Categorical Modelling of Structural Operational Rules

Case Studies

Daniele Turi
<dt@dcs.ed.ac.uk>

Department of Computer Science
Laboratory for Foundations of Computer Science
University of Edinburgh, The King's Buildings
Edinburgh EH9 3JZ, Scotland

To appear in *CTCS'97* proceedings, LNCS, Springer-Verlag.

Abstract. This paper aims at substantiating a recently introduced categorical theory of 'well-behaved' operational semantics. A variety of concrete examples of structural operational rules is modelled categorically illustrating the versatility and modularity of the theory. Further, a novel functorial notion of guardedness is introduced which allows for a general and formal treatment of guarded recursive programs.

Introduction

The predominant approach to operational semantics is Plotkin's *SOS* [13], which is based on structural rules. One finds in the literature various formats of structural rules which guarantee a good behaviour such as having adequate denotational models and behavioural equivalence (eg bisimulation) being a congruence.

In [17], it is shown that the rules in the best known of these formats, namely *GSOS* [5], are in 1-1 correspondence with natural transformations of a suitable type, depending on specific *functorial* notions of syntax and behaviour. This led to studying abstract rules as natural transformations parametric in the syntax and behaviour, and to a corresponding general theory which elegantly explains the good behaviour of the (abstract) rules in terms of *distributive laws* of monads over comonads.

The present study aims at substantiating the generality of the above theory. One way to do this would be to find formats other than GSOS which fit in the theory. But here a different direction is taken in order to illustrate a key feature of the theory: *modularity.* A few elementary examples of rules such as those for sequential and for parallel composition of programs are considered and the parametricity of the theory is exploited in order to distil abstract rules,

⋆ Research supported by *EuroFOCS* and the European Union *TMR* programme.

parametric in the behaviour, which – by retaining only the essential, intrinsic structure – can be instantiated to various situations.

For instance, it is shown how, by varying the choice of the behaviour, the same rules can be interpreted both deterministically and non-deterministically, both in uniform (ie with atomic actions) and non-uniform (ie with side-effects) settings, both with respect to bisimulation and to trace equivalence. Moreover the same rules are interpretable in several categories, such as categories of sets, cpos, metric spaces, semi-lattices, etc. All this while retaining the desirable properties typical of GSOS.

Languages without variable binding, but possibly multi-sorted, can be described as Σ-algebras [6]. This gives rise to a well-known functorial notion of syntax, parametric in the language constructs [2]. It is less clear what a general functorial notion of behaviour is. Here some examples are given; interestingly, most of them are *computational monads* in the sense of [12], but only the fact that the functors are strong is relevant.

Further, a novel functorial notion of *guard*, again parametric in syntax and behaviour, is introduced. This strongly generalizes the specific and somewhat informal notion of guardedness as used for defining recursive programs in languages à la *CCS* [10]. Unguarded recursion also fits nicely in this framework, interpreting the rules in Kleisli categories of lifting monads; but for lack of space this is left for a future presentation.

Contents. After a first section of preliminaries, Section 2 recalls the definition of abstract operational rules from [17], while the next four sections illustrate how a wide range of examples of structural rules can be modelled in such an abstract form. Following [17], Section 7 shows how both an operational and a denotational model can be canonically derived from the abstract rules; a simple process algebra is used as example and interpreted with respect to both bisimulation and trace equivalence. The last section is devoted to guarded recursion.

Thanks to: Gordon Plotkin for joint foundational work; Marcelo Fiore and Alex Simpson for discussions. Part of this work is based on the author's thesis; he wishes to thank Jaco de Bakker and Bart Jacobs for their guidance. Diagrams drawn using *XFig*.

1 Preliminaries and Notation

A non-standard notation first: given two maps $f : X \to Y$ and $g : Y \to Z$ in a category, the expression

$$\frac{\begin{array}{l} f : X \longrightarrow Y \\ g : \quad\;\; \longrightarrow Z \end{array}}{h : X \longrightarrow Z}$$

says that the map $h : X \to Z$ is defined to be the composite $g \circ f$. In general, the objects of a category are ranged over by X and Y. In Set, X itself is ranged over by x, y, and z.

Initial and final (ie terminal) objects are written as 0 and 1 respectively and the unique arrow from an object X to 1 is denoted by $! : X \to 1$. The **power** X^A denotes the product of $|A|$-copies of X. The i-th injection (projection) of an n-ary coproduct (product) is denoted by ι_i (π_i). For every set A, the **copower** $A \cdot X$ denotes the coproduct of $|A|$-copies of X, hence it comes with an injection $\iota_a : X \to A \cdot X$ for each a in A; if X is a set, $\iota_a(x)$ is more suggestively written as $a \cdot x$. In Set, the expression $a \cdot x$ can also be understood as the pair $\langle a, x\rangle$ since the copower $A \cdot X$ is isomorphic to the product $A \times X$; also, since the power X^A is isomorphic (in Set) to the set $\mathsf{Set}(A, X)$ of functions of type $A \to X$, the 'thick lambda' notation $\boldsymbol{\lambda} a.x$ is used to denote its elements.

The associativity, symmetry, and unit natural isomorphisms of a symmetric monoidal category (**SMC**) $\langle \mathcal{C}, \otimes, I\rangle$ are denoted by α, γ, and λ respectively. (The latter is often omitted.) The symbol $X^{\otimes n}$ denotes the n-fold monoidal power of X.

All categories in this paper are SMCs with finite products and coproducts and with tensors $\otimes$ distributing over coproducts; the distributivity natural isomorphism is denoted by δ. Two models are used, namely Set and the category $SL(\mathsf{Set})$ of semi-lattices and join-preserving functions, but other categories used in semantics, such as categories of cpos and of metric spaces, could have been chosen. In Set $\otimes$ and $\times$ coincide, while in $SL(\mathsf{Set})$ they are distinct ($\otimes$ is the classifier of bilinear functions). The categories $SL(\mathcal{C})$ considered here are **affine** SMCs, that is, the unit I of the tensor is the final object 1.

In order to ease the notation, natural transformations are written without subscript in this paper. For instance:

$$\lambda : I \otimes X \to X \cong X \qquad \text{instead of} \qquad \lambda_X : I \otimes X \to X \cong X \ .$$

Recall that a functor F between SMC's is **strong** [9] if there exists a **strength** natural transformation $st : X \otimes FY \to F(X \otimes Y)$ satisfying the unit and the associativity condition of [9]. For simplicity, the same symbol denotes also the map $F\gamma \circ st \circ \gamma : FX \otimes Y \to F(X \otimes Y)$. If F is a monad, there are two canonical **double strength** maps of type $FX \otimes FY \to F(X \otimes Y)$ (which are equal if the monad is commutative); here they are both denoted by dst. Finally, strong functors compose: $\langle F, st\rangle \circ \langle F', st'\rangle = \langle FF', F(st') \circ st\rangle$.

Semi-Lattices. Let $\mathcal{C}$ be a cartesian category. A **semi-lattice** in $\mathcal{C}$ is a pair $\langle X, \vee\rangle$ such that the 'carrier' X is an object of $\mathcal{C}$ and the 'join'

$$\vee : X \times X \to X$$

is a map of $\mathcal{C}$ satisfying the following three axioms.

$$\vee \circ (id \times \vee) = \vee \circ (\vee \times id) \circ \alpha \ , \qquad \vee = \vee \circ \gamma \ , \qquad \vee \circ \langle id, id\rangle = id \ .$$

If $\mathcal{C}$ is Set, then these axioms read as $x \vee (y \vee z) = (x \vee y) \vee z$, $x \vee y = y \vee x$, and $x \vee x = x$ respectively, for all x, y, z in X.

The category $SL(\mathcal{C})$ has semi-lattices in $\mathcal{C}$ as objects; the maps are the evident join-preserving maps of $\mathcal{C}$, also called linear maps.

Semi-lattices are used here to model non-deterministic programs, binary (hence finite, non-empty) joins modelling binary (and finite, non-empty) choices. Several variations can be considered according to the cardinality of the non-determinism available in the language under study. For instance might consider semi-lattices with a neutral element

$$e : 1 \to X$$

such that $id \cong \vee \circ (id \times e)$, which in Set reads as $x \vee \bot = x$. The corresponding category of finite joins semi-lattices in $\mathcal{C}$ is denoted by $SL_*(\mathcal{C})$.

