Modular Bialgebraic Semantics and Algebraic Laws

Technical Report ICIS–R13008, July 2013

Ken Madlener[1], Sjaak Smetsers[1], and Marko van Eekelen[1,2]
{K.Madlener,S.Smetsers,M.vanEekelen}@cs.ru.nl

[1] Institute for Computing and Information Sciences
Radboud University Nijmegen
[2] School of Computer Science
Open University of the Netherlands

Abstract. The ability to independently describe operational rules is indispensable for a modular description of programming languages. This paper introduces a format for open-ended rules and proves that conservatively adding new rules results in well-behaved translations between the models of the operational semantics. Silent transitions in our operational model are truly unobservable, which enables one to prove the validity of algebraic laws between programs. We also show that algebraic laws are preserved by extensions of the language and that they are substitutive. The work presented in this paper is developed within the framework of bialgebraic semantics.

1 Introduction

In order to scale to the complexity of real-world programming languages, a modular way of describing semantics is highly desirable. When dealing with incrementally constructed languages, one should anticipate on future extensions or changes to the language. Moreover, a concrete program seldom uses all the constructs provided by the language. When reasoning about a program it is convenient to narrow the semantics down to the part of the language which is actually used. True modularity offers the possibility to build an ad hoc semantics, easing the construction of correctness proofs.

Mosses [15] advocates to define higher-level language constructs out of so-called "funcons", language-independent fundamental programming constructs. It is highly desirable that algebraic rules between programs are preserved under the addition of new funcons, since this avoids the repetition of proofs.

The present paper provides a fundamental perspective on this issue, built on the framework of Turi and Plotkin's bialgebraic semantics [20]. One of the advantages of this work is that it can be implemented in a functional language such as Haskell as well as in a theorem prover like Coq. In fact, part of this work has already been formalized within Coq, based on [13].

Each operation corresponding to a funcon has a number of defining operational rules, which may manipulate the state, or invoke an external operation. For example, the rule for a condition-less loop would be $\mathsf{loop}\ x \Longrightarrow \mathsf{seq}\ x\ (\mathsf{loop}\ x)$. The double arrow indicates that the transition is deemed silent, it does not generate an observable side-effect. To handle two subsequential commands, loop invokes the external operation seq. This mechanism is comparable to interfaces in object-oriented languages. Thus, we consider the operational rules corresponding to some construct as open-ended, empowering true modularity in language descriptions. By commencing with an empty language and then incrementally extending this with new constructs, a full language is obtained.

Silent transitions are indispensable in providing independent descriptions of the operations. An alternative version of the previous rule, which avoids the use of a silent transition, can be defined by performing a "look-ahead", i.e. $x \xrightarrow{a} x' \vdash \mathsf{loop}\ x \xrightarrow{a} \mathsf{seq}\ x'\ (\mathsf{loop}\ x)$. The problem with this version is that the resulting rule is no longer modular. It makes implementation assumptions on seq, namely that seq always makes a step on its first statement. Such assumptions clearly violate the independency principle. On the other hand, representing silent transitions as distinguished labels does not make them truly unobservable, as $\mathsf{loop}\ x$ is no longer (behaviorally) equivalent to $\mathsf{seq}\ x\ (\mathsf{loop}\ x)$, unless one resorts to the more complex notion of weak bisimulations. Moreover, rules for silent transitions are often not purely structural. For example, the rule $\mathsf{seq}\ \mathsf{skip}\ x \Longrightarrow x$ inspects the first argument of the head operation before it can be applied.

In this paper we treat structural operational rules and rules for silent transitions as separate classes. A generalization of the categorical interpretation by Turi and Plotkin [20] of the GSOS rule format accommodates the structural rules. For the silent transitions we apply an altered construction of Klin [9].

The standard notion of bisimulation between computations expresses that both computations exhibit the same observational behavior. Unfortunately, standard bisimulation is not preserved by language extensions [16]. De Simone [5] introduced Formal-Hypothesis bisimulations, which take into account that variables in terms being evaluated may exhibit arbitrary behavior. A pair of FH-bisimilar (open) terms is called an algebraic law. We prove that our notion of language extension preserves algebraic laws. Moreover, we show that algebraic laws are substitutive, in the sense of [18]. This property eases reasoning about programs, since it allows program fragments to be replaced by other simpler fragments, provided these are FH-bisimilar.

In summary, the contributions of this paper are threefold:

- We introduce a rule format, called "open GSOS", which enables the modular description of operational semantics. Moreover, we provide a definition for conservative extensions of open GSOS rules, and show that there exists a well-behaved translation between the operational semantics described by open GSOS rules and the operational semantics described by conservative extensions of these rules.
- We add support for rules with silent transitions to open GSOS, in such a manner that silent transitions are truly unobservable while well-behavedness

of the translation between the base and extended operational semantics remains intact.
– We formalize the notion of algebraic laws within the bialgebraic framework, prove that these laws are preserved through conservative language extensions, and prove that they are substitutive. This transfers results from [16] and [18] to the setting of bialgebraic semantics.

Basic definitions are provided in Section 2, followed by three sections (Section 3, 4, and 5) corresponding to the above bullets. Section 6 shows how the resulting operational models can be executed. Related work is discussed in Section 7 and conclusions are drawn in Section 8. The reader is expected to have some familiarity with category theory and bialgebraic semantics.

2 Preliminaries

This section recalls some basic definitions. A good introduction to the field of bialgebraic semantics is provided in [10], further background can be found in [7].

The (open) terms TX, generated by an endofunctor F, where X acts as the variables, are the least solution to the equation $Y \cong X + FY$. This means that there is an isomorphism $\kappa_X : TX \to X + FTX$, and we call the left and right components of the inverse morphism $\eta_X : X \to TX$ and $\psi_X : FTX \to TX$ respectively. We will also use the auxiliary morphism $\phi_X := \psi_X \circ F\eta_X$. The functor F, called the *signature functor*, stands for the grammar of the language, and is specified by cases, e.g. $FX := \mathsf{skip} \mid \mathsf{seq}\ (x\ y : X) \mid \mathsf{loop}\ (x\ y : X)$.

The terms come with a principle which says that there is a unique morphism satisfying the following diagram, for any morphism $f : X \to Y$ and algebra $g : FY \to Y$:

$$\begin{array}{ccccc} X & \xrightarrow{\eta} & TX & \xleftarrow{\psi} & FTX \\ & \searrow^{f} & \downarrow{\scriptstyle fold\, f\, g} & & \downarrow{\scriptstyle F(fold\, f\, g)} \\ & & Y & \xleftarrow[g]{} & FY \end{array} \tag{1}$$

We have called this unique morphism *fold f g*, to emphasize that this corresponds to folding over terms, familiar from functional programming. One can show that the functor T is a monad, i.e. it has a unit, $\eta : Id \to T$, and a join operation $\mu : TT \to T$, subject to the following conditions:

$$\mu \circ T\mu = \mu \circ \mu_T \tag{2}$$

$$\mu \circ \eta_T = \mu \circ T\eta = id. \tag{3}$$

Algebras such as ψ_X play a crucial role in the syntax, and dually coalgebras play a crucial role for the behavior. A *B-coalgebra* consists of a state-space, i.e. a set X of states, together with a morphism $c : X \to BX$. Sometimes we will also call c itself a coalgebra. One calls the pair $\langle D, \pi \rangle$ a *copointed endofunctor*

if there is a natural transformation $\pi : D \to Id$. The leading example will be $DX := X \times BX$ (we assume that the underlying category has products).

