Combining Bialgebraic Semantics and Equations

Jurriaan Rot and Marcello Bonsangue

LIACS — Leiden University, Niels Bohrweg 1, Leiden, Netherlands
Centrum Wiskunde en Informatica, Science Park 123, Amsterdam, Netherlands
{j.c.rot, m.m.bonsangue}@liacs.leidenuniv.nl

Abstract. It was observed by Turi and Plotkin that structural operational semantics can be studied at the level of universal coalgebra, providing specification formats for well-behaved operations on many different types of systems. We extend this framework with non-structural assignment rules which can express, for example, the syntactic format for structural congruences proposed by Mousavi and Reniers. Our main result is that the operational model of such an extended specification is well-behaved, in the sense that bisimilarity is a congruence and that bisimulation-up-to techniques are sound.

1 Introduction

Structural operational semantics (SOS) is a framework for defining the semantics of programming languages and calculi in terms of transition system specifications [1]. By imposing syntactic restrictions, one can prove well-behavedness properties of transition systems at the meta-level of their specification. For instance, any specification in the GSOS format [4] has a unique operational model, on which bisimilarity is a congruence.

Traditionally, research in SOS has focused on labelled transition systems as the fundamental model of behaviour. Turi and Plotkin [27] introduced the *bialgebraic* approach to structural operational semantics, where in particular GSOS can be studied at the level of *universal coalgebra* [22]. The theory of coalgebras provides a mathematical framework for the uniform study of many types of state-based systems, including labelled transition systems but also, e.g., (non)-deterministic automata, stream systems and various types of probabilistic and weighted automata [23,11,3]. In the coalgebraic framework, there is a canonical notion of bisimilarity, which instantiates to the classical definition of (strong) bisimilarity in the case of labelled transition systems. It is shown in [27] that GSOS specifications can be generalised by certain natural transformations, which are called *abstract GSOS specifications*, and that these correspond to the categorical notion of *distributive laws*. This provides enough structure to prove at this general level that bisimilarity is a congruence. By instantiating the theory to concrete instances, one can then obtain congruence formats for systems such as probabilistic automata, weighted transition systems, streams, etc. — see [12] for an overview. Another advantage of abstract GSOS is that bisimulation up to

context is "compatible" [21,20], providing a sound enhancement of the bisimulation proof method which can be combined with other compatible enhancements such as bisimulation up to bisimilarity [24,19].

In this paper we add rules such as in (1) to this framework. The rule in (1) properly defines the replication operator in CCS[1]: intuitively $!x$ represents $x \mid x \mid x \mid \ldots$, i.e., the infinite parallel composition of x with itself. In fact, the above rule can be seen as assigning the behaviour of the term $!x \mid x$ to the simpler term $!x$, therefore we call it an *assignment rule*. Being inherently non-structural, such an assignment rule cannot directly be embedded in the bialgebraic framework of Turi and Plotkin, where the behaviour of terms is computed inductively. In this paper we show how to interpret assignment rules together with abstract GSOS specifications. As it turns out, this requires the assumption that the functor which represents the type of coalgebra is *ordered* as a complete lattice; for example, in the case of labelled transition systems this order is simply inclusion of sets of pairs (a, x) of a label a and a state x. The operational model on closed terms then is the *least* model such that every transition can either can be derived from a rule in the specification, or there is a rule assigning to an operator σ the behaviour of a term t in the model. To ensure the existence of such least models, we disallow negative premises by using *monotone* abstract GSOS specifications, a generalisation of the *positive GSOS format* for transition systems [8]. Positive GSOS can be seen as the greatest common divisor of GSOS and the tyft/tyxt format [2].

$$\frac{!x \mid x \xrightarrow{a} t}{!x \xrightarrow{a} t} \tag{1}$$

Our main result is that the interpretation of a monotone abstract GSOS specification together with a set of assignment rules is itself the operational model of another (typically larger) abstract GSOS specification. Like the interpretation of a GSOS specification with assignment rules, we construct this latter specification by fixpoint induction. As a direct consequence of this alternative representation of the interpretation, we obtain that bisimilarity is a congruence and that bisimulation up to context is sound and even compatible — properties that do not follow from bisimilarity being a congruence [19]. As an example application, we obtain the compatibility of bisimulation-up-to techniques for CCS with replication, which so far had to be shown with an ad-hoc argument [19].

A further contribution of this paper consists in combining *structural congruences* [16,17] with the bialgebraic framework using assignment rules. Structural congruences were introduced in the operational semantics of the π-calculus in [16]. The basic idea is that SOS specifications are extended with *equations* on terms, which are then linked by a special deduction rule. This rule essentially states that if two processes are equated by the congruence generated by the set of equations, then they can perform the same transitions. Prototypical examples are the specification of the parallel operator by combining a single rule with commutativity, and the specification of the replication operator by an equation,

[1] The simpler rule $\frac{x \to x'}{!x \to !x \mid x'}$ is problematic in the presence of the sum operator [19,26].

both shown below:

$$\frac{x \xrightarrow{a} x'}{x \mid y \xrightarrow{a} x' \mid y} \qquad x \mid y = y \mid x \qquad\qquad !x = !x \mid x \tag{2}$$

In [17] Mousavi and Reniers show how to interpret SOS rules with structural congruences in various equivalent ways. They exhibit very simple examples of equations and SOS rules for which bisimilarity is not a congruence, even when the SOS rules are in the tyft (or the GSOS) format. As a solution to this problem they introduce a restricted format for equations, called cfsc, for which bisimilarity is a congruence when combined with tyft specifications.

In the present paper we show how to interpret structural congruences at the general level of coalgebras, in terms of an operational model on closed terms. We prove that when the equations are in the cfsc format then they can be encoded by assignment rules, in such a way that their respective interpretations coincide up to bisimilarity. Consequently, not only is bisimilarity a congruence for monotone abstract GSOS combined with cfsc equations, but also bisimulation up to context and bisimilarity is compatible.

