Towards a theory of mathematical operational semantics

John Power

Laboratory for the Foundations of Computer Science, School of Informatics
University of Edinburgh
King’s Buildings, Edinburgh EH9 3JZ, SCOTLAND
email: ajp@inf.ed.ac.uk

Abstract
Turi and Plotkin gave a precise mathematical formulation of a notion of structural operational semantics in their paper “Towards a mathematical operational semantics.” Starting from that definition and at the level of generality of that definition, we give a mathematical formulation of some of the basic constructions one makes with structural operational semantics. In particular, given a single-step operational semantics, as is the spirit of their work, one composes transitions and considers streams of transitions in order to study the dynamics induced by the operational semantics. In all their leading examples, it is obvious that one can do that and it is obvious how to do it. But if their definition is to be taken seriously, one needs to be able to make such constructions at the level of generality of their definition rather than case-by-case. So this paper does so for several of the basic constructions associated with structural operational semantics, in particular those required in order to speak of a stream of transitions and hence of dynamics.

1 Introduction
Turi and Plotkin, in their paper “Towards a mathematical operational semantics” [13], gave a precise general mathematical formulation of a notion of structural operational semantics. They gave a little abstract development of their definition and they provided several examples. Over the years since then, other authors have refined their definition a little [4], given a little more abstract development of it [10,12], and provided further examples [2,9], some of them, such as those cited, going well beyond the examples originally studied. But although they gave a promising definition of structural operational

1 This work is supported by EPSRC grant GR/M56333: The structure of programming languages : syntax and semantics, and by AIST, Japan

©2003 Published by Elsevier Science B. V. Open access under CC BY-NC-ND license.
semantics in their setting, they did not give a mathematical formulation of the constructions one makes with structural operational semantics. For instance, extending foundational work on coalgebra [6], they gave a mathematical formulation of the notion of a transition function, yielding transitions \( t \rightarrow t' \), but they did not give a mathematical formulation of the induced function that yields a string of transitions \( t_0 \rightarrow t_1 \rightarrow \cdots t_n \). One needs to consider such strings in order to study dynamic issues such as safety and liveness. So, if one is to take Turi and Plotkin’s proposal seriously, one needs to make the construction of this induced function at the level of generality of their definition rather than on a case-by-case basis. So, in this paper, we start to provide a theory of Turi and Plotkin’s mathematical operational semantics by extending their definition with mathematical formulations of some of the basic constructions one makes with structural operational semantics, in particular those constructions that one makes in analysing the dynamic properties induced by a structural operational semantics.

Turi and Plotkin modelled GSOS, more precisely image-finite Generalised Structural Operational Semantics. They started with a base category \( C \) with finite products, a “syntax” endofunctor \( \Sigma \) on \( C \), and a “behaviour” endofunctor \( B \) on \( C \). They modelled a GSOS rule by an abstract operational rule, which they defined to be a natural transformation \( \Sigma(B \times \text{Id}) \Rightarrow BT \), where \( T \) is the free (syntax) monad on \( \Sigma \), which they assumed exists. They showed that this natural transformation determines a distributive law \( \lambda \) of the monad \( T \) over the comonad \( D \), where \( D \) is the cofree comonad on \( B \), which they also assumed exists. This gave them a category \( \lambda\text{-Bialg} \) of \( \lambda \)-bialgebras that provided a combined operational and denotational model for a language that generated their data. They made an (unnecessary) excursion into a special type of recursion in order to give the distributive law. This was later all expressed more elegantly and without that unnecessary excursion in [10].

In practice, their base category \( C \) was typically \( \text{Set} \), their syntax endofunctor was invariably generated by a signature of operations, each of finite arity, and their behaviour endofunctor was a variant of the endofunctor \( P_f(-)^A \), where \( P_f \) is the finite powerset functor, and \( A \) is a set of labels. So the data for an abstract operational rule was equivalent to giving, for each set \( X \), for each \( n \)-ary function symbol \( f \), and for behaviours for each of \( x_1 \) to \( x_n \), possibly including undefinedness, a behaviour for \( f(x_1, \cdots, x_n) \)

\[
\frac{(x_i \rightarrow^a y_{ij})_{a \in A}(x_i \rightarrow^b \perp)_{b \in A}}{f(x_1, \cdots, x_n) \rightarrow^a t}
\]

where \( t \) is a term generated by the signature with variables in \( X \), with the \( x \)'s and \( y \)'s all in \( X \), and with \( \perp \) denoting undefinedness, subject to an image-finiteness condition. This equivalence provided evidence of the computational naturality of their definition of an abstract operational rule as a natural transformation of the form \( \Sigma(B \times \text{Id}) \Rightarrow BT \), and that computational naturality has been further supported by the development of an assortment of examples,
sometimes going well beyond the original idea, for instance in [2,9]. But, a priori, a definition does not constitute a theory, which we seek.

