PROD. TYPE: COM
PP: 1-20 (col.fig.: nil)

TCS5266 DTD VER: 5.0.1

ED: Sudha
PAGN: Vish – SCAN: nil

ARTICLE IN PRESS

Available online at www.sciencedirect.com

SCIENCE DIRECT®

Theoretical Computer Science

ELSEVIER

Theoretical Computer Science ▮▮▮ (▮▮▮▮) ▮▮▮–▮▮▮

www.elsevier.com/locate/tcs

Category theory for operational semantics

Marina Lenisa[a,*,1], John Power[b,2], Hiroshi Watanabe[c,3]

[a] *Dipartimento di Matematica e Informatica,Università di Udine, I-33100 Udine, Italy*

[b] *Laboratory for the Foundations of Computer Science,University of Edinburgh, King's Buildings, Edinburgh EH9 3JZ, Scotland, UK*

[c] *Laboratory for Verification and Semantics, AIST, Amagasaki 661-0974, Japan*

Abstract

We use the concept of a distributive law of a monad over a copointed endofunctor to define and develop a reformulation and mild generalisation of Turi and Plotkin's notion of an abstract operational rule. We make our abstract definition and give a precise analysis of the relationship between it and Turi and Plotkin's definition. Following Turi and Plotkin, our definition, suitably restricted, agrees with the notion of a set of *GSOS*-rules, allowing one to construct both an operational model and a canonical, internally fully abstract denotational model. Going beyond Turi and Plotkin, we construct what might be seen as large-step operational semantics from small-step operational semantics and we show how our definition allows one to combine distributive laws, in particular accounting for the combination of operational semantics with congruences.

Keywords: ■; ■; ■

1. Introduction

In order to describe a programming language completely, one requires both operational semantics and denotational semantics [25]. Operational semantics describes the execution

* Corresponding author.

E-mail addresses: lenisa@dimi.uniud.it (M. Lenisa), ajp@inf.ed.ac.uk (J. Power), hirowata@ni.aist.go.jp (H. Watanabe).

[1] This work is supported by the MIUR Project COFIN 2001013518 COMETA and by the UE Project IST-2000-29001 TYPES.

[2] This work is supported by EPSRC grant GR/586372/01: A Theory of Effects for Programming Languages.

[3] This work is supported by MEXT through Grant-in-Aid for Young Scientists Category B, 14780251.

0304-3975/$ - see front matter © 2004 Published by Elsevier B.V.
doi:10.1016/j.tcs.2004.07.024

of programs, while denotational semantics allows one to reason about the mathematical entities that the programs are supposed to address. So one wants a syntax, an operational semantics, and a denotational semantics, together with a proof that the denotational semantics is *adequate*, i.e., that it is consistent with the operational semantics [23].

Operational semantics is typically described in terms of atomic, elementary transitions, describing local behaviour. Mathematically, the transitions are the elements of a relation, the intended operational model of the language. The transition relation is usually described by induction on the structure of the program, starting from operational rules for the basic constructs of the language [18].

Denotational semantics is a mapping of programs into a suitable semantic domain endowed with an operation for each basic construct of the language. For languages without variable binding but possibly multi-sorted, a denotational model is given by a Σ-algebra, where Σ is a signature consisting of the language constructs. The programs of the language form the initial Σ-algebra, and the induced unique homomorphism from the set of programs to the denotational model is called the *initial algebra semantics* of the language [9].

The carrier of a denotational model is often given by the final solution of a domain equation $X \cong B(X)$ for a behaviour functor B. The transition relation, and therefore the intended operational model of the language, forms a B-coalgebra. Finality induces a unique coalgebra map from the intended operational model to the denotational model, and that map is called the *final coalgebra semantics* of the language [22]. Under some assumptions on B, it is fully abstract with respect to behavioural equivalence. When the initial algebra and final coalgebra semantics agree, one has an adequate denotational semantics [22].

Adequacy is often difficult to prove, and so one seeks general criteria from which one can deduce it. For process algebras, as used for specifying non-deterministic and concurrent programs [2,17], one can give syntactic restrictions on the format of operational rules that force bisimulation [17] to be a congruence. Among such rules, *GSOS* rules [4] are among the most popular and general. So Turi and Plotkin [25] presented a category theoretic formulation of *GSOS* rules and proved a general adequacy result, allowing them to deduce a general congruence result. The central observation from which the rest of their analysis flowed was that image-finite sets of *GSOS* rules may be described, up to the obvious syntactic equivalence of rules, exactly as natural transformations between a pair of composite functors constructed from the signature Σ and the behaviour B. Their definition generalised from the base category *Set* to an axiomatically defined category, and also from specific classes of signatures and behaviour functors, allowing, for instance, analysis of probabilistic nondeterminism [3] and timed processes [14]. We recall Turi and Plotkin's characterisation of *GSOS* rules and their general definition of an *abstract operational rule* in Section 2.

Pursuing the direction proposed by Turi and Plotkin, we show, in Section 3, that under mild conditions on an axiomatically defined base category, their abstract operational rules amount exactly to distributive laws of the free monad T on the signature Σ over the cofree copointed endofunctor H on the behaviour endofunctor B. We can thus replace their definition of an abstract operational rule by the notion of a distributive law of a monad over a copointed endofunctor. Inherently, that replacement mildly generalises their setting, in that we need not restrict to freeness of T and cofreeness of H. The additional generality allows us to incorporate more sophisticated formulations of structural operational semantics such as

those involving a combination of transitions with congruences [5]. But, more importantly, while being similarly computationally natural, our reformulation is mathematically more elegant. In particular, it allows us more elegant proofs of some of their results and it yields a deeper understanding of the computational significance of constructs that naturally arise in developing their idea: the heart of their idea involves generalisation from the ordinary notion of a transition system to that of a B-coalgebra for an arbitrary endofunctor B on an a arbitrary category C, subject to axiomatically defined conditions; then to study the constructs associated with operational and denotational semantics at that much greater level of generality.

For an example of what we gain here, under conditions on the base category C and under mild conditions on the behaviour functor B, the latter generates a cofree comonad D. Every abstract operational rule extends uniquely to a distributive law of the monad T over the comonad D. Turi and Plotkin used the existence of that extension to prove a general adequacy result. But a particular construction of the cofree comonad D has computational significance in its own right: implicit in it is the process of passing from small-step operational semantics to large-step operational semantics. A distributive law $TH \Rightarrow HT$, where H is a copointed endofunctor, is, by the above, a generalisation of the notion of a transition function. Now consider the composite $THH \Rightarrow HTH \Rightarrow HHT$. One must introduce an equaliser into HH in order to force the target of one transition agree with the source of the following one, but subject to that, the composite exactly instantiates to the two-step transition function induced by the transition function in all the leading examples. In the limit, extending from two to an arbitrary number, one has constructed the cofree comonad D on the copointed endofunctor H, and one obtains the induced distributive law of the monad T over the comonad D. Sometimes, that limit does not exist although the induced distributive law does exist, cf [1,26], but, subject to mild conditions on C, the approximants always do exist and act as approximants to the distributive law. So the induced distributive law may be seen as the large-step operational semantics induced by the abstract operational rule. The details of this appear in Section 4.