Outline. In Section 2 we recall some preliminaries on (co)algebras and abstract GSOS. In Section 3, assignment rules and their interpretation are introduced. We show in Section 4 that this interpretation can be obtained as the operational model of another abstract GSOS specification. Section 5 contains the integration of structural congruence with the bialgebraic framework. In Section 6 we discuss related work, and in Section 7 we conclude with some directions for future work.

To fully understand the technical development in this paper, familiarity with basic notions in category theory, bialgebraic semantics and order theory is useful. However, many of the main results and definitions are illustrated with concrete examples, in particular on the familiar case of transition systems.

2 Coalgebras, Signatures and Bialgebraic Semantics

By Set we denote the category of sets and total functions. We write Id for the identity functor on Set.

Coalgebras. For an extensive treatment with many examples we refer to [22]. An *(F-)coalgebra* for a functor $F\colon \mathsf{Set} \to \mathsf{Set}$ consists of a set of states C and a map $\alpha\colon C \to FC$. Let (C, α) and (D, β) be two coalgebras. A function $f\colon C \to D$ is an *(F-coalgebra) homomorphism* if $Ff \circ \alpha = \beta \circ f$. A relation $R \subseteq C \times D$ is an *(F-)bisimulation* if R can be equipped with a transition structure $\gamma\colon R \to FR$ such that the two projection functions $\pi_1\colon R \to C$ and $\pi_2\colon R \to D$ are homomorphisms. The largest bisimulation between two systems α and β is called *bisimilarity*. Two F-coalgebras $\alpha, \beta\colon X \to FX$ (on a common carrier X) are *equal up to bisimilarity* if the diagonal on X progresses [21] to $\sim$.

Example 1. Labelled transition systems (LTSs) over a set of labels A are coalgebras for the functor $FX = (\mathcal{P}X)^A$. For an LTS $\alpha\colon X \to (\mathcal{P}X)^A$ we write

$x \xrightarrow{a} x'$ iff $x' \in \alpha(x)(a)$. Intuitively, for a state $x \in X$, $\alpha(x)(a)$ contains all the outgoing transitions from x labelled by a. Coalgebraic bisimulation instantiates to the classical definition by Milner and Park: a relation $R \subseteq X \times X$ is called a *bisimulation* provided that for all $(x, y) \in R$, if $x \xrightarrow{a} x'$ then there exists a state y' such that $y \xrightarrow{a} y'$ and $(x', y') \in R$, and vice versa.

Image-finite labelled transition systems are coalgebras for the functor $FX = (\mathcal{P}_\omega X)^A$, where $\mathcal{P}_\omega X$ is the set of all *finite* subsets of X.

Coalgebras for the functor $FX = \mathbb{R} \times X$, where $\mathbb{R}$ is the set of real numbers, are called *stream systems* (over the reals). In a stream system, for every state we can observe an output in $\mathbb{R}$ and a next state.

Signatures. A *signature* Σ is a (possibly infinite) collection of operator names $\sigma \in \Sigma$ with (finite) arities $|\sigma| \in \mathbb{N}$. Equivalently, it is a polynomial functor as in (3). In the sequel we write $\sigma(x_1, \ldots, x_n)$ instead of $(\sigma, (x_1, \ldots, x_n))$ for elements of ΣX. The functor ΣX acts on a map $f \colon X \to Y$ as follows: $(\Sigma f)(\sigma(x_1, \ldots, x_n)) = \sigma(f(x_1), \ldots, f(x_n))$. Above and in the sequel we abuse notation and use Σ to represent signatures as well as their associated functors.

$$\Sigma X = \coprod_{\sigma \in \Sigma} X^{|\sigma|} \tag{3}$$

A *Σ-algebra* consists of a set A and a function $\alpha \colon \Sigma A \to A$. This coincides with the standard notion of an algebra for the signature Σ. For a set of variables X and a signature Σ we denote by TX the set of *terms*, as defined by the grammar $t ::= \sigma(t_1, \ldots, t_n) \mid x$ where σ ranges over Σ, n is the arity of σ and x ranges over X. The special case $T\emptyset$ is the set of *closed terms*. Every set TX can be turned into the (free) Σ-algebra $\nu_X \colon \Sigma TX \to TX$ by defining $\nu_X(\sigma(t_1, \ldots, t_n)) = \sigma(t_1, \ldots, t_n)$. Note that $\nu_\emptyset$ is an isomorphism, since it is the initial Σ-algebra.

We note that T is the *free monad* for the signature Σ, without going into details. Of importance to our purposes is that it comes equipped with natural transformations $\eta \colon \mathsf{Id} \Rightarrow T$ and $\mu \colon TT \Rightarrow T$; for a set (of variables) X, η_X is the injection of variables into terms, and μ_X turns a term over terms into a single term in the expected manner. In the sequel, whenever the type can be deduced from the context, we omit subscripts from natural transformations to avoid notational clutter.

Bialgebraic operational semantics. See [12] for an overview of this topic. In the remainder of this paper, we assume some fixed signature Σ with associated term monad T, and a Set endofunctor F representing the type of behaviour. An *(abstract GSOS) specification* is a natural transformation of the form

$$\rho \colon \Sigma(F \times \mathsf{Id}) \Rightarrow FT\,.$$

As first observed by Turi and Plotkin [27], if F is the functor $(\mathcal{P}_\omega -)^A$ of image-finite labelled transition systems then specifications of the above type can be induced by specifications in the well-known GSOS format introduced in [4]. A

GSOS rule for an operator $\sigma \in \Sigma$ of arity n is of the form

$$\frac{\{x_{i_j} \xrightarrow{a_j} y_j\}_{j=1..m} \qquad \{x_{i_k} \overset{b_k}{\not\to}\}_{k=1..l}}{\sigma(x_1, \ldots, x_n) \xrightarrow{c} t} \tag{4}$$

where m is the number of positive premises, l is the number of negative premises, and $a_1, \ldots, a_m, b_1, \ldots, b_l, c \in A$ are labels. The variables $x_1, \ldots, x_n$, $y_1, \ldots, y_m$ are pairwise distinct, and t is a term over these variables.