In this paper, we develop their definition in several ways. The most important way in which we do so is by considering the dynamics induced by such operational semantics: a transition system generates streams of transitions, and one can speak of the streams of transitions from a given term. I.e., if \( t \) evolves to \( t' \), one can iterate the process until one reaches a final state or the process loops. It is not immediately clear how to explain that phenomenon in terms of Turi and Plotkin’s formulation. Direct application of their rule gives transitions from terms of the form \( f(x_1, \cdots, x_n) \) to arbitrary terms \( t \), subject to implicit universal quantification. Implicit in their study of the initial bialgebra is a transition from an arbitrary closed term, but they use the cofree comonad in order to describe the initial bialgebra, and the cofree comonad is an unnecessary distraction here. Moreover, they do not speak at all of the composition of transitions or of streams of transitions. So here we use a non-trivial reformulation of their definition as our mathematical foundation. The reformulation we use is given by a distributive law of the free monad \( T \) on an endofunctor \( \Sigma \) over the cofree copointed endofunctor on an endofunctor \( B \): that this is equivalent to Turi and Plotkin’s formulation was not quite shown in [10], but it follows by close inspection of the constructions therein, as shown in Section 2. That reformulation readily allows us to compose transitions and to consider streams of transitions.

Implicit in the question of dynamics is the process of passing from small-step operational semantics to large-step operational semantics. Implicit also is the process of taking an operational semantics as formulated by Turi and Plotkin and giving an operational semantics that applies to arbitrary terms. Further implicit is the notion of a composite of transitions. In giving a theory rather than a definition of structural operational semantics, we give mathematical formulations of all three of these computationally natural constructions at the level of abstraction proposed by Turi and Plotkin.

Given an abstract operational rule seen as a distributive law \( TH \Rightarrow HT \) of a monad \( T \) over a copointed endofunctor \( H \), one can treat two-step transition sequences by considering the composite

\[
THH \Rightarrow HTH \Rightarrow HHT
\]

One needs to introduce an equaliser into \( HH \) in order to make the target of one transition agree with the source of the following one, but the (pointwise) equalising property applied to the above composite immediately yields the desired two-step transition function in Turi and Plotkin’s leading examples, as shown in Section 3. The dual of that equaliser appears in [7], at the start of a construction of the free monad on a pointed endofunctor. That is no coincidence. Indeed, here, in the limit, assuming the relevant limit exists, it follows from a dual of a theorem in [7] that one recovers the cofree comonad \( D \) on the copointed endofunctor \( H \), and one can readily check that
one recovers the induced distributive law of the monad $T$ over the comonad $D$. The limit need not always exist in $C$, cf [1,14], but it does always exist in a larger universe, and the approximants to the limit act as approximants to the latter distributive law even if the limit does not exist. So one can consider the distributive law of $T$ over the comonad $D$ as the large-step operational semantics induced by the abstract operational rule given by the distributive law of $T$ over the copointed endofunctor $H$.

We further consider computationally natural constructions on the statics of a structural operational semantics, i.e., on its definition rather than on the streams of transitions it induces. One often has a combination of equational theory and operational semantics, as studied in general in [3]. Not only is that more general than Turi and Plotkin’s setting, but our equivalence theorem suggests it is a more natural setting for their definition. The more general setting is given by a signature subject to equations and an operational semantics that must respect the equations. Such operations and equations always generate monads, and in fact, all monads, subject to a size condition, arise from such [8]. Our reformulation of Turi and Plotkin’s definition extends to handle that situation as one can consider a distributive law for a monad $T$ over a copointed endofunctor $H$, where $T$ need not be free on an endofunctor. The constructions implicit in considering an equational theory rather than just a signature are that of adding further operations and that of subjecting operations to further equations. One wants the definition of an abstract operational rule to be robust under both constructions. We treat these constructions in Sections 4 and 5 respectively.