For another example, in Section 5, we consider two mathematical constructs that combine distributive laws. One is given by taking distributive laws $TH \Rightarrow HT$ and $T'H \Rightarrow HT'$ and inducing a distributive law of the form $(T+T')H \Rightarrow H(T+T')$: computationally, this shows how, at the level of generality essentially proposed by Turi and Plotkin, one can add operations. The other, more profoundly, is given by taking a coequaliser of T and inducing a distributive law of the coequalised monad $T[E]$ over H: computationally, that amounts to letting T be subject to equations E, and shows how one may combine operational semantics with a congruence at the level of generality espoused here, agreeing with [5].

An obvious question arising from the work of this paper is how to generalise it to more sophisticated notions of signature, such as those relating to computational effects (see for instance [10]). Also, one might explore the distinction between terms and states, as behaviour is ultimately about state rather than terms. A start in this direction was considered in [7]. And of course, one would like to apply this analysis to further computational examples (see for instance [3]) and to extend it to incorporate binders as advocated in [6,20,24]. Further, just as one has monads that are not freely generated by signatures, one can use the setting of this paper to consider comonads that are not cofreely generated by endofunctors, for instance in analysing timing [14,15].

This paper incorporates the workshop paper [19] with part of the workshop paper [16]. By the very nature of the paper, we owe particular thanks to Gordon Plotkin and Daniele Turi for the computational foundation on which it is written.

2. The motivating example: GSOS

This section recalls and mildly reorganises Turi and Plotkin's proof that *GSOS* rules amount exactly to a class of natural transformations, and gives their definition of an abstract operational rule. We need to recall their work in some detail here in order to make this paper complete and comprehensible.

Consider the language with signature Σ given by a constant symbol *nil*, a set of unary prefixing operators indexed by a finite set A of actions ranged over by a, and a binary parallel composition operator $\|$. This signature generates, for every set X of variables x, the set TX of terms t given by the abstract grammar

$$t ::= x \mid nil \mid a.t \mid t \parallel t.$$

The set TX is the carrier of the free Σ-algebra on X. Now let the operational rules R inductively defining the labelled transitions performable by the programs of the language be

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

The behaviour of the language is given by $BX = (P_{fi}X)^A$, where $P_{fi}X$ denotes the set of finite subsets of X. Let x and y range over X, β range over $(P_{fi}X)^A$, and write $a \mapsto \{x_1, \cdots, x_n\}$ for the function from A to $P_{fi}X$ sending a to $\{x_1, \cdots, x_n\}$ and sending everything else to the empty set. Then, for each operator σ of Σ, the corresponding rules yield the function

$$M(\sigma) : (X \times (P_{fi}X)^A)^{\mathrm{arity}(\sigma)} \longrightarrow (P_{fi}TX)^A$$

defined by

$$M(nil) = a \mapsto \emptyset,$$

$$M(a.)(x, \beta) = a \rightsquigarrow \{x\},$$

$$(x, \beta)M(\|)(y, \beta') = a \mapsto \{x' \parallel y \mid x' \in \beta(a)\} \cup \{x \parallel y \mid y \in \beta'(a)\}.$$

These can be combined into a single function of the form

$$M(R)_X : 1 \sqcup \left(\coprod_A (X \times BX)\right) \sqcup (X \times BX)^2 \longrightarrow BTX$$

and this function is natural in X because, for every renaming of variables, first renaming then applying the rules is the same as first applying the rules then renaming. This example motivates a general definition of a *GSOS* rule as follows.

Definition 1. Let A_i and B_i range over subsets of A. A *GSOS*-rule is a rule of the form

$$\frac{\{x_i \xrightarrow{a} y_{ij}^a\}_{1\leqslant j\leqslant m_i^a}^{1\leqslant i\leqslant n, a\in A_i} \{x_i \overset{b}{\not\rightarrow}\}_{b\in B_i}^{1\leqslant i\leqslant n}}{\sigma(x_1,\cdots,x_n)\xrightarrow{c} t}$$

such that the x_i and y_{ij}^a are all distinct, and those are the only variables that appear in the term t.

Two sets of *GSOS*-rules are called *equivalent* if they prove the same rules in the precise sense of Definition 2.5 of [8].

Now suppose we are given a set R of *GSOS*-rules that is *image finite* in the sense that there are finitely many rules for each operator σ in Σ and action c in A. For every set X, one can associate with R a function

$$M(R)_X : \coprod_{\sigma\in\Sigma} (X \times (P_{fi}X)^A)^{\mathrm{arity}(\sigma)} \longrightarrow (P_{fi}TX)^A$$

as follows: for all t in TX, c in A, x_i in X, and β_i in $(P_{fi}X)^A$, put

$$t \in M(R)_X(\sigma((x_1,\beta_1),\cdots,(x_n,\beta_n)))(c)$$

if and only if there exists a (possibly renamed) rule in R such that $\{y_{i1}^a,\cdots,y_{im_i^a}^a\}$ is a subset of $\beta_i(a)$ for a in A_i, and $\beta_i(b)$ is empty for b in B_i. Turi and Plotkin's central motivating theorem [25] is as follows.

Theorem 2. *The construction $M(-)$ is a bijection from equivalence classes of image finite sets of GSOS-rules for a signature Σ over a fixed denumerably infinite set of variables V to natural transformations of the form*

$$\coprod_{\sigma\in\Sigma} (X \times (P_{fi}X)^A)^{\mathrm{arity}(\sigma)} \longrightarrow (P_{fi}TX)^A.$$

Observe the generality of natural transformations of the form of $M(R)$. Every signature Σ generates an endofunctor on *Set*, also denoted by Σ, defined as follows:

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

The functor T is given by the free monad on the endofunctor Σ, i.e., it is determined by the left adjoint to the forgetful functor from Σ-*Alg* to *Set*. And one can generalise from the endofunctor $BX = (P_{fi}X)^A$ to an arbitrary endofunctor. Those observations, together with a general theorem about combined operational and denotational semantics, led to Turi and Plotkin's abstract category theoretic formulation of the notion of a set of *GSOS*-rules, vastly generalising the usual syntactic description, as follows.

Definition 3. Given a category with finite products C and endofunctors Σ and B on C such that a free monad T on Σ exists, an *abstract operational rule* is a natural transformation of

the form

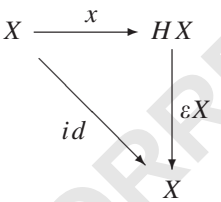
$$\rho : \Sigma(Id \times B) \Rightarrow BT.$$

We shall explore this definition in the next section, but observe the generality here. Even in the case of $C = Set$, the definition allows Σ and B to be arbitrary subject to the existence of T. So, implicit in the definition is a generalisation of the notion of transition system to that of B-coalgebra for an arbitrary endofunctor B: to give an image-finite transition system is to give a B-coalgebra for $B = (P_{fi}-)^A$. But arbitrary endofunctors can look very different to $(P_{fi}-)^A$, and so, a priori, the various intuitive ideas one has about transition systems might not lift at all well to this generality. For instance, as we shall explore later, one might ask, given an arbitrary endofunctor B and a B-coalgebra, which we consider as a generalised notion of a transition system, what could one mean by a two-step transition? Using the usual notions of transition system, it is obvious; but that does not, a priori, make it obvious here. The notion of signature has been more developed in this generality (see for instance [13]), but one might still ask how to understand a transition function in this generality rather than just in specific examples. We explore some of the possibilities in the following sections.