If we instantiate F to the functor $\mathbb{R} \times \mathsf{Id}$ of stream systems over the reals, specifications correspond to the format of *behavioural differential equations* [23] presented in [13]. By instantiating abstract GSOS specifications to other functors one can obtain formats for many types of systems, including, e.g., syntactic formats for probabilistic and weighted transition systems [3,11].

Each specification $\rho\colon \Sigma(F \times \mathsf{Id}) \Rightarrow FT$ induces a unique *operational model* $f\colon T\emptyset \to FT\emptyset$, also called *$\rho$-model* (on the initial algebra), with the following property:

$$f \circ \nu = F\mu \circ \rho \circ \Sigma\langle f, \mathsf{id}\rangle\,. \tag{5}$$

By this equation, to compute the behaviour of a term $\sigma(t_1, \ldots, t_n)$ in f, we may compute the behaviour of its subterms $t_1, \ldots, t_n$ and then instantiate a rule from ρ. For labelled transition systems, f is precisely the unique *supported model* corresponding to a GSOS specification ρ: every transition in f is derived from rules in the specification ρ and each derivable transition occurs in f (see [1,4]).

An important property of GSOS is that bisimilarity is a congruence on the operational model corresponding to a specification [4]. Turi and Plotkin [27] proved at the general level of abstract GSOS specifications that coalgebraic bisimilarity on the operational model is a congruence, extending the result of [4] to calculi for many other types of systems. Furthermore, these specifications guarantee the (strictly stronger property of) compatibility of *bisimulation-up-to context* on the operational model [20]. This yields an enhanced proof technique for bisimilarity in which one can use the syntactic structure of the terms to relate their successors.

3 Adding Assignment Rules

In this section we consider the interpretation of abstract GSOS specifications (without negative premises) together with *assignment rules* of the form

$$\sigma(x_1, \ldots, x_n) := t \tag{6}$$

where t is a term over the variables $x_1, \ldots, x_n$. These will be interpreted as a kind of rewriting rules: the behaviour of t induces behaviour of $\sigma(x_1, \ldots, x_n)$. An example is the replication operator given in equation (1) of the introduction; this can be given by $!x := !x \mid x$. Notice that the above rules, which we will call *assignment rules*, do not fit directly into the bialgebraic framework, since they are inherently non-structural. That is, they ruin the property of GSOS specifications

that the behaviour of terms in the operational model can be computed from the behaviour of their subterms.

In the case of labelled transition systems, given a GSOS specification and a set of rules of the above form, the desired interpretation is informally as follows: every transition from a term $\sigma(t_1, \ldots, t_n)$ should either be derived from the transitions of $t_1, \ldots, t_n$ and a rule in the specification, or from an assignment rule which has σ on the left-hand side. This suggests a natural extension of the fixpoint equation of the supported model (equation (5)) to incorporate the assignment rules. However, because of assignment rules there is not necessarily a *unique* supported model anymore, since now there may be infinite inferences. For example, the rule $\sigma(x) := \sigma(x)$ does not have a unique solution. In order to rule this out, one is interested in the *least* transition system on closed terms which satisfies the extended fixpoint equation. Such a least model does not exist in general because of negative premises, so we will use a formalization of the notion of *positive* GSOS at the level of abstract specifications, and restrict to such specifications in the remainder of this paper.

To interpret specifications which involve assignment rules at the general level of a functor F one needs a notion of order on F. In the case of labelled transition systems this order is clear and often left implicit: in that case $FX = (\mathcal{P}X)^A$, and it is simply the (pointwise) subset order. To allow the desired generalisation, we assume that our behaviour functor F is *ordered* [9], that is, it factors through CJSL, which is the category of complete (join semi-)lattices and join-preserving functions. Thus we assume a functor $\hat{F}\colon \mathsf{Set} \to \mathsf{CJSL}$ such that $U \circ \hat{F} = F$, where $U\colon \mathsf{CJSL} \to \mathsf{Set}$ is the forgetful functor that takes a complete lattice to its underlying set. Thus for every set X we have arbitrary joins in FX; in the sequel we denote the join of a set $S \subseteq FX$ by $\bigvee S$, and we write $\bot$ for $\bigvee \emptyset$ and $x \leq y$ if $x \vee y = y$, for $x, y \in FX$.

Example 2. As mentioned above, the functor $(\mathcal{P}-)^A$ of labelled transition systems has a natural complete lattice structure.

Any functor $F\colon \mathsf{Set} \to \mathsf{Set}$ can be extended to an ordered functor F' by taking $F'X = FX + 2$ where $2 = \{\bot, \top\}$: we then define, for $x, y \in FX$, $x \leq y$ iff $x = y$ and use $\bot$ and $\top$ as the least and greatest element respectively.

The functor for (possibly infinitely branching) weighted transition systems [11] over a complete lattice, is ordered. For example, one can take as weights the set $\mathbb{R} \cup \{\infty, -\infty\}$, i.e., the reals extended with top and bottom elements.

For any function $f\colon X \to Y$ we have $Ff = U \circ \hat{F}(f)$ and thus Ff is a join-preserving map: $f[\bigvee S] = \bigvee f[S]$. Consequently, Ff is also monotone, i.e., $x \leq y$ implies $f(x) \leq f(y)$. Given arbitrary sets X and Y, the complete lattice on FY lifts pointwise to a complete lattice on functions of type $X \to FY$, i.e., for a collection $\{f_i\}_{i \in I}$ of functions of the form $f_i\colon X \to FY$ we define $(\bigvee\{f_i\}_{i\in I})(x) = \bigvee_{i \in I}(f_i(x))$. This induces in particular a complete lattice on the set of all coalgebras on closed terms, which we denote by

$$\mathbb{M} = \{f \mid f\colon T\emptyset \to FT\emptyset\}.$$

The order on F lifts to an order on $F \times \mathsf{Id}$ by defining $(b_1, x_1) \leq (b_2, x_2)$ iff $b_1 \leq b_2$ and $x_1 = x_2$ for $(b_1, x_1), (b_2, x_2) \in FX \times X$. Moreover, the order lifts component-wise to ΣFX (and also to $\Sigma(FX \times X)$) for any set X, by defining, for any $\sigma, \tau \in \Sigma$ of arity n and m respectively, $\sigma(k_1, \ldots, k_n) \leq \tau(l_1, \ldots, l_m)$ iff $\sigma = \tau$ (so also $n = m$) and $k_i \leq l_i$ for all $i \leq n$.