Given a cartesian category $\mathcal{C}$, the corresponding **free semi-lattice monad** $\mathcal{P}_f$ exists if the forgetful functor $\langle X, \vee \rangle \mapsto X$ mapping a semi-lattice to its carrier has a left adjoint. Similarly, $\mathcal{P}_{fi}$ denotes the monad corresponding to $SL_*(\mathcal{C})$. In the sequel, it is assumed that the forgetful functor of semi-lattices is monadic and that both $\mathcal{P}_f$ and $\mathcal{P}_{fi}$ are commutative.

In Set, the monads $\mathcal{P}_f$ and $\mathcal{P}_{fi}$ are the (covariant) non-empty finite power-set and finite power-set respectively; for both, the unit is the function $x \mapsto \{x\}$ and the multiplication is the 'big union' function. Also, if $\mathcal{C}$ is the category of ω-algebraic cppos and continuous functions, then $\mathcal{P}_f$ is the Plotkin powerdomain. All these are examples of commutative monads. In Set, the double strength $dst : \mathcal{P}_f X \times \mathcal{P}_f Y \to \mathcal{P}_f(X \times Y)$ of $\mathcal{P}_f$ is $\langle \{x_i\}_I, \{y_j\}_J \rangle \mapsto \{\langle x_i, y_j \rangle\}_{I,J}$.

Algebras and Coalgebras. Given any endofunctor F on $\mathcal{C}$ (ie $F : \mathcal{C} \to \mathcal{C}$), an F**-algebra** is a pair $\langle X, h \rangle$, where the carrier X is an object and the structure $h : FX \to X$ is a map of $\mathcal{C}$. One often identifies an algebra $\langle X, h \rangle$ with its structure h. The F-algebras form a category $F\text{-}\mathsf{Alg}$, with homomorphisms $f : \langle X, h \rangle \to \langle X', h' \rangle$ the maps $f : X \to X'$ between the carriers such that $f \circ h = h' \circ (Ff)$. The carrier of the initial algebra (if it exists) is denoted by $\mu X.FX$; by Lambek's Lemma (see, eg, [15, Lemma 1]), its structure is an isomorphism.

Consider the evident forgetful functor $U^F : F\text{-}\mathsf{Alg} \to \mathcal{C}$: one can easily prove that if it has a left adjoint, then it is monadic. The corresponding monad T is called the **monad freely generated by** F. If $\mathcal{C}$ has finite coproducts, then $TX = \mu Y.X + FY$ for every object X of $\mathcal{C}$. A clash of notation: the same symbol μ is used for the initial algebra constructor and for the multiplication $\mu : T^2 \to T$ of the monad T.

Dually, one has a category $F\text{-}\mathsf{Coalg}$ of F**-coalgebras** $k : X \to FX$ with homomorphisms $f : \langle X, k \rangle \to \langle X', k' \rangle$ maps $f : X \to X'$ between the carriers such that $k' \circ f = (Ff) \circ k$. We write $\nu X.FX$ for the carrier of the final coalgebra and $\varphi : \nu X.FX \cong F(\nu X.FX)$ for its structure which, by the dual of Lambek's Lemma, is an isomorphism.

A Functorial Notion of Syntax. Consider the language with signature Σ consisting of a distinguished constant symbol 'nil', a set A of constant symbols ranged over by a, and two binary operators ' ; ' and '$\|$'. This signature freely generates, for every set X of variables ranged over by x, the set of terms t given by the following abstract grammar:

$$t ::= x \mid \mathsf{nil} \mid a \mid t \,;\, t \mid t \parallel t \quad . \tag{1}$$

One can model the above signature in any monoidal category with coproducts $\langle \mathcal{C}, \otimes, I \rangle$ as an endofunctor with the same name:

$$\Sigma X = I + A \cdot I + X \otimes X + X \otimes X \ . \tag{2}$$

For $\langle \mathcal{C}, \otimes, I \rangle = \langle \mathsf{Set}, \times, 1 \rangle$, one has that $TX = \mu Y.X + \Sigma Y$ is the above set of terms.

In general, for any signature Σ, one can define an endofunctor

$$\Sigma X = \coprod_{\phi \in \Sigma} X^{\otimes arity(\phi)} \tag{3}$$

on $\langle \mathcal{C}, \otimes, I \rangle$. (Cf [2, §10] for the case $\mathcal{C}$ is cartesian.) In the sequel, $\mathcal{C}$ is assumed to be such that every such Σ freely generates a monad T (which $\mathcal{C}$ 'thinks' as the syntax of the language).

For every object X, the structure of the free Σ-algebra over X is denoted by:

$$\psi : \Sigma TX \to TX \ .$$

It is natural in X. Let ϕ by a binary operator of a signature Σ and let f and g be two maps of type $X \to TX$. The mixed syntax/semantics notation $\phi(f, g)$ is used here to denote the map $X \otimes X \xrightarrow{f \otimes g} TX \otimes TX \xrightarrow{\iota_\phi} \Sigma TX \xrightarrow{\psi} TX$. This generalizes to every operator of any arity.

The unit $\eta : X \to TX$ of the monad T is the insertion-of-variables map; in Set, this is the formal operation mapping a variable x in X to a term x in TX. Using the above notation, one can form the maps $\eta\,;\eta$ and $\eta \parallel \eta$ of type $X \otimes X \to TX$.

There is a similar functorial notion of syntax for multi-sorted signatures: it suffices to consider endofunctors on power categories $\mathcal{C}^n$, where n is the number of sorts. It remains a major open question whether also variable binding constructs can be modelled functorially.

A Functorial Notion of Behaviour. In *initial algebra semantics* [6] Σ-algebras are regarded as denotational models and the unique homomorphism from the initial algebra of programs to a denotational model is the canonical compositional interpretation relative to that model. Dually, in *final coalgebra semantics* [1,14] one looks for a 'behaviour' endofunctor B such that its coalgebras are operational models. In particular, the final B-coalgebra is a solution of the domain equation $X \cong BX$ – the canonical domain of interpretation for programs with behaviour of type B.

The typical example of a behaviour endofunctor is $BX = \mathcal{P}(A \cdot X)$ on Set. Its coalgebras are in 1-1 correspondence with relations of type $X \times A \times X$, ie with **labelled transition systems** [13]. Indeed, writing $x \overset{a}{\rightsquigarrow} x'$ for $\langle x, a, x' \rangle \in \rightsquigarrow$, one has:

$$\frac{\rightsquigarrow \subseteq X \times A \times X}{k \ : X \to \mathcal{P}_{\mathrm{fi}}(A \cdot X)} \qquad\qquad x \overset{a}{\rightsquigarrow} x' \iff a \cdot x' \in k(x) \tag{4}$$

Note that the following isomorphisms hold.

$$\mathcal{P}(A \cdot X) \;\cong\; \mathcal{P}(A \times X) \;\cong\; \mathsf{Set}(A, \mathcal{P}X) \;\cong\; (\mathcal{P}X)^A \;. \tag{5}$$

For cardinality reasons, $\mathcal{P}$ has no final coalgebra, therefore one usually replaces it by the finite power-set functor $\mathcal{P}_{\mathrm{fi}}$;the correspondence in (4) cuts down then to 'finitely branching' transition systems. The final coalgebra is then the set of rooted, finitely branching trees, with branches labelled by actions a in A, quotiented by strong bisimulation. (Cf [4].)

Here more examples are considered, taken from the following class of endofunctor, not only on Set but in any SMC with enough structure to define them; the definition is parametric in a set S of states or actions.

$$B ::= \mathit{Id} \mid 1 + B \mid S \cdot B \mid B^S \mid \mathcal{P}_{\mathrm{f}} B \mid \mathcal{P}_{\mathrm{fi}} B \tag{6}$$

This is by no means intended as an exhaustive list of behaviour endofunctors. For concrete behaviours, β is used to range over BX.

Convention: in the two models Set and $\mathit{SL}(\mathsf{Set})$, the initial algebra and final coalgebra isomorphisms as well as the monad operations η, μ and the coproduct injections ι_1, ι_2 are often omitted.

2 Abstract Operational Rules

In [17], it is shown that if the operational rules for a language with signature Σ and behaviour of type B can be modelled as a map

$$\rho : \Sigma(X \times BX) \longrightarrow BTX \tag{7}$$

natural in X, then the rules enjoy several desirable properties. Further, it is shown that for the specific behaviour endofunctor $BX = (\mathcal{P}_{\mathrm{fi}} X)^A$ on Set there is an (essentially) 1-1 correspondence between natural transformations as in (7) and 'image finite' sets of rules in *GSOS* format [5] (relative to a signature Σ and a finite set of actions A).