3. Abstract operational rules as distributive laws

In this section, we see that Turi and Plotkin's definition of an abstract operational rule is equivalent to giving a distributive law of a monad over a copointed endofunctor. The easiest proof, albeit not the most direct one, involves use of the categories of coalgebras for an endofunctor B and coalgebras for a copointed endofunctor (H, ε). So we shall develop and use those notions here when convenient.

Definition 4. A *copointed endofunctor* on a category C is an endofunctor $H : C \longrightarrow C$ together with a natural transformation $\varepsilon : H \Rightarrow Id$. An (H, ε)-*coalgebra* is an object X of C together with a map $x : X \longrightarrow HX$ such that



commutes.

The evident definition of a map of (H, ε)-coalgebras yields the category (H, ε)-*Coalg* of (H, ε)-coalgebras.

Definition 5. Given a copointed endofunctor (H, ε) on C, the right adjoint to the forgetful functor

$$U : (H, \varepsilon) - Coalg \longrightarrow C$$

if it exists, is the *cofree comonad* on (H, ε).

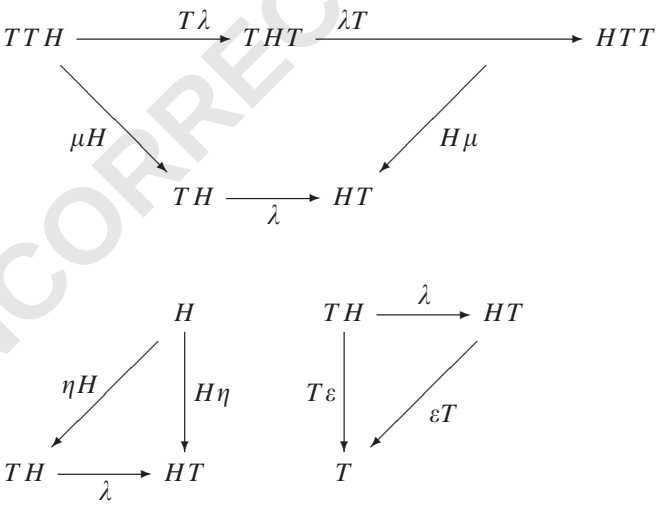
This definition is more subtle than it may appear. One could readily define the category $Cmd(C)$ of comonads on C and the category $PtdEnd(C)$ of endofunctors on C, the maps being natural transformations that respect the structure. There is a forgetful functor from $Cmd(C)$ to $PtdEnd(C)$. In general, that functor does not have a right adjoint. But if one has a particular pointed endofunctor (H, ε), the cofree comonad on (H, ε) satisfies the universal property required of a right adjoint for the particular object (H, ε) of $PtdEnd(C)$. The converse does not hold in general, i.e., this universal property is not sufficient to prove one has a cofree comonad in the sense in which we have defined it above [12], i.e., it might not satisfy the stronger property which we have used to define the notion of cofree comonad.

One can similarly define a category $End(C)$, and there is a forgetful functor from $PtdEnd(C)$ to $End(C)$. If C has finite coproducts, this functor has a right adjoint sending an endofunctor B to $(B \times Id, \pi_2)$. Moreover, the categories B-*Coalg* and $(B \times Id, \pi_2)$-*Coalg* are canonically isomorphic: note that B-*Coalg* is the category of coalgebras for the endofunctor B while $(B \times Id, \pi_2)$-*Coalg* is the category of coalgebras for the *copointed* endofunctor $(B \times Id, \pi_2)$. So, in contrast to the situation for a cofree comonad, we may use right adjointness to *define* the notion of cofreeness of a copointed endofunctor, and in the presence of finite products, we can construct it as above.

The notion of the cofree comonad on an endofunctor is defined in the same spirit as that of the cofree comonad on a copointed endofunctor. It follows that the cofree comonad on the endofunctor B agrees with the cofree comonad on the copointed endofunctor $(B \times Id, \pi_2)$, either existing if the other does: a small amount of care is required in regard to existence, as outlined above and as explained in [12], but mistakes in this setting are most unlikely.

We now move towards showing that the definition of an abstract operational rule is equivalent to giving a distributive law of the monad T over the cofree copointed endofunctor $(B \times Id, \pi_2)$ on B.

Definition 6. A *distributive law* of a monad (T, μ, η) over a copointed endofunctor (H, ε) is a natural transformation $\lambda : TH \Rightarrow HT$ that makes the following diagrams commute:



It is often difficult to calculate directly with monads, but the following theorem will allow us to deduce existence of useful constructions on abstract operational rules, using monads, without need for explicit calculation.

Theorem 7. *Given a monad T and a copointed endofunctor (H, ε), to give a distributive law of T over (H, ε) is equivalent to giving a lifting $(\bar{H}, \bar{\varepsilon})$ of (H, ε) to T-Alg.*

Proof. The constructions are routine, as is the proof of equivalence. For example, given a distributive law $\lambda : TH \Rightarrow HT$, the lifting of (H, ε) is given on objects by sending a T-algebra (X, h) to HX with action

$$THX \xrightarrow{\lambda X} HTX \xrightarrow{Hh} HX.$$

One routinely checks that this action satisfies the axioms for a T-algebra. The inverse is obtained by applying a lifting to the free T-algebra on X, i.e., to $(TX, \mu X)$.

Let (H, ε) be a copointed endofunctor on a category C. A natural transformation $\rho : \Sigma H \Rightarrow HT$ *respects the structure of the copointed endofunctor* (H, ε) if the following diagram commutes:

$$\begin{array}{ccc} \Sigma H & \xrightarrow{\rho} & HT \\ {\scriptstyle \Sigma\varepsilon}\downarrow & & \downarrow{\scriptstyle \varepsilon T} \\ \Sigma & \xrightarrow[\theta]{} & T \end{array}$$

where $\theta : \Sigma \Rightarrow T$ is the canonical natural transformation exhibiting T as the free monad on the endofunctor Σ. □

Proposition 8. *To give an abstract operational rule $\rho : \Sigma(B \times Id) \Rightarrow BT$ is equivalent to giving a natural transformation $\varrho : \Sigma(B \times Id) \Rightarrow (B \times Id)T$ that respects the structure of the copointed endofunctor $(B \times Id, \pi_2)$.*

Proof. For each natural transformation $\varrho : \Sigma(B \times Id) \Rightarrow (B \times Id)T$ that respects the structure of $(B \times Id, \pi_2)$, the second component must be

$$\Sigma(B \times Id) \xrightarrow{\Sigma\pi_2} \Sigma \xrightarrow{\theta} T. \tag{1}$$

