, E\rangle$-algebras are considered, ie Σ-algebras which satisfy some equations E on the operators derivable from the signature. The forgetful functor from the corresponding category $\mathbf{Set}^{\langle \Sigma, E\rangle}$ has a left adjoint and it is monadic, hence

$$\mathbf{Set}^{\langle \Sigma, E\rangle} \cong \mathbf{Set}^T \tag{6}$$

for the corresponding monad T. For instance, consider semi-lattices with a least element, ie let Σ contain only a binary operator $\vee$ and a constant $\perp$ and let E be the associativity, commutativity, and associativity axioms for $\vee$ and the unit axiom for $\perp$ wrt $\vee$. Then $\mathbf{Set}^{\langle \Sigma, E\rangle}$ is isomorphic to the category of algebras of the monad $<\mathcal{P}_{\mathsf{fi}}, \{\text{-}\}, \bigcup>$, where $\check{\mathcal{P}}X$ is the set of finite subsets of X.

If the operators of Σ are the constructs of a programming language then, an algebra $h : TY \to Y$ of the corresponding *syntactical* monad T is a *denotational* model of the language and it induces an *initial algebra semantics* [GTW78], namely the unique arrow $h^\# : T0 \to Y$ from the initial algebra $\mu_0 : T^2 0 \to T0$ to $h : TY \to Y$:

$$h^\#(\sigma(t_1, \ldots, t_n)) = h(\sigma(h^\# t_1, \ldots, h^\# t_n))$$

Coalgebras. Dually, let $\mathbf{C}_B$ denote the category of coalgebras of an endofunctor B on $\mathbf{C}$, having as objects arrows $k : X \to BX$ in $\mathbf{C}$ and as arrows $f : (X \xrightarrow{k} BX) \to (X' \xrightarrow{k'} BX')$ those arrows $f : X \to X'$ in $\mathbf{C}$ such that $f \circ k' = Bf \circ k$.

Every *finitely branching* labelled transition system [Plo81b]

$$\langle X, \{\xrightarrow{a}\}_{a \in \mathsf{Act}}\rangle$$

can be seen as a coalgebra $k : X \to BX$ of the *behaviour* endofunctor $BX = \mathcal{P}_{\mathsf{fi}}(\mathsf{Act} \times X)$ on $\mathbf{Set}$ [Acz88]:

$$x \xrightarrow{a} x' \iff < a, x' > \in k(x) \tag{7}$$

Notice that although the category of B-coalgebras has the same objects as the standard category of transition systems [WN95], the arrows are different.

The final B-coalgebra $D1 \cong B(D1)$ exists [AM89] and, correspondingly, every *operational* model, ie coalgebra (ie transition system) $[\![\text{-}]\!] : TX \to BTX$ on the syntax, coinduces a *final coalgebra semantics* [RT93], namely the unique arrow $[\![\text{-}]\!]^{@} : TX \to D1$ from $[\![\text{-}]\!]$ to the final coalgebra. Up to the isomorphism $D1 \cong B(D1)$,

$$[\![t]\!]^{@} = \{<a, [\![t']\!]^{@}> \mid <a, t'> \in [\![t]\!]\}$$

The existence of this final coalgebra is a corollary of the fact that the forgetful functor $U_B : \mathbf{Set}_B \to \mathbf{Set}$ has a *right* adjoint [Bar93]. In general, for any endofunctor on a complete category $\mathbf{C}$, if the forgetful functor $U_B : \mathbf{C}_B \to \mathbf{C}$ has a right adjoint then it is *comonadic*, ie the coalgebras of the corresponding comonad $D = < D, \varepsilon, \delta >$ are isomorphic to the B-coalgebras:

$$\mathbf{C}_B \cong \mathbf{C}_D \tag{8}$$

(A coalgebra of the comonad D is an arrow $k : X \to DX$ in $\mathbf{C}$ such that $\varepsilon_X \circ k = \mathsf{id}_X$ and $\delta_X \circ k = Dk \circ k$.) Correspondingly, the forgetful functor $U_D : \mathbf{C}_D \to \mathbf{C}$ has a right adjoint $X \mapsto (DX \xrightarrow{\delta_X} D^2X)$:

$$\begin{tikzcd} & X \arrow[dl, "f"'] \arrow[d, "f^\flat"] \\ X & DY \arrow[l, "\varepsilon_Y"] \end{tikzcd} \qquad \begin{tikzcd} X \arrow[r, "k"] \arrow[d, dashed, "f^\flat"'] & DX \arrow[d, "Df^\flat"] \\ DY \arrow[r, "\delta_Y"'] & D^2Y \end{tikzcd} \tag{9}$$

To every endofunctor B corresponds a notion of B-*bisimulation* [AM89] which, for $BX = \mathcal{P}_{\mathrm{fi}}(\mathsf{Act} \times X)$, specializes to the ordinary bisimulation [Par81]. Final coalgebras are *internally fully-abstract* in the sense that their greatest B-bisimulation (exists and) is an equality relation; moreover, if B (like the above behaviour) preserves weak pullbacks, then the kernel pair of the final coalgebra semantics is the greatest B-bisimulation (on the B-coalgebra under consideration) [RT93]. One can prove that the final coalgebra of the behaviour $BX = \mathcal{P}_{\mathrm{fi}}(\mathsf{Act} \times X)$ is the set of rooted finitely branching trees quotiented by its greatest bisimulation.

In general, for an endofunctor B to qualify as a *behaviour* its corresponding notion of bisimulation should be a significant notion of *observational equivalence*; moreover, it should preserve weak pullbacks, and the forgetful functor $U_B : \mathbf{C}_B \to \mathbf{C}$ should have a right adjoint. The corresponding comonad D is then an *observational comonad*.

Functorial Denotational Semantics

Given an observational comonad $D = < D, \varepsilon, \delta >$ and a syntactical monad $T = < T, \eta, \mu >$, a **functorial denotational semantics** is a comonad Ψ **lifting** the comonad D to the T-algebras:

$$\begin{tikzcd}
\mathbf{C}^T \arrow[r, "\Psi"] \arrow[d, "U^T"'] & \mathbf{C}^T \arrow[d, "U^T"] \\
\mathbf{C} \arrow[r, "D"'] & \mathbf{C}
\end{tikzcd}$$

That is, Ψ is a triple $<\Psi, \widetilde{\varepsilon}, \widetilde{\delta}>$ such that

$$\begin{array}{rcl}
U^T\Psi & = & DU^T : \mathbf{C}^T \to \mathbf{C} \\
U^T\widetilde{\varepsilon} & = & \varepsilon_{U^T} : DU^T \Rightarrow U^T \\
U^T\widetilde{\delta} & = & \delta_{U^T} : DU^T \Rightarrow D^2U^T
\end{array}$$

In other words, $U^T : \mathbf{C}^T \to \mathbf{C}$ is a 'map of monads'.

The second and third equation imply that the counit $\widetilde{\varepsilon}$ and comultiplication $\widetilde{\delta}$ of Ψ are the same as those of $D = < D, \varepsilon, \delta >$, because of the very definition of coalgebra arrows. Therefore:

$$\Psi = < \Psi, \varepsilon, \delta >$$

One can check that the three equations and the fact that the triple $< P, \varepsilon, \delta >$ is a comonad imply that also the triple $\Psi = < \Psi, \varepsilon, \delta >$ is a comonad. Also, the first equation is equivalent to Ψ being an **action** of T on $DU^T : \mathbf{C}^T \to \mathbf{C}$, ie a natural transformation

$$\Psi : TDU^T \Rightarrow DU^T$$

such that, for every T-algebra $h : TX \to X$, $\Psi h : TDX \to DX$ is also a T-algebra. (See, eg, [BW85] for the equivalence between liftings and actions.) Then, the second and third equations are equivalent to the fact that, for every $h : TX \to X$, the following diagram commutes.

$$\begin{tikzcd}
TX \arrow[d, "h"'] & TDX \arrow[l, "T\varepsilon_X"'] \arrow[r, "T\delta_X"] \arrow[d, "\Psi h"'] & TD^2X \arrow[d, "\Psi^2 h"] \\
X & DX \arrow[l, "\varepsilon_X"] \arrow[r, "\delta_X"'] & D^2X
\end{tikzcd}$$

That is,

$$\varepsilon_X \circ \Psi h = h \circ T\varepsilon_X \tag{10}$$

$$\delta_X \circ \Psi h = \Psi^2 h \circ T\delta_X \tag{11}$$

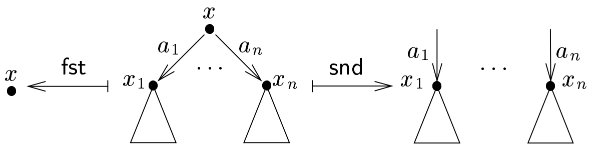
As an example, consider the following functorial denotational semantics for *basic process algebra* [BW90]. The base category $\mathbf{C}$ is $\mathbf{Set}$. The syntactical monad T is the one freely generated by the constructs $\Sigma = \{\mathsf{nil}, a\,.\,, \mathsf{or}\}$, ie

$$t ::= \mathsf{nil} \mid a\,.\,t \mid t\,\mathsf{or}\,t$$

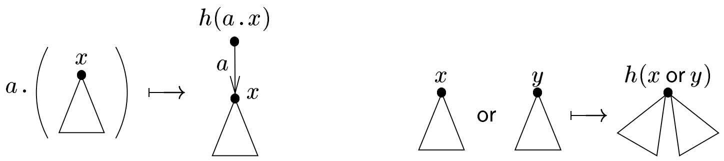
The observational comonad $D = < D, \varepsilon, \delta >$ is cofreely generated by the behaviour $BY = \mathcal{P}_{\text{fi}}(\mathsf{Act} \times Y)$. The set DX is the carrier of the final $(X \times B)$-coalgebra:

$$X \xleftarrow{\varepsilon_X = \mathsf{fst}_X} DX \cong X \times BDX \xrightarrow{\mathsf{snd}_X} BDX$$

and it is a set of (finitely branching) trees whose nodes are labelled by $x \in X$ and whose arcs are labelled by $a \in \mathsf{Act}$. The operation $\varepsilon_X = \mathsf{fst}_X : DX \to X$ gives the label of the root node for each tree in DX and the other operation $\mathsf{snd}_X : DX \to BDX = \mathcal{P}_{\text{fi}}(\mathsf{Act} \times DX)$ gives the remaining part of the tree (and it coinductively extends to give the counit $\delta : D \Rightarrow D^2$ of the comonad D):



Using (5), one can define Ψ as an action of Σ rather than of T. That is, $\Psi : \Sigma P U^{\Sigma} \Rightarrow DU^{\Sigma}$. Then, for every $h : \Sigma X \to X$, define the action of the constant nil as the tree with only one node and label $h(\mathsf{nil})$, and the action of '$a\,.$' and 'or' as follows.



Formally, using the meta-variables p and q to range over the elements of DTX, for every X, Ψ is defined as follows.

$$\begin{array}{lcl}
\mathsf{nil} & \mapsto & < h(\mathsf{nil}), \emptyset > \\
a\,.\,p & \mapsto & < h(a\,.\,(\mathsf{fst}p)), \{< a, p >\} > \\
p\,\mathsf{or}\,q & \mapsto & < h((\mathsf{fst}p)\,\mathsf{or}\,(\mathsf{fst}q)), (\mathsf{snd}p) \cup (\mathsf{snd}q) >
\end{array}$$

Therefore, the Σ-algebra $\Psi h : \Sigma DX \to DX$ is a pair, whose first component is simply the composite function $h \circ \Sigma\mathsf{fst}_X$, ie, up to (5), the equation (10) holds, because $\mathsf{fst}_X = \varepsilon_X$. Also (11) holds, because both $\delta_X \circ \Psi h$ and $\Psi^2 h \circ \Sigma\delta_X$ fit as the (unique!) pair $< \Psi h, B\delta_X \circ \mathsf{snd}_X \circ \Psi h >$. Therefore, $\Psi = < \Psi, \varepsilon, \delta >$ is a functorial denotational semantics for the above signature Σ and behaviour B.