For the former, i.e., adding operations, it is more general, more natural, and more elegant to consider two equational theories, equivalently monads, $T$ and $T'$ with operational semantics for each, and generate operational semantics for their sum. In computational terms, it is obvious that one can do that [3]; but if the spirit of Turi and Plotkin’s proposal is to be taken seriously, we need to describe a natural mathematical operation that, given abstract operational rules for each of $T$ and $T'$, provides an abstract operational rule for $T + T'$; and that mathematical operation should agree with the natural computational phenomenon. That requires some thought because the sum of the induced monads $T$ and $T'$ qua monads does not always exist, and it is rarely easy to describe: if one started with signatures, a term generated by the sum of signatures may involve a combination of function symbols from each of the two signatures. One not only must have the sum, but one must use that sum in describing a combined distributive law. We consider the sum in Section 4.

We treat the latter construction, i.e., the addition of equations, in Section 5. That amounts to considering a coequaliser of monads. As shown for instance in [8], given a monad, equivalently an equational theory, $T$ on $Set$, to subject the equational theory to further equations is equivalent to giving an endofunctor $E$ on $Set$ together with a pair of natural transformations $\tau_1, \tau_2 : E \Rightarrow T$. The monad induced by $T$ subject to the equations is
given by the coequaliser of the monad maps $\bar{\tau}_1, \bar{\tau}_2 : E^* \Rightarrow T$, where $E^*$ is the free monad on the endofunctor $E$ and $\bar{\tau}_1$ and $\bar{\tau}_2$ are the evident liftings. So, we want the definition of abstract operational rule to be robust under taking coequalisers of monads. That coequaliser is not given pointwise, and it is typically difficult to describe explicitly. So some effort is required here. For a popular calculus in which the combination of equations and operational semantics was discussed although not used, see Milner’s account of CCS [11]. And for a specific application of Turi and Plotkin’s idea applied to a monad that is not free on an endofunctor, see Kick’s work on timing, where he uses $T = M \times -$ for a monoid $M$ [9].

Natural further questions in the line of this paper are to give a mathematical formulation of typed structure and to describe the process of passing from an operational semantics to contextual equivalence. The latter seems likely to be difficult for the simple reason that it is a complex computational construction, but there is a start in [13].

The paper is organised as follows. In Section 2, we carefully reprise one of the main constructions of [10] to show that, although not stated in the technical sections of that paper, Turi and Plotkin’s abstract operational rules may be characterised as distributive laws. In Section 3, we address dynamics at the level of generality proposed by Turi and Plotkin. And in Sections 4 and 5, we prove robustness of the definition of abstract operational rule under the addition of operations and under the addition of equations respectively.

2 Turi and Plotkin’s abstract operational rules as distributive laws

In this section, following [10], 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 construction of the latter from the former appeared in [10], but it was not observed in the technical sections of that paper that that construction is an equivalence: it was mentioned in the introduction to the paper, but we did not appreciate its significance at the time, hence our ignoring it in the relevant technical sections. The heart of this paper’s technical content is an exploration of the computational significance of that characterisation, so we describe it in detail here. We need not only the characterisation of Theorem 2.5 but also intermediate results, notably Theorem 2.2, in later sections, the latter especially in Sections 4 and 5, and of course we need the various definitions throughout the paper.

Definition 2.1 A copointed endofunctor on a category $C$ is an endofunctor $H : C \to C$ together with a natural transformation $\epsilon : H \Rightarrow Id$. An $(H, \epsilon)$-
coalgebra is an object $X$ of $C$ together with a map $x : X \to HX$ such that

$$\begin{array}{ccc}
X & \xrightarrow{x} & HX \\
\downarrow{\text{id}} & & \downarrow{\epsilon X} \\
X & \xrightarrow{\text{id}} & X
\end{array}$$

commutes. The evident definition of a map of $(H, \epsilon)$-coalgebras yields the category $(H, \epsilon)\text{-Coalg}$ of $(H, \epsilon)$-coalgebras. The right adjoint to the forgetful functor