Definition 3. *Using the above lifting of the order to $\Sigma(F \times \mathsf{Id})$, a specification ρ is said to be* monotone *if all of its components are.*

Example 4. As stated in [8], for the functor $F = (\mathcal{P}-)^A$ of labelled transition systems, monotone specifications correspond to specifications in (an infinitary version of) the *positive GSOS* format.

Any abstract GSOS specification for a functor F induces a monotone GSOS specification for the discretely ordered functor $F + 2$ (see Example 2).

Assignment rules (6) can be formalised categorically in terms of natural transformations. These are independent of the behaviour functor F.

Definition 5. *An* assignment rule *is a natural transformation $d \colon \Sigma \Rightarrow T$.*

For example, the assignment rule for the replication operator is the natural transformation which sends $!x$ to $!x \mid x$ for any x, and is the identity on all other operators in Σ.

Assumption 6. *In the remainder of this paper we assume all our abstract GSOS specifications to be monotone. In particular we fix a monotone GSOS specification ρ and a set Δ of assignment rules.*

Now we have all the necessary tools to define a model on closed terms of an abstract GSOS specification together with a set of assignment rules.

Definition 7. *Let $\psi \colon \mathbb{M} \to \mathbb{M}$ be the (unique) function such that*

$$\psi(f) \circ \nu = F\mu \circ \rho \circ \Sigma\langle f, \mathsf{id}\rangle \vee \bigvee_{d \in \Delta} f \circ \mu \circ d\,.$$

A (ρ, Δ)-model is a coalgebra $f \in \mathbb{M}$ such that $\psi(f) = f$.

Notice that $\nu \colon \Sigma T\emptyset \to T\emptyset$ is an isomorphism, so ψ is uniquely defined. As argued above, in general there may be more than one model for a fixed ρ and Δ. We will be interested in the *least* supported model as the correct interpretation. In order to show that this exists we need the following:

Lemma 8. *$\psi \colon \mathbb{M} \to \mathbb{M}$ is monotone.*

By the Knaster-Tarski theorem, ψ has a least fixpoint (e.g., [25]).

Definition 9. *The* interpretation *of ρ and Δ is the least (ρ, Δ)-model.*

Example 10. For a GSOS specification on transition systems together with assignment rules, the interpretation is the least system where $\sigma(t_1, \ldots, t_n) \xrightarrow{a} t'$ iff it can be derived from a rule in the specification or there is an assignment of t to σ, and $t \xrightarrow{a} t'$. This is a recursive definition; being the least such transition system has the consequence that every derivation of a transition $t \xrightarrow{a} t'$ is finite.

4 Abstract GSOS Specifications for Assignment Rules

In the previous section we have seen how to interpret an abstract GSOS specification ρ together with a set of assignment rules Δ as a coalgebra on closed terms. In this section we will show that we can alternatively construct this coalgebra as the operational model of another specification (without assignment rules), which is constructed as a least fixpoint of a function on the complete lattice of specifications. The consequence of this alternative representation is well-behavedness properties, in particular bisimilarity being a congruence and the compatibility of bisimulation up to context, on the interpretation of ρ and Δ.

Let $\mathbb{S}$ be the set of all monotone abstract GSOS specifications. We turn $\mathbb{S}$ into a complete lattice by defining the order componentwise, i.e., for any $L \subseteq \mathbb{S}$ and any set X: $(\bigvee L)_X = \bigvee_{\rho \in L} \rho_X$. The join is well-defined:

Lemma 11. *For any $L \subseteq \mathbb{S}$: the family of functions $(\bigvee L)$ as defined above is a monotone specification.*

This provides a way of pointwise combining specifications.

Consider, for some assignment rule $d \in \Delta$ and specification τ, the following natural transformation:

$$\Sigma(F \times \mathsf{Id}) \xrightarrow{d} T(F \times \mathsf{Id}) \xrightarrow{\tau^*} FT \times T \xrightarrow{\pi_1} FT \tag{7}$$

Here, and in the sequel, we use τ^* to denote the inductive extension of τ to terms (e.g., [3]). Informally, the above natural transformation acts as follows. For an operator σ of arity n, given behaviour $k_1, \ldots, k_n \in FX \times X$ of its arguments, it first applies the assignment rule d to obtain a term $t(k_1, \ldots, k_n)$. Subsequently τ^* is used to compute the behaviour of t given the behaviour $k_1, \ldots, k_n$. In short, the above transformation computes the behaviour of an operator by using rules from τ and a single application of the rule d.

Example 12. Suppose ρ is the specification of CCS, without any rules for the replication operator $!x$. Moreover suppose d is the assignment rule associated to the replication. Then the natural transformation in (7) is a specification which has for the replication operator the rule (for any label a) $\frac{x \xrightarrow{a} x'}{!x \xrightarrow{a} !x \mid x'}$ and is unchanged on all other operators. This rule is deduced from the behaviour of the parallel operator; since a process $!t$ can not make any transitions by the rules in ρ, after the construction in (7), $!t$ can make precisely the transitions that t can.

To obtain the correct specification of the replication operator we will need to apply such a construction recursively, which we will do below. First we define a function φ on $\mathbb{S}$ which uses the above construction to build, from an argument specification τ, the specification containing all rules from our fixed specification ρ and all rules which can be formed as in (7).