A Dual Lifting: Functorial Operational Semantics

The definition of **functorial operational semantics** is the dual of the one of functorial denotational semantics: it is a monad $\Phi = <\Phi, \eta, \mu>$ lifting the syntactical monad $T = <T, \eta, \mu>$ to the D-coalgebras. That is, a **coaction**

$$\Phi : TU_D \Rightarrow DTU_D$$

of the comonad D on $TU_D : \mathbf{C}_D \to \mathbf{C}$ such that, for every D-coalgebra $k : X \to DX$, the following diagram commutes.

$$\begin{tikzcd} X \arrow[r, "\eta_X"] \arrow[d, "k"'] & TX \arrow[d, "\Phi k"'] & T^2X \arrow[l, "\mu_X"'] \arrow[d, "\Phi^2 k"] \\ DX \arrow[r, "D\eta_X"'] & DTX & DT^2X \arrow[l, "D\mu_X"] \end{tikzcd}$$

The Basic Property. Every functorial denotational semantics Ψ defines a functorial operational semantics whose action $\Psi^\# : TU_D \Rightarrow DTU_D$ is defined by means of the adjunction (4) as follows.

$$\begin{tikzcd} X \arrow[r, "\eta_X"] \arrow[d, "k"'] & TX \arrow[d, "\Psi^\# k"'] \\ DX \arrow[r, "D\eta_X"'] & DTX \end{tikzcd} \qquad \begin{tikzcd} TX \arrow[d, dashed, "\Psi^\# k = (D\eta_X \circ k)^\sharp"'] & T^2X \arrow[l, "\mu_X"'] \arrow[d, "T\Psi^\# k"] \\ DTX & TDTX \arrow[l, "\Psi\mu_X"] \end{tikzcd}$$

That is,

$$\Psi^\# = (D\eta \circ \_)^\sharp$$

Dually, every functorial operational semantics Φ defines, by means of (9) a functorial denotational semantics

$$\Phi^@ = (\_ \circ T\varepsilon)^\flat$$

Proof. Naturality follows from universality. Next, $\Psi^\# k \circ \mu_X$ is equal to $D\mu_X \circ \Psi^{\#2} k$ because both fit as the (unique!) T-algebra arrow

$$(\Psi^\# k)^\sharp : (T^2X \xrightarrow{\mu_{TX}} TX) \to (TDTX \xrightarrow{\Psi k} DTX)$$

obtained by taking the left adjunct of $\Psi^\# k$ wrt the adjunction (4). Similarly, $\Psi^\# k : TX \to DTX$ is a D-coalgebra: to prove that $\varepsilon_{TX} \circ \Psi^\# k$ is equal to the identity id_{TX} on TX notice that both fit as $\eta_X^\sharp$, since $\eta_X = \eta_X \circ \mathsf{id}_X = \eta_X \circ (\varepsilon_X \circ k)$; and to prove that $D\Psi^\# k \circ \Psi^\# k$ is equal to $\delta_{TX} \circ \Psi^\# k$ notice that both fit as $(D^2\eta_X \circ (Dk \circ k))^\sharp = (D^2\eta_X \circ (\delta_X \circ k))^\sharp$. $\square$

For the denotational semantics Ψ in the above example one has the following induced operational semantics $\Psi^\#$. Using the isomorphisms (5) (to move from Σ- to

T-actions) and (8) (to move from D- to B-coactions), obtain $\Psi^{\#} : TU_B \Rightarrow BTU_B$. Consider, for simplicity, the case of the 'empty' coalgebra $0 : 0 \to B0$ given by the initial function into $B0$ and put

$$[\![\text{-}]\!]_\Psi = \Psi^{\#}(0) : T0 \to BT0$$

Up to (7), this is the transition system induced by Ψ on the closed program $t \in T0$ of basic process algebra. Spelling out the details, one can obtain

$$\begin{array}{lcl}
[\![\mathsf{nil}]\!]_\Psi & = & \emptyset \\
[\![a\,.\,t]\!]_\Psi & = & \{<a, t>\} \\
[\![t_1 \,\mathsf{or}\, t_2]\!]_\Psi & = & [\![t_1]\!]_\Psi \cup [\![t_2]\!]_\Psi
\end{array}$$

Using (7), this really yields basic process algebra:

$$\begin{array}{l}
\mathsf{nil} \not\rightarrow \\
a\,.\,t \xrightarrow{a} t \\
t_1 \,\mathsf{or}\, t_2 \xrightarrow{a} t \quad \text{if } t_1 \xrightarrow{a} t \text{ or } t_2 \xrightarrow{a} t
\end{array}$$

Operational is Denotational

The mapping $\Phi \mapsto \Phi^{@}$ is a bijection between operational monads and denotational comonads with $\Psi \mapsto \Psi^{\#}$ as inverse:

$$\begin{array}{ccc}
\Phi & \begin{tikzcd} \mathbf{C}_D \arrow[r, "\Phi"] \arrow[d, "U_D"'] & \mathbf{C}_D \arrow[d, "U_D"] \\ \mathbf{C} \arrow[r, "T"'] & \mathbf{C} \end{tikzcd} & \Psi^{\#} \\
\big\downarrow & \overline{\overline{\qquad\qquad\qquad}} & \big\uparrow \\
\Phi^{@} & \begin{tikzcd} \mathbf{C}^T \arrow[r, "\Psi"] \arrow[d, "U^T"'] & \mathbf{C}^T \arrow[d, "U^T"] \\ \mathbf{C} \arrow[r, "D"'] & \mathbf{C} \end{tikzcd} & \Psi
\end{array}$$

Proof. Everything in sight in the following diagram commutes.

$$\begin{tikzcd}
TX \arrow[rrr, "\Phi k"] & & & DTX \\
TX \arrow[u, equal] \arrow[r, "T\eta_X"] \arrow[d, "Tk"'] \arrow[rrr, bend right=80, "(\Phi^{@})^{\#} k"', to path={}] & TDX \arrow[ul, "\mu_X"'] \arrow[r, "\Phi^2 k"] \arrow[d, "T\Phi k"] & DT^2X \arrow[ur, "D\mu_X"'] \arrow[d, "DT\Phi k"] \arrow[dr, phantom, "="] & \\
TDX \arrow[r, "TD\eta_X"'] & TDTX \arrow[r, "\Phi\delta_{TX}"'] & DTDTX \arrow[r, "DT\varepsilon_{TX}"'] & DT^2X \arrow[uu, "D\mu_X"']
\end{tikzcd}$$

Indeed, the value of the unit of the adjunction (9) at a coalgebra $\langle X, k\rangle$ is its structure $k : X \to DX$, thus, in particular, its value at $\langle TDX, \Phi\delta_X\rangle$ is $\Phi\delta_X : TDX \to DTDX$, hence:

$$\Phi^{@}h = (h \circ T\varepsilon_X)^{\flat} = D(h \circ T\varepsilon_X) \circ \Phi\delta_X = Dh \circ DT\varepsilon_X \circ \Phi\delta_X$$

Dually:

$$\Psi^{\#}k = \Psi\mu_X \circ TD\eta_X \circ Tk$$

Therefore:

$$\begin{aligned}(\Phi^{@})^{\#}k &= \Phi^{@}\mu_X \circ TD\eta_X \circ Tk \\ &= D\mu_X \circ DT\varepsilon_{TX} \circ \Phi\delta_{TX} \circ TD\eta_X \circ Tk\end{aligned}$$

This proves the commutativity of the lower subdiagram in the above diagram. The other non-immediate fact is the commutativity of the subdiagram in the middle, but this follows from the fact that it is the image under the functor Φ of one of the two D-coalgebra laws for the structure $\Phi k : TX \to DTX$. That is,

$$\begin{tikzcd} TX \arrow[r, "\Phi k"] \arrow[d, "\Phi k"'] & DTX \arrow[d, "\delta_{TX}"] \\ DTX \arrow[r, "D\Phi k"'] & D^2TX \end{tikzcd} \quad \overset{\Phi}{\longmapsto} \quad \begin{tikzcd} T^2X \arrow[r, "T\Phi k"] \arrow[d, "\Phi^2 k"'] & TDTX \arrow[d, "\Phi\delta_{TX}"] \\ DT^2X \arrow[r, "DT\Phi k"'] & DTDTX \end{tikzcd}$$

This proves that $\Phi k = (\Phi^{@})^{\#}k$ and, by duality, $\Psi h = (\Psi^{\#})^{@}h$.

Φ-Algebras are $\Phi^{@}$-coalgebras

The algebras of an operational monad Φ and the coalgebras of its coinduced denotational comonad $\Phi^{@}$ are respectively of the form

$$\begin{tikzcd} TX \arrow[r, "\Phi k"] \arrow[d, "h"'] & DTX \arrow[d, "Dh"] \\ X \arrow[r, "k"'] & DX \end{tikzcd} \qquad \begin{tikzcd} TX \arrow[r, "Tk"] \arrow[d, "h"'] & TDX \arrow[d, "\Phi^{@}h"] \\ X \arrow[r, "k"'] & DX \end{tikzcd}$$

where $h : TX \to X$ is a T-algebra and $k : X \to DX$ is a D-coalgebra. For both, the arrows are those between their carriers (hence in $\mathbf{C}$) which are simultaneously T-algebra (hence in $\mathbf{C}^T$) and D-coalgebra (hence in $\mathbf{C}_D$) arrows.

The claim is that $Dh \circ \Phi k$ is equal to $\Phi^{@}h \circ Tk$. Indeed, everything in sight in the following diagram commutes.

$$\begin{tikzcd} TX \arrow[rr, "\Phi k"] \arrow[dd, "Tk"'] & & DTX \arrow[dl, "DTk"'] \arrow[d, equal] \\ & DTDX \arrow[r, "DT\varepsilon_X"'] & DTX \arrow[d, "Dh"] \\ TDX \arrow[ur, "\Phi\delta_X"'] \arrow[rr, "\Phi^{@}h"'] & & DX \end{tikzcd}$$

The only non-trivial sub-diagram is the one corresponding to the upper left corner but this is the image under the functor Φ of one of the two D-coalgebra laws for the structure $k : X \to DX$. That is,

$$\begin{tikzcd} X \arrow[r, "k"] \arrow[d, "k"'] & DX \arrow[d, "\delta_X"] \\ DX \arrow[r, "Dk"'] & D^2X \end{tikzcd} \quad \overset{\Phi}{\longmapsto} \quad \begin{tikzcd} TX \arrow[r, "Tk"] \arrow[d, "\Phi k"'] & TDX \arrow[d, "\Phi\delta_X"] \\ DTX \arrow[r, "DTk"'] & DTDX \end{tikzcd}$$

Thus, up to the permutation $\langle X, k, h\rangle \mapsto \langle X, h, k\rangle$, for any monad Φ lifting a monad T to the coalgebras of a comonad D, the two categories of Φ-algebras and $\Phi^@$-coalgebras are the same:

$$\mathbf{C}_D{}^{\Phi} = \mathbf{C}^T{}_{\Phi^@}$$

Dually,

$$\mathbf{C}^T{}_{\Psi} = \mathbf{C}_D{}^{\Psi^\#}$$

that is, Ψ**-coalgebras are** $\Psi^\#$**-algebras**.

Adequacy

If Φ is an operational monad, then the category $\mathbf{C}_D{}^{\Phi} = \mathbf{C}^T{}_{\Phi^@}$ can be seen as the category of **models of** Φ:

$$\Phi\text{-}\mathbf{Mod} = \mathbf{C}_D{}^{\Phi} = \mathbf{C}^T{}_{\Phi^@}$$

This category has both an initial and a final object which are 'lifted' from the initial T-algebra and the final D-coalgebra, respectively.

Lemma. The forgetful functor $\widetilde{U_D} : \mathbf{C}_D{}^{\Phi} \to \mathbf{C}^T$ with action $(TX \to X \to DX) \mapsto (TX \to X)$ has a right adjoint, namely

$$\begin{tikzcd} TX \arrow[d, "h"'] \\ X \end{tikzcd} \quad \overset{\widetilde{G_D}}{\longmapsto} \quad \begin{tikzcd} TDX \arrow[r, "\Phi\delta_X"] \arrow[d, "\Phi^@ h"] & DTDX \arrow[d, "D\Phi^@ h"] \\ DX \arrow[r, "\delta_X"'] & D^2X \end{tikzcd}$$

Dually, $\widetilde{U^T} : \mathbf{C}^T{}_{\Psi} \to \mathbf{C}_D$ has a left adjoint

$$(X \xrightarrow{k} DX) \overset{\widetilde{F^T}}{\longmapsto} (T^2X \xrightarrow{\mu_X} TX \xrightarrow{\Psi^\# k} DTX)$$

Proof. The counit of the adjunction is simply the counit ε of D, ie it is lifted from the adjunction (9). $\square$

Thus there are two adjunctions for the category of Φ-models, namely

$$\mathbf{C}^T \underset{\widetilde{G_D}}{\overset{\widetilde{U_D}}{\underset{\perp}{\leftrightarrows}}} \mathbf{C}_D{}^{\Phi} = \Phi\text{-}\mathbf{Mod} = \mathbf{C}^T{}_{\Phi^@} \underset{\widetilde{U^T}}{\overset{\widetilde{F^T}}{\underset{\perp}{\leftrightarrows}}} \mathbf{C}_D$$

Hence, $\widetilde{F^T}$ maps the (trivial) initial D-coalgebra to the *initial* Φ-*model*:

$$(0 \xrightarrow{0} D0) \overset{\widetilde{F^T}}{\longmapsto} (T^2 0 \xrightarrow{\mu_0} T0 \xrightarrow{\Phi 0} DT0)$$

Dually, $\widetilde{G_D}$ maps the (trivial) final T-algebra to the *final* Φ-*model*:

$$(T1 \xrightarrow{1} 1) \overset{\widetilde{G_D}}{\longmapsto} (TD1 \xrightarrow{\Phi^@ 1} D1 \xrightarrow{\delta_1} D^2 1)$$

Then, by definition of Φ-algebra (alias Φ-model) arrow, the following holds.

Adequacy Theorem. The unique (both by initiality and finality) arrow from $\widetilde{F^T}(0)$ to $\widetilde{G_D}(1)$ is both the *initial algebra semantics* from the closed programs $T0$ to the domain $D1$ with denotations $\Phi^@1$, and the *final coalgebra semantics* from the transition system $\Phi 0$ on the closed programs to the set of most abstract observations $D1$. □

Since by 'pulling back' this final coalgebra semantics one obtains the greatest B-bisimulation, the fact that it is also an initial algebra semantics gives the following

Corollary. B-bisimulation is a congruence wrt Φ.