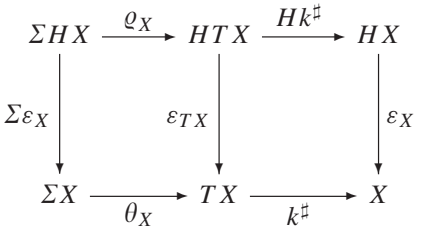
So, to give a natural transformation $\varrho : \Sigma(B \times Id) \Rightarrow (B \times Id)T$ that respects the structure of $(B \times Id, \pi_2)$ is equivalent to giving its first component $\Sigma(B \times Id) \Rightarrow BT$, i.e., an abstract operational rule. □

Proposition 9. *For any copointed endofunctor (H, ε), to give a natural transformation $\varrho : \Sigma H \Rightarrow HT$ respecting the structure of (H, ε) is equivalent to giving a distributive law of the free monad T on Σ over (H, ε).*

Proof. Given ϱ, we first show that the endofunctor H lifts to an endofunctor $\bar{H}$ on the category Σ-Alg, and the natural transformation $\varepsilon : H \Rightarrow Id$ lifts to $\bar{\varepsilon} : \bar{H} \Rightarrow Id$.

Define the action of $\bar{H} : \Sigma\text{-}Alg \to \Sigma\text{-}Alg$ as follows: a Σ-algebra $k : \Sigma X \to X$ is sent to $Hk^\sharp \circ \varrho_X$, where $k^\sharp : TX \to X$ is the corresponding Eilenberg–Moore algebra for the monad (T, μ, η) under the isomorphism $\Sigma\text{-}Alg \cong T\text{-}Alg$. An arrow f of Σ-algebras from $k : \Sigma X \to X$ to $l : \Sigma Y \to Y$, i.e., an arrow $f : X \to Y$ in C satisfying $f \circ k = l \circ \Sigma f$, is sent to $Hf : HX \to HY$. The functor $\bar{H} : \Sigma\text{-}Alg \to \Sigma\text{-}Alg$ is a lifting of H.

Next, for each Σ-algebra $k : \Sigma X \to X$, observe that the X component $\varepsilon_X : HX \to X$ of ε is a morphism of Σ-algebras from $\bar{H}k$ to k, i.e., $\varepsilon_X \circ \bar{H}k = k \circ \Sigma\varepsilon_X$: since the natural transformation ϱ respects the structure of (H, ε), both squares in the following diagram commute:



Since the bottom arrow of the diagram is $k^\sharp \circ \theta_X = k$ and the top arrow is $\bar{H}k$, the arrow $\varepsilon_X : HX \to X$ is a morphism of Σ-algebras from $\bar{H}k$ to k. So we may define $\bar{\varepsilon} : \bar{H} \Rightarrow Id$ by defining its $k : \Sigma X \to X$ component to be ε_X. Its naturality follows from naturality of ε. It is evidently a lifting of ε to Σ-Alg.

Because Σ-Alg is isomorphic to T-Alg, both the functor $\bar{H}$ and the natural transformation $\bar{\varepsilon} : \bar{H} \Rightarrow Id$ are liftings of H and ε to T-Alg. By Theorem 7, to give such a lifting is equivalent to giving a distributive law of the monad (T, μ, η) over the copointed endofunctor (H, ε).

For the converse construction, compose such a distributive law with the canonical natural transformation from Σ to T that exhibits T as the free monad on Σ. The two constructions are routinely verified to be inverse. □

The construction of Proposition 9 gives us the unique canonical extension of an abstract operational rule to all terms. From Propositions 8 and 9, we conclude the following.

Theorem 10. *To give an abstract operational rule $\rho : \Sigma(B \times Id) \Rightarrow BT$ is equivalent to giving a distributive law $\lambda : T(B \times Id) \Rightarrow (B \times Id)T$ of the monad (T, μ, η) over the copointed endofunctor $(B \times Id, \pi_2)$.*

This result is not only elegant in its own right, but it also suggests possible greater generality in a direction that is computationally significant: one could consider a distributive law of an arbitrary monad T over an arbitrary copointed endofunctor H, without insisting that T be free on an endofunctor or that H be cofree on one. The former possibility is implicit in the work of [5], where a combination of operational semantics with congruences is considered. The work of [13] gives a syntactic way to construct arbitrary finitary monads T from a signature together with equations between derived terms, and one can readily extend that to construct distributive laws. We start to develop the significance of that idea in Section 5.

The theorem is also computationally natural: one can pass similarly easily between *GSOS* rules and distributive laws as between *GSOS* rules and abstract operational rules. Abstract operational rules describe the behaviour of terms of the form $f(x_1, \cdots, x_n)$ for symbols f of Σ, while distributive laws describe behaviour of arbitrary terms t: the description for arbitrary t trivially restricts to that for each $f(x_1, \cdots, x_n)$, while the other direction is given by induction on the complexity of t.

4. The cofree comonad

Given an abstract operational rule, equivalently a distributive law of a monad T over the copointed endofunctor $B \times Id$ by Theorem 10, one can readily derive a distributive law of T over the cofree comonad D on B, whenever the latter exists. There have been many theorems giving conditions on C and B that force the cofree comonad to exist, see for instance [1,11]. Suffice it to say here that all our leading examples are covered. So we shall simply assume henceforth that a cofree comonad D on B does exist. One easy, albeit indirect argument, that yields the unique extension of a distributive law of T over the cofree copointed endofunctor H on B to a distributive law of T over the cofree comonad D on B is as follows.

Proposition 11. *Given a monad T and a copointed endofunctor (H, ε), every distributive law of T over (H, ε) induces a lifting $\bar{T}$ of T to (H, ε)-Coalg.*

The construction is routine, given by the dual of that for Theorem 7. We do not assert a converse. But we do have the following (see for instance [21]).

Theorem 12. *Given a monad T and a comonad D, to give a distributive law of T over D is equivalent to giving a lifting $\bar{T}$ of T to D-Coalg.*

The constructions and proofs extend the dual of those for Theorem 7. Combining Proposition 11 with Theorem 12 under the assumption that D is the cofree comonad on the copointed endofunctor (H, ε), i.e., assuming D is the right adjoint to the forgetful functor from (H, ε)-*Coalg* to C, we immediately have the result we seek, as follows.

Corollary 13. *Given a monad T and a copointed endofunctor H, and assuming that a cofree comonad D on H exists, every distributive law of T over H extends uniquely to a distributive law of T over D.*

Turi and Plotkin use this result to give a general account of adequacy [25], but we shall take a closer look at a particular construction of the cofree comonad. This construction does not always exist, but it is of direct computational significance when it does, specifically in regard to dynamics.

Dynamics are fundamental to programming, as one considers safety and liveness issues for example. Moreover, our leading class of examples arise from concurrency constructs, with our analysis of nondeterminism and a parallel operator. So it is a natural, relevant question how to generate dynamic structures from an abstract operational rule, equivalently

a distributive law. It is, of course, routine to consider dynamic constructs on a case-by-case basis, but if one is to take the definition of abstract operational rule seriously, one wants constructs at that level of generality. And, in particular examples, those constructs should agree with extant ones, suggest interesting alternatives to extant ones, or suggest possibilities in cases that have never previously been considered.