A relation $R \subseteq X \times X$ is a *bisimulation relation* between the coalgebras c, d if there exists a morphism γ such that the following diagram commutes:

$$\begin{array}{ccccc} X & \xleftarrow{\pi_1} & R & \xrightarrow{\pi_2} & X \\ {\scriptstyle c}\downarrow & & \downarrow{\scriptstyle \exists\gamma} & & \downarrow{\scriptstyle d} \\ BX & \xleftarrow[B\pi_1]{} & BR & \xrightarrow[B\pi_2]{} & BX \end{array}$$

Here, R is considered to be an object of the underlying category. When B is a polynomial functor, it is equivalent to say that any pair $\langle x, y\rangle \in R$ implies that $\langle c\, x, d\, y\rangle \in Rel(B)(R)$, where $Rel(B)(R) \subseteq BX \times BX$ is the *lifting of R to B*, see Chapter 3 in [7].

In Section 4 we will need to make the additional assumption that the underlying category is **CPPO**-enriched. This holds true when the homsets, V say, are cppo's (i.e. (small) posets with a least element and closed under LUBs and omega chains), and that composition is continuous in both arguments. Tarski's theorem asserts that for every continuous $\Psi : V \to V$, if $\Psi\, f \geq f$, then the least fixpoint of Ψ exists, and it is equivalent to $\Psi^*\, f := \bigsqcup_{n\in\mathbb{N}} \Psi^n\, f$.

3 Rule Format

In [20] it was shown that the operational rules in the GSOS format can be understood as a natural transformation $FD \to BT$. In this section we introduce a more general rule format, which we call "open GSOS", tailored to the independent description of operational rules.

3.1 Open GSOS

As an example, consider the constructs in Figure 1. We can define the higher-level construct while by combining these:

$$\mathsf{while}\ c\ b := \mathsf{catch}\ (\mathsf{loop}\ (\mathsf{if}\ c\ b\ \mathsf{break})).$$

In this section we will strictly consider rules for non-silent transitions. For the behavior functor, set $BX := S \to B_0X$, where $B_0X := 1 + S \times X$, and S represents the states. In this case, the type for the GSOS format is isomorphic to $FD \times S \to 1 + S \times T$. The option "1" is for operations which do not have any defining operational rules, i.e. skip, stuck, true, false. Such operations are called *values*, as they are only to be inspected by the rules [4]. We can send values to 1, to ensure that the rules are completely defined, e.g. $\mathsf{skip} \mapsto 1$.

The rule for seq can be defined as follows:

$$\langle \mathsf{seq}\ \langle x, x_B\rangle\ \langle y, y_B\rangle, s\rangle \longmapsto B_0\,(\lambda x', \mathsf{seq}\ x'\ y)\,(x_B\, s),$$

$$\frac{}{\mathsf{if\ true}\ x\ y \Longrightarrow x} \qquad \frac{}{\mathsf{if\ false}\ x\ y \Longrightarrow y} \qquad \frac{x \xrightarrow{a} x'}{\mathsf{if}\ x\ y\ z \xrightarrow{a} \mathsf{if}\ x'\ y\ z}$$

$$\frac{}{\mathsf{seq\ skip}\ x \Longrightarrow x} \qquad \frac{x \xrightarrow{a} x'}{\mathsf{seq}\ x\ y \xrightarrow{a} \mathsf{seq}\ x'\ y} \qquad \frac{}{\mathsf{loop}\ x \Longrightarrow \mathsf{seq}\ x\ (\mathsf{loop}\ x)}$$

$$\frac{}{\mathsf{catch\ skip} \Longrightarrow \mathsf{skip}}$$

$$\frac{x \xrightarrow{\{\mathsf{ex}'=\mathsf{false},\ldots\}} x'}{\mathsf{catch}\ x \xrightarrow{\{\mathsf{ex}'=\mathsf{false},\ldots\}} \mathsf{catch}\ x'} \qquad \frac{x \xrightarrow{\{\mathsf{ex}'=\mathsf{true},\ldots\}} x'}{\mathsf{catch}\ x \xrightarrow{\{\mathsf{ex}'=\mathsf{false},\ldots\}} \mathsf{skip}} \qquad \frac{}{\mathsf{break} \xrightarrow{\{\mathsf{ex}'=\mathsf{true},-\}} \mathsf{stuck}}$$

Fig. 1. Example operational rules.

where the argument pairs stand for the variable and the behavior of that variable, respectively, thus $x, y : X$ and $x_B, y_B : BX$. The signature functor is $FX := \mathsf{seq}\ (x\ y : X)$.

The rules for catch, which catches loop breaks, have been provided in MSOS notation [14]. The curly brackets indicate the pattern the label is matched on. Primed component names (e.g. ex') indicate an update of the state. We can interpret the catch rules as follows:

$$\langle \mathsf{catch}\ \langle x, x_B\rangle, s\rangle \longmapsto \begin{cases} B_0\,(\lambda x',\ \mathsf{catch}\ x')\,(x_B\ s) & \text{if } \mathsf{is\_ex}\,(x_B\ s) = \mathsf{false} \\ B_0\,(\lambda x',\ \mathsf{skip})\,(\mathsf{reset\_ex}\,(x_B\ s)) & \text{if } \mathsf{is\_ex}\,(x_B\ s) = \mathsf{true} \end{cases}$$

This rule requires that states come with a component $s.\mathsf{ex} \in \{\mathsf{true}, \mathsf{false}\}$. To query whether an exception has been thrown, we use $\mathsf{is\_ex} : BX \to \{\mathsf{true}, \mathsf{false}\}$, defined by

$$\begin{aligned} 1 &\longmapsto \mathsf{false} \\ \langle s, x\rangle &\longmapsto s.\mathsf{ex}, \end{aligned}$$

and $\mathsf{reset\_ex} : B_0X \to B_0X$ to reset the exception component, inherited from $\mathsf{reset\_ex\_S} : S \to S$:

$$\begin{aligned} 1 &\longmapsto 1 \\ \langle s, x\rangle &\longmapsto \langle \mathsf{reset\_ex\_S}\ s, x\rangle. \end{aligned}$$

We have defined a rule for the signature $FX := \mathsf{catch}\ (x : X)$, however, the result points to skip, which is not included in F. We introduce a generalization of the GSOS rule format that permits such a discrepancy between the set of defined operations, the *ingoing* signature functor F, and the resulting terms T', which are generated by the *outgoing* signature functor F'.

Definition 1 (Open GSOS). *Suppose that we have functors F, F', B, and that T' is the free monad generated by F'. A rule in* open GSOS format *is a natural transformation $\rho : FD \to BT'$.*

We will assume the existence of a natural transformation $\iota_{F,F'} : F \to F'$ between signature functors. The intuition is that $\iota_{F,F'}$ corresponds to the set

inclusion of the operations (function symbols) corresponding to each of the signatures, and henceforth we shall call this morphism an *inclusion*, but formally all we require is that $\iota_{F,F'}$ is natural. It is straightforward to extend $\iota_{F,F'}$ to the terms by induction, yielding a monad morphism $\iota_{T,T'} : T \to T'$. Likewise we have an inclusion for the behavior functors and the obvious extension to the copointed behavior functors. When the types are obvious, we will omit the subscripts.

3.2 Operational model

The following is a generalization of [20]. In this section, the monads T and T' are the free monads over F and F', respectively.