GSOS is Functorial

First a preliminary remark. Notice that, in the operational semantics $[\![\text{-}]\!]_\Psi$ given above for basic process algebra, the construct or behaves as the join ∪ of the semi-lattice $\mathcal{P}_{fi}(\mathsf{Act} \times T)$. Thus the above Ψ can also be seen as a lifting of B to the $\langle \Sigma, E\rangle$-algebras, where E are the semi-lattice laws for the binary operator or (ie or is required to be associative, commutative, and absorptive). For simplicity, let us keep the notation $T =< T, \eta, \mu >$ also for the monad corresponding to the $\langle \Sigma, E\rangle$-algebras.

(Thus TX is now the quotient wrt (the congruence relation generated by) E of the (previous) free algebra of terms over X; thus one cannot distinguish in this syntax between, for instance, the terms $t_1 \,\mathsf{or}\, t_2$ and $t_2 \,\mathsf{or}\, t_1$. Keeping this quotient in mind, one can still regard the elements of TX as terms, that is, one can use representatives rather than equivalence classes.)

One can then embed the behaviour $BX = \mathcal{P}_{fi}(\mathsf{Act} \times X)$ into this new syntax T by mapping ∅ to nil, $\{<a, x>\}$ to $a.x$, and ∪ to or. This defines a natural transformation

$$\gamma : B \Rightarrow T$$

injective in each component. It is a **retraction** for the above basic process algebra $\Psi^{\#}$, in the sense that the composite $\Psi^{\#} \circ \mu \circ \gamma_T : BT \Rightarrow BT$ is the identity natural transformation on BT.

This retraction γ is important because it permits to regard every set $\mathcal{R}$ of 'GSOS' rules containing basic process algebra as a natural transformation $\lceil \mathcal{R} \rceil : \Sigma B \Rightarrow BT$. A **GSOS rule** specifies one possible transition for terms of the form $\sigma(u_1, \ldots u_l)$, for σ a given program construct of arity l:

$$\frac{\{u_i \xrightarrow{a_{ij}} v_{ij}\}^{1 \leq i \leq l}_{1 \leq j \leq m_i} \quad \{u_i \overset{b_{ij}}{\not\longrightarrow}\}^{1 \leq i \leq l}_{1 \leq j \leq n_i}}{\sigma(u_1, \ldots, u_l) \xrightarrow{a} C[\overrightarrow{u}, \overrightarrow{v}]} \tag{12}$$

The a_{ij}'s and b_{ij}'s are actions in Act; the u_i's and v_{ij}'s are all distinct (meta) variables ranging over terms, the expression $C[\overrightarrow{u}, \overrightarrow{v}]$ is a term formed by the context $C[\overrightarrow{-}]$ and some (meta) variables contained in the set of u_i's and v_{ij}'s.

Clearly, the rules of basic process algebra are in GSOS. Let us assume that every set $\mathcal{R}$ of GSOS rules *conservatively extends* basic process algebra. Therefore, the corresponding syntactical monad T contains terms $\mathsf{nil}, a\,.\,t, t_1 \,\mathsf{or}\, t_2$, and or is a semi-lattice join, hence the above retraction $\gamma : B \Rightarrow T$ is a retraction also for (the operational semantics induced by the rules) $\mathcal{R}$.

Then, using the meta-variables r_i to range over the elements of $BX = \mathcal{P}_{\mathrm{fi}}(\mathsf{Act} \times X)$, define $\lceil \mathcal{R} \rceil_X : \Sigma BX \to BTX$ by putting, for every rule (12) in $\mathcal{R}$,

$$< a, C[\overrightarrow{\gamma_X r}, \overrightarrow{x}] > \in \lceil \mathcal{R} \rceil_X (\sigma(r_1, \ldots, r_l))$$

if $\{< a_{ij}, x_{ij} > \in r_i\}^{1 \leq i \leq l}_{1 \leq j \leq m_i}$ and, for every $x \in X$, $\{< b_{ij}, x > \notin r_i\}^{1 \leq i \leq l}_{1 \leq j \leq n_i}$

The claim now is that this really defines a natural transformation

$$\lceil \mathcal{R} \rceil : \Sigma B \Rightarrow BT$$

ie for every 'variable renaming' $f : X \to Y$, $BTf \circ \lceil \mathcal{R} \rceil_X \; = \; \lceil \mathcal{R} \rceil_Y \circ \Sigma Bf$.

Consider the case of negative premises: if there is no pair $< b_{ij}, x >$ in $r_i \in BX$ for any $x \in X$, then there is also no pair $< b_{ij}, y >$ in $(Bf)(r_i) \in BY$ for arbitrary $y \in Y$. Assume thus only positive premises in the rule. Then the following lemma suffices to prove the claim.

Substitution Lemma. $(Tf)(C[\overrightarrow{\gamma_X r}, \overrightarrow{x}]) = C[\overrightarrow{\gamma_Y (Bf)(r)}, \overrightarrow{fx}]$

Proof. It is an immediate consequence of the naturality of the retraction γ from B to T and of the GSOS condition that the variables of $C[\overrightarrow{u}, \overrightarrow{v}]$ are contained in the set of u_i's and $v_{i,j}$'s (hence $(Tf)C[\ldots] = C[(Tf)\ldots]$). □

Next, this natural transformation $\lceil \mathcal{R} \rceil : \Sigma B \Rightarrow BT$ can be made into an action of Σ on BT:

$$\Sigma BT \overset{\lceil \mathcal{R} \rceil_T}{\Longrightarrow} BT^2 \overset{B\mu}{\Longrightarrow} BT$$

This family of Σ-algebras validates the semi-lattice laws, thus, using (6), it can also be seen as an action

$$\phi^{\mathcal{R}} : TBT \Rightarrow BT$$

of the syntactical monad T. Then, like in the basic property, one can obtain an operational monad Φ lifting the monad T to the B-coalgebras (instead of to the D-coalgebras) by putting $\Phi = (B\eta \circ \_)^\sharp$, ie

$$\begin{tikzcd}
X \arrow[r, "\eta_X"] \arrow[d, "k"'] & TX \arrow[d, "\Phi k"] \\
BX \arrow[r, "B\eta_X"'] & BTX
\end{tikzcd}
\qquad
\begin{tikzcd}
TX \arrow[d, dashed, "\Phi k = (B\eta_X \circ k)^\sharp"'] & T^2X \arrow[l, "\mu_X"'] \arrow[d, "T\Phi k"] \\
BTX & TBTX \arrow[l, "\phi^{\mathcal{R}}_X"]
\end{tikzcd}$$

(Notice the coalgebra $k : X \to BX$ can be seen, by (7), as a set of "*δ-rules*" in the sense of [BIM88].)

Theorem.

The operational semantics induced by $\mathcal{R}$ is *observationally equivalent* to Φ.

Proof. Consider, without loss of generality, the case of closed terms $T0$. Call $\llbracket\text{-}\rrbracket_{\mathcal{R}} : T0 \to BT0$ the coalgebra corresponding, via (7), to the transition system induced by $\mathcal{R}$ starting from the empty transition system (ie from the coalgebra $0 : 0 \to B0$). Similarly, put $\llbracket\text{-}\rrbracket_{\lceil\mathcal{R}\rceil} = \Phi 0 : T0 \to BT0$. The claim is that the final coalgebra semantics $\llbracket\text{-}\rrbracket^{@}_{\mathcal{R}} : (T0 \xrightarrow{\llbracket\text{-}\rrbracket_{\mathcal{R}}} BT0) \to (D1 \cong BD1)$ and $\llbracket\text{-}\rrbracket^{@}_{\lceil\mathcal{R}\rceil} : (T0 \xrightarrow{\llbracket\text{-}\rrbracket_{\lceil\mathcal{R}\rceil}} BT0) \to (D1 \cong BD1)$ are the same. The idea is that the final coalgebra semantics abstracts from the actual name of the states and just looks at the actions which can be performed. Then, since $<a, t'> \in \llbracket t\rrbracket_{\lceil\mathcal{R}\rceil}$ iff there exists a context $C[\overrightarrow{u}, \overrightarrow{v}]$ and terms u_i's and v_{ij}'s such that $<a, C[\overrightarrow{u}, \overrightarrow{v}]> \in \llbracket t\rrbracket_{\mathcal{R}}$ and $t' = C[\overrightarrow{\gamma_{T0}\llbracket u\rrbracket_{\lceil\mathcal{R}\rceil}}, \overrightarrow{v}]$, the theorem follows from the following

Lemma. $\llbracket C[\overrightarrow{\gamma_{T0}\llbracket u\rrbracket_{\lceil\mathcal{R}\rceil}}, \overrightarrow{v}]\rrbracket^{@}_{\lceil\mathcal{R}\rceil} = \llbracket C[\overrightarrow{u}, \overrightarrow{v}]\rrbracket^{@}_{\lceil\mathcal{R}\rceil}$

Proof. From the second corollary of the main theorem, the final coalgebra semantics $\llbracket\text{-}\rrbracket^{@}_{\lceil\mathcal{R}\rceil} : T0 \to D1$ is also an initial algebra semantics (wrt the denotations $\Phi^{@}1$), hence $\llbracket\text{-}\rrbracket^{@}_{\lceil\mathcal{R}\rceil}$ is compositional and the lemma can be reduced to

$$\llbracket\gamma_{T0}\llbracket u_i\rrbracket_{\lceil\mathcal{R}\rceil}\rrbracket^{@}_{\lceil\mathcal{R}\rceil} = \llbracket u_i\rrbracket^{@}_{\lceil\mathcal{R}\rceil}$$

which is a consequence of the fact that γ is a retraction for (basic process algebra and hence, as one can check, for) $\llbracket\text{-}\rrbracket_{\lceil\mathcal{R}\rceil}$. $\square$

Notice that $\llbracket\text{-}\rrbracket^{@}_{\lceil\mathcal{R}\rceil} : T0 \to D1$ is the unique arrow from the initial to the final Φ-algebra.

GSOS models are Φ-models. Spelling out the definition of Φ-models (alias Φ-algebras) for the operational monad Φ corresponding to a set $\mathcal{R}$ of GSOS rules, one obtains those

$$TX \xrightarrow{h} X \xrightarrow{k} BX$$

such that $h : TX \to X$ validates the T-algebra laws and such that

$$< a, x' > \in (k \circ h)(\sigma(x_1, \ldots, x_l))$$

iff there exists a rule (12) in $\mathcal{R}$ such that $< a_{ij}, y_{ij} > \in k(x_i)$, $x' = h(C[\overrightarrow{x}, \overrightarrow{y}])$, and, for all $y \in X$, $< b_{ij}, y > \notin k(x_i)$. Up to the isomorphisms (5) and (7), this is the definition of **GSOS models** given in [Sim95].

Guarded Recursion, coalgebraically. Every set of terms (mutually) recursively defined by means of equations in some variables $x_i \in X$

$$x_1 = t_1[X],\ x_2 = t_2[X], \ldots$$

where $t_i[X]$ are elements of TX (hence might contain variables from X), can be seen as a T-coalgebra $k : X \to TX$ by putting $k(x_i) = t_i[X]$. (And vice versa.) In order to interpret the recursive terms $x_i = t_i[X]$ operationally, the usual requirement is that they are **guarded**, that is, every term t_i is of the form $(a_{i_1} . t_{i_1})$ or $\ldots$ or $(a_{i_n} . t_{i_n})$. Notice then, that if all terms in a recursive definition are guarded, the corresponding coalgebra $k : X \to TX$ always factorizes through a BT-coalgebra $g : X \to BTX = \mathcal{P}_{\mathrm{fi}}(\mathsf{Act} \times TX)$ as follows.

$$k = \mu_X \circ \gamma_{TX} \circ g : X \to TX$$

Clearly, $g(x_i) = \{< a_{i_1}, t_{i_1} >, \ldots, < a_{i_n}, t_{i_n} >\}$. Conversely, every BT-coalgebra can be seen as a set of mutually recursive definitions.

Now, one can take the left adjunct wrt the adjunction (4) of every $g : X \to BTX$ using a given set of GSOS rules $\mathcal{R}$:

$$\begin{tikzcd} X \arrow[r, "\eta_X"] \arrow[dr, "g"'] & TX \arrow[d, "\llbracket\text{-}\rrbracket^g_{\lceil\mathcal{R}\rceil}"] \\ & BTX \end{tikzcd} \qquad \begin{tikzcd} TX \arrow[d, dashed, "\llbracket\text{-}\rrbracket^g_{\lceil\mathcal{R}\rceil} = g^\sharp"'] & T^2X \arrow[l, "\mu_X"'] \arrow[d, "T\llbracket\text{-}\rrbracket^g_{\lceil\mathcal{R}\rceil}"] \\ BTX & TBTX \arrow[l, "\phi^{\mathcal{R}}_X"] \end{tikzcd}$$

Then the final coalgebra semantics $(\llbracket\text{-}\rrbracket^g_{\lceil\mathcal{R}\rceil})^{@} : TX \to D1$ from the resulting coalgebra $\llbracket\text{-}\rrbracket^g_{\lceil\mathcal{R}\rceil} : TX \to BTX$ to the final coalgebra $D1 \cong BD1$ gives the desired interpretation of g as a recursive process. Notice that no variable binding operator (like, eg, 'fix' in [BIM88]) is (explicitly) needed here.