In order to address dynamic issues, one needs to consider a generalised notion of a stream of transitions. Formally, that generalised notion will be far more general than the usual notion of a stream of transitions: the reason being that the generalised notion of transition system we have implicitly adopted, i.e., a coalgebra for an arbitrary endofunctor, is far more general than the usual notion of transition system. In particular, it means that our generalised notion includes not only the usual notion of stream of transitions but also, when one invokes the finite powerset functor, the notion of a tree of choices of them. Our level of generality also means that it is not possible to reduce our notion to something resembling a standard notion, just as it is not possible to reduce the notion of an arbitrary endofunctor to one of the standard examples. It may be possible to find general theorems that characterise streams relative to the endofunctor given as a parameter, but for the present, the best we have is as follows.

In order to give a notion of a stream of transitions, one first needs to be able to describe two-step transitions $t_0 \to t_1 \to t_2$ generated by a transition function, i.e., generated by an abstract operational rule, equivalently a distributive law of a monad T over a copointed endofunctor (H, ε), with leading examples having (H, ε) cofree on an endofunctor B. A behaviour functor B a priori allows one to speak of one step of a transition system. A transition system is *defined* to be a coalgebra $x : X \longrightarrow BX$, where an element of BX is a potential result of one step of the transition system, corresponding to all possible first steps in the usual operational sense: note that nondeterminism is normally present in the leading examples. An element of BBX gives the result of two steps of the transition system represented by the coalgebra (X, x), by considering the composite

$$X \xrightarrow{x} BX \xrightarrow{Bx} BBX.$$

But, in the leading class of examples, that does not agree with the composite of transitions in the usual sense as it does not record the intermediate state.

Example 14. Let $BX = X^A$. Then $BBX = X^{A\times A}$. Given a coalgebra (X, x), consider the composite

$$X \xrightarrow{x} X^A \xrightarrow{x^A} X^{A\times A}.$$

An element of X is sent to an element of X with label (a, a'), but the composite does not record which intermediate state was visited, i.e., which state was visited after the a-transition and before the a'-transition. So the information given by BBX does not agree with the usual notion of two steps of the transition system.

In order to avoid examples such as this, we need something more sophisticated than BB. The next obvious idea is to consider HH, where H is the cofree copointed endofunctor on B.

As we shall see later, that also has a mathematical advantage of giving an obvious possible composite for a distributive law, i.e.,

$$THH \Rightarrow HTH \Rightarrow HHT$$

which is encouraging, and it does record intermediate states. But it too is not quite right but for the opposite reason, creating difficulty for the other leading example of a behaviour.

Example 15. Let $B = P_{fi}$, the finite powerset functor. We thus have the cofree copointed endofunctor given by $HX = P_{fi}X \times X$. So the composite is $HHX = P_{fi}(P_{fi}X \times X) \times P_{fi}X \times X$. But this gives too much freedom: for a two-step transition, one needs an element of the second component of the product to agree with an element of the second subcomponent of the first component of the product in order to make the target of the first transition agree with the source of the second one. So although the composite

$$X \xrightarrow{(x, id)} P_{fi}X \times X \xrightarrow{(P_{fi}(x), id)} P_{fi}(P_{fi}X \times X) \times P_{fi}X \times X$$

is right, its codomain is not, posing difficulty in iterating to a third step.

In order to avoid this example, one needs to introduce an equaliser. That equaliser is a remarkably simple one: we put H_2X equal to the equaliser of the maps

$$H\varepsilon X, \varepsilon HX : HHX \longrightarrow HX.$$

And we define $\varepsilon_2 : H_2 \Rightarrow Id$ by composition.

Proposition 16. *For any (H, ε)-coalgebra (X, x), the composite*

$$X \xrightarrow{x} HX \xrightarrow{Hx} HHX$$

is an (H_2, ε_2)-coalgebra.

Proof. One needs check that the composite composed with $H\varepsilon X$ and εHX is the same, but that follows directly from the naturality of the copoint and from the definition of an (H, ε)-coalgebra. Satisfaction of the coherence condition, i.e., that composition with the counit yields the identity, is immediate. □

Example 17. We continue our consideration of the finite powerset functor $B = P_{fi}$. Recall that the cofree copointed endofunctor H on B is given by $HX = P_{fi}X \times X$, with $\varepsilon X : HX \longrightarrow X$ being the second coprojection. The set H_2X is defined to be an equaliser of two maps with domain $HHX = P_{fi}(P_{fi}X \times X) \times P_{fi}X \times X$. With a small amount of calculation, one can verify, by the equalising condition and by considering finite unions, that an element of H_2X may be characterised as an element t_0 of X, a finite subset T_1 of X, and the assignment to each element t_1 of T_1, of a further finite subset T_2 of X: one can prove directly that the collection of these acts as an equaliser. Given a B-coalgebra (X, x), one can readily prove that the composite

$$X \xrightarrow{(x, id)} P_{fi}X \times X \xrightarrow{(P_{fi}(x), id)} P_{fi}(P_{fi}X \times X) \times P_{fi}X \times X$$

makes $H\varepsilon X$ and εHX equal. This composite sends an element t of X to the triple for which an element can be characterised by the element $t_0 = t$ of T, the set T_1 given by taking one step in the transition system x from t_0, together with, for each t_1 in T_1, the set of further transitions one can take from t_1. Thus we have precisely the two-step transitions.

With some thought, the process of constructing H_2 from H can be iterated: one can define H_n axiomatically such that, in the leading example, given a B-coalgebra, the induced n-fold composite function from X to H_nX sends an element t of X to the set of n-step transitions one can make from t. We shall spell out the details shortly, but first we remark that although, in this particular example, the process converges at ω, that is not true in general: the construction we define need not converge at ω owing to lack of uniformity [1,26]; we shall consider an example later. The construction we need here is the dual of a construction hidden deep inside Kelly's paper [12]. We describe the dual, i.e, the form we want, here.

Definition 18. Given a copointed endofunctor $(H, \varepsilon{:}H \Rightarrow Id)$ on a base category C with all limits, and given an object X, put

$$X_0 = X \qquad X_1 = HX \qquad x_0 = id_{HX} : X_1 \to HX_0.$$

Now define $X_{\beta+2}$ and $x_{\beta+1} : X_{\beta+2} \longrightarrow HX_{\beta+1}$ by the equaliser of

$$HX_{\beta+1} \xrightarrow{Hx_\beta} H^2X_\beta \xrightarrow{\varepsilon HX_\beta} HX_\beta,$$

with

$$HX_{\beta+1} \xrightarrow{Hx_\beta} H^2X_\beta \xrightarrow{H\varepsilon X_\beta} HX_\beta.$$

For arbitrary β, we define $X_\beta^{\beta+1}{:}X_{\beta+1} \longrightarrow X_\beta$ to be $\varepsilon X_\beta \cdot x_\beta$. Then, for a limit ordinal α, we define $X_\alpha = lim_{\beta<\alpha}X_\beta$ with the X_β^α being the generators of the limit cone for the co-chain, and we define $X_{\alpha+1}$ and $x_\alpha : X_{\alpha+1} \longrightarrow HX_\alpha$ by the equaliser of

$$HX_\alpha \xrightarrow{\varepsilon X_\alpha} X_\alpha = lim_{\beta<\alpha}X_{\beta+1} \xrightarrow{lim_{\beta<\alpha}x_\beta} lim_{\beta<\alpha}HX_\beta$$

with

$$HX_\alpha \longrightarrow lim_{\beta<\alpha}HX_\beta,$$

where the unlabelled map is canonically induced by the limiting property. We say that the sequence converges at α if $X_\alpha^{\alpha+1}$ is an isomorphism.