$$U : (H, \epsilon) \text{-Coalg} \to C$$

if it exists, is the cofree comonad on $(H, \epsilon)$. A distributive law of a monad $(T, \mu, \eta)$ over a copointed endofunctor $(H, \epsilon)$ is a natural transformation $\lambda : TH \Rightarrow HT$ that makes the following diagrams commute:

$$\begin{array}{cccc}
TTH & \xrightarrow{T\lambda} & THT & \xrightarrow{\lambda T} & HTT \\
\downarrow{\mu H} & & \downarrow{H\mu} & & \\
TH & \xrightarrow{\lambda} & HT \\
\downarrow{\eta H} & & \downarrow{H\eta} & & \\
TH & \xrightarrow{\lambda} & HT
\end{array}$$

We require the following result for our proof of the characterisation of Turi and Plotkin’s definition. But beyond that, we shall use this result in later sections too. It is often difficult to calculate directly with monads, but the following theorem, appearing in [10], allows us to deduce existence of useful constructions on distributive laws (see Sections 4 and 5), equivalently on abstract operational rules, using monads, without need for explicit calculation.

**Theorem 2.2** Given a monad $T$ and a copointed endofunctor $(H, \epsilon)$, to give a distributive law of $T$ over $(H, \epsilon)$ is equivalent to giving a lifting $(\bar{H}, \bar{\epsilon})$ of $(H, \epsilon)$ to $T\text{-Alg}$.

**Proof.** The constructions are given by the evident variants of those for a monad and a comonad [10,13]. The proof of equivalence is routine. \qed
It is routine to verify that, for an endofunctor $B$ on a category with finite products $C$, the cofree copointed endofunctor on $B$ is given by $(B \times Id, \pi_2)$, and 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 $(B \times Id, \pi_2)$. 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 explained in [7], but mistakes in this setting are most unlikely.

Given a category $C$ with finite products, an endofunctor $\Sigma$ and a (behaviour) endofunctor $B$ on $C$, with $\Sigma$ freely generating the (syntax) monad $(T, \mu, \eta)$, Turi and Plotkin showed in [13] that, using functoriality of GSOS, each image-finite GSOS rule can be modelled by an abstract operational rule, which they defined to be a natural transformation $\rho : \Sigma(B \times Id) \Rightarrow BT$. Moreover, in their leading class of examples, for $C = Set$, they exhibited a converse, yielding an equivalence between image-finite GSOS-rules and abstract operational rules. They gave a class of examples, and they gave some abstract development of the idea. We now show that Turi and Plotkin’s definition of an abstract operational rule is equivalent, under the conditions they cited, to giving a distributive law of the monad $T$ over the cofree copointed endofunctor $(B \times Id, \pi_2)$ on $B$.

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

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

(1)

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

**Proposition 2.3** To give an abstract operational rule $\rho : \Sigma(B \times Id) \Rightarrow BT$ is equivalent to giving a natural transformation $\varrho : \Sigma(B \times Id) \Rightarrow (B \times Id)T$ which 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

$$
\begin{array}{ccc}
\Sigma(B \times Id) & \xrightarrow{\Sigma \pi_2} & \Sigma \\
\downarrow & \xrightarrow{\theta} & \downarrow \\
\Sigma & \xrightarrow{} & T
\end{array}
$$

(2)

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 the first component \(\Sigma(B \times Id) \Rightarrow BT\), i.e., an abstract operational rule.

**Proposition 2.4** For any copointed endofunctor \((H, \epsilon)\), to give a natural transformation \(\varrho: \Sigma H \Rightarrow HT\) respecting the structure of \((H, \epsilon)\) is equivalent to giving a distributive law of the free monad \(T\) on \(\Sigma\) over \((H, \epsilon)\).

**Proof.** Given \(\varrho\), we first show that the endofunctor \(H\) lifts to an endofunctor \(\breve{H}\) on the category \(\Sigma-\text{alg}\), and the natural transformation \(\epsilon: H \Rightarrow Id\) lifts to \(\breve{\epsilon}: \breve{H} \Rightarrow Id\).