Definition 13. *Given our fixed ρ and Δ, the map $\varphi \colon \mathbb{S} \to \mathbb{S}$ is defined as*

$$\varphi(\tau) = \rho \vee \bigvee_{d \in \Delta} (\pi_1 \circ \tau^* \circ d)\,.$$

For well-definedness, we need to check that φ preserves monotonicity.

Lemma 14. *The function $\varphi\colon \mathbb{S} \to \mathbb{S}$ is monotone. Moreover, if τ is a monotone specification, then $\varphi(\tau)$ is monotone as well.*

As a consequence of φ being monotone, it has a least fixpoint, which we denote by $\mathsf{lfp}\,\varphi$. Moreover, since φ preserves monotonicity we obtain monotonicity of $\mathsf{lfp}\,\varphi$ by transfinite induction (the base case and limit steps are rather easy). This proof technique, which we also use several times below, is justified by the fact that the least fixpoint of a monotone function in a complete lattice can be constructed as the supremum of an ascending chain obtained by iterating the function over the ordinals (see, e.g., [25]).

Corollary 15. *$\mathsf{lfp}\,\varphi$ is monotone.*

Informally, $\mathsf{lfp}\,\varphi$ is the specification consisting of rules from ρ and Δ. By

$$M\colon \mathbb{S} \to \mathbb{M}$$

we denote the function which assigns to a specification its unique operational model (5) (Section 2). We proceed to prove that the operational model of the least fixpoint of φ is precisely the interpretation of ρ and Δ, i.e., that $M(\mathsf{lfp}\,\varphi) = \mathsf{lfp}\,\psi$. First, we show that $M(\mathsf{lfp}\,\varphi)$ is a fixpoint of ψ.

Lemma 16. *$M(\mathsf{lfp}\,\varphi)$ is a (ρ, Δ)-model.*

We proceed to show that $M(\mathsf{lfp}\,\varphi) \leq \mathsf{lfp}\,\psi$. The main step is that any fixpoint of ψ is "closed under ρ", i.e., that in such a model, each transition which we can derive by the specification is already there. This result is the contents of Lemma 17 below. For the proof, one shows that $F\mu \circ h \circ \Sigma\langle f, \mathsf{id}\rangle \leq f \circ \nu$ holds for any approximation h of $\mathsf{lfp}\,\varphi$, by transfinite induction.

Lemma 17. *Let $f \in \mathbb{M}$ be a fixpoint of ψ. Then $F\mu \circ \mathsf{lfp}\,\varphi \circ \Sigma\langle f, \mathsf{id}\rangle \leq f \circ \nu$.*

This allows to prove our main result.

Theorem 18. *$M(\mathsf{lfp}\,\varphi) = \mathsf{lfp}\,\psi$, i.e., the interpretation of ρ and Δ coincides with the operational model of the specification $\mathsf{lfp}\,\varphi$.*

Proof. By Lemma 16, $M(\mathsf{lfp}\,\varphi)$ is a fixpoint of ψ. To show it is the least one, let f be any fixpoint of ψ; we proceed to prove $M(\mathsf{lfp}\,\varphi) \leq f$ by structural induction on closed terms. Suppose $\sigma \in \Sigma$ is an operator of arity n, and suppose we have $t_1, \ldots, t_n \in T\emptyset$ such that $M(\mathsf{lfp}\,\varphi)(t_i) \leq f(t_i)$ for all i with $1 \leq i \leq n$ (note that this trivially holds in the base case, when $n = 0$). Then $M(\mathsf{lfp}\,\varphi)(\sigma(t_1, \ldots, t_n)) = F\mu \circ \mathsf{lfp}\,\varphi \circ \Sigma\langle M(\mathsf{lfp}\,\varphi), \mathsf{id}\rangle(\sigma(t_1, \ldots, t_n)) \leq F\mu \circ \mathsf{lfp}\,\varphi \circ \Sigma\langle f, \mathsf{id}\rangle(\sigma(t_1, \ldots, t_n)) \leq f(\sigma(t_1, \ldots, t_n))$ where the first inequality holds by assumption and monotonicity of $F\mu$ and $\mathsf{lfp}\,\varphi$ (Corollary 15) and the second by Lemma 17. □

As a consequence, the interpretation of ρ and Δ is well-behaved:

Corollary 19. *Bisimilarity is a congruence on the interpretation of ρ and Δ, and bisimulation up to context is compatible.*

Example 20. The operators of CCS can be given by a positive GSOS specification, and equation (1) of the introduction contains a rule for the replication operator. Thus, by the above Corollary, bisimilarity is a congruence on the operational model of CCS with replication, and bisimulation up to context is compatible; this is proved and used in [25], but here we obtain it directly from the format and the above results.

5 Structural Congruences (as Assignment Rules)

The assignment rules considered in the theory of the previous sections copy behaviour from a term to an operator, but this assignment goes one way only. In this section we consider the combination of abstract GSOS specifications with actual *equations*, which are elements of $TV \times TV$ (where V is an arbitrary set of variables). Any set of equations $E \subseteq TV \times TV$ induces a *congruence* $\equiv_E$:

Definition 21. *Let $E \subseteq TV \times TV$ be a set of equations. The* congruence closure *$\equiv_E$ of E is the least relation $\equiv \subseteq T\emptyset \times T\emptyset$ satisfying the following rules:*

$$\frac{t\,E\,u \quad s\colon V \to T\emptyset}{s^\sharp(t) \equiv s^\sharp(u)} \qquad \frac{}{t \equiv t} \qquad \frac{u \equiv t}{t \equiv u} \qquad \frac{t \equiv u \qquad u \equiv v}{t \equiv v}$$

$$\frac{t_1 \equiv u_1 \qquad \dots \qquad t_n \equiv u_n}{\sigma(t_1,\dots,t_n) \equiv \sigma(u_1,\dots,u_n)} \qquad \text{for each } \sigma \in \Sigma, n = |\sigma|$$

where $s^\sharp$ is the extension of s to terms (defined by substitution).