Definition 2. *Suppose that there exists a natural transformation $\iota : T \to T'$ between monads T and T'. An* open distributive law of T, T' over the copointed functor D *is a natural transformation $\Lambda : TD \to DT'$, subject to the following three coherence conditions:*

$$\begin{array}{ccc} D \xrightarrow{\eta_D} TD \\ \searrow^{D\eta'} \downarrow{\scriptstyle \Lambda} \\ & DT' \end{array} \qquad \begin{array}{ccccc} TTD & \xrightarrow{T\Lambda} & TDT' & \xrightarrow{\Lambda_{T'}} & DT'T' \\ {\scriptstyle \mu_D}\downarrow & & & & \downarrow{\scriptstyle D\mu'} \\ TD & & \xrightarrow[\Lambda]{} & & DT' \end{array} \qquad \begin{array}{ccc} TD & \xrightarrow{\Lambda} & DT' \\ {\scriptstyle T\pi_1}\downarrow & & \downarrow{\scriptstyle (\pi_1)_{T'}} \\ T & \xrightarrow[\iota]{} & T' \end{array}$$

From left to right, the first condition says that the law should behave trivially on variables, the second condition characterizes the compositionality of the semantics, and the third condition says that the first component of the result is essentially the input, included into T'.

Proposition 1. *There exists a map $\rho \mapsto \Lambda^\rho$, which is a one-to-one correspondence between natural transformations $\rho : FD \to BT'$ and open distributive laws $\Lambda^\rho : TD \to DT'$.*

Proof. From an open GSOS rule ρ_X we obtain a morphism $\Lambda^\rho_X : TDX \to DT'X := \mathit{fold}\,(D\eta'_X)\,(D\mu'_X \circ \widetilde{\rho}_{T'X})$ by recursion, using the auxiliary morphisms

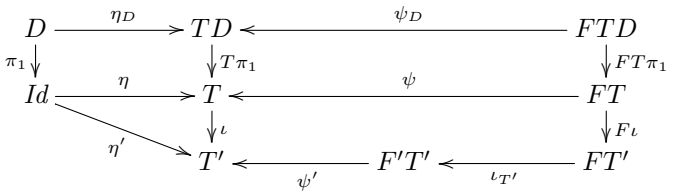
$$\widetilde{\psi} \;\frac{}{FD \xrightarrow{F(\eta\circ\pi_1)} FT \xrightarrow{\psi} T} \qquad \widetilde{\rho} \;\frac{}{FD \xrightarrow{\langle\widetilde{\psi},\rho\rangle} T \times BT' \xrightarrow{\iota_{T,T'}\times id} DT'}.$$

The inverse is:

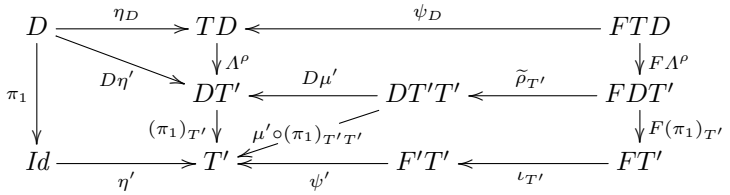
$$\begin{aligned} \Lambda^\rho_X \circ \phi_{DX} &= \Lambda^\rho_X \circ \psi_{DX} \circ F\eta_{DX} \\ &= D\mu'_X \circ \widetilde{\rho}_{T'X} \circ F\Lambda^\rho_X \circ F\eta_{DX} \\ &= D\mu'_X \circ \widetilde{\rho}_{T'X} \circ F(\Lambda^\rho_X \circ \eta_{DX}) \\ &= D\mu'_X \circ \widetilde{\rho}_{T'X} \circ FD\eta'_X \\ &= D\mu'_X \circ DT'\eta'_X \circ \widetilde{\rho}_{T'X} \\ &= D(\mu'_X \circ T'\eta'_X) \circ \widetilde{\rho}_{T'X} \\ &= Did \circ \widetilde{\rho}_{T'X} \\ &= \widetilde{\rho}_{T'X}. \end{aligned}$$

The first coherence condition holds by the definition of Λ^ρ. The verification of the second condition is entirely analogous to Lemma 3.5.2i in [2]. In order to verify the third coherence condition for Λ^ρ, we show that the next two diagrams commute. We do so by applying induction on the terms, i.e. (1).

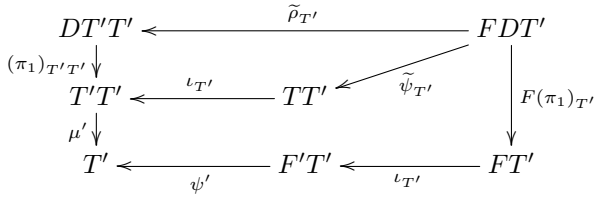
First, consider the following commuting diagram:



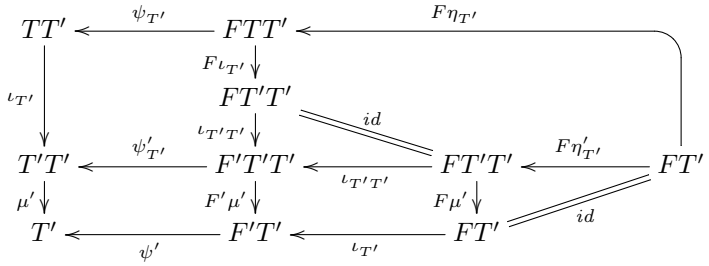
Now, consider the following diagram:



Everything but the bottom right pentagon commutes trivially. Note that by using the definition of $\widetilde{\rho}$, we can split up the pentagon as follows:



The bottom region follows from the commutativity of the following diagram, in which we have unfolded the definition of $\widetilde{\psi}$:



This completes the proof. □

In the proof, Λ^ρ is obtained from ρ by induction over the terms.

Any open distributive law Λ, whether obtained from an open GSOS rule or not, induces an *operational model*:

$$op_\Lambda \; \frac{X \xrightarrow{h} BX}{TX \xrightarrow{T\langle id,h\rangle} TDX \xrightarrow{\Lambda_X} DT'X \xrightarrow{(\pi_2)_{T'X}} BT'X}.$$

The operational model takes an environment h (hypotheses about the behavior of variables) and maps it over the terms, and then applies the distributive law. The projection π_2 leaves us with the resulting behavior. Throughout the rest of this paper we will use the notation $h_+ := (\iota_{B,B_+})_X \circ h$, to denote the inclusion of the environment h into the extended behavior.

In light of Proposition 1, it is also possible to derive the operational model directly from a GSOS rule.

First we need the following lemma.

Lemma 1. *For all morphisms k, h, g as in the diagram below, there exists a unique morphism f that makes the diagram commute:*

$$\begin{array}{ccccc} X & \xrightarrow{\eta_X} & TX & \xleftarrow{\psi_X} & FTX \\ & \searrow^{k} & \downarrow f & & \downarrow F\langle g,f\rangle \\ & & B & \xleftarrow[h]{} & F(A\times B) \end{array}$$

Proof. Remark that $h \circ F\langle g, f\rangle = h \circ F(g \times id) \circ F\langle id, f\rangle$. Then apply the structural recursion theorem with accumulators, i.e. Theorem 5.1 in [20]. □

Lemma 2. *The operational model $op_{\Lambda^\rho} h$, where the distributive law is obtained from a GSOS rule ρ, is equivalent to the unique morphism $op'_\rho h$ in following the diagram:*

$$\begin{array}{ccccccc} X & \xrightarrow{\eta_X} & TX & & \xleftarrow{\qquad\psi_X\qquad} & & FTX \\ {\scriptstyle h}\downarrow & & \downarrow{\scriptstyle op'_\rho\, h} & & & & \downarrow{\scriptstyle F\langle \iota_X, op'_\rho\, h\rangle} \\ BX & \xrightarrow[B\eta'_X]{} & BT'X & \xleftarrow[B\mu'_X]{} & BT'T'X & \xleftarrow[\rho_{T'X}]{} & FDT'X \end{array}$$