Define the action of \(\breve{H}: \Sigma-\text{alg} \rightarrow \Sigma-\text{alg}\) as follows: a \(\Sigma\)-algebra \(k: \Sigma X \rightarrow X\) is sent to \(Hk^\sharp \circ \varrho_X\), where \(k^\sharp: TX \rightarrow X\) is the corresponding Eilenberg-Moore algebra for the monad \((T, \mu, \eta)\) under the isomorphism \(\Sigma-\text{alg} \cong T-\text{Alg}\).

An arrow \(f\) of \(\Sigma\)-algebras from \(k: \Sigma X \rightarrow X\) to \(l: \Sigma Y \rightarrow Y\), i.e., an arrow \(f: X \rightarrow Y\) in \(C\) satisfying \(f \circ k = l \circ \Sigma f\), is sent to \(Hf: HX \rightarrow HY\). The functor \(\breve{H}: \Sigma-\text{alg} \rightarrow \Sigma-\text{alg}\) is a lifting of \(H\).

Next, for each \(\Sigma\)-algebra \(k: \Sigma X \rightarrow X\), observe that the \(X\) component \(\epsilon_X: HX \rightarrow X\) of \(\epsilon\) is a morphism of \(\Sigma\)-algebras from \(\breve{H}k\) to \(k\), i.e., \(\epsilon_X \breve{H}k = k \epsilon_X\). Since the natural transformation \(\varrho\) respects the structure of \((H, \epsilon)\), both squares in the following diagram commute:

\[
\begin{array}{ccc}
\Sigma HX & \xrightarrow{\varrho_X} & HTX \\
\downarrow \Sigma \epsilon_X & & \downarrow \epsilon_{TX} \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \Quad Since the bottom arrow of the diagram is \(k^\sharp \circ \theta_X = k\) and the top arrow is \(\breve{H}k\), the arrow \(\epsilon_X: HX \rightarrow X\) is a morphism of \(\Sigma\)-algebras from \(\breve{H}k\) to \(k\). So we may define \(\breve{\epsilon}: \breve{H} \Rightarrow Id\) by defining its \(k: \Sigma X \rightarrow X\) component to be \(\epsilon_X\). Its naturality follows from naturality of \(\epsilon\). It is evidently a lifting of \(\epsilon\) to \(\Sigma-\text{alg}\).

Because \(\Sigma-\text{alg}\) is isomorphic to \(T-\text{Alg}\), both the functor \(\breve{H}\) and the natural transformation \(\breve{\epsilon}: \breve{H} \Rightarrow Id\) are liftings of \(H\) and \(\epsilon\) to \(T-\text{Alg}\). By Theorem 2.2, to give such a lifting is equivalent to giving a distributive law of the monad \((T, \mu, \eta)\) over the copointed endofunctor \((H, \epsilon)\).

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

From Propositions 2.3 and 2.4, we conclude that

**Theorem 2.5** 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 \((T, \mu, \eta)\) over the copointed endofunctor \((B \times Id, \pi_2)\).
One can readily derive the distributive law of the monad $T$ over the cofree comonad $D$ on $B$ from Theorem 2.5 (see [10]).

The construction in Theorem 2.5, which is essentially that of Proposition 2.4, is inherently of computational interest, especially in analysing dynamics. To analyse dynamics, we need to consider streams of transitions $t_0 \rightarrow t_1 \rightarrow t_2 \cdots$, so even if $t_0$ only involves one function symbol, $t_1$ will typically be more complicated, and one needs to apply the transition function in order to know how it may evolve. The construction of Proposition 2.4 gives us the unique canonical extension of an abstract operational rule to all terms. And it does so without reference to a cofree comonad, which is irrelevant, cf., [13].

3 Dynamics

In this section, we begin an analysis of dynamic constructs at the level of generality proposed by Turi and Plotkin. Dynamics are fundamental to programming, as one considers safety and liveness issues for example. Moreover, Turi and Plotkin’s leading class of examples arise more from concurrency constructs than from functional languages, in particular with their analysis of nondeterminism and a parallel operator. So it is a natural, relevant question how to generate dynamic structures from their definition. It is, of course, routine to consider dynamic constructs on a case-by-case basis, but if one is to take their proposal seriously, one wants constructs at their 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 adopted by Turi and Plotkin, i.e., a coalgebra for an arbitrary endofunctor, is far more general than the usual notion of transition system, which is given by a specific choice of endofunctor, and that greater generality inevitably leads to greater generality in the notion of a stream of transitions. 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 $t_0 \rightarrow t_1 \rightarrow t_2$ 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 $(H, \epsilon)$, with leading examples having $(H, \epsilon)$ 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 $x : X \rightarrow BX$, and 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 3.1** 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 $B$. That 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.