In the sequel, various examples of rules involving behaviours other than the one for GSOS are modelled as in (7). The endofunctor Σ is as in (3), therefore it suffices to model the rules for each operator ϕ separately as a map

$$[\![\phi]\!] : \underbrace{(X \times BX) \otimes \cdots \otimes (X \times BX)}_{\mathit{arity}(\phi)} \longrightarrow BTX$$

natural in X, and then define ρ as the coparing of such maps.

3 Exceptions and Side-Effects

As a first example, let us consider the rules for the sequential composition of programs involving a simple example of exceptions, namely explicit termination.

Furthermore, let us consider programs with side-effects, assuming a given set S of states ranged over by σ. The rules are:

$$\frac{\langle x,\, \sigma\rangle \rightsquigarrow \langle x',\, \sigma'\rangle}{\langle x \,;\, y,\, \sigma\rangle \rightsquigarrow \langle x' \,;\, y,\, \sigma'\rangle}\;, \qquad \frac{\langle x,\, \sigma\rangle \rightsquigarrow \sigma'}{\langle x \,;\, y,\, \sigma\rangle \rightsquigarrow \langle y,\, \sigma'\rangle}\;.$$

Let $\langle \mathcal{C}, \otimes, 1\rangle$ be an affine SMC, with finite products and coproducts, powers, and tensors distributing over coproducts. If the programs are deterministic, then one can use the endofunctor

$$BX = (S \cdot (1 + X))^S \tag{8}$$

on $\mathcal{C}$ as behaviour. Note that in Set the isomorphism $S \cdot (1 + X) \cong S + S \cdot X$ holds, but the expression on the left hand side can be interpreted also in categories where S is not an object (such as categories of semi-lattices and of pointed partial orders). Also, note that the endofunctor $X \mapsto (S \cdot X)^S$ is strong; in Set the corresponding strength $st : (S \cdot X)^S \times Y \to (S \cdot (X \times Y))^S$ is the map $\langle(\boldsymbol{\lambda}\sigma.(\sigma' \cdot x)), y\rangle \mapsto \boldsymbol{\lambda}\sigma.(\sigma' \cdot \langle x, y\rangle)$. Then sequential composition can be modelled as follows.

$$\begin{array}{cl}
\pi_2 \otimes \pi_1 & : (X \times (S \cdot (1+X))^S) \otimes (X \times (S \cdot (1+X))^S) \\
& \qquad\qquad\qquad \longrightarrow (S \cdot (1+X))^S \otimes X \\
st & : \qquad\qquad\qquad \longrightarrow (S \cdot ((1+X) \otimes X))^S \\
(S \cdot \delta)^S & : \qquad\qquad\qquad \cong \ (S \cdot (X + (X \otimes X))^S \\
(S \cdot [\eta, (\eta \,;\, \eta)])^S & : \qquad\qquad\qquad \longrightarrow (S \cdot TX)^S \\
(S \cdot \iota_2)^S & : \qquad\qquad\qquad \longrightarrow (S \cdot (1 + TX))^S \\
\hline
[\![\;;\;]\!] & : (X \times (S \cdot (1+X))^S) \otimes (X \times (S \cdot (1+X))^S) \\
& \qquad\qquad\qquad \longrightarrow (S \cdot (1 + TX))^S
\end{array}$$

Note that this map in natural because all its components are. The same holds for all the rules modelled in this paper.

In Set:

$$(x, \beta)[\![\;;\;]\!](y, \beta') = \boldsymbol{\lambda}\sigma.\begin{cases} \sigma' \cdot (x' \,;\, y) & \text{if } \beta(\sigma) = \sigma' \cdot x' \\ \sigma' \cdot y & \text{if } \beta(\sigma) = \sigma' \end{cases}\;.$$

4 Non-Determinism

Consider the following rules for the parallel composition operator of CCS [10], relative to a set A of actions ranged over by a, with a distinguished 'silent step' action τ, and a dual action $\overline{a}$ for every a in A different from τ.

$$\frac{x \stackrel{a}{\rightsquigarrow} x'}{x \parallel y \stackrel{a}{\rightsquigarrow} x' \parallel y}\;, \qquad \frac{y \stackrel{a}{\rightsquigarrow} y'}{x \parallel y \stackrel{a}{\rightsquigarrow} x \parallel y'}\;, \qquad \frac{x \stackrel{a}{\rightsquigarrow} x' \quad y \stackrel{\overline{a}}{\rightsquigarrow} y'}{x \parallel y \stackrel{\tau}{\rightsquigarrow} x' \parallel y'}\;. \tag{9}$$

One possible behaviour endofunctor for these rules is

$$BX = (\mathcal{P}_{\mathrm{fi}} X)^A \tag{10}$$

on a distributive category $\mathcal{C}$ with powers and a commutative free semi-lattice monad $\mathcal{P}_{\mathrm{fi}}$. Note that $BX = (\mathcal{P}_{\mathrm{fi}}X)^A$ inherits from $\mathcal{P}_{\mathrm{fi}}X$ a finite semi-lattice structure, say $\vee : (\mathcal{P}_{\mathrm{fi}}X)^A \times (\mathcal{P}_{\mathrm{fi}}X)^A \to (\mathcal{P}_{\mathrm{fi}}X)^A$ and $e : 1 \to (\mathcal{P}_{\mathrm{fi}}X)^A$, natural in X. Further, B is the composition of strong endofunctors, hence it is strong.

Let us model each rule separately and then join them using the semi-lattice structure of BTX. The first rule can be modelled as:

$$\begin{array}{rl}
\pi_2 \times \pi_1 & : (X \times (\mathcal{P}_{\mathrm{fi}}X)^A) \times (X \times (\mathcal{P}_{\mathrm{fi}}X)^A) \longrightarrow (\mathcal{P}_{\mathrm{fi}}X)^A \times X \\
st & : \qquad\qquad\qquad\qquad\qquad\qquad\qquad\quad \longrightarrow (\mathcal{P}_{\mathrm{fi}}(X \times X))^A \\
(\mathcal{P}_{\mathrm{fi}}(\eta \parallel \eta))^A & : \qquad\qquad\qquad\qquad\qquad\qquad\qquad\quad \longrightarrow (\mathcal{P}_{\mathrm{fi}}TX)^A \\
\hline
\textit{l-merge} & : (X \times (\mathcal{P}_{\mathrm{fi}}X)^A) \times (X \times (\mathcal{P}_{\mathrm{fi}}X)^A) \longrightarrow (\mathcal{P}_{\mathrm{fi}}TX)^A
\end{array}$$

The second rule is symmetric, hence it can be modelled as symply the composite $\textit{l-merge} \circ \gamma$. As for the third rule, let us denote its modelling by *synch*. Since the rule only contributes to the τ-component of $(\mathcal{P}_{\mathrm{fi}}TX)^A$, we can take $\pi_a \circ \textit{synch}$ to be the composite map $(X \times (\mathcal{P}_{\mathrm{fi}}X)^A) \times (X \times (\mathcal{P}_{\mathrm{fi}}X)^A) \xrightarrow{!} 1 \xrightarrow{e} \mathcal{P}_{\mathrm{fi}}TX$. Next, the τ-component of *synch* is the join over all actions a different from τ of the following family of maps $\textit{synch}_a$, where *dst* is the double strength of the commutative monad $\mathcal{P}_{\mathrm{fi}}$. (Note that a join of cardinality $|A|$ is needed here, hence if A is infinite one needs to replace $\mathcal{P}_{\mathrm{fi}}$ by a free monad for semi-lattices with joins of cardinality greater than or equal to $|A|$.)

$$\begin{array}{rl}
\pi_2 \times \pi_2 & : (X \times (\mathcal{P}_{\mathrm{fi}}X)^A) \times (X \times (\mathcal{P}_{\mathrm{fi}}X)^A) \longrightarrow (\mathcal{P}_{\mathrm{fi}}X)^A \times (\mathcal{P}_{\mathrm{fi}}X)^A \\
\pi_a \times \pi_{\overline{a}} & : \qquad\qquad\qquad\qquad\qquad\qquad\qquad\quad \longrightarrow \mathcal{P}_{\mathrm{fi}}X \times \mathcal{P}_{\mathrm{fi}}X \\
dst & : \qquad\qquad\qquad\qquad\qquad\qquad\qquad\quad \longrightarrow \mathcal{P}_{\mathrm{fi}}(X \times X) \\
\mathcal{P}_{\mathrm{fi}}(\eta \parallel \eta) & : \qquad\qquad\qquad\qquad\qquad\qquad\qquad\quad \longrightarrow \mathcal{P}_{\mathrm{fi}}TX \\
\hline
\textit{synch}_a & : (X \times (\mathcal{P}_{\mathrm{fi}}X)^A) \times (X \times (\mathcal{P}_{\mathrm{fi}}X)^A) \longrightarrow \mathcal{P}_{\mathrm{fi}}TX
\end{array}$$