In the context of structural operational semantics, equations are often interpreted by the *structural congruence rule*:

$$\frac{t \equiv_E u \qquad u \xrightarrow{a} u' \qquad u' \equiv_E v}{t \xrightarrow{a} v} \tag{8}$$

Informally, this rule states that we can deduce transitions modulo the congruence generated by the equations. In fact, removing the part $u' \equiv_E v$ from the premise (and writing u' instead of v in the conclusion) does not affect the behaviour, modulo bisimilarity [17]. See [17] for details on the interpretation of structural congruences in the context of transition systems.

We denote by $(T\emptyset)/{\equiv_E}$ the set of equivalence classes, and by $q\colon T\emptyset \to (T\emptyset)/{\equiv_E}$ the quotient map of $\equiv_E$. Thus $q(t) = q(u)$ iff $t \equiv_E u$. Assuming the axiom of choice, we further have $t \equiv_E u$ iff there is a right inverse $r\colon (T\emptyset)/{\equiv_E} \to T\emptyset$ of q such that $q \circ r(t) = u$. This is exploited in the operational interpretation of a specification together with a set of equations.

Definition 22. *Let $\theta\colon \mathbb{M} \to \mathbb{M}$ be the (unique) function such that*

$$\theta(f) \circ \nu = F\mu \circ \rho \circ \Sigma\langle f, \mathsf{id}\rangle \vee \bigvee_{r \in R} f \circ r \circ q \circ \nu\,.$$

where R is the set of right inverses of q. A (ρ, E)-model is a coalgebra $f \in \mathbb{M}$ such that $\theta(f) = f$.

Lemma 23. *θ is monotone.*

Definition 24. *The* interpretation *of ρ and E is the least (ρ, E)-model.*

Example 25. Consider the specification of the parallel operator $x \mid y$ as given in (2) in the introduction, i.e., by a single rule and commutativity. In the interpretation, if $t \xrightarrow{a} t'$ then $t \mid u \xrightarrow{a} t' \mid u$ simply by the SOS rule. But also $u \mid t \xrightarrow{a} t' \mid u$, since $t \mid u \equiv_E u \mid t$. As for the definition of the replication operator by the equation $!x = !x \mid x$, for a term t the interpretation contains the least set of transitions from $!t$ which satisfy the equation, as desired.

On stream systems, abstract GSOS specifications correspond to behavioural differential equations, which are guarded, that is, for each operator (or constant) one defines its initial value concretely. For example, one can define $\mathsf{zip}(x, y) = o(x) : (y, x')$, where $o(x)$ denotes the initial value of x and x' its derivative (its tail); and, e.g., $\mathsf{zeros} = 0 : \mathsf{zeros}$ and $\mathsf{ones} = 1 : \mathsf{ones}$ define the streams consisting only of zeros and ones, respectively. Taking the discrete order on the functor (Example 2) we can now add *equations* to such specifications. For instance, the paper folding sequence can be defined by the equations[2] $\mathsf{h} = \mathsf{zip}(\mathsf{ones}, \mathsf{zeros})$ and $\mathsf{pf} = \mathsf{zip}(\mathsf{h}, \mathsf{pf})$. In the interpretation, pf then defines the paper folding sequence.

Unfortunately, bisimilarity is not a congruence when equations are added [17]. For convenience we recall the counterexample on transition systems.

Example 26 ([17]). Consider rules $\dfrac{}{p \xrightarrow{a} p}$ and $\dfrac{}{q \xrightarrow{a} p}$ and the single equation $p = \sigma(q)$, where p, q are constants, σ is a unary operator and a is an arbitrary label. In the interpretation, p is bisimilar to q, but $\sigma(p)$ is not bisimilar to $\sigma(q)$.

The solution of [17] is to introduce a restricted format of equations, called cfsc. It is then shown that for any tyft specification combined with cfsc equations, bisimilarity is a congruence.

Definition 27. *A set of equations $E \subseteq TV \times TV$ is in cfsc format with respect to ρ if every equation is of one of the following forms:*

1. *A σx-*equation*: $\sigma_1(x_1, \ldots, x_n) = \sigma_2(y_1, \ldots, y_n)$, where $\sigma_1, \sigma_2 \in \Sigma$ are of arity n (possibly $\sigma_1 = \sigma_2$), $x_1, \ldots, x_n$ are distinct variables and $y_1, \ldots, y_n$ is a permutation of $x_1, \ldots, x_n$.*
2. *A* defining equation*: $\sigma(x_1, \ldots, x_n) = t$ where $\sigma \in \Sigma$ and t is an arbitrary term (which may involve σ again); $x_1, \ldots, x_n$ are distinct variables, and all variables that occur in t are in $x_1, \ldots, x_n$. Moreover σ does not appear in any other equation in E, and $\rho_X(\sigma(u_1, \ldots, u_n)) = \bot$ for any set X and any $u_1, \ldots, u_n \in FX \times X$.*

[2] This example was taken from a presentation by Jörg Endrullis.

A σx-*equation* allows to assign simple algebraic properties to operators which already have behaviour; the prototypical example here is commutativity, like in the specification of the parallel operator in (2). With a *defining equation*, one can define the behaviour of an operator. Examples are $!x = !x \mid x$ and $\mathsf{pf} = \mathsf{zip}(\mathsf{h}, \mathsf{pf})$ (pf and h are constants). Associativity of $\mid$ is neither a σx-equation nor a defining one; see [17] for a discussion why the cfsc format cannot be trivially extended. The cfsc format depends on an abstract GSOS specification: operators at the left hand side of a defining equation should not get any behaviour in the specification.

In [17], σx-equations are a bit more liberal in that they do not require the arities of σ and σ' to coincide, and do allow variables which only occur on one side of the equation. But in the interpretation these variables are quantified universally over closed terms; thus, we can encode this using infinitely many equations. We work with the simpler format above for technical convenience.