**Example 3.2** Let $B = P_f$, the finite powerset functor. We thus have the cofree copointed endofunctor given by $HX = P_f(X \times X)$. So the composite is $HHX = P_f(P_f(X \times X)) \times P_f(X \times X)$. 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

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

266
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

\[ H\epsilon X, \epsilon HX : HHX \rightarrow HX \]

And we define \( \epsilon_2 : H_2 \Rightarrow Id \) by composition.

**Proposition 3.3** For any \((H, \epsilon)\)-coalgebra \((X, x)\), the composite

\[
\begin{array}{ccc}
X & \xrightarrow{x} & HX \\
& & \xrightarrow{Hx} \\
& & HHX
\end{array}
\]

is an \((H_2, \epsilon_2)\)-coalgebra.

**Proof.** One needs only check that the composite composed with \( H\epsilon X \) and \( \epsilon HX \) is the same, but that follows directly from the naturality of the copoint and from the definition of \((H, \epsilon)\)-coalgebra. \(\square\)

**Proposition 3.4** Given a distributive law of a monad \( T \) over a copointed endofunctor \((H, \epsilon)\), the composite

\[ THH \Rightarrow HTH \Rightarrow HHT \]

induces a distributive law of \( T \) over the copointed endofunctor \((H_2, \epsilon_2)\).

**Proof.** The equalising property for the (composite) map into \( HHTX \) follows from the preservation of \( \epsilon \) in the definition of a distributive law. For each \( X \), that yields the required map \( TH_2X \rightarrow H_2TX \). 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. \(\square\)

The definition of \((H_2, \epsilon_2)\) is therefore as desired, agreeing with the examples, and the distributive law

\[ TH_2 \Rightarrow H_2T \]

of the monad \( T \) over the copointed endofunctor \( H_2 \) induced by the composite

\[ THH \Rightarrow HTH \Rightarrow HHT \]

is also as desired, yielding the normal two-step transition function in the leading examples.

That yields two steps. With some thought, the process can be iterated. It is not, in general, sufficient to stop after \( \omega \) steps for the usual reason that the limit one often has to characterise the terminal coalgebra need not converge after \( \omega \) steps \([1, 14]\), i.e., because of lack of uniformity. But, assuming a (typically transfinite) limit does exist, what does this yield? An answer is given by taking the dual of a construction hidden deep inside Kelly’s paper \([7]\). We describe the dual, i.e, the form we want, here.
**Definition 3.5** Given a copointed endofunctor \((H, \epsilon: H \Rightarrow Id)\) on a base category \(C\) with all limits, and given an object \(X\), put
\[
X_0 = X \quad X_1 = HX \quad x_0 = id_{HX} : X_1 \to HX_0
\]
Now define \(X_{\beta+2}\) and \(x_{\beta+1} : X_{\beta+2} \to HX_{\beta+1}\) by the equaliser of
\[
HX_{\beta+1} \xrightarrow{Hx_\beta} H^2X_\beta \xrightarrow{\epsilonHX_\beta} HX_\beta
\]
with
\[
HX_{\beta+1} \xrightarrow{Hx_\beta} H^2X_\beta \xrightarrow{H\epsilonX_\beta} HX_\beta
\]
For a limit ordinal \(\alpha\), put \(X_\alpha = \lim_{\beta<\alpha} X_\beta\) with the \(X_\beta\) being the generators of the limit cone, and define \(X_{\alpha+1}\) and \(x_\alpha : X_{\alpha+1} \to HX_\alpha\) by the equaliser of
\[
HX_\alpha \xrightarrow{\epsilonX_\alpha} X_\alpha = \lim_{\beta<\alpha} X_{\beta+1} \xrightarrow{\lim_{\beta<\alpha} x_\beta} \lim_{\beta<\alpha} HX_\beta
\]
with
\[
HX_\alpha \xrightarrow{\epsilonX_\alpha} \lim_{\beta<\alpha} HX_\beta
\]
where the unlabelled map is canonically induced by the limiting property.