Then:

$$[\![\parallel]\!] = \textit{l-merge} \vee (\textit{l-merge} \circ \gamma) \vee \textit{synch} \ .$$

In Set:

$$\begin{array}{rl}
(x, \beta)[\![\parallel]\!](y, \beta') = & (a \mapsto (\{x' \parallel y \mid x' \in \beta(a)\} \cup \{x \parallel y' \mid y' \in \beta'(a)\})) \cup \\
& \cup (\tau \mapsto \{x' \parallel y' \mid \exists a : \, x' \in \beta(a), \, y' \in \beta'(\overline{a})\})
\end{array} \ .$$

Another simple example of an operator modellable in this setting is the usual binary choice operator 'or':

$$\begin{array}{rl}
\pi_2 \times \pi_2 & : (X \times (\mathcal{P}_{\mathrm{fi}}X)^A) \times (X \times (\mathcal{P}_{\mathrm{fi}}X)^A) \longrightarrow (\mathcal{P}_{\mathrm{fi}}X)^A \times (\mathcal{P}_{\mathrm{fi}}X)^A \\
\vee & : \qquad\qquad\qquad\qquad\qquad\qquad\qquad\quad \longrightarrow (\mathcal{P}_{\mathrm{fi}}X)^A \\
(\mathcal{P}_{\mathrm{fi}}\eta)^A & : \qquad\qquad\qquad\qquad\qquad\qquad\qquad\quad \longrightarrow (\mathcal{P}_{\mathrm{fi}}TX)^A \\
\hline
[\![\mathsf{or}]\!] & : (X \times (\mathcal{P}_{\mathrm{fi}}X)^A) \times (X \times (\mathcal{P}_{\mathrm{fi}}X)^A) \longrightarrow (\mathcal{P}_{\mathrm{fi}}TX)^A
\end{array}$$

In Set: $(x, \beta)[\![\mathsf{or}]\!](y, \beta') = \boldsymbol{\lambda} a.(\beta(a) \cup \beta'(a))$.

Negative Premises. In order to model rules with negative premises one can use the following variant of the above behaviour endofunctor:

$$BX = (1 + \mathcal{P}_f X)^A \quad . \tag{11}$$

(In Set, $1 + \mathcal{P}_f X$ is isomorphic to $\mathcal{P}_{fi} X$, but this is not true in general.)

As an example consider the following rules describing a unary operator '$\tau \gg$' which gives priority to τ-steps:

$$\frac{x \overset{\tau}{\leadsto} x'}{(\tau \gg x) \overset{\tau}{\leadsto} x'} \quad , \qquad \frac{x \overset{\tau}{\not\leadsto} \qquad x \overset{a}{\leadsto} x'}{(\tau \gg x) \overset{a}{\leadsto} x'} \; a \neq \tau \quad . \tag{12}$$

For every action a different from τ, the second rule can be modelled as:

$$\begin{array}{rcl}
\pi_2 & : & X \times (1 + \mathcal{P}_f X)^A \longrightarrow (1 + \mathcal{P}_f X)^A \\
\langle id, id \rangle & : & \longrightarrow (1 + \mathcal{P}_f X)^A \times (1 + \mathcal{P}_f X)^A \\
\pi_a \times \pi_\tau & : & \longrightarrow (1 + \mathcal{P}_f X) \times (1 + \mathcal{P}_f X) \\
\delta & : & \longrightarrow 1 + \mathcal{P}_f X + \mathcal{P}_f X + (\mathcal{P}_f X)^2 \\
id + id + [!, !] & : & \longrightarrow 1 + \mathcal{P}_f X + 1 \\
\gamma & : & \cong \; 1 + 1 + \mathcal{P}_f X \\
[id, id] + \mathcal{P}_f \eta & : & \longrightarrow 1 + \mathcal{P}_f T X \\
\hline
\pi_a \circ [\![\tau \gg]\!] & : & (X \times (1 + \mathcal{P}_f X)^A) \longrightarrow 1 + \mathcal{P}_f T X
\end{array}$$

while the first rule yields: $\pi_\tau \circ [\![\tau \gg]\!] = (1 + \mathcal{P}_f \eta) \circ \pi_\tau \circ \pi_2$.

The examples in this section should convince the reader that GSOS rules can be modelled using the behaviour B as in (11) on any distributive category with powers and a commutative free semi-lattice monad $\mathcal{P}_f$. (Cf [17, Theorem 2.1].)

5 Modularity

One of the advantages of the present approach to opertional semantics is that, by decomposing each rule in a series of natural transformations, one brings to light the amount of structure of the behaviour which is essential for the rule. As a consequence, one can interpret the same rules modularly with respect to different behaviours with the same structure.

Parallel Composition. For modelling the first rule in (9), all one needs is B to be a strong endofunctor. Correspondingly, the map *l-merge* can be abstracted to:

$$\textit{l-merge}_B = B(\eta \parallel \eta) \circ st \circ (\pi_2 \otimes \pi_1) : (X \times BX) \otimes (X \times BX) \to BTX \quad ,$$

parametric in B. Therefore, if BX is endowed with a join $\vee : BX \times BX \to BX$, one can abstract the modelling of the first two rules for parallel composition (thus without synchronization!) as:

$$[\![\parallel]\!]_{B,\vee} = \textit{l-merge}_B \vee (\textit{l-merge}_B \circ \gamma) \quad . \tag{13}$$

In §6, it is shown how this map $[\![\ \|\ ]\!]_{B,\vee}$ can be instantiated to the behaviour on semi-lattices as used for trace equivalence.

A simple variation of the above $\mathit{l\text{-}merge}_B$ is obtained by considering behaviours of the form $BX = B'(1+X)$:

$$\mathit{l\text{-}merge}_{B'(1+\_)} = B'(\iota_2) \circ B'([\eta, (\eta \parallel \eta)]) \circ B'(\delta) \circ st \circ (\pi_2 \otimes \pi_1) \ .$$

One can then instantiate this rule to $B'X = (\mathcal{P}_f(S \cdot X))^S$. In Set:

$$\mathit{l\text{-}merge}((x,\beta),(y,\beta')) = \boldsymbol{\lambda}\sigma.(\{\sigma' \cdot (x' \parallel y) \mid (\sigma' \cdot x') \in \beta(\sigma)\} \cup \{\sigma' \cdot y \mid \sigma' \in \beta(\sigma)\})$$

which corresponds to the usual rules of the left merge operator for non-deterministic programs with side-effects and explicit termination.

Sequential Composition. Also the modelling $[\![\ ;\ ]\!]$ of the rules for sequential composition in §3 can be made parametric in $BX = B'(1+X)$, for some strong endofunctor B'. This simply yields the above $\mathit{l\text{-}merge}_{B'(1+\_)}$ with '$\parallel$' replaced by ' ; ':

$$[\![\ ;\ ]\!]_{B'(1+\_)} = B'(\iota_2) \circ B'([\eta, (\eta\ ;\ \eta)]) \circ B'(\delta) \circ st \circ (\pi_2 \otimes \pi_1) \ .$$

For $B'X = (\mathcal{P}_f(S \cdot (X)))^S$ on Set one has then the non-deterministic version of the map $[\![\ ;\ ]\!]$ in §3.