Proof. Consider the following expansion of op_{Λ^ρ}:

$$\begin{array}{ccccccc} X & \xrightarrow{\eta_X} & TX & & \xleftarrow{\qquad\psi_X\qquad} & & FTX \\ {\scriptstyle \langle id,h\rangle}\downarrow & & \downarrow{\scriptstyle T\langle id,h\rangle} & & & & \downarrow{\scriptstyle FT\langle id,h\rangle} \\ DX & \xrightarrow{\eta_{DX}} & TDX & & \xleftarrow{\qquad\psi_{DX}\qquad} & & FTDX \\ & \searrow^{D\eta'_X} & \downarrow{\scriptstyle \Lambda^\rho_X} & & & & \\ {\scriptstyle \pi_2}\downarrow & & DT'X & & & & \downarrow{\scriptstyle F\Lambda^\rho_X} \\ & & \downarrow{\scriptstyle \pi_2} & & & & \\ BX & \xrightarrow[B\eta'_X]{} & BT'X & \xleftarrow[B\mu'_X]{} & BT'T'X & \xleftarrow[\rho_{T'X}]{} & FDT'X \end{array}$$

The triangle commutes by the first coherence condition of Λ^ρ, the rest follows by definition. Since the above diagram commutes, we can conclude by Lemma 1 that the operational models are equivalent. □

In Section 4 we will introduce a Λ which is a distributive law, depending on some conditions. In that case, the above lemma does not apply, and we will prefer to make use of the original version. In Section 5, the distributivity of Λ will be essential, and we make use of the alternative version to prove the main result of that section.

3.3 Operational conservative extensions

A language extension relates two open GSOS rules, the *base language* and the *extended language*. We call an extension *conservative* when the base language, as included in the extended language, retains its original behavior, see also [1]. In the rest of this paper we will omit the word "conservative", as everything we do is in this spirit.

As a convention, we will write F_+, F'_+ (T_+, T'_+) for the in- and outgoing signatures (terms) of the extended language, respectively, and B_+ (D_+) for the (copointed) behavior functor.

Definition 3. *Let $\rho : FD \to BT'$ and $\rho_+ : F_+D_+ \to B_+T'_+$ be open GSOS rules. Then ρ_+* is a rule extension of *ρ if the diagram below holds.*

$$\begin{array}{ccccc} FD & \xrightarrow{\iota_D} & F_+D & \xrightarrow{F_+\iota} & F_+D_+ \\ {\scriptstyle\rho}\downarrow & & & & \downarrow{\scriptstyle\rho_+} \\ BT' & \xrightarrow[B\iota]{} & BT'_+ & \xrightarrow[\iota_{T'_+}]{} & B_+T'_+ \end{array}$$

Let $\Lambda : TD \to DT'$ and $\Lambda_+ : T_+D \to D_+T_+$ be natural transformations. Then Λ_+ is a law extension *of Λ if the diagram below holds.*

$$\begin{array}{ccccc} TD & \xrightarrow{\iota_D} & T_+D & \xrightarrow{T_+\iota} & T_+D_+ \\ {\scriptstyle\Lambda}\downarrow & & & & \downarrow{\scriptstyle\Lambda_+} \\ DT' & \xrightarrow[D\iota]{} & DT'_+ & \xrightarrow[\iota_{T'_+}]{} & D_+T'_+ \end{array}$$

We view the full language as a closed set of rules that is obtained by gradually extending a base language with new rules. If we take the liberty to assume a category of partial functions as the underlying category, we can also view the rule extension as the inequality $\rho_X \leq (\rho_+)_X$ between the two families of morphisms $\{\rho_X\}_{X\in C}$ and $\{(\rho_+)_X\}_{X\in C}$, and the full language would be the join of all sub-languages. However, this is not general enough for Section 4.

Proposition 2. *Suppose that the signature inclusions satisfy:*

$$\iota_{F',F'_+} \circ \iota_{F,F'} = \iota_{F_+,F'_+} \circ \iota_{F,F_+}.$$

Then, if ρ_+ is an extension of ρ, then Λ^{ρ_+} is an extension of Λ^ρ.

Proof. The proof proceeds by induction on the terms. At the core of the induction proof, one is required to show that $\widetilde{\rho'}_{U'}$ is an extension of $\widetilde{\rho}_{T'}$ (recall from Proposition 1 that $\widetilde{\rho}$ is used as an intermediate step to obtain the distributive law Λ^ρ from a rule ρ). This can be proved by considering the two cases where we post-compose the equality to be proved with the projections π_1 and π_2. In the first case, the commutativity can be proved making use of the assumption about the signature inclusions. In the second case, one makes use of the assumption that ρ' is an extension of ρ. The full proof has been carried out in Coq. □

Proposition 3. *Suppose that Λ_+ is an extension of Λ. Let $h : X \to BX$ be arbitrary. Then it holds that op_{Λ_+} is an extension of op_Λ, i.e.*

$$\begin{array}{ccccc} TX & & \xrightarrow{\quad\iota_X\quad} & & T_+X \\ {\scriptstyle op_\Lambda\, h}\downarrow & & & & \downarrow{\scriptstyle op_{\Lambda_+}\, h_+} \\ BT'X & \xrightarrow[B\iota_X]{} & BT'_+X & \xrightarrow[\iota_{T'_+X}]{} & B_+T'_+X \end{array}$$

Proof. We prove that the following diagram commutes.

$$\begin{array}{ccccc} TX & & \xrightarrow{\quad\iota_X\quad} & & T_+X \\ {\scriptstyle T\langle id,h\rangle}\downarrow & & & \swarrow{\scriptstyle T_+\langle id,h\rangle} & \downarrow{\scriptstyle T_+\langle id,h_+\rangle} \\ TDX & \xrightarrow[\iota_{DX}]{} & T_+DX & \xrightarrow[T_+\iota_X]{} & T_+D_+X \\ {\scriptstyle \Lambda_X}\downarrow & & & & \downarrow{\scriptstyle (\Lambda_+)_X} \\ DT'X & \xrightarrow[D\iota_X]{} & DT'_+X & \xrightarrow[\iota_{T'_+X}]{} & D_+T'_+X \end{array}$$

The square in the top part follows by naturality of ι_{T,T_+}, the triangle follows using the definition of ι_{D,D_+}, the bottom region follows from Proposition 2. The theorem follows trivially by making use of naturality of π_2. □

4 Silent Transitions

It is trivial to represent silent transitions by adjusting the behavior functor to $X + BX$. However, the problem with this approach is that for example the terms seq skip x and x have different semantics, since in this case the silent transitions are not truly unobservable, therefore and bisimilarity does not hold.

In this section, we introduce a merging of silent transition rules with an existing open distributive law, resulting in an operational model where silent transitions are truly unobservable. We will need to assume that the underlying category is **CPPO**-enriched, to ensure the existence of a least fixpoint construction. Specifically, the examples are aimed at a category of partial maps.