Observe that the first three steps of this construction, i.e., the definitions of X_0, X_1, and X_2 agree with our constructions regarding zero (implicitly), one, and two-step transitions. The higher ordinals generalise this to arbitrary steps. Transfiniteness arises owing to lack of uniformity: as we shall see, it does not involve consideration of streams of transfinite length.

Example 19. We continue with our leading example of $B = P_{fi}$, the finite powerset functor. By induction, we may assume that the n-step transition system generated by a

transition system, i.e., generated by a B-coalgebra (X, x), is given by the n-fold composite from X to $H_n X$. Now consider $H_{n+1} X$. It is defined to be an equaliser of a pair of maps with domain $H H_n X$, thus an equaliser of a pair of functions with domain $P_{fi} H_n X \times H_n X$. Again using unions, as we did for the case of $n = 1$, and by induction on n, we can characterise an element of the equaliser as an element of $H_n X$ together with an assignment to the last term of a further finite subset of X. By induction, that is exactly what we seek. Moreover, also by induction, the composite induced by x satisfies the equalising property and sends an element t of X to the set of $(n + 1)$-step transitions from t.

Example 20. Consider the replacement of finite powersets by countable powersets in our leading example. We want to allow this possibility as our central assertion is that one can give an axiomatic treatment of transition systems and then of operational semantics in terms of an endofunctor B, subject to axiomatic conditions. A variant of this example arises naturally anyway when one extends the finite powerset functor to allow a countable set of labels on a transition system. The sequence of Definition 18 does not converge at ω. Transfiniteness of the sequence still yields only the usual notion of stream owing to the well-orderedness of all ordinals, with the transfiniteness only arising because of the breadth rather than depth of possibilities.

If the sequence does converge, the dual of Theorem 17.3 of [12] yields a characterisation of it.

Theorem 21. *If the sequence X_β converges at α, the cofree comonad D on H, applied to X, is given by $DX = X_\alpha$ with co-action $x_\alpha : X_\alpha = X_{\alpha+1} \longrightarrow H X_\alpha$.*

There are reasonable conditions under which the sequence does converge, some such conditions being implicit in [1,26] for example.

We now extend from the axiomatic study of transition systems to the axiomatic study of operational semantics. We have characterised approximants to the cofree comonad D on B axiomatically in terms of the passage from a one-step transition system to the induced multi-step transition system. We now check that coheres with the presence of a monad T and a distributive law of T over D. The move from one step to two steps works as follows.

Proposition 22. *Given a distributive law of a monad T over a copointed endofunctor (H, ε), the composite*

$$THH \Rightarrow HTH \Rightarrow HHT$$

induces a distributive law of T over the copointed endofunctor (H_2, ε_2).

Proof. The equalising property for the (composite) map into $HHTX$ follows from the preservation of ε in the definition of a distributive law. For each X, that yields the required map $T H_2 X \longrightarrow H_2 T X$. Its naturality and its respect for the structure of T follow from the unicity part of the notion of equaliser together with the axioms for a distributive law.

We have already checked that our axiomatic definition of (H_2, ε_2) agrees with the examples, in particular with our leading example of finite powersets. The distributive law

$$TH_2 \Rightarrow H_2T$$

of the monad T over the copointed endofunctor H_2 induced by the composite

$$THH \Rightarrow HTH \Rightarrow HHT$$

also agrees with the examples, yielding the normal two-step transition function as follows. □

Example 23. Let $BX = (P_{fi}X)^A$, take Σ to be generated by an arbitrary signature, let T be the free monad on Σ, and suppose we are given a set of *GSOS*-rules

$$\frac{\{x_i \xrightarrow{a} y_{ij}^a\}_{1 \leqslant j \leqslant m_i^a}^{1 \leqslant i \leqslant n, a \in A_i} \{x_i \overset{b}{\not\rightarrow}\}_{b \in B_i}^{1 \leqslant i \leqslant n}}{\sigma(x_1, \cdots, x_n) \xrightarrow{c} t}.$$

Passing from H to H_2 is given by the systematic replacement of all single steps in each rule by a composite pair of steps. So each x_i is assigned both its transitions and the transitions of its transitions, and $\sigma(x_1, \cdots, x_n)$ is also assigned both its transitions and the transitions of its transitions. The distributive law for H determined by the *GSOS*-rules in turn determines a distributive law for H_2 by the formula we have given. It works as follows. The two-step behaviour of $\sigma(x_1, \cdots, x_n)$ is determined by taking a first step based upon the first-step transitions from the x_i's, then taking a second step based upon the second-step transitions of the x_i's.

This process iterates. Given a distributive law of T over (H, ε), by induction on n and by consideration of the composite

$$THH_n \Rightarrow HTH_n \Rightarrow HH_nT$$

one obtains a distributive law, for every n, of T over H_n. One can extend this to arbitrary ordinals by simple use of the properties of category-theoretic limits and limit ordinals. The examples iterate likewise, yielding the multi-step transition function determined by a single step one.

Theorem 24. *If the sequence X_β converges at α, applying the construction of Proposition 22 iterated on ordinals of size less than α to a distributive law of a monad T over a copointed endofunctor (H, ε) yields a distributive law of the monad T over the cofree comonad D on (H, ε). Moreover, that distributive law agrees with the canonical one.*

Proof. The main statement here follows from a tedious inductive proof. The second statement can be seen in several ways, perhaps most easily by the characterisation of distributive laws in terms of liftings to *T-Alg*. □

As this characterises the distributive law, if the sequence converges, we regard this result as implying that, in the particular sense we have described above, the distributive law of the

monad T over the cofree comonad D can reasonably be regarded, at the level of generality we have proposed, as the large-step operational semantics induced by small-step semantics. There are, of course, alternative descriptions and constructions of the cofree comonad and the induced distributive law, but our point here is that, from the particular perspective we have outlined, the construction we have described suggests a computational interpretation of the induced distributive law as being a general construction of large-step operational semantics from small-step operational semantics.