The rules for sequential composition with atomic actions instead of side-effects are slightly different. One can use negative premises as in the previous section or a distinguished transition, say $x \rightsquigarrow *$, to express explicit termination, in which case the rules are:

$$\frac{x \overset{a}{\rightsquigarrow} x'}{x\ ;\ y \overset{a}{\rightsquigarrow} x'\ ;\ y} \ , \qquad \frac{x \rightsquigarrow * \quad y \overset{a}{\rightsquigarrow} y'}{x\ ;\ y \overset{a}{\rightsquigarrow} y'} \ , \qquad \frac{x \rightsquigarrow * \quad y \rightsquigarrow *}{x\ ;\ y \overset{a}{\rightsquigarrow} *} \ . \tag{14}$$

These rules can be modelled parametrically in a strong endofunctor B' on an affine SMC with finite products and coproducts, and tensors distributing over coproducts:

$$\begin{array}{ll}
\pi_2 \otimes id & : (X \times (1+B'X)) \otimes (X \times (1+B'X)) \\
 & \qquad\quad \longrightarrow (1+B'X) \otimes (X \times (1+B'X)) \\
\delta & : \qquad\quad \cong\ (X \times (1+B'X)) + ((B'X \otimes X) \times (1+B'X)) \\
\pi_2 + \pi_1 & : \qquad\quad \longrightarrow (1+B'X) + (B'X \otimes X) \\
id + st & : \qquad\quad \longrightarrow 1 + B'X + B'(X \otimes X) \\
id + [B'\eta, B'(\eta\ ;\ \eta)] & : \qquad\quad \longrightarrow 1 + B'TX \\
\hline
[\![\ ;\ ]\!]_{1+B'} & : ((X \times (1+B'X)) \otimes (X \times (1+B'X)) \longrightarrow 1 + B'TX
\end{array}$$

In Set, for $B'X = (\mathcal{P}_f X)^A$:

$$(x,\beta)[\![\ ;\ ]\!]_{1+B'}(y,\beta') = \begin{cases} * & \text{if } \beta = * = \beta' \\ \boldsymbol{\lambda}a.(\{y' \mid y' \in \beta'(a)\}) & \text{if } \beta = *,\ \beta' \neq * \\ \boldsymbol{\lambda}a.(\{x'\ ;\ y \mid x' \in \beta(a)\}) & \text{otherwise} \end{cases} \ .$$

Choice. The map $[\![\,\mathsf{or}\,]\!]$ modelling the choice operator of CCS can be abstracted and made parametric in any endofunctor B on an affine SMC (so that the tensor comes with two projections, say $\pi_1^{\otimes} : X \otimes Y \to X$ and $\pi_2^{\otimes} : X \otimes Y \to Y$) such that BX is endowed with a semi-lattice structure $\vee$:

$$[\![\,\mathsf{or}\,]\!]_{B,\vee} = B\eta \circ \vee \circ (\pi_2 \times \pi_2) \circ \langle \pi_1^{\otimes}, \pi_2^{\otimes} \rangle : (X \times BX) \otimes (X \times BX) \longrightarrow BTX \ . \tag{15}$$

Actions as Outputs. An alternative to the behaviour $B_1 X = (\mathcal{P}_{\mathrm{fi}} X)^A$ is the behaviour $B_2 X = \mathcal{P}_{\mathrm{fi}}(A \cdot X)$. In Set, if A is finite, these two behaviours are isomorphic, but this is not true in general, although there is a canonical map *embed* : $\mathcal{P}_{\mathrm{fi}}(A \cdot X) \to (\mathcal{P}_{\mathrm{fi}} X)^A$ (which can be used to interpret the rule for synchronization – ie the third rule in (9) – using B_2 instead of B_1). The endofunctor B_1 with actions as powers corresponds to consider actions as (testable!) inputs, while the endofunctor B_2 with actions as copowers corresponds to actions as outputs. In the presence of negative premises (and exceptions in general) it seems the use of powers is really needed, but otherwise copowers seem more natural. Take, for instance, the axiom $a \overset{a}{\rightsquigarrow} \mathsf{nil}$ for atomic actions a. Using B_1 one has to map all actions different from a to the neutral element of $B_1 TX$, while using B_2 one can define $[\![A]\!] : A \cdot 1 \to \mathcal{P}_{\mathrm{fi}}(A \cdot TX)$ to be the composition of the unit of the monad $\mathcal{P}_{\mathrm{fi}}$ with B_2 applied to the insertion of the constant nil into TX (which is denoted here by nil as well). More generally, for every pointed endofunctor $\langle H, \nu \rangle$, one can define:

$$[\![A]\!]_H = \nu \circ (A \cdot \mathsf{nil}) : A \cdot 1 \to H(A \cdot TX) \ . \tag{16}$$

Clearly, also $[\![\ \|\ ]\!]_{B,\vee}$ and $[\![\,\mathsf{or}\,]\!]_{B,\vee}$ can be instantiated to $BX = \mathcal{P}_{\mathrm{fi}}(A \cdot X)$ as well as to $BX = 1 + \mathcal{P}_{\mathrm{f}}(A \cdot X)$. The latter behaviour can be used also for $[\![\ ;\ ]\!]_{1+B'}$.

Replication. Another example of a modular modelling is the one for a unary loop operator. It can be defined parametrically in a strong endofunctor B and a natural transformation $\otimes : X \times Y \to X \otimes Y$ as follows:

$$\begin{array}{ll} \otimes & : X \times BX \longrightarrow X \otimes BX \\ \gamma & : \qquad\qquad\ \longrightarrow BX \otimes X \\ st & : \qquad\qquad\ \longrightarrow B(X \otimes X) \\ B(\eta \ ;\ \mathsf{loop}(\eta)) & : \qquad\qquad\ \longrightarrow BTX \\ \hline [\![\mathsf{loop}]\!]_B^{\otimes} & : X \times BX \longrightarrow BTX \end{array}$$

In Set, for $BX = 1 + \mathcal{P}_{\mathrm{f}}(A \cdot X)$ and for $\otimes$ simply the identity, one has:

$$[\![\mathsf{loop}]\!](x, \beta) = \begin{cases} * & \text{if } \beta = * \\ \{a \cdot (x' \ ;\ \mathsf{loop}(x)) \mid a \cdot x \in \beta\} & \text{otherwise} \end{cases}$$

which models the rules:

$$\frac{x \overset{a}{\rightsquigarrow} x'}{\mathsf{loop}(x) \overset{a}{\rightsquigarrow} x' \ ;\ (\mathsf{loop}(x))} \ , \qquad \frac{x \rightsquigarrow *}{\mathsf{loop}(x) \rightsquigarrow *} \ . \tag{17}$$

Note that, by using parallel instead of sequential composition, the operator loop has essentially the same behaviour as the replication operator of the π-calculus [11].

6 Non-Determinism II: Trace Equivalence

In [7], non-determinism with side-effects and explicit termination is treated by simply interpreting the (deterministic) behaviour $BX = (S \cdot (1 + X))^S$ in a category of semi-lattices, instead of using the behaviour $BX = (\mathcal{P}_f(S \cdot (1 + X)))^S$. In terms of behavioural equivalences, this amounts to consider (completed) *trace equivalence* instead of bisimulation.

When modelling rules in categories of semi-lattices one needs the full generality of monoidal rather than cartesian categories. Indeed, although, if $\mathcal{C}$ is cartesian, the category $SL(\mathcal{C})$ inherits from it the cartesian structure (which is useful for applying the structural recursion theorem with accumulators), for modelling the syntax one needs tensors rather than cartesian products: the natural interpretation of a binary operators in a category of semi-lattice is not as linear but as *bilinear* maps, ie as maps which are linear in each component separately; the tensor product $\otimes$ is the classifier of bilinear maps. (Cf [7].)

As a first example of trace equivalence semantics, consider the map $[\![\ ;\ ]\!]$ defined in §3: clearly, it can be interpreted in categories of semi-lattices. For instance, in $SL(\mathsf{Set})$, one has the following, where $x \leq y$ stands for $x \vee y = y$:

$$(x, \beta)[\![\ ;\ ]\!](y, \beta') = \boldsymbol{\lambda}\sigma. \left(\bigvee_{(\sigma' \cdot x') \leq \beta(\sigma)} \sigma' \cdot (x'\ ;\ y) \vee \bigvee_{\sigma' \leq \beta(\sigma)} \sigma' \cdot y\right) \quad .$$

Note that copowers $S \cdot X$ in $SL(\mathcal{C})$ are right-linear, hence in $SL(\mathsf{Set})$:

$$\sigma \cdot (x \vee y) = (\sigma \cdot x) \vee (\sigma \cdot y)$$

for every σ in S, and for all x, y in X. Therefore, one could write $\sigma \cdot x$ as the more familiar $\{\sigma\} \otimes x$ of [7], where $\otimes : X \times Y \to X \otimes Y$ is the universal bilinear natural transformation associated with the tensor.