The \(X_\beta\)'s form a (typically infinitary) cochain whose non-limit intermediary maps \(X_{\beta+1} : X_{\beta+1} \to X_\beta\) are given by \(\epsilonX_\beta \cdot x_\beta\). We say that the sequence converges at \(\alpha\) if \(X^{\alpha+1}_\alpha\) is an isomorphism.

One can immediately 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. Moreover, the rest of the construction here gives the evident generalisation to arbitrary steps, possibly transfinitely owing to lack of uniformity: this does not imply a consideration of streams of transfinite length. The cofree comonad, by this construction, gives all the intermediate steps of the streams of transitions from a given state, so agrees with the usual notion.

With some effort, one can check that the dual of Theorem 17.3 of [7] implies the following:

**Theorem 3.6** If the sequence \(X_\beta\) converges at \(\alpha\), the cofree comonad of \(H\) applied to \(X\) is given by \(X_\alpha\) with co-action \(x_\alpha : X_\alpha = X_{\alpha+1} \to HX_\alpha\)

There are reasonable conditions under which the convergence does hold. Some such conditions seem to be implicit in [1,14] for example.

**Theorem 3.7** If the sequence \(X_\beta\) converges at \(\alpha\), applying the construction of Proposition 3.4 iterated on ordinals of size less than \(\alpha\) to a distributive law
of a monad $T$ over a copointed endofunctor $(H, \epsilon)$ yields a distributive law of the monad $T$ over the cofree comonad $D$ on $(H, \epsilon)$. Moreover, that distributive law agrees with the canonical one [10,13].

**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. $\square$

As this characterises the distributive law, if the sequence converges, we regard this result as implying that the distributive law of the monad $T$ over the cofree comonad $D$ can reasonably be regarded, at the level of generality of Turi and Plotkin, as the large-step operational semantics induced by their 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 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.

4 Adding operations

In this section, we consider the addition of operations to a signature at the level of generality of Turi and Plotkin. Rather than doing so directly, it is more elegant and conceptual, as well as being more general, in accord with [3], to suppose we have two monads $T$ and $T'$, distributive laws of each of $T$ and $T'$ over a copointed endofunctor $H$, and seek to combine them into a distributive law of $T + T'$, if it exists, over $H$. One can recover the situation of the addition of operators by considering free monads on signatures. But the more general setting accords with the use of an equational theory combined with operational semantics, as studied in [3], agrees with our central definition, and keeps our account consistent with that of Section 3, where we studied dynamics. Although not essential, it is simplest to appeal to Theorem 2.2.

Given an object $X$ of $C$, consider the functor $X^{C(-,X)} : C \to C$. It sends an object $Y$ to the product of $C(Y,X)$ copies of $X$. For an arbitrary endofunctor $F : C \to C$, it follows from the Yoneda lemma that to give a natural transformation

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

is equivalent to giving a map $x : FX \to X$. One can readily prove that the functor $X^{C(-,X)}$ possesses a natural monad structure, and one has the following equivalence, as used extensively for instance in [8].

**Proposition 4.1** For a monad $T$ on $C$, 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 $X$. 269
Using that proposition, one can immediately prove the following.

**Proposition 4.2** For monads $T$ and $T'$ on $C$, 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

\end{array}
\begin{array}{ccc}
& & \\
\downarrow & & \\
T' - Alg & \underset{U'}{\longrightarrow} & C
\end{array}
$$

**Theorem 4.3** Given monads $T$ and $T'$ and a copointed endofunctor $(H, \epsilon)$, and given distributive laws $\lambda : TH \Rightarrow HT$ and $\lambda' : T'H \Rightarrow HT'$, there is a canonical distributive law of $T + T'$ over $(H, \epsilon)$ if the sum of monads $T + T'$ exists.

**Proof.** This follows from the combination of Theorem 2.2 with Proposition 4.2. By the former, the two distributive laws give liftings of $(H, \epsilon)$ 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 $(H, \epsilon)$. So by an application of the converse part of Theorem 2.2, we have the distributive law of $T + T'$ over $(H, \epsilon)$ that we seek.