We proceed to show that the interpretation of an abstract GSOS specification ρ and a set of equations E in cfsc equals the operational model of a certain other specification, up to bisimilarity. This is done by encoding equations in this format as assignment rules, and using the theory of the previous section to obtain the desired result.

First, notice that for any σx-equation $\sigma_1(x_1, \ldots, x_n) = \sigma_2(y_1, \ldots, y_n)$, the variables on one side are a permutation of the variables on the other, and thus it can equivalently be represented as a triple (σ_1, σ_2, p) where $p\colon \mathsf{Id}^n \to \mathsf{Id}^n$ is the natural transformation corresponding to the permutation given by the equation. Below, we use $t[x_1, \ldots, x_n := t_1, \ldots, t_n]$ to denote the simultaneous substitution of variables $x_1, \ldots, x_n$ by terms $t_1, \ldots, t_n$ in a term t.

Definition 28. *A set of equations E in cfsc defines a set of assignment rules Δ^E as follows:*

1. *For every σx-equation (σ_1, σ_2, p) we define d and d' on a component X as*

$$d_X(\sigma(u_1, \ldots, u_n)) = \begin{cases} \sigma_2(p_X(u_1, \ldots, u_n)) & \text{if } \sigma = \sigma_1 \\ \sigma(u_1, \ldots, u_n) & \text{otherwise} \end{cases}$$

   for all $u_1, \ldots, u_n \in X$, and d' is similarly defined using the inverse permutation p^{-1}, and σ_1 and σ_2 swapped.
2. *For every defining equation $\sigma_1(x_1, \ldots, x_n) = t$ we define a corresponding assignment rule* $d_X(\sigma(u_1, \ldots, u_n)) = \begin{cases} t[x_1, \ldots, x_n := u_1, \ldots, u_n] & \text{if } \sigma = \sigma_1 \\ \sigma(u_1, \ldots, u_n) & \text{otherwise} \end{cases}$
   for any set X and all $u_1, \ldots, u_n \in X$.

If $\sigma(x_1, \ldots, x_n) = t$ is a defining equation of a set of equations in the cfsc format, then the behaviour of $\sigma(x_1, \ldots, x_n)$ will be the same as that of t.

Lemma 29. *Let E be a set of equations in cfsc format w.r.t. ρ, and let ψ be the function of Definition 7 for ρ and Δ^E. Then for any defining equation $\sigma(x_1, \ldots, x_n) = t$ and any terms $t_1, \ldots, t_n \in T\emptyset$: $(\mathsf{lfp}\, \psi)(\sigma(t_1, \ldots, t_n)) = (\mathsf{lfp}\, \psi)(t[x_1, \ldots, x_n := t_1, \ldots, t_n])$.*

The following lemma is the main step towards the correctness of the encoding.

Lemma 30. *Let E and ψ be as above. If $t \equiv_E u$ then $Fq \circ (\mathsf{lfp}\,\psi)(t) = Fq \circ (\mathsf{lfp}\,\psi)(u)$, where q is the quotient map of $\equiv_E$.*

This allows to prove that $\mathsf{lfp}\,\psi$ and $\mathsf{lfp}\,\theta$ coincide "up to $\equiv_E$".

Lemma 31. *Let ψ and q be as above. Then $Fq \circ (\mathsf{lfp}\,\theta) = Fq \circ (\mathsf{lfp}\,\psi)$.*

This implies that $\mathsf{lfp}\,\theta$ and $\mathsf{lfp}\,\psi$ are *behaviourally equivalent* up to $\equiv_E$. It is well-known that behavioural equivalence coincides with bisimilarity whenever the functor F preserves weak pullbacks [22], a mild condition satisfied by most functors used in practice, including, e.g., transition systems and stream systems. Under this assumption one can prove that $\mathsf{lfp}\,\theta$ is equal to $\mathsf{lfp}\,\psi$ up to bisimilarity, and by Theorem 18 we then obtain our main result of this section.

Theorem 32. *Suppose E is a set of equations which is in* cfsc *format w.r.t. ρ, and suppose the behaviour functor F preserves weak pullbacks. Then the interpretation of ρ and E equals the operational model of some abstract GSOS specification, up to bisimilarity. Bisimilarity is a congruence, and bisimulation up to context and bisimilarity is compatible.*

6 Related Work

The main work on structural congruences [17] focuses on labelled transition systems, whereas our results apply to the more general notion of coalgebras. As for transition systems, the basic rule format in [17] is tyft/tyxt[3], which is strictly more general than positive GSOS since it allows lookahead. However, while [17] proves congruence of bisimilarity this does not imply the compatibility (or even soundness) of bisimulation up to context [19], which we obtain in the present work (and is in fact problematic in the presence of lookahead).

In the bialgebraic setting, Klin [10] showed that by moving to CPPO-enriched categories, one can interpret recursive constructs which have a similar form as our assignment rules. Technically our approach, based on ordered functors, is different; it allows us to stay in the familiar category of sets and apply the coalgebraic bisimulation-up-to techniques of [20], which are based in this category. Further, in [10] each operator is either specified by an equation or by operational rules, disallowing a specification such as that of the parallel operator in equation (2). Plotkin proposed to move to CPPO in [18] to model recursion. The recent [15] applies the theory of [10] to add silent transitions to their concept of open GSOS laws. This is then used to show that equations are preserved by conservative extensions.

In [14] various constructions on distributive laws are presented. Their Example 32 discusses the definition of the parallel operator as in (2) above, but

[3] In [17] it is sketched how to extend the results to the ntyft/ntyxt, which involves however a complicated integration of the cfsc format with the notion of stable model.

a general theory for structural congruence is missing. In [5] it is shown how to obtain a distributive law for a monad that is the quotient of another one by imposing extra equations, under the condition that the distributive law respects the equations. However, this condition requires that the equations already hold semantically, which is fundamentally different from the present paper where we define behaviour by imposing equations on an operational specification. Similarly in [6,7] it is shown how to lift calculi with structural axioms to coalgebraic models, but under the assumption, again, that the equations already hold.