4.1 Unfolding rules and their conservative extensions

A rule for a silent transition typically consists of an operation of the base language applied to a computed value, e.g. seq skip $x \Longrightarrow x$. In Section 3 we had

two kinds of signatures: the ingoing and the outgoing signature. We add a third signature functor F'', which consists of the operations of the original ingoing signature functor F, together with the computed values. Rules for silent transitions will be regarded as maps $T'' \to TT''$, where T'' is the free monad generated by F''. For example, the aforementioned rule has the corresponding mapping $\mathsf{seq}\ \mathsf{skip}\ x \longmapsto x$. We call these maps *unfolding rules* if they unfold variables in a trivial way:

Definition 4. *An* unfolding rule *is a natural transformation $r : T'' \to TT''$, subject to the condition $r \circ \eta'' = T\eta'' \circ \eta$.*

Set $v : T \to Id := [id, \bot] \circ \kappa$. This auxiliary morphism is called a *variable classifier*, used to query whether a given term is a variable. The infinite unfolding of r is $\overline{r} := \Phi^*_X\,(\eta_X \circ v''_X)$, the least fixpoint of Φ:

$$\Phi\ \frac{T'' \xrightarrow{f} T}{T'' \xrightarrow{r} TT'' \xrightarrow{Tf} TT \xrightarrow{\mu} T}.$$

One can show that this is a well-formed definition in a **CPPO**-enriched category, and that it is a natural transformation, see [9]. We compute a few examples:

- $\overline{r}\ (\mathsf{if}\ \mathsf{true}\ (\mathsf{if}\ \mathsf{false}\ x\ y)\ z) = y$
- $\overline{r}\ (\mathsf{loop}\ x) = \mathsf{seq}\ x\ (\mathsf{seq}\ x\ (\mathsf{seq}\ x\ \ldots))$
- $\overline{r}\ (\mathsf{skip}) = \bot$

We also wish that $\overline{r}$ behaves as the identity on terms which have no silent transition rules acting on them, e.g. $\overline{r}\ (\mathsf{seq}\ x\ y) = \mathsf{seq}\ x\ y$. This is not warranted by Definition 4, but can be solved by requiring that r is based on a *decomposition structure* [9].

Definition 5. *Suppose that we have unfolding rules $r : T'' \to TT''$ and $r_+ : T''_+ \to T_+T''_+$. Then r_+ is an* unfolding rule extension *of r if the following condition is satisfied:*

$$\begin{array}{ccccc} T'' & & \xrightarrow{\quad\iota\quad} & & T''_+ \\ {\scriptstyle r}\downarrow & & & & \downarrow{\scriptstyle r_+} \\ TT'' & \xrightarrow[T\iota]{} & TT''_+ & \xrightarrow[\iota_{T''_+}]{} & T_+T''_+ \end{array}$$

Lemma 3. $v_{T''} \circ \iota_{T,T''} = v_T$.

Proof. As follows:

$$\begin{aligned} v_{T''} \circ \iota_{T,T''} &= [id, \bot] \circ \kappa'' \circ \iota_{T,T''} \\ &= [id, \bot] \circ (id + (\iota_{F,F''} \circ F\iota_{T,T''})) \circ \kappa \\ &= [id, \bot] \circ \kappa \\ &= v_T. \end{aligned}$$

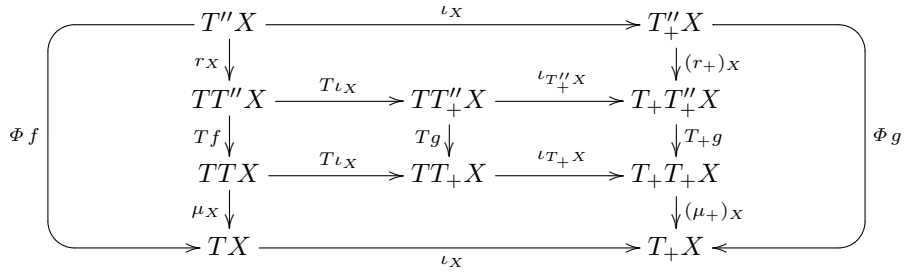
Note that the second step follows by the definition of $\iota_{T,T''}$. □

Lemma 4. $\overline{r_+} \circ \iota_{T'',T''_+} = \iota_{T,T_+} \circ \overline{r}$.

Proof. We proceed by fixpoint induction on Φ.

Base case. Making use of Lemma 3 in the second step: $\eta_+ \circ v''_+ \circ \iota_{T'',T''_+} = \eta_+ \circ v'' = \iota_{T,T_+} \circ \eta \circ v''$.

Induction step. Assume that $(\iota_{T,T_+})_X \circ f = g \circ (\iota_{T'',T''_+})_X$ for some $f : T''X \to TX$ and $g : T''_+X \to T_+X$. Consider the following diagram:



The top and bottom squares follow from the definition of Φ, the pentagon on the left is the assumption of this lemma, the top middle square by the induction hypothesis, the bottom middle square by naturality of ι_{T,T_+}, and the pentagon on the right by the fact that ι_{T,T_+} is a monad morphism.

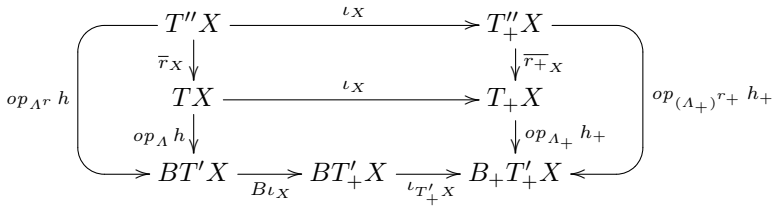
This completes the proof. □

4.2 An open distributive law for silent transitions

We incorporate the infinite unfolding into a law of type $T''D \to DT'$ by setting $\Lambda^r := \Lambda \circ \overline{r}_D$. One can view Λ^r as an extension of Λ, in the sense of Definition 3. In this situation, the inclusion of the ingoing terms is $\overline{r}$, and since the behavior functors and outgoing signatures are equal, the inclusion of the outgoing terms is the identity. With $(\Lambda_+)^{r+}$ being the usual extension counterpart, we can prove the following theorem.

Theorem 1. *$op_{(\Lambda_+)^{r+}}$ is an extension of op_{Λ^r}.*

Proof. As in Proposition 3 we need to prove the statement for an arbitrary $h : X \to BX$. We show that everything in the following diagram commutes:



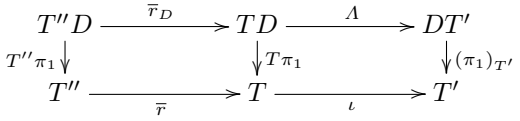
The top square commutes by Lemma 4, and the other regions commute by Proposition 3. Note that for the two side regions we instantiate $\iota_{T'',T}$ with $\overline{r}$ and $\overline{r_+}$. This is permitted, as the only requirement about $\iota_{T'',T}$ by Proposition 3 is that it is a natural transformation. □

Lemma 5. *Λ^r is a natural transformation, and satisfies the first and third coherence condition of open distributive laws.*

Proof. The first point is obvious, since Λ^r is a composition of two natural transformations.

The second condition follows by $\Lambda^r \circ \eta''_D = \Lambda \circ \overline{r}_D \circ \eta''_D = \Lambda \circ \eta_D = D\eta'$, making use of Lemma 11 in [9], and the first coherence condition of Λ.

The third condition can be proved, with $\overline{r} \circ \iota_{T,T'}$ being the inclusion morphism. Making use of naturality of $\overline{r}$ and the third coherence condition of Λ we can see that the following diagram commutes:



This concludes the proof. □

If r contains "non-regular" rules such as $\mathsf{seq}\ \mathsf{skip}\ x \Longrightarrow x$, then Λ^r fails to be compositional, i.e. it does not meet the second coherence condition. This can be seen by considering the layering of $\mathsf{seq}\ \mathsf{skip}\ x$ as an instance of $TTDX$ into $\mathsf{seq}\ (\eta\ \mathsf{skip})\ (\eta\ x)$. Then the upper leg of the diagram results in $\bot$, since $\Lambda^r_X\ \mathsf{skip} = \bot$, while the lower leg does not result in $\bot$. By requiring that r is regular, a notion due to Klin [9], we can show that Λ^r is an open distributive law.

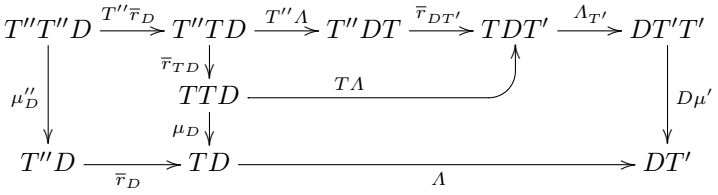
Definition 6. *An unfolding rule r is* regular *if it can be generated by recursion from a rule $r_0 : F'' \to TT''$ such that $r_0 \circ \iota_{F,F''} = T\eta'' \circ \phi$.*

Thus, the previous rule is not regular, but the rule for loop (see Figure 1) is.