5. Combining distributive laws

In this section, we consider two constructions that allow us to combine distributive laws. Computationally, the first shows how one can add operations axiomatically at the level of generality espoused in this paper, while the other amounts to adding equations. This work axiomatises activity that is already taking place in examples in the literature: it is just a matter of exploiting the fact that abstract operational rules can be characterised as distributive laws, and that the various constructions can be made axiomatically at the level of generality of distributive laws. The axiomatic development often sheds light on previously existing activity. For an example that we do not investigate in this section, primarily because it remains ongoing, see the work on timed processes, its combination with ordinary (non-timed) operational semantics, and how an axiomatic approach in terms of distributive laws informs that [14,15].

The central technical result we need to support the results of this section is as follows. Suppose the category C has products. Given an object X of C, consider the functor $X^{C(-,X)} : C \longrightarrow C$. It sends an object Y to the product of $C(Y, X)$ copies of X. Given an arbitrary endofunctor $\Sigma : C \longrightarrow C$, and given a map $x : \Sigma X \longrightarrow X$, one obtains a natural transformation

$$\chi : \Sigma \Rightarrow X^{C(-,X)}$$

whose Y-component is given, using of the definition of product, by considering the function

$$C(Y, X) \xrightarrow{\Sigma} C(\Sigma Y, \Sigma X) \xrightarrow{C(\Sigma Y, x)} C(\Sigma Y, X).$$

It follows from the Yoneda lemma that this correspondence is an equivalence, i.e., every natural transformation of the form

$$\chi : \Sigma \Rightarrow X^{C(-,X)}$$

arises uniquely via this construction from a map $x : \Sigma X \longrightarrow X$. Summarising:

Proposition 25. *Given an endofunctor Σ on a category C with products, and given an object X of C, to give a natural transformation*

$$\chi : \Sigma \Rightarrow X^{C(-,X)}$$

is equivalent to giving a map $x : \Sigma X \longrightarrow X$, i.e., a Σ-algebra structure (X, x) on the object X.

One can further prove that the functor $X^{C(-,X)}$ possesses a natural monad structure: the unit is easy to understand, its Y-component being given by the evident map from Y to $X^{C(Y,X)}$. The multiplication of the monad is somewhat more complex, given by unfolding a product as above and using two applications of evaluation [13]. The proposition and its proof extend to the following:

Proposition 26. *For a monad T on a category C with products, given an object X of C, to give a map of monads*

$$\chi : T \Rightarrow X^{C(-,X)}$$

is equivalent to giving a T-algebra structure (X, x) on the object X.

We now use this infrastructure to study the situation of two monads T and T', and distributive laws of each of T and T' over a copointed endofunctor H, and we seek to combine them into a distributive law of $T + T'$, if it exists, over H.

Example 27. Let T be the free monad on a signature Σ, and let T' be the free monad on a signature Σ'. Then, the sum $T + T'$ of monads is the free monad on the disjoint union of Σ and Σ', assuming existence: that follows from an argument we shall outline shortly. So, in Turi and Plotkin's terms, given an abstract operational rule of each of Σ and Σ' over B, we shall give an abstract operational rule of the disjoint union of Σ and Σ' over B. Syntactically, that combined abstract operational rule is given by the disjoint union of the set of rules for Σ with the set of rules for Σ'.

We shall work in somewhat greater generality than the example suggests, as we shall not demand that T and T' be freely generated by signatures. This greater generality accords with the use of an equational theory combined with operational semantics, as studied in [5] and as we shall investigate later in this section, and it agrees with our central definition and development of the paper, i.e. a distributive law that need not require that the monad be freely generated (see Theorem 7).

From Proposition 26, one can immediately prove the following.

Proposition 28. *For monads T and T' on a category C with products, if the sum of monads $T + T'$ exists, the category of algebras $(T + T')$-Alg is canonically isomorphic to the pullback*

$$\begin{array}{ccc} P & \longrightarrow & T - Alg \\ \downarrow & & \downarrow U \\ T' - Alg & \xrightarrow[U']{} & C \end{array}$$

This result justifies our assertion in Example 27 that if T and T' are the free monads on signatures Σ and Σ' respectively, the sum $T + T'$ is the free monad on the disjoint union

of Σ and Σ': for an algebra for the disjoint union is given by an object of C together with a Σ-structure and a Σ'-structure on it.

Theorem 29. *Given monads T and T' and a copointed endofunctor (H, ε), and given distributive laws $\lambda : TH \Rightarrow HT$ and $\lambda' : T'H \Rightarrow HT'$, the characterisation of the category of algebras for $T + T'$ as a pullback generates a canonical distributive law of $T + T'$ over (H, ε) whenever the sum of monads $T + T'$ exists.*

Proof. This follows from the combination of Theorem 7 with Proposition 28. By the former, the two distributive laws give liftings of (H, ε) to T-*Alg* and T'-*Alg*, respectively. By the latter, these liftings yield a copointed endofunctor on $(T+T')$-*Alg*, as it is the pullback category P, and that copointed endofunctor necessarily lifts (H, ε). So by an application of the converse part of Theorem 7, we have the distributive law of $T + T'$ over (H, ε) that we seek.

By construction, this combination of distributive laws is associative with an evident unit. To calculate the combined distributive law tends to be complicated because construction of the monad $T + T'$ is usually complex, involving the intertwining of operations generating T with those generating T'. But there is an easy description of the sum if one of the monads is free on an endofunctor [10]: it is $T(\Sigma T)^*$ for monad T and endofunctor Σ, where S^* denotes the free monad on an endofunctor S. In our leading class of examples, as described in Example 27, T and T' are generated by signatures Σ and Σ', the sum of monads $T + T'$ is given by the free monad on the disjoint union of Σ and Σ', and the combined distributive law is generated by the disjoint union of each set of rules.

The imposition of equations upon a signature is modelled by taking a coequaliser in the category of monads on C, just as adding operations was modelled by considering coproducts in the category of monads on C. The central result that supports this perspective is another immediate consequence of Proposition 26, as follows. □

Proposition 30. *Given monads T and T' and monad maps $\tau_1, \tau_2 : T \Rightarrow T'$, if the coequaliser $T'[\tau_1, \tau_2]$ exists, the category of algebras $T'[\tau_1, \tau_2]$-Alg is canonically isomorphic to the equaliser of the pair of functors τ_1-Alg and τ_2-Alg from T'-Alg* to *T-Alg.*

To add equations to an equational theory qua monad T is equivalent to giving an endofunctor $E : C \longrightarrow C$ and a pair of natural transformations $\tau_1, \tau_2 : E \Rightarrow T$ as explained in [13] and as we shall illustrate below. The equational theory generated by T subject to these additional equations is given by the monad obtained by the coequaliser $T[\bar{\tau}_1, \bar{\tau}_2]$, in the category of monads, of the monad maps $\bar{\tau}_1, \bar{\tau}_2 : E^* \Rightarrow T$ where E^* is the free monad on E and $\bar{\tau}_1$ and $\bar{\tau}_2$ are the induced maps. It is routine to see that the proposition supports this by consideration of the algebras of the theory.

We now extend this view of equations to cohere with the axiomatic formulation of operational semantics that we have been developing. The following theorem provides support for a natural extension.