Other examples are the maps $[\![\ \|\ ]\!]_{B,\vee}$, $[\![\ ;\ ]\!]_{1+B'}$, $[\![\,\mathsf{or}\,]\!]_{B,\vee}$, $[\![A]\!]_H$, and $[\![\mathsf{nil}]\!]$ which can be interpreted in $SL(\mathsf{Set})$ with respect to $BX = 1 + A \cdot X$, $B'X = A \cdot X$, and $HX = 1 + X$ as follows:

$$(x, \beta)[\![\ \|\ ]\!](y, \beta') = \begin{cases} * \vee \bigvee_{a \cdot x' \leq \beta} a \cdot (x' \parallel y) \vee \bigvee_{a \cdot y' \leq \beta'} a \cdot (x \parallel y') & \text{if } \beta \geq * \leq \beta' \\ \bigvee_{a \cdot x' \leq \beta} a \cdot (x' \parallel y) \vee \bigvee_{a \cdot y' \leq \beta'} a \cdot (x \parallel y') & \text{otherwise} \end{cases}$$

$$(x, \beta)[\![\ ;\ ]\!](y, \beta') \ = \begin{cases} * \vee \bigvee_{a \cdot y' \leq \beta'} a \cdot y' \vee \bigvee_{a \cdot x' \leq \beta} a \cdot (x'\ ;\ y) & \text{if } \beta \geq * \leq \beta' \\ \bigvee_{a \cdot y' \leq \beta'} a \cdot y' \vee \bigvee_{a \cdot x' \leq \beta} a \cdot (x'\ ;\ y) & \text{if } \beta \geq * \not\leq \beta' \\ \bigvee_{a \cdot x' \leq \beta} a \cdot (x'\ ;\ y) & \text{otherwise} \end{cases}$$

$$(x, \beta)[\![\,\mathsf{or}\,]\!](y, \beta') = \beta \vee \beta'$$

$$[\![a]\!] = a \cdot \mathsf{nil} \qquad\qquad [\![\mathsf{nil}]\!] = *$$

7 From the Rules to the Models

In this section, it is shown how both an operational and a denotational model can be derived from abstract operational rules $\rho : \Sigma(Id \times B) \stackrel{.}{\longrightarrow} BT$. General results from [17] ensure that the denotational model is always adequate with respect to the operational one. All this is exemplified using the language in (1) and the natural transformation

$$\rho = [\, [\![\mathsf{nil}]\!], [\![A]\!]_H, [\![\ \|\ ]\!]_{B,\vee}, [\![\ ;\ ]\!]_{1+B'}\,] : \Sigma(X \times BX) \longrightarrow BTX \tag{18}$$

interpreted either in Set with respect $BX = 1 + \mathcal{P}_f(A \cdot X)$, $B'X = \mathcal{P}_f(A \cdot X)$, and $HX = 1 + \mathcal{P}_f X$, or in $SL(\mathsf{Set})$ with respect to $BX = 1 + A \cdot X$, $B'X = A \cdot X$, and $HX = 1 + X$. Note the isomorphism $1 + \mathcal{P}_f(A \cdot X) \cong \mathcal{P}_{fi}(A \times X)$ which holds in Set; this is used extensively in the sequel without further mention.

The main tool in this section is the following 'folklore' structural recursion theorem with accumulators, ie with terms as extra parameters of the recursion. (Cf [17].)

Theorem 1 (Structural Recursion with Accumulators). Let T be the monad freely generated by an endofunctor F on a cartesian category $\mathcal{C}$ and let $\psi : FTX \to TX$ be the structure of the free F-algebra over an object X of $\mathcal{C}$. For all maps $f : X \to Y$ and $h : F(TX \times Y) \to Y$ in $\mathcal{C}$ there exists a unique map $f^\sharp : TX \to Y$ in $\mathcal{C}$ such that

$$\begin{tikzcd} X \arrow[r, "\eta"] \arrow[dr, "f"'] & TX \arrow[d, dashed, "f^\sharp"] & FTX \arrow[l, "\psi"'] \arrow[d, "F\langle id, f^\sharp\rangle"] \\ & Y & F(TX \times Y) \arrow[l, "h"] \end{tikzcd}$$

commutes. □

A special case is when X is the initial object 0. Then the left triangle is vacuous, ψ is an isomorphism, and the unique map from $T0$ to Y is called the **iterator** (or the inductive extension) of h:

$$\begin{tikzcd} T0 \arrow[d, dashed, "it(h)"'] & FT0 \arrow[l, "\psi"'] \arrow[d, "F\langle id, it(h)\rangle"] \\ Y & F(T0 \times Y) \arrow[l, "h"] \end{tikzcd}$$

In this case it suffices that $\langle T0, \psi\rangle$ is the initial F-algebra, rather than the stronger hypothesis that F freely generates T.

7.1 The Induced Operational Model

Notation. [17] For Σ, B, and T as above and for every natural transformation $\rho : \Sigma(Id \times B) \stackrel{.}{\longrightarrow} BT$, put:

$$\varrho = B\mu \circ \rho_T : \Sigma(T \times BT) \stackrel{.}{\longrightarrow} BT \ . \tag{19}$$

Note that the mapping $\rho \mapsto \varrho$ is a 1-1 correspondence between natural transformations of type $\Sigma(Id \times B) \stackrel{.}{\longrightarrow} BT$ and those of type $\Sigma(T \times BT) \stackrel{.}{\longrightarrow} BT$.

The **canonical operational model** $T_\rho(0)$ induced by some abstract operational rules $\rho : \Sigma(Id \times B) \stackrel{.}{\longrightarrow} BT$ is the iterator of ϱ at 0:

$$\begin{tikzcd}
T0 \arrow[d, dashed, "T_\rho(0)"'] & \Sigma T0 \arrow[l, "\psi"'] \arrow[d, "\Sigma\langle id, T_\rho(0)\rangle"] \\
BT0 & \Sigma(T0 \times BT0) \arrow[l, "\varrho"]
\end{tikzcd}$$

In Set, chasing the above diagram one obtains:

$$\begin{aligned}
T_\rho(0)(t_1 \parallel t_2) \ &= \{\langle a, t_1' \parallel t_2\rangle \mid \langle a, t_1'\rangle \in T_\rho(0)(t_1)\} \cup \\
&\quad \cup \{\langle a, t_1 \parallel t_2'\rangle \mid \langle a, t_2'\rangle \in T_\rho(0)(t_2)\} \\
T_\rho(0)(t_1 \ ; t_2) \ &= \{\langle a, t_1' \ ; t_2\rangle \mid \langle a, t_1'\rangle \in T_\rho(0)(t_1)\} \cup \\
&\quad \cup \{\langle a, t_2'\rangle \mid T_\rho(0)(t_1) = \emptyset, \ \langle a, t_2'\rangle \in T_\rho(0)(t_2)\} \\
T_\rho(0)(\mathsf{nil}) = \emptyset \quad & T_\rho(0)(a) = \{\langle a, \mathsf{nil}\rangle\}
\end{aligned}$$

Using the restriction of the correspondence (4) to the endofunctor $\mathcal{P}_{\mathrm{fi}}$, one can check that this is the intended transition relation indeed.

In $SL(\mathsf{Set})$, since left adjoints preserve colimits, $T0$ is simply the free semilattice over the usual closed terms in Set and $T_\rho(0) : T0 \to 1 + A \cdot T0$ can be transposed across the adjunction between sets and semi-lattices, yielding a transition system (with explicit termination):

$$t \overset{a}{\rightsquigarrow} t' \iff a \cdot t' \leq T_\rho(0) \quad \text{ and } \quad t \rightsquigarrow * \iff * \leq T_\rho(0) \ ,$$

for all closed terms t and t'. This should help in understanding the following: $T_\rho(0)(\mathsf{nil}) = *$, $T_\rho(0)(a) = a \cdot \mathsf{nil}$,

$$T_\rho(0)(t_1 \parallel t_2) = \begin{cases}
\begin{array}{l} * \vee \bigvee_{a \cdot t_1' \leq T_\rho(0)(t_1)} a \cdot (t_1' \parallel t_2) \vee \\ \quad \vee \bigvee_{a \cdot t_2' \leq T_\rho(0)(t_2)} a \cdot (t_1 \parallel t_2') \end{array} & \text{if } T_\rho(0)(t_1) \geq * \leq T_\rho(0)(t_2) \\
\begin{array}{l} \bigvee_{a \cdot t_1' \leq T_\rho(0)(t_1)} a \cdot (t_1' \parallel t_2) \vee \\ \quad \vee \bigvee_{a \cdot t_2' \leq T_\rho(0)(t_2)} a \cdot (t_1 \parallel t_2') \end{array} & \text{otherwise}
\end{cases}$$

and a similar clause for $T_\rho(0)(t_1 \ ; t_2)$.