Example. Let $\mathcal{R}$ be basic process algebra together with the rules for (simple) *interleaving*

$$\frac{u \xrightarrow{a} u'}{u \parallel v \xrightarrow{a} u' \parallel v} \qquad \frac{v \xrightarrow{a} v'}{u \parallel v \xrightarrow{a} u \parallel v'}$$

and let g be the BT-coalgebra corresponding to the guarded recursive definition

$$x = a \,.\, x \qquad y = (a \,.\, y)\ \mathsf{or}\ (b \,.\, x)$$

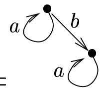
in $X = \{x, y\}$. (Notice that the x's and y's in the interleaving rules are *meta*-variables not to be confused with the actual variables x, y used in the recursive definition.) Writing, for simplicity,

$$\llbracket\text{-}\rrbracket = \llbracket\text{-}\rrbracket^{g}_{\lceil\mathcal{R}\rceil} : TX \to BTX$$

and letting $\llbracket\text{-}\rrbracket^{@} : TX \to D1$ be the unique Φ-algebra arrow from $T^2X \xrightarrow{\mu_X} TX \xrightarrow{\llbracket\text{-}\rrbracket} BTX$ to the final Φ-algebra $TD1 \xrightarrow{\Phi^{@}1} D1 \xrightarrow{\sim} BD1$, one has, omitting the insertion-of-variables η_X and the final coalgebra isomorphism $D1 \cong BD1$,

$$\llbracket x\rrbracket^{@} = \{<a, \llbracket x\rrbracket^{@}>\} =$$

$$\llbracket y\rrbracket^{@} = \{<a, \llbracket y\rrbracket^{@}>, <b, \llbracket x\rrbracket^{@}>\} =$$



$$\begin{array}{lcl}
\llbracket a \,.\, t\rrbracket^{@} & = & \{<a, \llbracket \gamma_{TX}\llbracket t\rrbracket\rrbracket^{@}>\} = \{<a, \llbracket t\rrbracket^{@}>\} \\
\llbracket t_1\ \mathsf{or}\ t_2\rrbracket^{@} & = & \{<a, \llbracket t_1'\rrbracket^{@}> \mid t_1 \xrightarrow{a} t_1'\} \cup \\
 & & \{<a, \llbracket t_2'\rrbracket^{@}> \mid t_2 \xrightarrow{a} t_2'\} \\
 & = & \llbracket t_1\rrbracket^{@} \cup \llbracket t_2\rrbracket^{@} = \Phi^{@}1(\llbracket t_1\rrbracket^{@}\ \mathsf{or}\ \llbracket t_2\rrbracket^{@}) \\
\llbracket t_1 \parallel t_2\rrbracket^{@} & = & \{<a, \llbracket t_1' \parallel t_2\rrbracket^{@}> \mid t_1 \xrightarrow{a} t_1'\} \cup \\
 & & \{<a, \llbracket t_1 \parallel t_2'\rrbracket^{@}> \mid t_2 \xrightarrow{a} t_2'\} \\
 & = & \Phi^{@}1(\llbracket t_1\rrbracket^{@} \parallel \llbracket t_2\rrbracket^{@})
\end{array}$$

Final Remarks. The retraction $\gamma : B \Rightarrow T$ gives a general way of dealing with guarded recursion, but it is not clear whether its use and the assumption that the rules conservatively extend basic process algebra are really necessary to present GSOS functorially. At the moment, basic process algebra, with its natural *denotational* definition, seems to be *the* language for the behaviour $BX = \mathcal{P}_{\mathrm{fi}}(\mathsf{Act} \times X)$ (somewhat like the untyped lambda-calculus is *the* language for a suitable function space functor [Sco80]), while all other GSOS rules seem to be intrinsically *operational* and in a less direct correspondence with the behaviour, although denotationally well-behaved.

V

Sets like Recursive Processes

Synopsis

This part is devoted to a coalgebraic presentation of Peter Aczel's theory of "non-well-founded sets" [Acz88]. A categorical duality is proved between the 'anti-foundation axiom' giving non-well-founded sets and the 'foundation axiom': it is shown that the former is equivalent to postulating that 'the universe $V = \mathcal{P}_S V$ is a final coalgebra', while the latter is equivalent to '$V = \mathcal{P}_S V$ is an initial algebra'. (The endofunctor $\mathcal{P}_S$ maps a class to the class of its (small) subsets.)

The semantic motivation for the use of anti-foundation is that it permits to prove the "Special Final Coalgebra Theorem" [Acz88] which states that, under mild assumptions, the greatest fixed point of an endofunctor on (possibly non-well-founded) sets is a final coalgebra.

The special final coalgebra theorem is stated in terms of the "Solution Lemma" [Acz88]. The final coalgebra presentation of anti-foundation adopted here renders this lemma (and its equivalence with anti-foundation) trivial. Correspondingly, the 'uniformity on maps' condition which an endofunctor has to satisfy in order for the special final coalgebra theorem to hold can be formulated in a more transparent way than in [Acz88].

Note. A preliminary version of this part has appeared as [RT93, §4].

Basic Set Theory

One way of understanding the abstract notion of *set* is as a collection x such that its elements have "no internal structure whatsoever" and x itself has "no internal structure except for equality and inequality of pairs of elements". (Cf [Law76, page 119].) Axiomatically, this corresponds to taking the membership relation '$\in$' as the only primitive notion of set theory and to postulating the following 'extensionality axiom', the first axiom of set theory.

Extensionality:

Two sets are equal iff they have the same elements.

Next, for every property P in a (first-order) language with membership and equality only, one would like the collection $\{x \mid \mathsf{P}(x)\}$ of sets which have the property P to be a set. However, Russel's paradoxical set $\{x \mid x \notin x\}$ shows that this 'strong comprehension axiom' cannot be stated in its full generality. One needs to consider properties *relative* to the elements of an already defined set. This leads to the 'comprehension axiom', the second axiom of set theory.

Comprehension:

For every property P and every set v, the collection

$$\{x \mid \mathsf{P}(x) \wedge x \in v\}$$

is a set.

As comprehension can be applied only to members of already defined sets, it is necessary to postulate the existence of some sets, either primitive or derived by applying some basic operators:

Empty Set:

There exists a set 0 with no elements.

Paring, Union, Power Set:

$\{x, y\}$, $\bigcup x$, $\mathcal{P}(x)$ are all sets, for x, y sets.

As usual, $\bigcup x$ and $\mathcal{P}(x)$ stand for the collection of all members of members of x and the collection of all subsets of x, respectively. In turn, the subset relation '$\subseteq$' can be derived from the membership relation:

$$x \subseteq y \iff \forall v\ (v \in x \Rightarrow v \in y)$$

By means of the union operator one can define an operator s acting as successor as follows: $s(x) = x \cup \{x\}$. The existence of an infinite set can be stated by postulating the existence of a set containing the natural numbers. That is:

Infinity:

> There exists a set containing 0 and closed under the successor operator s.

(The axioms above, as well as those given in the sequel, are written for convenience in natural language but note that they can also be expressed in the language of set theory – see, eg, [Lev79].)

Further useful notions can be derived from the above axioms, like, for instance, the notion of *ordered pair*:

$$<x, y> = \{x, \{x, y\}\}$$

A formal definition of function can then be given as a collection f of ordered pairs such that for every x there exists a unique y with $<x, y> \in f$. Two more axioms about functions are then usually added:

Replacement:

> The image of a set under a function is a set.

Choice:

> Every surjective function has a 'right inverse'.

A *right inverse* for a function $f : a \to b$ is a function $g : b \to a$ such that $f \circ g$ is the identity on b. The above axiom of choice is equivalent to postulate that for every set a there exists a *choice function*, that is, a function f such that, for every $x \in a$, $f(x) \in x$.

The above axioms (extensionality, comprehension, empty set, pairing, union, power set, infinity, replacement, choice) are the basic axioms of set theory; let us call the theory associated with (ie, the collection of all sentences derivable from) them **basic set theory** and the corresponding category of sets and functions **Set**. (Basic set theory is usually called *ZFC*$^-$ in the literature – see, eg, [Lev79].)

Classes

Even though the collection $\{x \mid \mathsf{P}(x)\}$ of all sets x having a given property P might not be a set, it can still be of interest for set theory. Such 'specifiable' collections are called **classes**. Clearly, a set is a class, but the converse is not true, in which case one speaks of a **proper class**. (Also the terminology '*large set*', vs '*small set*', is used.) In the sequel, lower case letters are used for (small) sets and capital letters for classes.

The equality between classes is determined by their *small* elements. That is, two classes $X = \{x \mid \mathsf{P}(x)\}$ and $Y = \{x \mid \mathsf{P}'(x)\}$ are equal if and only if P and P' hold for the same (small) sets.

An example of a proper class is the **universe of sets**, namely the collection of all sets:

$$V \;=\; \{x \mid x = x\}.$$

(Since the property $x = x$ trivially holds for all sets, the class V is the collection of all sets indeed.) Notice that different properties may specify the same class. For instance, any property other than '$x = x$' which holds for all sets can be used to specify the universe.

Next, let **SET** be the category of classes and (class) functions corresponding to basic set theory. The claim is that the universe V can be seen as the carrier of both an algebra and a coalgebra structure of a suitable power-set endofunctor $\mathcal{P}_S$ on **SET**.

Recall, from Section 10, that semi-lattices with sets as carriers and with arbitrary sets of joins give rise (by adjunction) to the power-set endofunctor $\mathcal{P} : \mathbf{Set} \to \mathbf{Set}$ and that, similarly, semi-lattices with finite joins give rise to the finite power-set functor $\mathcal{P}_{\text{fi}} : \mathbf{Set} \to \mathbf{Set}$. By considering semi-lattices with classes as carriers and joins of sets of classes one obtains then the following endofunctor on **SET**:

$$\mathcal{P}_S : \mathbf{SET} \to \mathbf{SET} \qquad\qquad X \mapsto \{x \mid x \text{ is a set } \wedge\ x \subseteq X\}$$

Notice that only (small) sub*sets* are taken into consideration. This makes possible that V be a fixed point of the power-set functor (which, by cardinality reasons, would not be the case if one would consider the collection of all sub*classes* of a given class):

> **The universe V is a fixed point** $V = \mathcal{P}_S V$.
>
> Indeed, V is the largest class. Thus, since $\mathcal{P}_S V$ is itself a class, $\mathcal{P}_S V \subseteq V$. For the converse it is sufficient to prove that every set x is a subset of V. That is, for every $y \in x$, y is also in V. This is immediate from the fact that y is a set.

Therefore, the identity on V can be seen both as a $\mathcal{P}_S$-algebra and as a $\mathcal{P}_S$-coalgebra structure for V.

Well-Founded Sets and Foundation

From the axioms of basic set theory alone it is not possible to draw a canonical picture of what the universe looks like, a picture independent of the specific interpretation one might give to the theory. This was felt as a problem already in the early developments of set theory. The solution was found in the '*foundation axiom*', which was then added to basic set theory. This axiom restricts the universe to the 'smallest' of all possible ones. Then the picture arises of a universe in which sets are hereditarily constructed from the empty set, by iterative applications of the power-set operator. Every set has a *rank*, namely the stage at which it appears in such a '*cumulative hierarchy*'.

In this section it is proved that the foundation axiom is equivalent to postulating that the universe $V = \mathcal{P}_S V$ is the initial algebra of the power-set endofunctor $\mathcal{P}_S$ on **SET**.

A set x is **well-founded** wrt the membership relation '$\in$' if either it is empty or has a least element wrt $\in$. In other words, there is no infinitely descending chain of elements starting from x. Correspondingly, let the class

$$W = \{x \mid x \text{ is well-founded wrt the relation} \in \}$$

be the **universe of well-founded sets**.

The 'foundations axiom' amounts to postulating that all sets in the universe V are well-founded, that is,

Foundation Axiom:

$$V = W$$

Now, notice that the class $\mathcal{P}_S W$ of (small) subsets of well-founded sets is the same as W, because the elements of a well-founded set are themselves well-founded. Thus

$$\mathcal{P}_S W = W$$

and the identity on W can be seen as a $\mathcal{P}_S$-algebra structure.

The universe of well-founded sets is the initial $\mathcal{P}_S$-algebra.

For every $\mathcal{P}_S$-algebra structure $h : \mathcal{P}_S X \to X$ there exists a unique function $h^{\#} : W \to X$ such that the following diagram commutes.

$$\begin{tikzcd}
\mathcal{P}_S W \arrow[r, "\mathcal{P}_S(h^{\#})"] \arrow[d, equal] & \mathcal{P}_S X \arrow[d, "h"] \\
W \arrow[r, dashed, "h^{\#}"'] & X
\end{tikzcd}$$

That is,

$$\begin{array}{rcl} h^{\#}(0) & = & h(0) \\ h^{\#}\{x_i\}_I & = & h\{h^{\#}(x_i)\}_I \end{array}$$

The proof is by straightforward induction on the (well-founded!) membership relation $\in$.

An immediate consequence of the initiality of W is the existence of a 'rank' function, mapping every well-founded set to a suitable 'ordinal'. An **ordinal** is a well-founded set which is *totally* ordered by the membership relation and which is 'transitive'. (A **transitive set** is a set x such that every element $y \in x$ is also a subset $y \subseteq x$.) Correspondingly, one can form the class $\mathbb{O}$ of all ordinals, which is a subclass of W.