Proposition 4. *If r is a regular unfolding rule, then Λ^r is an open distributive law.*

Proof. What remains to verify is that Λ^r satisfies the second coherence condition, as the other conditions have already been proved in Lemma 5.

Theorem 30 in [9] says that when r is regular, then $\overline{r}$ is a monad morphism from T'' to T. Consider the following diagram in which we have unfolded the definition of Λ^r:



The region on the left commutes due to $\overline{r}$ being a monad morphism, the square commutes by the naturality of $\overline{r}$, and the remaining region commutes due to the second coherence condition of Λ. □

5 Algebraic Laws

It is desirable that bisimulation relations remain intact under extensions of the language, so that bisimilarity proofs have to be checked only once. As it turns out, whether this holds depends on the precise definition of bisimulation one chooses to work with. In this section we consider a variant of the standard notion of bisimulations, called Formal-Hypothesis bisimulations, tailored to the fact that the state-space of the operational model consists of the terms. The perhaps most straightforward choice, also called Closed-Instance bisimilarity, demands that all substitutions of variables with closed terms are again bisimulations. However, CI-bisimulations are not preserved by language extensions [16]. The reason for this is that only substitutions with closed terms from the base language are considered by the hypotheses.

FH-bisimulations, introduced by De Simone [5], take into account that the variables of the terms in question may exhibit arbitrary behavior. CI- and FH-bisimulations have only been studied in the context of transition systems, and not of a generic B-coalgebra. We introduce FH-bisimulations here, adapted to our coalgebraic setting.

Definition 7 (FH-bisimulation). *A relation $R \subseteq TX \times TX$ is an* FH-bisimulation relation *on op_Λ if for every environment $h : X \to BX$, it holds that R is a bisimulation relation on $op_\Lambda\, h$.*

Since FH-bisimulations relate terms, we call a pair of such terms an *algebraic law*. Some obvious examples can be found by formulating silent transition rules as algebraic laws: $\mathsf{seq}\ \mathsf{skip}\ x = x$ and $\mathsf{catch}\ \mathsf{skip} = \mathsf{skip}$. Note that if the terms are closed, then the definition coincides with standard bisimulations.

5.1 The preservation of algebraic laws

We can prove that algebraic laws are preserved by conservative language extensions, by making use of the fact that the operational behavior is preserved from Theorem 1.

Theorem 2. *Suppose that $R \subseteq TX \times TX$ is an FH-bisimulation relation on op_Λ, and that $B = B_+$. Then*

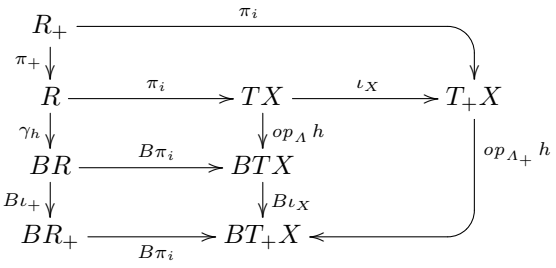
$$R_+ := \{\langle (\iota_{T,T_+})_X\, x,\, (\iota_{T,T_+})_X\, y\rangle \mid x\, R\, y\}$$

is an FH-bisimulation on op_{Λ_+}.

Proof. First note that R_+ and R are isomorphic; we will denote the corresponding morphisms by $\pi_+ : R_+ \to R$ and $\iota_+ : R \to R_+$. Let $h : X \to BX$ be given. The assumption of the theorem says that for any h, there exists a morphism γ_h satisfying the bisimulation diagram in Section 2. We claim that the following composition is the required morphism to prove that R_+ is a bisimulation relation:

$$\frac{R \xrightarrow{\gamma_h} BR}{R_+ \xrightarrow{\pi_+} R \xrightarrow{\gamma_h} BR \xrightarrow{B\iota_+} BR_+}.$$

We verify this by the commuting diagram below for $i = 1, 2$, making use of Theorem 1.



Thus, the claimed bisimulation mapping is correct, which finishes the proof. □

The premise of Theorem 2 essentially means that the extended language can not add any new effects. As a simple example to see how this fails without this premise, first consider the law $\mathsf{catch}\ x = x$ in the absence of exceptions (i.e. for every state s, set $s.\mathsf{ex} = \mathsf{false}$) as the base language, and then add states with exceptions to the extended language.

5.2 Combining algebraic laws

Just as in the situation with standard bisimulations, there exists a greatest FH-bisimulation relation, notation $\overset{\text{FH}}{\leftrightarrow}$, which is the union of all FH-bisimulation relations. It is straightforward to show that $\overset{\text{FH}}{\leftrightarrow}$ itself is an FH-bisimulation relation, and that it is an equivalence relation.

Lemma 6. *There is a greatest FH-bisimulation relation, notation $\overset{\text{FH}}{\leftrightarrow}$. It is the union of all FH-bisimulation relations.*

Proof. Suppose that we have $v \overset{\text{FH}}{\leftrightarrow} w$. Then there exists an FH-bisimulation relation R such that $v\, R\, w$. This means that for all environments h, that $op_\Lambda\, h\, v$ and $op_\Lambda\, h\, w$ are in the lifted relation $Rel(B)(R)$, which implies that they are also in $Rel(B)(\overset{\text{FH}}{\leftrightarrow})$, for all h, since R is included in the greatest FH-bisimulation relation. □

Define the substitution of a function $f : X \to TX$ in a term t as $t[f] := \mu_X(Tf\,t)$. We will be concerned with proving that the following relation is an FH-bisimulation relation:

$$R := \{\langle t[f_1],\, u[f_2]\rangle \mid t \overset{\text{FH}}{\leftrightarrow} u\},$$

in which $f : X \to \overset{\text{FH}}{\leftrightarrow}$, a function which assigns to variables an FH-bisimilar pair of terms, and $f_i := \pi_i \circ f$ for $i = 1, 2$. Barring that this is valid, and knowing that $\mathsf{loop}\ x = \mathsf{seq}\ x\ (\mathsf{loop}\ x)$ is a valid algebraic law, we can use the substitution $f : x \mapsto \langle \mathsf{loop}\ x, \mathsf{seq}\ x\ (\mathsf{loop}\ x)\rangle$ to derive that for example $\mathsf{loop}\ x = \mathsf{seq}\ x\ (\mathsf{seq}\ x\ (\mathsf{loop}\ x))$. We call this property of FH-bisimulations *being substitutive* [18]. We prove this in two steps by considering the special cases preservation by *instantiation* ($t = u$) and preservation by *insertion* ($f_1 = f_2$).

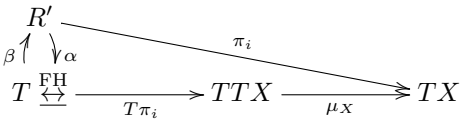
The proofs below make essential use of the assumption that Λ is a distributive law. In the light of distributive laws obtained from mergings with unfolding rules as in Section 4, this section only applies to the situation where the unfolding rules are regular. This assumption provides a well-known property of bialgebraic semantics, which says that the operational model is compositional.

Lemma 7 (Compositionality). $B\mu_X \circ op_\Lambda\,(op_\Lambda\,h) = op_\Lambda\,h \circ \mu_X$.