By construction, this combination of distributive laws is associative with an evident unit. It is canonically induced by the structure of $T + T'$ as a sum of monads, and one can characterise it by a universal property, but that property is a little unnatural, so we do not state it here. Calculating the combined distributive law is typically difficult, but only because the monad $T + T'$ is typically difficult to describe. There is a relatively easy description of that sum if one of the monads is free on an endofunctor [5]: it is $T(\Sigma T)^*$ for monad $T$ and endofunctor $\Sigma$, where $S^*$ denotes the free monad on an endofunctor $S$.

## 5 Adding equations

Adding equations is a matter of coequalisers in the category of monads on $C$ just as adding operations was a matter of coproducts in the category of monads on $C$ as we saw in Section 4. We use essentially the same techniques as in Section 4 to see that the notion of an abstract operational rule qua distributive law of a monad $T$ over a copointed endofunctor $(H, \epsilon)$ is robust under the addition of equations as for instance in [3]. The relevant abstract category theoretical basis for this appears in [8]. It should be clear in the case that $C = Set$.  

270
To add equations to an equational theory qua monad $T$ is equivalent to giving an endofunctor $E : C \rightarrow C$ and a pair of natural transformations $\tau_1, \tau_2 : E \Rightarrow T$. The equational theory generated by $T$ subject to these additional equations is given by the monad obtained by the coequaliser $T[E]$ 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.

Mimicking the work of Section 4 but replacing coproducts by coequalisers and replacing the pullback of categories by an equaliser of categories, we may deduce the following result.

**Theorem 5.1** Distributive laws of $T$ and $E^*$ over $(H, \epsilon)$ that are respected by $\bar{\tau}_1$ and $\bar{\tau}_2$ induce a distributive law of $T[E]$ over $(H, \epsilon)$.

The condition of the theorem may look a little complex, but it is exactly the condition required and it agrees with the leading examples. The reason for the complexity is that a primitive equation between terms may be sent by the operational semantics to a derived equation rather than just a primitive one. Obviously, further work needs to be done here in giving examples explicitly.

**References**

[1] Adamek, J., and V. Koubek, *On the greatest fixed point of a set functor*, Theoretical Computer Science 150 (1995), 57–75.

[2] Bartels, F., *GSOS for Probabilistic Transition Systems*, Electronic Notes in Theoretical Computer Science 65 (2002).

[3] Corradini, A., R. Heckel, and U. Montanari, *From SOS Specifications to Structured Coalgebras: How to make Bisimulation a Congruence*, Electronic Notes in Theoretical Computer Science 19 (1999).

[4] Fiore, M., and D. Turi, *Semantics of Name and Value Passing*, “Proceedings of 16th Symposium on Logic in Computer Science,” IEEE Computer Society Press, 2001, 93–104.

[5] Hyland, M., G.D. Plotkin, and A.J. Power, *Combining effects: sum and tensor*, (submitted).

[6] Jacobs, B., and J. Rutten. *A tutorial on (co)algebras and (co)induction*, Bulletin of the European Association of Theoretical Computer Science 62 (1997), 222–259.

[7] Kelly, G.M., *A unified treatment of transfinite constructions for free algebras, free monoids, colimits, associated sheaves, and so on*, Bulletin Australian Mathematical Society 22 (1980), 1–83.

[8] Kelly, G.M., and A.J. Power, *Adjunctions whose counits are coequalizers, and presentations of finitary monads*, Journal of Pure and Applied Algebra 89 (1993), 163–179.
Power

[9] Kick, M., *Bialgebraic modelling of timed processes*, “Proceedings of ICALP 2002,” Lecture Notes in Computer Science **2380**, 2002, 525–536.

[10] Lenisa, M., A.J. Power, and H. Watanabe, *Distributivity for endofunctors, pointed and co-pointed endofunctors, monads and comonads*, Electronic Notes in Theoretical Computer Science **33** (2000).

[11] Milner, R., “Communication and Concurrency,” Prentice-Hall International Series in Computer Science, 1989.

[12] Power, A.J., and H. Watanabe, *Combining a monad and a comonad*, Theoretical Computer Science **280** (2002), 137–162.

[13] Turi, D., and G.D. Plotkin, *Towards a mathematical operational semantics*, “Proceedings of 12th Symposium on Logic in Computer Science,” IEEE Computer Science Press, 1997, 280–291.

[14] Worell, James, *Terminal sequences for accessible endofunctors*, Electronic Notes in Theoretical Computer Science **19** (1999).