If α and β are two ordinals such that $\beta \in \alpha$, one usually writes $\beta < \alpha$. The first ordinals are: 0, $s(0)$, $s^2(0)$, etc. The first limit ordinal is $\omega = \bigcup_{n \in N} s^n(0)$, which, by the infinity axiom, is indeed a set. In general, because every ordinal is totally ordered by $\in$, the union $\bigcup \{\alpha_i\}_I$ of a set $\{\alpha_i\}_I$ of ordinals is the least upper bound of the α_i's. As a consequence, the union operator is a $\mathcal{P}_S$-algebra structure on the class $\mathbb{O}$ of ordinals:

$$\bigcup : \mathcal{P}_S(\mathbb{O}) \to \mathbb{O} \qquad\qquad \{\alpha_i\}_I \mapsto \bigcup\{\alpha_i\}_I$$

The inductive extension $\mathsf{rank} = \bigcup^{\#} : W \to \mathbb{O}$ of this algebra structure on is the function assigning a 'rank' to every well-founded set. This can be thought of as the stage at which a well-founded set is constructed in an ideal construction starting from the empty set and then iteratively applying the power-set functor $\mathcal{P}_S$:

$$\begin{array}{rcl} \mathsf{rank}(0) & = & 0 \\ \mathsf{rank}\{x_i\}_I & = & \bigcup\{\mathsf{rank}(x_i)\}_I \end{array}$$

Another consequence of the initiality of W is that $W = \mathcal{P}_S W$ is the least (pre-) fixed point for $\mathcal{P}_S$:

$$W = \mathsf{lfp}[\mathcal{P}_S]$$

That is, for every class X such that $\mathcal{P}_S X \subseteq X$, one has that $W \subseteq X$. Indeed, regarding the inclusion of $\mathcal{P}_S X$ into X as a function $\kappa : \mathcal{P}_S X \hookrightarrow X$, one has that its inductive extension $\kappa^{\#} : W \to X$ is of the following form.

$$\begin{array}{rcl} \kappa^{\#}(0) & = & 0 \\ \kappa^{\#}\{x_i\}_I & = & \kappa\{\kappa^{\#}(x_i)\}_I \end{array}$$

Then, to see that $\kappa^\#$ is the inclusion of W into X, it suffices to notice that the power-set functor

$$\mathcal{P}_S \text{ **'preserves inclusion functions'**}$$

that is, if $\iota : X \hookrightarrow Y$ is the inclusion of a subclass X of Y into Y, then the function $\mathcal{P}_S(\iota) : \mathcal{P}_S X \to \mathcal{P}_S Y$ is the inclusion of $\mathcal{P}_S X$ into $\mathcal{P}_S Y$.

Usually, initial algebras are unique up to isomorphism, but in this setting one has a stronger result:

$$\mathcal{P}_S X = X \text{ is the initial } \mathcal{P}_S\text{-algebra} \iff X = W$$

That is, any other initial algebra which is a (strict) fixed point of $\mathcal{P}_S$ is not only isomorphic but equal to W. In order to prove this, ie the non-trivial implication from left to right, one can use very much the same argument as the one used above to prove that W is the least fixed point of $\mathcal{P}_S$.

Therefore, by replacing X by Y in the above equivalence, one has that the foundation axiom '$V = W$' is equivalent to postulating that the universe V is the initial algebra of the power-set functor:

Foundation is Initiality:

$$V = W \iff \mathcal{P}_S V = V \text{ is the initial } \mathcal{P}_S\text{-algebra.}$$

Anti-Foundation and Finality

Not all sets occurring in the mathematical practice are well-founded. A typical example is given by recursive processes as occurring in the semantics of programming languages. (Cf Section 5.) In order to ensure the existence of *non-well-founded sets*, one can postulate the 'anti-foundation axiom'.

In this section, 'anti-foundation' is shown to be the dual of the initial algebra formulation of 'foundation':

Foundation: $\mathcal{P}_S V = V$ is an initial $\mathcal{P}_S$-algebra.

Anti-Foundation: $V = \mathcal{P}_S V$ is a final $\mathcal{P}_S$-coalgebra.

That is, anti-foundation postulates that the universe is the 'largest' possible one, while foundation postulates that it is the 'smallest'.

Let us consider the existence of the final coalgebra for the endofunctor

$$\mathcal{P}_S : \mathbf{SET} \to \mathbf{SET} \qquad X \mapsto \{x \mid x \text{ is a set } \wedge\ x \subseteq X\}$$

where, recall **SET** is the category of classes (ie large sets) which are definable within basic set theory. The proof that a final coalgebra for this functor exists can be carried out very much the same way as for the finite power-set functor

$$\mathcal{P}_{\text{fi}} : \mathbf{Set} \to \mathbf{Set} \qquad x \mapsto \{y \mid y \text{ is finite } \wedge\ y \subseteq x\}$$

As shown in Section 13, the coalgebras of this finite power-set functor are the same as the directed finitely branching graphs and the final coalgebra is the set of rooted finitely branching trees (possibly of infinite depth) quotiented by $\mathcal{P}_{\text{fi}}$-bisimulation.

Correspondingly, the coalgebras of the power-set functor $\mathcal{P}_S$ are the same as the directed 'locally small' graphs and the final coalgebra is the class of rooted 'locally small' trees (possibly of infinite depth) quotiented by $\mathcal{P}_S$-bisimulation. A (possibly large) graph is **locally small** if the collection of children of every node is a (small) set. Thus locally small graphs are in between *large graphs* (with a class of nodes each possibly having a class of children) and *small graphs* (with a set of nodes and a set of arcs).

Peter Aczel's original formulation of the anti-foundation axiom is in terms of small graphs and 'decorations'. A **decoration** for (the graph corresponding to) a

$\mathcal{P}_S$- coalgebra $\langle X, k\rangle$ is a coalgebra arrow from $\langle X, k\rangle$ to $V = \mathcal{P}_S V$

$$\begin{tikzcd} X \arrow[r, "f"] \arrow[d, "k"'] & V \arrow[d, equal] \\ \mathcal{P}_S X \arrow[r, "\mathcal{P}_S f"'] & \mathcal{P}_S V \end{tikzcd}$$

That is, a function f from X to the universe V such that, for every $x \in X$,

$$f(x) = \{f(x') \mid x' \in k(x)\}$$

In terms of graphs, this corresponds to a function mapping every node to a set in the following way.

$$f(x) = \{f(x') \mid x \longrightarrow x'\}$$

Therefore, by definition of final coalgebra, the coalgebra $V = \mathcal{P}_S V$ is final if and only if every (directed) locally small graph has a unique decoration. Now, the claim is that 'locally small' can be replaced by 'small' in the above equivalence. That is, every *locally small* graph has a unique decoration if (and only if) every *small* graph has a unique decoration. Indeed:

> (By contradiction.) Assume that every small graph has a unique decoration and that there are two distinct decorations f and g of (a coalgebra $\langle X, k\rangle$ corresponding to) a locally small graph. Then there is a node $x \in X$ such that
>
> $$f(x) \neq g(x)$$
>
> Now, the subgraph of $\langle X, k\rangle$ accessible from x is not only locally small but also (totally) small, that is, there are only set-many nodes accessible from x, because every node has only set-many children. But then f and g are both decorations for this small subgraph, which, by hypothesis, implies that
>
> $$f(x) = g(x)$$
>
> (The same argument can be used to prove that the class of small $\mathcal{P}_S$-coalgebras forms a *generating class* for the $\mathcal{P}_S$-coalgebras. (Cf Section 13.))

As a consequence, the postulate '$V = \mathcal{P}_S V$ is a final $\mathcal{P}_S$-coalgebra' is equivalent to Peter Aczel's original formulation of anti-foundation:

Anti-Foundation Axiom:

Every directed small graph has a unique decoration.

That is,

Anti-Foundation is Finality:

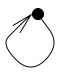
> Every directed small graph has a unique decoration if and only if $V = \mathcal{P}_S V$ is a final $\mathcal{P}_S$-coalgebra.

Notice that no axiom is needed in order to obtain a unique decoration for a *well-founded* graph: One can check that the class WG of well-founded directed small graphs is a (strict) fixed point for the power-set functor $\mathcal{P}_S$, and, moreover, that $\mathcal{P}_S(\mathsf{WG}) = \mathsf{WG}$ is an initial $\mathcal{P}_S$-algebra. Therefore WG is isomorphic to the universe of well-founded sets W and the image under this isomorphism of a well-founded graph is its unique decoration. (Cf "*Mostowski's collapsing lemma*" in [Acz88].)

When anti-foundation is postulated also non-well-founded graph have a unique decoration, but the converse is not true anymore. That is, there exist (non-well-founded) sets which 'decorate' different graphs. An example is the archetypal non-well-founded set, namely the *self-singleton* set

$$\Omega = \{\Omega\}$$

which is a member (and the only member) of itself. If anti-foundation is assumed, then both the root of the graph with one node and one arc



and the root of the graph consisting in one infinite path

$$\bullet \longrightarrow \bullet \longrightarrow \bullet \longrightarrow \cdots$$

are necessarily mapped to Ω by the corresponding unique decorations.

Notes. Aczel's anti-foundation axiom is equivalent to Forti and Honsell's "X_1-axiom" [FH83].

Besides applications in the semantics of programming languages (eg, [Acz88, Muk91, RT93, Acz94, Bal94, HL95, Har96]), non-well-founded sets have been extensively used in *Situation Theory* (eg, [BE87]), where they are better known as *hypersets*. (Correspondingly, models of the universe of non-well-founded sets are also called *hyperuniverses*.)

Reasoning about non-well-founded sets: bisimulation. By the extensionality axiom, the equality between two sets is determined by the membership relation. One of the consequences of *foundation* is that, since then the membership relation is well-founded, one can use *induction* to reason about (the equality between) sets. Categorically, this induction principle follows from the fact that foundation postulates that the universe is an initial algebra. Dually, anti-foundation, by postulating that the universe is a final coalgebra, gives a *coinduction* principle for reasoning about (possibly non-well-founded) sets.

Now, as shown in Section 12, if an endofunctor *preserves weak pullbacks* then coinduction (wrt its final coalgebra) can be 'pulled back' to the corresponding coalgebraic notion of *bisimulation*. In particular, the power-set functor $\mathcal{P}_S$ does preserve weak pullbacks; the proof is essentially the same as the one given in Section 12 for the behaviour $BX = \check{\mathcal{P}}(1 + \mathsf{Act} \times X)$. Therefore, two sets are equal if and only they are $\mathcal{P}_S$-bisimilar. (Cf [Acz88] for this "*Strong extensionality*".)

By instantiating the general definition of coalgebraic bisimulation (Section 12) to the $\mathcal{P}_S$-coalgebras one has that a (possibly large) relation on the carrier X of a coalgebra $\langle X, k\rangle$ *lifts to a $\mathcal{P}_S$-bisimulation* when, for all $x_1, x_2 \in X$ such that $x_1 R x_2$,

- if $x_1 \to x_1'$ then $x_2 \to x_2'$ for some x_2' such that $x_1' R\, x_2'$
- and, conversely, if $x_2 \to x_2'$ then $x_1 \to x_1'$ for some x_1' such that $x_1' R\, x_2'$.

(Here the notation $x \to x'$ stands for 'there is an arc from x to x' in the graph corresponding to the coalgebra $\langle X, k\rangle$'.)

In particular, a relation R on the universe V lifts to a $\mathcal{P}_S$-bisimulation if, for every set x and y such that xRy, for every $x' \in x$ there exists a $y' \in y$ such that $x' R\ y'$ and, conversely, for every $y' \in y$ there exists an $x' \in x$ such that $x' R\ y'$. Therefore, by strong extensionality,

$$x = y \quad\Longleftrightarrow\quad \text{there exists a relation } R \text{ such that:}\quad \begin{array}{ll} \bullet & x R\, y \\ \bullet & \forall x' \in x,\ \exists y' \in y,\ x' R\ y' \\ \bullet & \forall y' \in y,\ \exists x' \in x,\ x' R\ y' \end{array}$$

Systems of Set-Equations as Coalgebras

The self-singleton non-well-founded set $\Omega = \{\Omega\}$ can be seen as the unique solution of the 'set-equation'

$$x = \{x\}$$

In general, all non-well-founded sets arise from systems of set-equations with, on the left hand side, variables $x \in X$, and, on the right hand side, *well-founded* sets, possibly containing variables from X. This is the content of the "Solution Lemma".

In this section an elementary presentation of the solution lemma is given by means of the coalgebraic account of anti-foundation (and the initial algebra presentation of well-founded sets). This follows the coalgebraic treatment of recursive programs given in Section 5.

The definition of the universe of well founded sets W can be made parametric: for every (possibly large) set X, the **expanded universe of well-founded sets** WX is the class of all well-founded sets with variable $x \in X$. That is, every set in WX is either empty, or an element of X, or it has a least element wrt the membership relation $\in$. For $X = 0$ this yields the standard universe $W0$ of well-founded sets. Thus, in the sequel, W stands for an operator mapping a (large) set to the corresponding expanded universe of well-founded sets, rather than for the simple universe of well-founded sets.