7.2 The Coinduced Denotational Model

The construction of the model $T_\rho(0)$ extends to a functor $T_\rho : B\text{-}\mathsf{Coalg} \to B\text{-}\mathsf{Coalg}$ as follows. For every coalgebra $k : X \to BX$, the coalgebra $T_\rho(k) : TX \to BTX$ is the map:

$$\begin{tikzcd}
X \arrow[rr, "\eta"] \arrow[dr, "k"'] & & TX \arrow[dd, dashed, "T_\rho(k)"] & \Sigma TX \arrow[l, "\psi"'] \arrow[dd, "\Sigma\langle id, T_\rho(k)\rangle"] \\
& BX \arrow[dr, "B\eta"'] & & \\
& & BTX & \Sigma(TX \times BTX) \arrow[l, "\varrho"]
\end{tikzcd}$$

given by the structural recursion theorem. The naturality of ρ ensures this is a functor. (Moreover, T_ρ is a monad lifting the monad T to the B-coalgebras [17].) In particular, T_ρ can be applied to the final B-coalgebra

$$\varphi : \nu X.BX \cong B(\nu X.BX)$$

yielding the coalgebra $T_\rho(\varphi) : T(\nu X.BX) \to BT(\nu X.BX)$. The coalgebra $T_\rho(\varphi)$ is a 'conservative extension' of both the final coalgebra φ and the canonical operational model $T_\rho(0)$, because of the commutativity of the triangle in the above diagram and of the functoriality of T_ρ. In Set, this means that:

$$\langle a, p'\rangle \in \varphi(p) \iff \langle a, p'\rangle \in T_\rho(\varphi)(p) \ , \ \langle a, t'\rangle \in T_\rho(0)(t) \iff \langle a, t'\rangle \in T_\rho(\varphi)(t)$$

for all elements p, p' of $\nu X.BX$ and for all closed terms t, t'.

Most importantly, the final coalgebra can be endowed with a canonical T-algebra structure, namely the **final coalgebra semantics** [1,14] $\mathcal{M}_\varphi$ of the operational model $T_\rho(\varphi)$:

$$\begin{tikzcd}
T(\nu X.BX) \arrow[r, dashed, "\mathcal{M}_\varphi"] \arrow[d, "T_\rho(\varphi)"'] & \nu X.BX \arrow[d, "\varphi", "\cong"'] \\
BT(\nu X.BX) \arrow[r, "B\mathcal{M}_\varphi"'] & B(\nu X.BX)
\end{tikzcd}$$

Thus $\mathcal{M}_\varphi$ is the unique homomorphism from $T_\rho(\varphi)$ to the final coalgebra φ.

The theory in [17] shows that $\mathcal{M}_\varphi$, which is written there as $D_\rho(1)$, qualifies as the **canonical denotational model** for ρ and that it is dual, in a suitable sense, to the operational model $T_\rho(0)$. One can derive from it a denotation for every n-ary operator ϕ of the language, ie a map

$$\widehat{\phi} : \underbrace{\nu X.BX \otimes \cdots \otimes \nu X.BX}_{n} \longrightarrow \nu X.BX$$

by simply precomposing $\mathcal{M}_\varphi$ with the insertion of $(\nu X.BX)^{\otimes n}$ in $T(\nu X.BX)$, ie $\widehat{\phi}(p_1, \ldots, p_n) = \mathcal{M}_\varphi(\phi(p_1, \ldots, p_n))$ for all elements $p_1, \ldots, p_n : 1 \to \nu X.BX$.

For instance, in Set:

$$\begin{aligned} p_1 \mathbin{\widehat{\|}} p_2 &= \mathcal{M}_\varphi(p_1 \parallel p_2) \\ &= \{\langle a, \mathcal{M}_\varphi(p_1' \parallel p_2)\rangle \mid \langle a, p_1'\rangle \in \varphi(p_1)\} \cup \\ &\qquad \cup \{\langle a, \mathcal{M}_\varphi(p_1 \parallel p_2')\rangle \mid \langle a, p_2'\rangle \in \varphi(p_2)\} \end{aligned} \quad .$$

8 Guarded Recursion

Let X be a set (of variables) $x_1, x_2, \ldots$ and let $t_1, t_2, \ldots$ be elements of TX, hence terms which might contain variables from X. Every system

$$\Theta \quad = \quad \begin{cases} x_1 = t_1 \\ x_2 = t_2 \\ \quad \vdots \end{cases} \tag{20}$$

of mutually recursive term-equations in X can be seen as a T-coalgebra having as carrier the set X of variables appearing in the system and as structure the function

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

Conversely, every T-coalgebra defines a system of term-equations.

In structural operational semantics, the usual way to deal with recursive programs in Set is to use 'guarded terms'. This is a rather informal notion and the aim of this section is to use the categorical modelling of the rules to make it formal and general.

Typically, a term is called guarded if it is of the form $a\,;t$, that is, if it starts with an action. (Cf [5,3].) For instance, following [3], the guarded terms corresponding to the language in (1) are:

$$g ::= a \mid g\,;t \mid g \parallel g \ . \tag{21}$$

The idea is that this can be generalized by expressing, in a formal way, that a term is guarded if its behaviour does not depend on the behaviour of its variables.

First note that the functor $T_\rho : B\text{-Coalg} \to B\text{-Coalg}$ defined in the previous section can be factorized through the functor mapping a B-coalgebra $k : X \to BX$ to the coalgebra $B\eta \circ k : X \to BTX$ of the composite endofunctor BT. The other component of the factorization is denoted, with a slight abuse of notation, also as $T_\rho : BT\text{-Coalg} \to B\text{-Coalg}$. It is defined as:

$$\begin{tikzcd} X \arrow[r, "\eta"] \arrow[dr, "k"'] & TX \arrow[d, dashed, "T_\rho(k)"] & \Sigma TX \arrow[l, "\psi"'] \arrow[d, "\Sigma\langle id, T_\rho(k)\rangle"] \\ & BTX & \Sigma(TX \times BTX) \arrow[l, "\varrho"] \end{tikzcd}$$

Next, let $U : BT\text{-Coalg} \to \mathcal{C}$ be the forgetful functor from the BT-coalgebras to their carriers. Following [8], one can prove there is a 1-1 correspondence between 'liftings of functors' and natural transformations of a suitable type which allows one to regard the functor $T_\rho : BT\text{-}\mathsf{Coalg} \to B\text{-}\mathsf{Coalg}$ as a natural transformation $T_\rho : TU \stackrel{\cdot}{\longrightarrow} BTU$. (Cf 'Liftings as Coactions' in [16, §4].)

Definition 2. A **guard** for some abstract operational rules ρ as in (7) is a 'U-cone' over the induced natural transformation $T_\rho : TU \stackrel{\cdot}{\longrightarrow} BTU$, ie a span

$$\begin{tikzcd} & G \arrow[dl, "\xi"'] \arrow[dr, "\zeta"] & \\ T & & BT \end{tikzcd} \qquad \text{such that} \qquad \begin{tikzcd} & GU \arrow[dl, "\xi_U"'] \arrow[dr, "\zeta_U"] & \\ TU \arrow[rr, "T_\rho"'] & & BTU \end{tikzcd}$$

commutes. □

Given a guard $\langle G, \xi, \zeta \rangle$, the coalgebras $\Theta : X \to GX$ can be understood as systems of guarded term-equations. Indeed The second leg ζ of the span can be used to turn Θ into a BT-coalgebra and then the operational model $T_\rho(\zeta \circ \Theta)$ can be defined:

$$\begin{tikzcd} X \arrow[rr, "\eta"] \arrow[dr, "\Theta"'] & & TX \arrow[dd, dashed, "T_\rho(\zeta \circ \Theta)"] & \Sigma TX \arrow[l, "\psi"'] \arrow[dd, "\Sigma\langle id, T_\rho(\zeta \circ \Theta)\rangle"] \\ & GX \arrow[dr, "\zeta"'] & & \\ & & BTX & \Sigma(TX \times BTX) \arrow[l, "\varrho"] \end{tikzcd}$$

The corresponding final coalgebra semantics

$$\begin{tikzcd} TX \arrow[r, dashed, "\mathcal{M}_\Theta"] \arrow[d, "T_\rho(\zeta \circ \Theta)"'] & \nu X.BX \arrow[d, "\varphi", "\cong"'] \\ BTX \arrow[r, "B\mathcal{M}_\Theta"'] & B(\nu X.BX) \end{tikzcd}$$

gives the desired recursive interpretation of the system Θ.