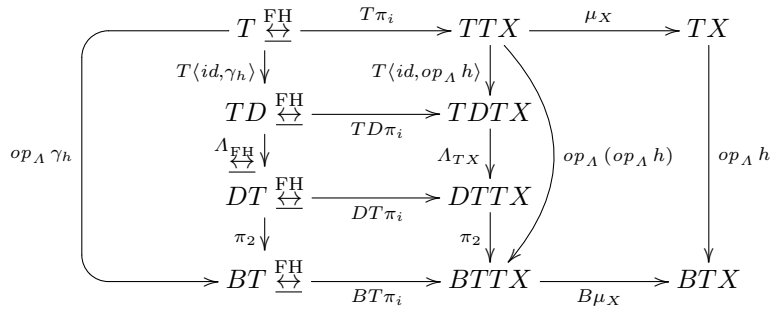
Proof. Straightforward, making use of the naturality of μ and the second coherence condition of Λ. □

Proposition 5. *The FH-bisimilarity relation is preserved by instantiation.*

Proof. We need to prove that $R' := \{\langle t[f_1], t[f_2]\rangle\}$ is an FH-bisimulation, in which $f : X \to \overset{\text{FH}}{\leftrightarrow}$. Remark that we have an isomorphism $R' \underset{\beta}{\overset{\alpha}{\rightleftarrows}} T \overset{\text{FH}}{\leftrightarrow}$, such that:



for $i = 1, 2$. Suppose that $h : X \to BX$ is arbitrary. The commuting diagram below in conjunction with the above remark shows that R' is an FH-bisimulation, with bisimulation mapping $B\beta \circ op_\Lambda\,\gamma_h \circ \alpha$.



The region on the right is exactly compositionality of the operational model. The squares, from top to bottom respectively, make use of the fact that $\overset{\text{FH}}{\leftrightarrow}$ itself is an FH-bisimulation relation, and that Λ and π_2 are natural transformations. □

Definition 8. *Given a coalgebra $d : Y \to BY$ and $f : X \to Y$, d* can be simulated via *f if there exists a coalgebra $c : X \to BX$ and a morphism $f' : X \to Y$ such that $d \circ f = Bf' \circ c$.*

For the next proposition, we will need to make the assumption that for any h, $op_\Lambda\, h$ can be simulated via f. To see that this is a reasonable assumption, as an example, for X take an infinitely large set of variables, and $BX := A \times X$, for some label set A. Gödel numberings provide a way to assign a unique number to each well-formed term [19]. This means that there exist $\mathsf{enc} : TX \to X$ and $\mathsf{dec} : X \to TX$, which form a bijection between X and TX. Then the validity of the assumption for this choice of B is witnessed by

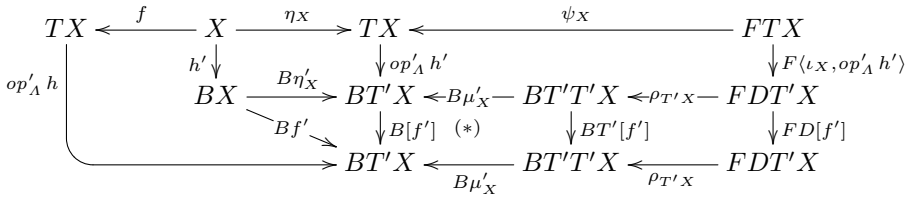
$$h'\, x := B\mathsf{enc}\,(op_\Lambda\, h\,(f\, x)), \qquad f'\, x_B := B\mathsf{dec}\,(x_B).$$

We will need the following technical lemma to prove preservation by insertion.

Lemma 8. *If for every $h : X \to BX$, $op_\Lambda\, h$ can be simulated via $f : X \to TX$, then $op_\Lambda\, h$ can be simulated via $[f]$ by $op_\Lambda\, h'$ for some $h' : X \to BX$, in particular, the following diagram commutes:*

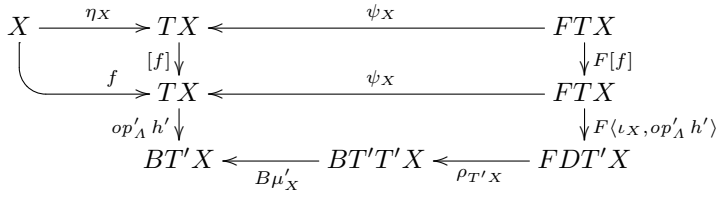
$$\begin{array}{ccc} TX & \xrightarrow{[f]} & TX \\ {\scriptstyle op_\Lambda\, h'}\downarrow & & \downarrow{\scriptstyle op_\Lambda\, h} \\ BT'X & \xrightarrow[B[f']]{} & BT'X \end{array}$$

Proof. We use the alternative version of the operational model in this proof, i.e. $op'_\Lambda\, h = op_\Lambda\, h$, see Lemma 2. Assume that ρ corresponds to Λ, as in Proposition 1. Using Lemma 1, we can formulate both legs of the above diagram as unique morphisms, from which it follows that they are equal. For the upper leg:



The region on the left follows from the assumption of the lemma. The fact that the square marked by $(*)$ commutes can be seen by unfolding $[f'] = \mu_X \circ Tf'$, applying the naturality of μ', and (2), since T' is a monad. The rest of the diagram commutes trivially.

For the lower leg, consider the following commuting diagram:



The top part follows from the fact that $[f] = \mathit{fold}\, f\, \psi_X$, using (1).

This concludes the proof. □

Proposition 6. *The FH-bisimulation relation is preserved by insertion.*

Proof. We need to prove that $R' := \{\langle t[f], u[f]\rangle \mid t \overset{\text{FH}}{\leftrightarrow} u\}$ is an FH-bisimulation relation, where $f : X \to TX$. This means that we need to show that for any set of hypotheses $h : X \to BX$, and terms $t_0, u_0 : TX$ such that $t_0\, R\, u_0$, it holds that $\langle op_\Lambda\, h\, t_0, op_\Lambda\, h\, u_0\rangle \in \mathit{Rel}(B)(R')$.

It follows by the definition of R' that $op_\Lambda\, h\, t_0 = op_\Lambda\, h\, (t[f])$ for some t, and likewise for u_0. By the assumption that $op_\Lambda\, h$ can be simulated via any morphism, by Lemma 8 there exist $f' : X \to TX$ and $h' : X \to BX$, such that $op_\Lambda\, h \circ [f] = B[f'] \circ op_\Lambda\, h'$.

Thus, what remains to prove is that

$$\langle B[f']\, (op_\Lambda\, h'\, t), B[f']\, (op_\Lambda\, h'\, u)\rangle \in \mathit{Rel}(B)(R').$$

It is enough to show that

$$\langle op_\Lambda\, h'\, t, op_\Lambda\, h'\, u\rangle \in \mathit{Rel}(B)(\lambda\, x\, y, (x[f'])\, R'\, (y[f'])),$$

which is equivalent to saying that the above pair is included in $\mathit{Rel}(B)(R')$, which is true by the fact that $t \overset{\text{FH}}{\leftrightarrow} u$. □

Theorem 3. *The FH-bisimilarity relation is substitutive.*

Proof. By the previous two propositions, $t[f_1] \overset{\text{FH}}{\leftrightarrow} t[f_2] \overset{\text{FH}}{\leftrightarrow} u[f_2]$, and thus by transitivity of $\overset{\text{FH}}{\leftrightarrow}$ we can conclude that FH-bisimilarity is substitutive. □

6 Running the operational semantics

This section highlights another application of Theorem 1.