The fact that $W0$ is the least (strict) fixed point of the power-set functor $\mathcal{P}_S$ and that $\mathcal{P}_S W0 = W0$ is an initial $\mathcal{P}_S$-algebra generalizes as follows: the class WX is the least (strict) fixed point of the endofunctor $X+\mathcal{P}_S(\text{-})$ on **SET** and

$$X + \mathcal{P}_S WX = WX$$

is an initial algebra for this endofunctor. As usual, this initiality can be used to extend the operator W to a functor (cf Section 1):

$$\begin{tikzcd}
X \arrow[r, "\eta_X = \mathsf{inl}_X"] \arrow[d, "f"'] & WX = X + \mathcal{P}_S WX \arrow[d, dashed, "Wf = [\eta_Y \circ f, \mathsf{inr}_Y]^{\#}"] & \mathcal{P}_S WX \arrow[l, "\mathsf{inr}_X"'] \arrow[d, "\mathcal{P}_S Wf"] \\
Y \arrow[r, "\eta_Y = \mathsf{inl}_Y"'] & WY = Y + \mathcal{P}_S WY & \mathcal{P}_S WY \arrow[l, "\mathsf{inr}_Y"]
\end{tikzcd}$$

That is, for every function $f : X \to Y$, the function $Wf : WX \to WY$ is the inductive extension of the algebra structure $\mathsf{inr}_Y : \mathcal{P}_S WY \to WY$ along the composite

$\eta_Y \circ f : X \to WY$, where the left injection $\eta_Y = \mathsf{inl}_Y : Y \to WY$ is the usual insertion-of-variables function. In other words,

$$W \text{ is } \textit{freely generated} \text{ by } \mathcal{P}_S.$$

Now, the idea is that a system of 'set-equations' like, eg,

$$\begin{array}{rcl} x & = & \{x, \{y\}\} \\ y & = & \{y, 0\} \end{array}$$

can be seen as a function k mapping the variables $x, y, \ldots \in X$ of the system to elements of $\mathcal{P}_S WX$, ie sets of well-founded sets possibly with variables in X. For instance, the above system corresponds to a function $k : \{x, y\} = X \to \mathcal{P}_S WX$ mapping x to $\{x, \{y\}\}$ and y to $\{y, 0\}$. Therefore, in general, a **system of set-equations in** X is a coalgebra $\langle X, k\rangle$ of the composite endofunctor $\mathcal{P}_S W$ on **SET**.

In order to solve a system of set-equations $\langle X, k\rangle$ one can (postulate anti-foundation and) use the finality of the universe $V = \mathcal{P}_S V$. For this, one first needs to extend the $\mathcal{P}_S W$-coalgebra structure $k : X \to \mathcal{P}_S WX$ to a $\mathcal{P}_S$-coalgebra structure as follows. Since $WX = X + \mathcal{P}_S WX$ is a coproduct, one can form the copair of k and the identity id on $\mathcal{P}_S WX$

$$\begin{tikzcd}
X \arrow[r, tail] \arrow[dr, "k"'] & WX \arrow[d, dashed, "{[k, \mathsf{id}]}"] & \mathcal{P}_S WX \arrow[l, tail] \arrow[dl, equal] \\
& \mathcal{P}_S WX &
\end{tikzcd}$$

This is a $\mathcal{P}_S$-coalgebra structure behaving as k on $x \in X$ and as the identity on $v \in \mathcal{P}_S WX$. Its coinductive extension $\overline{k} = [k, \mathsf{id}]^{@} : WX \to V$ wrt the final $\mathcal{P}_S$-coalgebra $V = \mathcal{P}_S V$ is then the (unique) **solution of the system** $k : X \to \mathcal{P}_S WX$ **of set-equations**:

$$\begin{tikzcd}
X \arrow[r, "\eta_X"] \arrow[dr, "k"'] & WX \arrow[r, dashed, "{\overline{k} = [k, \mathsf{id}]^{@}}"] \arrow[d, "{[k, \mathsf{id}]}"] & V \arrow[d, equal] \\
& \mathcal{P}_S WX \arrow[r, "\mathcal{P}_S \overline{k}"'] & \mathcal{P}_S V
\end{tikzcd}$$

Omitting, as usual, the injections, and letting v and v' range over objects of type $\mathcal{P}_S W$, one has that

$$\overline{k}(x) = \{\overline{k}(v) \mid v \in hx\}$$

and

$$\overline{k}(v) = \{\overline{k}(v') \mid v' \in v\}$$

For example, the unique solution of equation $k(x) = \{x\}$ is the self-singleton (non-well-founded) set

$$\overline{k}(x) = \{\overline{k}(x)\}$$

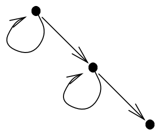
that is, $\overline{k}(x) = \Omega$. Similarly, the solution of the above system

$$k(x) = \{x, \{y\}\} \quad k(y) = \{y, 0\}$$

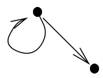
is

$$\begin{aligned} \overline{k}(x) &= \{\overline{k}(x), \{\overline{k}(y)\}\} \\ \overline{k}(y) &= \{\overline{k}(y), 0\} \end{aligned}$$

In terms of graphs, the sets $\overline{k}(x)$ and $\overline{k}(y)$ correspond to



and



respectively.

> **The Solution Lemma is equivalent to Anti-Foundation.** The above property that every system of set-equations has a unique solution, is called the **solution lemma** in [Acz88]. (See also [BE87, Chapter 3].) It is obtained assuming the anti-foundation axiom. Conversely, postulating the solution lemma, one can prove that $V = \mathcal{P}_S V$ is the final $\mathcal{P}_S$-coalgebra. Indeed, for every $\mathcal{P}_S$-coalgebra $\langle X, k\rangle$, one obtains
>
> $$\begin{tikzcd} X \arrow[r, "\eta_X"] \arrow[d, "k"'] & WX \arrow[r, dashed, "\overline{\mathcal{P}_S(\eta_X) \circ k}"] \arrow[d, "{[\mathcal{P}_S(\eta_X) \circ k, \mathsf{id}]}"] & V \arrow[d, equal] \\ \mathcal{P}_S X \arrow[r, "\mathcal{P}_S(\eta_X)"'] & \mathcal{P}_S W X \arrow[r, "\mathcal{P}_S(\overline{\mathcal{P}_S(\eta_X) \circ k})"'] & \mathcal{P}_S V \end{tikzcd}$$
>
> The desired coinductive extension of the coalgebra structure $k : X \to \mathcal{P}_S X$ is given by the composite coalgebra arrow
>
> $$k^{@} = \overline{\mathcal{P}_S(\eta_X) \circ k} \circ \eta_X : X \to V$$

Notice that, assuming anti-foundation, the upper rectangle in the following diagram commutes, because all other sub-diagrams commute.

$$
\begin{tikzcd}
 & & \mathcal{P}_S WX \arrow[dl, "\mathsf{inr}_X"'] \arrow[rr, "\mathcal{P}_S\overline{k}"] & & \mathcal{P}_S V \arrow[dl, equal] \\
X \arrow[r, "\eta_X"] \arrow[dr, "k"'] & WX \arrow[rr, "\overline{k}"] \arrow[d, "{[k,\mathsf{id}]}"'] & & V \arrow[d, equal] & \\
 & \mathcal{P}_S WX \arrow[rr, "\mathcal{P}_S\overline{k}"'] & & \mathcal{P}_S V &
\end{tikzcd}
$$

Therefore, the solution $\overline{k} : WX \to V$ of a system of set equations $\langle X, k\rangle$ is not only a $\mathcal{P}_S$-coalgebra arrow but also a $\mathcal{P}_S$-algebra arrow from $\langle WX, \mathsf{inr}_X\rangle$ to $\mathcal{P}_S V = V$. The algebra $\langle WX, \mathsf{inr}_X\rangle$ is a *free* $\mathcal{P}_S$*-algebra* over X.

The Substitution Lemma from Freeness. In the present approach, the proof of the solution lemma is trivial. The original proof, instead, makes use of a **substitution lemma** [Acz88]. This lemma asserts that, for every function $f : X \to V$, there exists a unique extension $f^\sharp : WX \to V$ of f to $WX = X + \mathcal{P}_S WX$ such that, omitting the injections,

$$f^\sharp(x) = f(x)$$

and

$$f^\sharp(v) = \{f^\sharp(v') \mid v' \in v\}$$

Now, also this becomes trivial here, because of the initial algebra presentation of the expanded universe of well-founded sets WX. Indeed, the desired function $f^\sharp : WX \to V$ is the inductive extension of the $\mathcal{P}_S$-algebra structure $\mathcal{P}_S V = V$ along $f : X \to V$. That is:

$$
\begin{tikzcd}
X \arrow[r, "\eta_X"] \arrow[dr, "f"'] & WX \arrow[d, dashed, "f^\sharp"] & \mathcal{P}_S WX \arrow[l, "\mathsf{inr}_X"'] \arrow[d, "\mathcal{P}_S f^\sharp"] \\
 & V \arrow[r, equal] & \mathcal{P}_S V
\end{tikzcd}
$$

Notice that, in contrast with [Acz88], anti-foundation is *not* used here.

Notes. In general, every free $\mathcal{P}_S$-algebra over a (possibly large) set X can be used to model the universe of Zermelo-Fraenkel set theory expanded with elements of X as *atoms*. This fact can be seen as an instance of a more general result in [JM95] (Theorem II.5.5) stated in terms of free "*Zermelo-Fraenkel algebras*" and intuitionistic set theory.

From Greatest Fixed Points to Final Coalgebras

The greatest (strict) fixed point $V = \mathcal{P}_S V$ of the power-set functor $\mathcal{P}_S$ can be seen as the final coalgebra of the restriction of the functor $\mathcal{P}_S$ to the subcategory $\mathbf{SET}_\subset$ of inclusion functions. Anti-foundation postulates that this final coalgebra lifts to a final coalgebra in **SET**. If an endofunctor is 'uniform on maps', then, assuming anti-foundation, its final coalgebra in the subcategory $\mathbf{SET}_\subset$ also lifts to a final coalgebra in **SET**. This is the content of the "Special Final Coalgebra Theorem".

In this section, a new formalization of the notion of uniformity on maps in terms of natural transformations is given. The proof of the theorem is then rephrased in terms of this definition.

Let F be an endofunctor on **SET**. A post-fixed point $X \subseteq FX$ for F can be seen as an inclusion function $X \hookrightarrow FX$, hence as an F-coalgebra structure on X. If the endofunctor F preserves inclusion functions, ie F applied to $X \hookrightarrow Y$ is an inclusion $FX \hookrightarrow FY$, then one can restrict F to the subcategory $\mathbf{SET}_\subset$ of classes and inclusion functions. The post-fixed points of F are then its coalgebras in this subcategory. In particular, the final F-coalgebra in $\mathbf{SET}_\subset$, if it exists, is the greatest (post-)fixed point

$$\mathsf{gfp}[F] = F(\mathsf{gfp}[F])$$

of F. The claim is that if F is 'uniform on maps' then, assuming anti-foundation, $\mathsf{gfp}[F] = F(\mathsf{gfp}[F])$ is also a final coalgebra.

Intuitively, an endofunctor on **SET** is uniform on maps if it is completely determined by its action on objects (ie classes). Most of endofunctors are thus uniform on maps. For instance, consider the endofunctor $X \mapsto A \times X$ mapping a class X to its product with a fixed class A. Given a function $f : X \to Y$, the value of $A \times f$ at an element $<a, x>$ of $A \times X$ is the pair $<a, f(x)> \in A \times Y$ which is obtained by applying f to the $x \in X$ in $A \times X$. This suggests that the class X should be regarded as a class of variables and that, in general, the action of a functor F uniform on maps on a function f should simply be the substitution of the variables x occurring in FX by $f(x)$.

Formally, this can be expressed by means of the expanded universe of well-founded sets $WX = X + \mathcal{P}_S WX$. What one needs is a natural transformation

$$\rho : F \Rightarrow \mathcal{P}_S W$$

which, for every X, 'embeds' FX into $\mathcal{P}_S WX$ – the class of sets of (well-founded) sets having $x \in X$ as variables.

Naturality amounts to having, for every function $f : X \to Y$, the following diagram commute.

$$\begin{tikzcd}
FX \arrow[r, "Ff"] \arrow[d, "\rho_X"'] & FY \arrow[d, "\rho_Y"] \\
\mathcal{P}_S W X \arrow[r, "\mathcal{P}_S W f"'] & \mathcal{P}_S W Y
\end{tikzcd}$$

It should be an 'embedding' in the sense that, for every X and for every $v \in FX$, by 'forgetting' the distinction between variables and sets in $\rho_X(v) \in \mathcal{P}_S W X$ one should get back the original set v. This operation of forgetting the distinction between variables and sets in objects of type $\mathcal{P}_S W$ can be made formal as follows.

Consider the inductive extension $\varepsilon_V : WV \to V$ of the $\mathcal{P}_S$-algebra structure $\mathcal{P}_S V = V$ along the identity on V:

$$\begin{tikzcd}
V \arrow[r, "\eta_V"] \arrow[dr, equal] & WV \arrow[d, dashed, "\varepsilon_V"] & \mathcal{P}_S WV \arrow[l, "\mathsf{inr}_V"'] \arrow[d, "\mathcal{P}_S(\varepsilon_V)"] \\
 & V \arrow[r, equal] & \mathcal{P}_S V
\end{tikzcd}$$

Omitting, as usual, the injections, one has that, for every $v \in WV$, $\varepsilon_V(v) = v$ if v is a variable and $\varepsilon_V(v) = \{\varepsilon_V(v_i)\}_I$ if $v = \{v_i\}_I$. Then, an endofunctor $F : \mathbf{SET} \to \mathbf{SET}$ is **uniform on maps** if there exists a natural transformation

$$\rho : F \Rightarrow \mathcal{P}_S W$$

such that

$$\begin{tikzcd}
FV \arrow[r, hook] \arrow[d, "\rho_V"'] & V \arrow[d, equal] \\
\mathcal{P}_S WV \arrow[r, "\mathcal{P}_S(\varepsilon_V)"'] & \mathcal{P}_S V
\end{tikzcd}$$

commutes.