How does one prove that this notion specialises to the above example? Firstly, note that the functorial notion of syntax corresponding to the grammar in (21) is the functor

$$\Sigma_G : \mathcal{C} \times \mathcal{C} \to \mathcal{C} \qquad (X_1, X_2) \mapsto A \cdot I + X_1 \otimes X_2 + X_1 \otimes X_1 \ .$$

Indeed, for every X, the initial algebra

$$GX = \mu Y.\Sigma_G(Y, TX)$$

specialises in Set to the set of terms given by the grammar in (21). Further, note that composing Σ_G with the diagonal $\Delta : \mathcal{C} \to \mathcal{C} \times \mathcal{C}$ one obtains a subfunctor of the endofunctor in (2) freely generating the monad $TX = \mu Y.X + \Sigma Y$ corresponding to the grammar in (1); let $i : \Sigma_G \circ \Delta \stackrel{\cdot}{\longrightarrow} \Sigma$ be the corresponding insertion map. The first leg $\xi : G \stackrel{.}{\longrightarrow} T$ of the span can then be defined recursively using the initiality of GX with respect to the endofunctor $\Sigma_G(\_\,, TX)$:

$$\begin{tikzcd}
\Sigma_G(GX,TX) \arrow[r, "\Sigma_G(\xi,TX)"] \arrow[dd, "\cong"'] & \Sigma_G(TX,TX) \arrow[d, tail] \\
& \Sigma TX \arrow[d, "\psi"] \\
GX \arrow[r, dashed, "\xi"'] & TX
\end{tikzcd}$$

Next, let the categorical modelling ρ of the rules for the language (1) be as in (18) with respect to an arbitrary B such that $B = 1 + B'$ and both B and B' are endowed with a semi-lattice structure. Consider the following restricted version of sequential composition:

$$[\![\ ;\ ]\!]_{B'} = B'(\eta\ ;\ \eta) \circ st \circ (\pi_2 \otimes id) : (X \times B'X) \otimes X \to B'TX \ .$$

Then, for:

$$\rho' = [\, [\![A]\!]_H, [\![\ \|\ ]\!]_{B',\vee}, [\![\ ;\ ]\!]_{B'}\,] : \Sigma_G(X \times B'X, X) \longrightarrow B'TX$$

the following diagram commutes.

$$\begin{tikzcd}
\Sigma_G(X \times B'X, X \times B'X) \arrow[r, "\Sigma_G\Delta(id \times \iota_2)"] \arrow[d, "\Sigma_G(id,\pi_1)"'] & \Sigma_G(X \times BX, X \times BX) \arrow[r, tail, "i"] & \Sigma(X \times BX) \arrow[d, "\rho"] \\
\Sigma_G(X \times B'X, X) \arrow[r, "\rho'"'] & B'TX \arrow[r, "\iota_2"'] & 1 + B'TX = BTX
\end{tikzcd}$$

Since the variables in GX are those stemming from the second component of Σ_G, the fact that the restriction of the rules ρ to $\Sigma_G(X \times B'X, X \times B'X)$ factorizes through $\Sigma_G(id, \pi_1)$ means that the behaviour of the variables in a guarded term is irrelevant for its behaviour. In particular, this allows us to define the second leg of the span as $\zeta = \iota_2 \circ \zeta' : G \stackrel{.}{\longrightarrow} BT$, where ζ' is the iterator of the composite structure $\varrho' \circ \Sigma_G(\xi \times B'TX, TX)$ with respect to the initial algebra GX:

$$\begin{tikzcd}
\Sigma_G(GX,TX) \arrow[r, "\Sigma_G(\langle id,\zeta'\rangle,TX)"] \arrow[ddd, "\cong"'] & \Sigma_G(GX \times B'TX, TX) \arrow[d, "\Sigma_G(\xi \times B'TX,TX)"] \\
& \Sigma_G(TX \times B'TX, TX) \arrow[dd, "\varrho'"] \\
& \\
GX \arrow[r, dashed, "\zeta'"'] & B'TX
\end{tikzcd}$$

Proposition 3. The above $\langle G, \xi, \zeta \rangle$ is a guard for the abstract operational rules ρ as in (18).

Proof. Some diagram chasing shows that, for every BT-coalgebra k, that both $T_\rho(k) \circ \xi$ and ζ fit as the unique iterator of the composite structure

$$\varrho \circ i \circ \Sigma_G(\xi \times id, \langle id, T_\rho(k)\rangle) : \Sigma_G(GX \times BTX, TX) \longrightarrow BTX$$

with respect to the initial algebra $GX = \mu Y.\Sigma_G(Y, TX)$. $\square$

Thus, for instance, the following system Θ of term-equations in $X = \{x, y\}$ is guarded with respect to the above guard and $BX = 1 + \mathcal{P}_f(A \cdot X)$ on Set:

$$x = a \,;\, x \ , \qquad y = (a \,;\, y) \parallel (b \,;\, x) \ .$$

Omitting, as usual, the insertion-of-variables η and the final coalgebra isomorphism, one has:

$$\mathcal{M}_\Theta(x) = \{\langle a, \mathcal{M}_\Theta(x)\rangle\} \ , \qquad \mathcal{M}_\Theta(y) = \{\langle a, \mathcal{M}_\Theta(y)\rangle, \langle b, \mathcal{M}_\Theta(x)\rangle\}$$

which is the intended interpretation of Θ as a set of recursive processes. Similarly, for $BX = 1 + A \cdot X$ on $SL(\mathsf{Set})$:

$$\mathcal{M}_\Theta(x) = a \cdot \mathcal{M}_\Theta(x) \ , \qquad \mathcal{M}_\Theta(y) = (a \cdot \mathcal{M}_\Theta(y)) \vee (b \cdot \mathcal{M}_\Theta(x)) \ .$$

References

1. P. Aczel. *Non-well-founded sets.* Number 14 in Lecture Notes. CSLI, 1988.
2. M.A. Arbib and E.G. Manes. *Arrows, Structures, and Functors.* Academic Press, 1975.
3. J.W. de Bakker and J.-J.Ch. Meyer. Metric semantics for concurrency. *BIT*, 28:504–529, 1988.
4. M. Barr. Terminal coalgebras in well-founded set theory. *Theoretical Computer Science*, 144(2):299–315, 1993.
5. B. Bloom, S. Istrail, and A.R. Meyer. Bisimulation can't be traced. *Journal of the ACM*, 42(1):232–268, jan 1995. A preliminary report appeared in *Proc. 3rd LICS*, pages 229-239, 1988.
6. 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.
7. M.C.B. Hennessy and G.D. Plotkin. Full abstraction for a simple parallel programming language. In J. Bečvář, editor, *Proc. 8th MFCS*, volume 74 of *LNCS*, pages 108–120. Springer-Verlag, 1979.
8. P.T. Johnstone. Adjoint lifting theorems for categories of algebras. *Bull. London Math. Soc.*, 7:294–297, 1975.
9. A. Kock. Strong functors and monoidal monads. *Arch. Math. (Basel)*, 23:113–120, 1972.
10. R. Milner. *A Calculus of Communicating Systems*, volume 92 of *LNCS*. Springer-Verlag, 1980.
11. R. Milner. Functions as processes. In M.S. Paterson, editor, *Proc. of 17th ICALP*, 1990.

12. E. Moggi. Notions of computation and monads. *Information and Computation*, 93:55–92, 1991.
13. G.D. Plotkin. A structural approach to operational semantics. Technical Report DAIMI FN-19, Computer Science Department, Aarhus University, 1981.
14. 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.
15. M. Smyth and G. Plotkin. The category-theoretic solution of recursive domain equations. *SIAM J. Comput.*, 11:761–783, 1982.
16. D. Turi. *Functorial Operational Semantics and its Denotational Dual*. PhD thesis, Free University, Amsterdam, June 1996. Accessible from `<http://www.dcs.ed.ac.uk/home/dt/>`.
17. D. Turi and G.D. Plotkin. Towards a mathematical operational semantics. In *Proc. 12*$^{\mathrm{th}}$ *LICS Conf.* IEEE, Computer Society Press, 1997.