Theorem 31. *Given monads T and T', monad maps $\tau_1, \tau_2 : T \Rightarrow T'$, and a copointed endofunctor (H, ε), and given distributive laws $\lambda : TH \Rightarrow HT$ and $\lambda' : T'H \Rightarrow HT'$*

that respect τ_1 *and* τ_2, *the characterisation of the category of algebras for* $T'[\tau_1, \tau_2]$ *as an equaliser generates a canonical distributive law of* $T'[\tau_1, \tau_2]$ *over* (H, ε) *whenever the coequaliser* $T'[\tau_1, \tau_2]$ *exists*.

The proof of the theorem is essentially the same as that of Theorem 29, and the theorem duly specialises to the situation where the domain monad is freely generated by an endofunctor of the form E as above.

The axioms on the data for the theorem can be complex to check in examples, but they do hold of the leading examples and the theorem does provide a natural and reasonable condition. In fact, the result holds under a milder condition given by taking more seriously the fact of $T'[\tau_1, \tau_2]$ being a coequaliser: we could generalise from the distributive law $\lambda' : T'H \Rightarrow HT'$ to a natural transformation with codomain $HT'[\tau_1, \tau_2]$, subject to evident natural axioms. In practice, e.g., in [5] and in the example below, the interaction between equations and operational semantics is often implicit.

Example 32. Consider the motivating example of Section 2. We had an abstract grammar

$$t ::= x \mid nil \mid a.t \mid t \parallel t$$

with operational rules given as follows:

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

But we could have re-organised the rules, and they sometimes are reorganised in practice.

For instance, we could have kept these rules while, redundantly, imposing symmetry and associativity axioms on the operator $\parallel$ for parallel composition. Had we done so, starting with the first two rules, we could see the third rule as yielding an immediate proof that the distributive law respects the symmetry axiom. With some effort, but essentially repeating work that is already well known in this example, we could further prove that these rules also respect the associativity axiom. But one would not normally present such a combination of rules, as it is well known that to do so involves redundancy.

Alternatively, and more sensibly, we could consider the first two rules, dispense with the third rule, and add the assertion that the parallel operator is to satisfy axioms for symmetry and associativity. The operational semantics would then respect symmetry and associativity because it is *defined* to do so. That is how the information is presented in [5], and our analysis here gives axiomatic support to that.

One can of course extend this example, as in [5], for instance by adding a nondeterministic operator, satisfying axioms such as idempotence, symmetry, and associativity.

References

[1] J. Adamek, V. Koubek, On the greatest fixed point of a set functor, Theoret. Comput. Sci. 150 (1995) 57–75.
[2] J.C.W. Baeten, W.P. Weijland, Process Algebra, Cambridge University Press, Cambridge, 1990.
[3] F. Bartels, GSOS for probabilistic transition systems, Proc. Workshop on Coalgebraic Methods in Computer Science 2002, Electron. Notes in Theoret. Comput. Sci. 65 (2002).

[4] B. Bloom, S. Istrail, A.R. Meyer, Bisimulation can't be traced, J. ACM 42 (1995) 232–268.
[5] A. Corradini, R. Heckel, U. Montanari, From SOS specifications to structured coalgebras: how to make bisimulation a congruence, Proc. Workshop on Coalgebraic Methods in Computer Science 1999, Electron. Notes in Theoret. Comput. Sci. 19 (1999).
[6] M. Fiore, G.D. Plotkin, D. Turi, Abstract syntax and variable binding, Proc. Logic in Comput. Sci. 14 (1999) 193–202.
[7] M. Fiore, D. Turi, Semantics of name and value passing, Proc. Logic in Comput. Sci. 16 (2001) 93–104.
[8] W. Fokkink, R. van Glabbeek, Ntyft/ntyxt rules reduce to ntree rules, Inform. and Comput. 126 (1996) 1–10.
[9] J.A. Goguen, J.W. Thatcher, E.G. Wagner, An initial algebra approach to the specification, correctness and implementation of abstract data types, Current Trends Prog. Methodol. 4 (1978) 80–149.
[10] M. Hyland, G.D. Plotkin, A.J. Power, Combining Effects: Sum and Tensor, submitted.
[11] P.T. Johnstone, A.J. Power, T. Tsujishita, H. Watanabe, J. Worrell, The structure of categories of coalgebras, Theoret. Comput. Sci. 260 (2001) 87–117.
[12] G.M. Kelly, A unified treatment of transfinite constructions for free algebras, free monoids, colimits, associated sheaves, and so on, Bull. Austral. Math. Soc. 22 (1980) 1–83.
[13] G.M. Kelly, A.J. Power, Adjunctions whose counits are coequalisers, and presentations of finitary monads, J. Pure Appl. Algebra 89 (1993) 163–179.
[14] M. Kick, Bialgebraic modelling of timed processes, Proc. ICALP 2002, Lecture Notes in Comput. Sci. 2380 (2002) 525–536.
[15] M. Kick, A.J. Power, Modularity of behaviours for mathematical operational semantics, Workshop on Coalgebraic Methods in Computer Science 2004, Electron. Notes in Theoret. Comput. Sci., to appear.
[16] M. Lenisa, A.J. Power, H. Watanabe, Distributivity for endofunctors, pointed and co-pointed endofunctors, monads and comonads, Proc. Workshop on Coalgebraic Methods in Computer Science 2000, Electron. Notes in Theoret. Comput. Sci. 33 (2000).
[17] R. Milner, A calculus for communicating systems, Lecture Notes in Comput. Sci. 92 (1980).
[18] G.D. Plotkin, A structural approach to operational semantics, Technical Report DAIMI FN-19, Computer Science Department, Aarhus University, 1981.
[19] A.J. Power, Towards a theory of mathematical operational semantics, Proc. Workshop on Coalgebraic Methods in Computer Science 2003, Electron. Notes in Theoret. Comput. Sci. 82.1 (2003).
[20] A.J. Power, A unified category-theoretic approach to variable binding, Proc. MERLIN 03, 2003, to appear.
[21] A.J. Power, H. Watanabe, Combining a monad and a comonad, Theoret. Comput. Sci. 280 (2002) 137–162.
[22] J. Rutten, D. Turi, Initial algebra and final coalgebra semantics for concurrency, Proc. REX Workshop: a Decade of Concurrency—reflections and perspectives, Lecture Notes in Comput. Sci. 803 (1994) 530–582.
[23] D. Scott, Outline of a mathematical theory of computation, Proc. 4th Annual Princeton Conference on Inf. Sciences and Systems, 1970, pp. 169–170.
[24] M. Tanaka, Abstract syntax and variable binding for linear binders, Proc. Mathematical Foundations of Computer Science 00, Lecture Notes in Comput. Sci. 1893 (2000) 670–679.
[25] D. Turi, G.D. Plotkin, Towards a mathematical operational semantics, Proc. Logic in Comput. Sci. 12 (1997) 280–291.
[26] J. Worrell, Terminal sequences for accessible endofunctors, Proc. Workshop on Coalgebraic Methods in Computer Science 1999, Electron. Notes in Theoret. Comput. Sci. 19 (1999).