Before setting out to prove the special final coalgebra theorem, notice that, since W is freely generated by $\mathcal{P}_S$, the forgetful functor mapping $\mathcal{P}_S$-algebras to their carriers is right adjoint to the functor mapping a class X to the (free) $\mathcal{P}_S$-algebra with carrier WX and structure

$$\mathsf{inr}_X : \mathcal{P}_S W X \to WX$$

(Cf Section 2.) The other injection $\eta_X = \mathsf{inl}_X : X \to WX$ is the unit of the adjunction at X, while the value of the counit at an algebra $\langle Y, h \rangle$ is given by the inductive extension of the right injection $\mathsf{inr}_Y : \mathcal{P}_S WY \to WY$ along the identity on Y.

$$\begin{tikzcd}
Y \arrow[r, "\eta_Y"] \arrow[dr, equal] & WY \arrow[d, dashed, "\varepsilon_{\langle Y,h\rangle}"] & \mathcal{P}_S WY \arrow[l, "\mathsf{inr}_Y"'] \arrow[d, "\mathcal{P}_S(\varepsilon_{\langle Y,h\rangle})"] \\
& Y & \mathcal{P}_S Y \arrow[l, "h"]
\end{tikzcd}$$

Thus, in particular, the above function $\varepsilon_V : WV \to V$ is the value of the counit at the algebra $\mathcal{P}_S V = V$. (Formally, $\varepsilon_V = U\varepsilon_{(\mathcal{P}_S V = V)} = \varepsilon_{(\mathcal{P}_S V = V)}$, where U is the forgetful functor mapping algebras to their carriers.) By adjunction, there is a bijection (natural in X and $\langle Y, h\rangle$) between functions $f : X \to Y$ and $\mathcal{P}_S$-algebra arrows $g : \langle WX, \mathsf{inr}_X\rangle \to \langle Y, h\rangle$. This bijection maps f to

$$f^\sharp = \varepsilon_{\langle Y,h\rangle} \circ Wf$$

and g to

$$g^\flat = Ug \circ \eta_X = g \circ \eta_X$$

> **The Special Final Coalgebra Theorem.** Let F be a endofunctor on **SET** which cuts down to an endofunctor on the subcategory $\mathbf{SET}_\subset$ of inclusion functions.
>
> If F is uniform on maps, then, assuming anti-foundation, its final coalgebra
>
> $$\mathsf{gfp}[F] = F(\mathsf{gfp}[F])$$
>
> in $\mathbf{SET}_\subset$ lifts to a final F-coalgebra in **SET**.

Proof: Consider an F-coalgebra structure

$$k : X \to FX$$

By uniformity on maps, there exists a function $\rho_X : FX \to \mathcal{P}_S WX$, hence k can be made into a system of set-equations in X by composing it with ρ_X. Take its solution $\overline{\rho_X \circ k} : WX \to V$ and define a function f from X to V as the right adjunct of this solution wrt the above adjunction; that is,

$$f = (\overline{\rho_X \circ k})^\flat = \overline{\rho_X \circ k} \circ \eta_X : X \to V$$

Diagrammatically:

$$\begin{tikzcd}
X \arrow[r, "\eta_X"'] \arrow[d, "k"'] \arrow[rr, bend left, "f = (\overline{\rho_X \circ k})^\flat"] & WX \arrow[r, dashed, "\overline{\rho_X \circ k}"'] \arrow[d, "{[\rho_X \circ k, \mathsf{id}]}"'] & V \arrow[d, equal] \\
FX \arrow[r, "\rho_X"'] & \mathcal{P}_S WX \arrow[r, "\mathcal{P}_S(\overline{\rho_X \circ k})"'] & \mathcal{P}_S V
\end{tikzcd}$$

The claim is that, under the above hypotheses, f is an F-coalgebra arrow from $\langle X, k \rangle$ to $\mathsf{gfp}[F] = F(\mathsf{gfp}[F])$, that is, the diagram

$$\begin{tikzcd}
X \arrow[r, "f"] \arrow[d, "k"'] & \mathsf{gfp}[F] \arrow[d, equal] \\
FX \arrow[r, "Ff"'] & F(\mathsf{gfp}[F])
\end{tikzcd}$$

commutes. More precisely: Let Y be the image under f of X. The function $f : X \to V$ can be factorized, like every function in **SET**, as

$$X \overset{f}{\twoheadrightarrow} Y \hookrightarrow V$$

The claim is then as follows.

> The class Y is a post-fixed point for F, ie $Y \subseteq FY$, and f is a coalgebra arrow from $\langle X, k \rangle$ to $Y \hookrightarrow FY$, ie
>
> $$\begin{tikzcd}
X \arrow[r, two heads, "f"] \arrow[d, "k"'] & Y \arrow[d, hook] \\
FX \arrow[r, "Ff"'] & FY
\end{tikzcd}$$
>
> commutes.

If the above holds, since F cuts down to an endofunctor on the subcategory $\mathbf{SET}_{\subset}$ of inclusions, the composition of f the inclusion $Y \hookrightarrow \mathsf{gfp}[F]$ of Y into the greatest fixed point of F is an F-coalgebra arrow:

$$\begin{tikzcd}
X \arrow[r, two heads, "f"] \arrow[d, "k"'] & Y \arrow[r, hook] \arrow[d, hook] & \mathsf{gfp}[F] \arrow[d, equal] \\
FX \arrow[r, "Ff"'] & FY \arrow[r, hook] & F(\mathsf{gfp}[F])
\end{tikzcd}$$

In order to prove the above claim, notice that everything in sight in the following diagram commutes.

$$\begin{tikzcd}
X \arrow[r, "\eta_X"'] \arrow[rrr, bend left, "f"] \arrow[dd, "k"'] & WX \arrow[rr, "f^\sharp"] \arrow[dr, "Wf"] \arrow[dd, "{[\rho_X \circ k, \mathsf{id}]}"'] & & V \arrow[dd, equal] \\
& & WV \arrow[ur, "\varepsilon_V"'] & \\
FX \arrow[r, "\rho_X"] \arrow[dr, "Ff"'] & \mathcal{P}_S WX \arrow[rr, "\mathcal{P}_S(f^\sharp)"] \arrow[dr, "\mathcal{P}_S Wf"'] & & \mathcal{P}_S V \\
& FV \arrow[r, "\rho_V"'] \arrow[uuurr, hook, bend right] & \mathcal{P}_S WV \arrow[ur, "\mathcal{P}_S \varepsilon_V"'] &
\end{tikzcd}$$

In particular, the outer diagram does commute, hence:

$$\begin{tikzcd}
X \arrow[r, two heads, "f"] \arrow[dd, "k"'] & Y \arrow[dr, hook] & \\
& & V \\
FX \arrow[r, "Ff"'] & FY \arrow[ur, hook] &
\end{tikzcd}$$

Therefore, for all $x \in X$,

$$f(x) = (Ff \circ k)(x)$$

which implies that the image Y of X under f is included in the image of FX under Ff, hence

$$Y \subseteq FY$$

and f is a coalgebra arrow from $\langle X, k\rangle$ to $Y \hookrightarrow FY$.

Therefore, for every F-coalgebra $\langle X, k\rangle$, there exists a coalgebra arrow to $\mathsf{gfp}[F] = F(\mathsf{gfp}[F])$. Moreover, this arrows is unique. Indeed, the above arguments also show that every coalgebra arrow from $\langle X, k\rangle$ to $\mathsf{gfp}[F] = F(\mathsf{gfp}[F])$ fits as the right adjunct $(\overline{\rho_X \circ k})^\flat$ of the unique solution of a system of set-equations, hence it is unique. This concludes the proof.

Notes. An alternative (but more restrictive) form of the special final coalgebra theorem in the standard category of ordinary sets is presented in [Pau95].

The special final coalgebra theorem is the 'dual' of the standard fact that least (strict) fixed points of most endofunctors on **SET** are initial algebras. (Cf [Acz88, Theorem 7.6].) It gives an elementary way of finding final coalgebras, at the price of assuming anti-foundation. For instance, under foundation, the endofunctor $BX = \mathsf{Act} \times X$ has the empty set 0 as the unique fixed point, while, under anti-foundation, the empty set is the least fixed point and the set Act^ω of infinite words over the alphabet Act is the greatest fixed point of B: the special final coalgebra theorem tells then that $\mathsf{Act}^\omega = \mathsf{Act} \times \mathsf{Act}^\omega$ is a final B-coalgebra.

Notice that one can prove the (non-strict!) fixed point $\mathsf{Act}^\omega \cong \mathsf{Act} \times \mathsf{Act}^\omega$ is a final B-coalgebra in **Set**, *independently* of the use of anti-foundation. In general, as shown in [AM89], endofunctors to which the special final coalgebra theorem applies always have a final coalgebra in the category of ordinary (possibly large) sets. Thus, unless one is really interested in strict fixed points $\widehat{B} = B\widehat{B}$ rather than fixed points up to isomorphism $\widehat{B} \cong B\widehat{B}$, the interest can be shifted from non-well-founded sets and greatest fixed points to ordinary sets and final coalgebras.

Bibliography

[Abr90] Samson Abramsky. The lazy lambda calculus. In D.A. Turner, editor, *Research Topics in Functional Programming*, pages 65–116. Addison-Wesley, 1990.

[Acz80] Peter Aczel. Frege structures and the notions of proposition, truth and set. In J. Barwise et al., editors, *The Kleene Symposium*, pages 31–60. North-Holland, 1980.

[Acz88] Peter Aczel. *Non-well-founded sets.* Number 14 in Lecture Notes. CSLI, 1988.

[Acz94] Peter Aczel. Final universes of processes. In *Mathematical Foundations of Programming Semantics, Proc. 9th Int. Conf., New Orleans, LA, USA, April 1993*, volume 802 of *LNCS*, pages 1–28. Springer-Verlag, 1994.

[AM80] M.A. Arbib and E.G. Manes. The greatest fixpoint approach to data types. In *Proc. of the 3rd workshop meeting on Categorical and Algebraic Methods in Computer Science and System Theory*, 1980. See also [AM82].

[AM82] M.A. Arbib and E.G. Manes. Parametrized data types do not need highly constrained parameters. *Information and Control*, 52:139–158, 1982.

[AM89] P. Aczel and N. Mendler. A final coalgebra theorem. In D.H. Pitt et al., editors, *Proc. category theory and computer science*, volume 389 of *LNCS*, pages 357–365. Springer-Verlag, 1989.

[AR89] P.H.M. America and J.J.M.M. Rutten. Solving reflexive domain equations in a category of complete metric spaces. *Journal of Computer and System Sciences*, 39(3):343–375, 1989.

[Bad87] Eric Badouel. Une construction systématique de modéles à partir de spécifications opérationnelles structurelles. Technical Report INRIA 764, IRISA, Rennes, 1987.

[Bal94] M. Baldamus. A non-well-founded sets semantics for observation congruence over full CCS. Tech. rep. Berlin University of Technology, 1994.

[Bar92] Michael Barr. Algebraically compact functors. *Journal of Pure and Applied Algebra*, 82:211–231, 1992.

[Bar93] Michael Barr. Terminal coalgebras in well-founded set theory. *Theoretical Computer Science*, 144(2):299–315, 1993.

[BB92] G. Berry and G. Boudol. The Chemical Abstract Machine. *Theoretical Computer Science*, 96:217–248, 1992.

[BE87] J. Barwise and J. Etchemendy. *The Liar: An Essay in Truth and Circularity.* Oxford University Press, 1987.

[Bec69] Jon Beck. Distributive laws. In B. Eckmann, editor, *Seminar on Triples and Categorical Homology Theory*, volume 80 of *Lecture Notes in Mathematics*, pages 119–140. Springer-Verlag, 1969.

[BG92] S. Brookes and S. Geva. Computational comonads and intensional semantics. In M.P. Fourman, P.T. Johnstone, and A.M. Pitts, editors, *Applications of categories in computer science*, volume 177 of *London Mathematical Society Lecture Note Series.* Cambridge University Press, 1992.

[BIM88] B. Bloom, S. Istrail, and A.R. Meyer. Bisimulation can't be traced: preliminary report. In *Proc. Third IEEE Symp. on Logic In Computer Science*, pages 229–239, 1988.

[BR92] J.W. de Bakker and J. Rutten, editors. *Ten Years of Concurrency Semantics.* World Scientific, 1992. Selected papers of the Amsterdam Concurrency Group.

[BV96] J.W. de Bakker and E. de Vink. *Control Flow Semantics.* The MIT Press, 1996.

[BW85] M. Barr and C. Wells. *Toposes, Triples and Theories.* Springer-Verlag, 1985.

[BW90] J.C.M. Baeten and W.P. Weijland. *Process Algebra.* Number 18 in Cambridge Tracts in Theoretical Computer science. Cambridge University Press, 1990.

[BZ82] J.W. de Bakker and J.I. Zucker. Processes and the denotational semantics of concurrency. *Information and Control*, 54:70–120, 1982.

[Cro93] R.L. Crole. *Categories for Types.* Cambridge Mathematical Textbooks. Cambridge Univ. Press, 1993.

[DG87] Ph. Darondeau and B. Gamatié. A fully observational model for infinite behaviours of communicating systems. In *Proc. TAPSOFT/CAAP'87*, volume 249 of *LNCS*, pages 153–168. Springer-Verlag, 1987. See also [DG90].

[DG90] Ph. Darondeau and B. Gamatié. Infinitary behaviours and infinitary observations. *Fundamenta Informaticae*, XIII:353–386, 1990.

[dS85] R. de Simone. Higher level synchronising devices in MEIJE-SCCS. *Theoretical Computer Science*, 37:245–267, 1985.