Final coalgebras, i.e. pairs $\langle Z, \zeta : Z \to BZ\rangle$, enjoy the property that there exists an operator $\mathit{unfold} : (X \to BX) \to X \to Z$ which takes a coalgebra as its argument and returns a morphism $X \to Z$ which maps the state-space of the argument to the final state-space. This morphism $\mathit{unfold}\, c$ is the unique

coalgebra homomorphism from c to ζ. Here Z is the greatest solution to the equation $Z \cong BZ$.[3]

When the rules are closed, then $op_\Lambda\, h$ (where $h : X \to BX$) is a coalgebra and we can "run" the operational semantics by unfolding it, i.e. $run_\Lambda\, h := unfold(op_\Lambda\, h)$. Set $\iota_{Z,Z_+} := unfold((\iota_{B,B_+})_Z \circ \zeta)$. We can prove that running the extended operational model is faithful to running the base model.

Proposition 7. *Suppose that Λ and Λ_+ are closed distributive laws, and that Λ_+ is an extension of Λ. Then for all $h : X \to BX$ it holds that $\iota_{Z,Z_+} \circ run_\Lambda\, h = run_{\Lambda_+}\, h_+ \circ (\iota_{T,T_+})_X$.*

Proof. Both sides of the equation are coalgebra homomorphisms from the coalgebra $c := (\iota_{B,B_+})_{TX} \circ op_\Lambda\, h$ to the final coalgebra ζ_+. For the LHS this follows easily from the definitions, while for the RHS we make use of Theorem 1. By the fact that ζ_+ is final, there is at most one coalgebra homomorphism from c to ζ_+, and thus the equality holds. □

7 Related Work

The results in this paper were developed within the bialgebraic semantics framework, a body of research initiated by Turi and Plotkin [20].

The theorems in Section 5, which prove that FH-bisimulations are preserved by conservative extensions, and that FH-bisimulations are substitutive, transfer the original results, obtained in the more traditional set-theoretic approach to SOS by Mosses et al. [16] and Rensink [18] respectively, to the bialgebraic framework. FH-bisimulations were originally introduced by De Simone [5].

The dichotomy between value terms and computational terms was emphasized by Churchill and Mosses [4], who introduce a rule format built on the tyft format, which has built-in rules to deal with silent transitions. They provide a variant of bisimilarity, and prove that it is a congruence in the resulting transition system. The distributive law Λ^r of Section 4 has similar characteristics, through the infinite unfolding of silent transitions. This law is a variant of the one introduced by Klin [9].

An alternative to considering only free monads as in the present paper, is to quotient the term monad by the algebraic laws. Bonsangue et al. [3] prove that if Λ respects the algebraic laws, then there is a unique distributive law Λ' such that the quotient map is a well-behaved translation from Λ to Λ'.

A modular variant of GSOS has been provided by Jaskelioff et al. [8] as part of a Haskell implementation of the bialgebraic framework. They distinguish ingoing from outgoing signatures, as in the present paper, but consider the outgoing signature as an abstract parameter of each modular rule, and add

[3] For $BX := S \to 1 + S \times X$, as in Section 3, Z is given by the set of partial functions

$$\{f : S^* \to S^* \mid f \text{ is length and prefix preserving}\}.$$

type-class constraints to ensure the inclusion of certain operations in the outgoing signature.

Watanabe [21] provides an interpretation of operational conservative extensions [1] in terms of distributive laws, and proves a variant of Theorem 3, but does not treat the difference between ingoing and outgoing signatures.

8 Conclusions

We have provided an operational rule format, tailored to the modular description of programming languages. The semantics supports truly unobservable transitions, as generated by rules for silent transitions. We have proved that algebraic laws are preserved by conservative extensions of the operational semantics, and that algebraic laws are substitutive. Our work has been developed within the bialgebraic framework [20], making it amenable to implementation in a theorem prover [13].

In future work we wish to ease the condition in Section 5.2 on the distributive law, enabling the substitutivity of algebraic laws for a wider range of silent transition rules. We would also like to explore applications to software verification. In particular, one can view the operational rules of programming languages as pointwise extensions in the sense of [6]. We expect that this will lead to a structured way to obtain sound Hoare logics for trace-based semantics such as [17].

Acknowledgments. The inspiration for this work arose from consultation with Peter D. Mosses by the author. Without his support this work would not have been possible. The authors also wish to thank the anonymous reviewers for their sharp comments.

References

1. L. Aceto, W. Fokkink, and C. Verhoef. Structural operational semantics. In J. Bergstra, A. Ponse, and S. Smolka, editors, *Handbook of Process Algebra*, pages 197–292. Elsevier, 1999.
2. F. Bartels. *On generalised coinduction and probabilistic specification formats*. PhD thesis, Vrije Universiteit Amsterdam, 2004.
3. M. M. Bonsangue, H. H. Hansen, A. Kurz, and J. Rot. Presenting distributive laws. In *Proceedings of CALCO'13*, LNCS. Springer. to appear.
4. M. Churchill and P. D. Mosses. Modular bisimulation theory for computations and values. In F. Pfenning, editor, *Proceedings of FoSSaCS'13*, volume 7794 of *LNCS*, pages 97–112. Springer, 2013.
5. R. De Simone. Higher-level synchronising devices in Meije-SCCS. *Theor. Comp. Sci.*, 37:245–267, 1985.
6. H. H. Hansen and B. Klin. Pointwise extensions of GSOS-defined operations. *Math. Struct. in Comp. Sci.*, 21(2):321–361, 2011.
7. B. Jacobs. *Introduction to coalgebra: Towards mathematics of states and observations*. In preparation, version 2.0. 2012. `http://www.cs.ru.nl/B.Jacobs/CLG/JacobsCoalgebraIntro.pdf`.

8. M. Jaskelioff, N. Ghani, and G. Hutton. Modularity and implementation of mathematical operational semantics. *Elec. Notes in Theor. Comp. Sci.*, 229(5):75–95, 2011.
9. B. Klin. Adding recursive constructs to bialgebraic semantics. *J. of Logic and Alg. Prog.*, 60:259–286, 2004.
10. B. Klin. Bialgebras for structural operational semantics: An introduction. *Theor. Comp. Sci.*, 412(38):5043–5069, 2011.
11. M. Lenisa, J. Power, and H. Watanabe. Distributivity for endofunctors, pointed and co-pointed endofunctors, monads and comonads. *Elec. Notes in Theor. Comp. Sci.*, 33:230–260, 2000.
12. M. Lenisa, J. Power, and H. Watanabe. Category theory for operational semantics. *Theor. Comp. Sci.*, 327(1-2):135–154, 2004.
13. K. Madlener and S. Smetsers. GSOS formalized in Coq. In *Proceedings of TASE'13*, pages 199–206. IEEE, 2013.
14. P. D. Mosses. Modular structural operational semantics. *J. of Logic and Alg. Prog.*, 60:195–228, 2004.
15. P. D. Mosses. Component-based semantics. In *Proceedings of SAVCBS'09*, pages 3–10. ACM, 2009.
16. P. D. Mosses, M. R. Mousavi, and M. A. Reniers. Robustness of equations under operational extensions. In S. Fröschle and F. D. Valencia, editors, *Proceedings of EXPRESS'10*, EPTCS, pages 106–120, 2010.
17. K. Nakata and T. Uustalu. A Hoare logic for the coinductive trace-based big-step semantics of While. In A. D. Gordon, editor, *Proceedings of ESOP'10*, volume 6012 of *LNCS*, pages 488–506. Springer, 2010.
18. A. Rensink. Bisimilarity of open terms. *Inf. and Comp.*, 156(1):345–385, 2000.
19. T. A. Sudkamp and A. Cotterman. *Languages and machines: An introduction to the theory of computer science.* Addison-Wesley, third edition, 2006.
20. D. Turi and G. D. Plotkin. Towards a mathematical operational semantics. In *Proc. of LICS'97*, pages 280–291. IEEE, 1997.
21. H. Watanabe. Well-behaved translations between structural operational semantics. *Elec. Notes in Theor. Comp. Sci.*, 65(1):337–357, 2002.