[FH83] M. Forti and F. Honsell. Set theory with free construction principles. *Annali Scuola Normale Superiore, Pisa*, X(3):493–522, 1983.

[Fio93] Marcelo Fiore. A coinduction principle for recursive data types based on bisimulation. In *Proc. Eighth IEEE Symp. on Logic In Computer Science*, 1993. To appear in *Information and Computation.*

[Fio96] Marcelo Fiore. Axiomatic domain theory in categories of partial maps. To be published by Cambridge University Press in the Distinguished Dissertations Series. (Ph.D. thesis, 1994. Technical report ECS-LFCS-94-307, Department of Computer Science, University of Edinburgh), 1996.

[FP92] M.P. Fiore and G.D. Plotkin. On compactness and **Cpo**-enriched categories. In G. Winskel, editor, *Proceedings of the CLICS Workshop (23-27 March 1992)*, volume 397-II of *DAIMI PB*, pages 571–584. Computer Science Department, Aarhus University, May 1992.

[FP94] M.P. Fiore and G.D. Plotkin. An axiomatisation of computationally adequate domain-theoretic models of FPC. In *9*$^{\text{th}}$ *LICS Conf.*, pages 92–102. IEEE, Computer Society Press, 1994.

[Fre90] Peter Freyd. Recursive types reduced to inductive types. In *Proc. Fifth IEEE Symp. on Logic In Computer Science*, pages 498–507. IEEE Computer Society Press, 1990.

[Fre91] Peter Freyd. Algebraically complete categories. In A. Carboni, M.C. Pedicchio, and G. Rosolini, editors, *Category Theory - Proc. of the Int'l Conf. held in Como, Italy, July 1990*, volume 1488 of *Lecture Notes in Mathematics*, pages 95–104. Springer-Verlag, 1991.

[Fre92] Peter Freyd. Remarks on algebraically compact categories. In H.P. Fourman, P.T. Johnstone, and A.M. Pitts, editors, *Applications of Category Theory in computer science*, volume 177 of *London Mathematical Society Lecture Notes Series*, pages 95–106. Cambridge University Press, 1992.

[FS90] Peter Freyd and Andre Scedrov. *Categories, Allegories.* North-Holland, 1990.

[Gro93] J.F. Groote. Transition system specifications with negative premises. *Theoretical Computer Science*, 118(2):263–299, 1993.

[GTW78] J.A. Goguen, J.W. Thatcher, and E.G. Wagner. An initial algebra approach to the specification, correctness and implementation of abstract data types. In R.T. Yeh, editor, *Current Trends in Programming Methodology*, volume IV, pages 80–149. Prentice Hall, 1978.

[GV92] J.F. Groote and F. Vaandrager. Structured operational semantics and bisimulation as a congruence. *Information and Computation*, 100(2):202–260, 1992.

[Har96] Chrysafis Hartonas. Semantics for finite delay. To appear in *Theoretical Computer Science*, 1996.

[Her93] C. Hermida. *Fibrations, Logical Predicates and Indeterminates.* PhD thesis, Univ. Edinburgh, 1993. Techn. rep. LFCS-93-277. Also available as Aarhus Univ. DAIMI Techn. rep. PB-462.

[Hes88] W.H. Hesselink. Deadlock and fairness in morphisms of transition systems. *Theoretical Computer Science*, 59:235–257, 1988.

[HJ95a] C. Hermida and B. Jacobs. An algebraic view of structural induction. In L. Pacholski and J. Tiuryn, editors, *Computer Science Logic 1994*, volume 933 of *LNCS*, pages 412–426. Springer-Verlag, 1995.

[HJ95b] C. Hermida and B. Jacobs. Induction and coinduction via subset types and quotient types. In *Informal Proceedings of the Joint CLICS-TYPES Workshop on Categories and Type Theory.* Prog. Meth. Group, Report 85, Göteborg Univ. and Chalmers Univ. of Techn., 1995.

[HL95] F. Honsell and M. Lenisa. Final semantics for untyped λ-calculus. In M. Dezani-Ciancaglini and G.D. Plotkin, editors, *Typed Lambda calculi and applications : second international conference*, volume 902 of *LNCS*. Springer-Verlag, 1995.

[HP79] M.C.B. Hennessy and G.D. Plotkin. Full abstraction for a simple parallel programming language. In J. Bečvář, editor, *Proc. 8th Int'l Symp. on Mathematical Foundations of Computer Science*, volume 74 of *LNCS*, pages 108–120. Springer-Verlag, 1979.

[Jac95] Bart Jacobs. Mongruences and cofree coalgebras. In V.S. Alagar and M. Nivat, editors, *Algebraic Methods and Software Technology*, volume 936 of *LNCS*, pages 245–260. Springer-Verlag, 1995.

[JM95] A. Joyal and I. Moerdijk. *Algebraic Set Theory*, volume 220 of *London Mathematical Society Lecture Note Series.* Cambridge University Press, 1995.

[JNW93] A. Joyal, M. Nielsen, and G. Winskel. Bisimulation and open maps. In *Proc. Eighth IEEE Symp. on Logic In Computer Science*, pages 418,427, 1993.

[Joh75] P.T. Johnstone. Adjoint lifting theorems for categories of algebras. *Bull. London Math. Soc.*, 7:294–297, 1975.

[Ken87] R.E. Kent. The metric closure powerspace construction. In M. Main et al., editors, *Mathematical Foundations of Programming Semantics, Proc. 3rd Int. Conf.*, volume 298 of *LNCS*, pages 173–199. Springer-Verlag, 1987.

[Klo80] Jan Willem Klop. *Combinatory Reduction Systems*, volume 127 of *Mathematical Centre Tracts.* CWI, Amsterdam, 1980. PhD Thesis.

[KR90] J.N. Kok and J.J.M.M. Rutten. Contractions in comparing concurrency semantics. *Theoretical Computer Science*, 76:179–122, 1990.

[Law69] F. William Lawvere. Adjointness in foundation. *Dialectica*, 23(3/4):281–296, 1969.

[Law76] F. William Lawvere. Variable quantities and variable structure in topoi. In A. Heller and M. Tierney, editors, *Algebra, Topology, and Category Theory*, pages 101–131. Academic Press, 1976.

[Lev79] Azriel Levy. *Basic Set Theory.* Perspectives in Mathematical Logic. Springer-Verlag, 1979.

[LS86] J. Lambek and P.J. Scott. *Introduction to higher order categorical logic*, volume 7 of *Cambridge studies in advanced mathematics.* Cambridge University Press, 1986.

[Mac71] Saunders Mac Lane. *Categories for the Working Mathematician*, volume 5 of *Graduate Texts in Mathematics.* Springer-Verlag, 1971.

[Mac86] Saunders Mac Lane. *Mathematics: Form and Function.* Springer-Verlag, 1986.

[Man76] E.G. Manes. *Algebraic Theories*, volume 26 of *Graduate Texts in Mathematics.* Springer-Verlag, 1976.

[Mei92] Karl Meinke. Universal algebra in higher types. *Theoretical Computer Science*, 100:385–417, 1992.

[MG85] J. Meseguer and J.A. Goguen. Initiality, induction, and computability. In M. Nivat and J.C. Reynolds, editors, *Algebraic Methods in Semantics*, pages 459–541. Cambridge University Press, 1985.

[Mil80] R. Milner. *A Calculus of Communicating Systems*, volume 92 of *LNCS*. Springer-Verlag, 1980.

[Mil90] R. Milner. Functions as processes. In M.S. Paterson, editor, *Proc. of 17th ICALP*, 1990.

[Mog91] Eugenio Moggi. Notions of computation and monads. *Information and Computation*, 93:55–92, 1991.

[Mos90] Peter D. Mosses. Denotational semantics. In J. van Leeuwen, editor, *Handbook of Theoretical Computer Science*, volume B: Formal Models and Semantics, pages 575–631. Elsevier, Amsterdam and The MIT Press, Cambridge, 1990.

[MT92] K. Meinke and J.V. Tucker. Universal algebra. In S. Abramsky et al., editors, *Handbook of logic in computer science*, volume 1, pages 189–411. Clarendon Press, Oxford, 1992.

[Muk91] Kuniaki Mukai. *Constraint Logic Programming and the Unification of Information.* PhD thesis, Tokio Institute of Technology, April 1991.

[Niv79] M. Nivat. Infinite words, infinite trees, infinite computations. In J.W. de Bakker and J. van Leeuwen, editors, *Foundations of Computer Science III, Part 2*, volume 109 of *Math. Centre Tracts*, pages 3–52, 1979.

[Ole82] Frank J. Oles. *A category-theoretic approach to the semantics of programming languages.* PhD thesis, School of Computer and Information Science, Syracuse University, August 1982.

[Par81] D.M.R. Park. Concurrency and automata on infinite sequences. In P. Deussen, editor, *Proceedings 5th GI Conference*, volume 104 of *LNCS*, pages 167–183. Springer-Verlag, 1981.

[Pau95] Lawrence C. Paulson. A concrete final coalgebra theorem for ZF set theory. In P. Dybjer et al., editors, *Types for Proofs and Programs '94*, volume 996 of *LNCS*, pages 120–139. Springer-Verlag, 1995.

[Pit93] Andrew Pitts. Relational properties of domains. Technical Report 321, Cambridge Univ. Computer Laboratory, December 1993. To appear in *Information and Computation.*

[Pit94a] Andrew Pitts. A co-induction principle for recursively defined domains. *Theoretical Computer Science*, 124:195–219, 1994.

[Pit94b] Andrew Pitts. Computational adequacy via 'mixed' inductive definitions. In *Mathematical Foundations of Programming Semantics, Proc. 9th Int. Conf., New Orleans, LA, USA, April 1993*, volume 802 of *LNCS*, pages 72–82. Springer-Verlag, 1994.

[Pit94c] Andrew Pitts. Some notes on inductive and co-inductive techniques in the semantics of functional programs. Notes Series BRICS-NS-94-5, BRICS, Department of Computer Science, University of Aarhus, December 1994.

[Plo76] Gordon Plotkin. A powerdomain construction. *SIAM J. Comput.*, 5:452–487, 1976.

[Plo81a] Gordon Plotkin. Post-graduate lecture notes in advanced domain theory (incorporating the "Pisa Notes"). Department of Computer Science, Univ. of Edinburgh, 1981.

[Plo81b] Gordon Plotkin. A structured approach to operational semantics. Technical Report DAIMI FN-19, Computer Science Department, Aarhus University, 1981.

[Plo85] Gordon Plotkin. Lectures on predomains and partial functions. CSLI, 1985.

[Plo90] Gordon Plotkin. An illative theory of relations. In R Cooper et al., editors, *Situation Theory and its Applications*, number 22 in CSLI Lecture Notes, pages 133–146. Stanford University, 1990.

[Poi92] Axel Poigné. Basic category theory. In S. Abramsky et al., editors, *Handbook of logic in computer science*, volume 1, pages 413–640. Clarendon Press, Oxford, 1992.

[RT93] J. Rutten and D. Turi. On the foundations of final semantics: non-standard sets, metric spaces, partial orders. In J. de Bakker et al., editors, *Proc. of the REX workshop Semantics – Foundations and Applications*, volume 666 of *LNCS*, pages 477–530. Springer-Verlag, 1993.

[RT94] J. Rutten and D. Turi. Initial algebra and final coalgebra semantics for concurrency. In J. de Bakker et al., editors, *Proc. of the REX workshop A Decade of Concurrency – Reflections and Perspectives*, volume 803 of *LNCS*, pages 530–582. Springer-Verlag, 1994.

[Rut90] J. Rutten. Deriving denotational models for bisimulation from Structured Operational Semantics. In M. Broy and C.B. Jones, editors, *Programming concepts and methods, proceedings of the IFIP Working Group 2.2/2.3 Working Conference*, pages 155–177. North-Holland, 1990.

[Rut92] Jan Rutten. Processes as terms: non-well-founded models for bisimulation. *Mathematical Structures in Computer Science*, 2:257–275, 1992.

[San95] Davide Sangiorgi. On the proof method for bisimulation. In J. Wiedermann and P. Háiek, editors, *Proc. MFCS'95*, volume 969 of *LNCS*, pages 479–488. Springer-Verlag, 1995.

[Sco70] Dana Scott. Outline of a mathematical theory of computation. In *Proc. 4th Annual Princeton Conference on Inf. Sciences and Systems*, pages 169–176, 1970.

[Sco80] Dana Scott. Relating theories of the λ-calculus. In J.R. Hindley and J.P. Seldin, editors, *To H.B. Curry: Essays on combinatory logic, lambda calculus and formalism*, pages 403–450. Academic Press, 1980.

[Sim95] Alex Simpson. Compositionality via cut-elimination: Hennessy-Milner logic for an arbitrary GSOS. In *Proc. Tenth IEEE Symp. on Logic In Computer Science*, 1995.

[SP82] M. Smyth and G. Plotkin. The category-theoretic solution of recursive domain equations. *SIAM J. Comput.*, 11:761–783, 1982.

[TJ93] D. Turi and B. Jacobs. On final semantics for applicative and non-deterministic languages. Fifth Biennal Meeting on Category Theory and Computer Science, Amsterdam, September 1993.

[Win93] Glynn Winskel. *The Formal Semantics of Programming Languages.* The MIT Press, 1993.

[WN95] G. Winskel and M. Nielsen. Models for concurrency. In S. Abramsky et al., editors, *Handbook of logic in computer science*, volume 4. Clarendon Press, Oxford, 1995.

Index