7 Conclusions

We extended Turi and Plotkin's bialgebraic approach to operational semantics with non-structural assignment rules and structural congruence, providing a general coalgebraic framework for monotone abstract GSOS with equations. Our main result is that the interpretation of a specification involving assignment rules is well-behaved, in the sense that bisimilarity is a congruence and bisimulation-up-to techniques are sound. This result carries over to specifications with structural congruence in the cfsc format proposed in [17].

There are several promising directions for future work. First, one could extend our techniques to allow lookahead in premises by using cofree comonads (e.g., [12]). While in general the combined use of cofree comonads and free monads in specifications is known to be problematic, we expect that these problems do not arise when considering positive (monotone) specifications. In fact, this could form the basis for a bialgebraic account of the tyft format. Unfortunately, the compatibility results of bisimulation-up-to do not hold in a setting with lookahead. Second, in the current work we only consider free monads. One can possibly incorporate equations which already hold, by using the theory of [5]. Finally, it is worthwhile to consider categorical generalisations to allow, e.g., to study structural congruences for calculi with names. It could also lead to congruence formats for notions of equivalence other than bisimilarity.

Acknowledgments. We would like to thank Daniel Gebler, Bartek Klin, Mohammad Reza Mousavi and Jan Rutten for helpful suggestions and discussions. Our research has been funded by the Netherlands Organisation for Scientific Research (NWO), CoRE project, dossier number: 612.063.920.

References

1. L. Aceto, W. Fokkink, and C. Verhoef. Structural operational semantics. In *Handbook of Process Algebra*, pages 197–292. Elsevier Science, 2001.
2. J. C. M. Baeten and C. Verhoef. A congruence theorem for structured operational semantics with predicates. In E. Best, editor, *CONCUR*, volume 715 of *LNCS*, pages 477–492. Springer, 1993.
3. F. Bartels. *On generalised coinduction and probabilistic specification formats*. PhD thesis, CWI, Amsterdam, April 2004.

4. B. Bloom, S. Istrail, and A. Meyer. Bisimulation can't be traced. *J. ACM*, 42(1):232–268, 1995.
5. M. M. Bonsangue, H. H. Hansen, A. Kurz, and J. Rot. Presenting distributive laws. In R. Heckel and S. Milius, editors, *CALCO*, volume 8089 of *LNCS*, pages 95–109. Springer, 2013.
6. M. G. Buscemi and U. Montanari. A first order coalgebraic model of pi-calculus early observational equivalence. In L. Brim et. al., editor, *CONCUR*, volume 2421 of *LNCS*, pages 449–465. Springer, 2002.
7. A. Corradini, R. Heckel, and U. Montanari. Compositional SOS and beyond: a coalgebraic view of open systems. *Theor. Comput. Sci.*, 280(1-2):163–192, 2002.
8. M. Fiore and S. Staton. Positive structural operational semantics and monotone distributive laws. In *CMCS Short Contributions*, page 8, 2010.
9. J. Hughes and B. Jacobs. Simulations in coalgebra. *Theor. Comput. Sci.*, 327(1-2):71–108, 2004.
10. B. Klin. Adding recursive constructs to bialgebraic semantics. *JLAP*, 60-61:259–286, 2004.
11. B. Klin. Structural operational semantics for weighted transition systems. In J. Palsberg, editor, *Semantics and Algebraic Specification*, volume 5700 of *LNCS*, pages 121–139. Springer, 2009.
12. B. Klin. Bialgebras for structural operational semantics: An introduction. *TCS*, 412(38):5043–5069, 2011.
13. C. Kupke, M. Niqui, and J. Rutten. Stream differential equations: concrete formats for coinductive definitions. Tech. Report No. RR-11-10, Oxford University, 2011.
14. M. Lenisa, J. Power, and H. Watanabe. Category theory for operational semantics. *Theor. Comput. Sci.*, 327(1-2):135–154, 2004.
15. K. Madlener, S. Smetsers, and M. C. J. D. van Eekelen. Modular bialgebraic semantics and algebraic laws. In A. R. Du Bois and P. Trinder, editors, *SBLP*, volume 8129 of *LNCS*, pages 46–60. Springer, 2013.
16. R. Milner. Functions as processes. *MSCS*, 2(2):119–141, 1992.
17. M. R. Mousavi and M. A. Reniers. Congruence for structural congruences. In V. Sassone, editor, *FoSSaCS*, volume 3441 of *LNCS*, pages 47–62. Springer, 2005.
18. G. D. Plotkin. Bialgebraic semantics and recursion (extended abstract). *Electr. Notes Theor. Comput. Sci.*, 44(1):285–288, 2001.
19. D. Pous and D. Sangiorgi. Enhancements of the bisimulation proof method. In *Advanced Topics in Bisimulation and Coinduction*, pages 233–289. Cambridge University Press, 2012.
20. J. Rot, F. Bonchi, M.M. Bonsangue, D. Pous, J.J.M.M. Rutten, and A. Silva. Enhanced coalgebraic bisimulation. `http://www.liacs.nl/~jrot/up-to.pdf`.
21. J. Rot, M. Bonsangue, and J. Rutten. Coalgebraic bisimulation-up-to. In P. van Emde Boas, F. Groen, G. Italiano, J. Nawrocki, and H. Sack, editors, *SOFSEM*, volume 7741 of *LNCS*, pages 369–381. Springer, 2013.
22. J. Rutten. Universal coalgebra: a theory of systems. *TCS*, 249(1):3–80, 2000.
23. J. Rutten. Behavioural differential equations: a coinductive calculus of streams, automata, and power series. *TCS*, 308(1-3):1–53, 2003.
24. D. Sangiorgi. On the bisimulation proof method. *MSCS*, 8(5):447–479, 1998.
25. D. Sangiorgi. *An introduction to Bisimulation and Coinduction*. Cambridge University Press, 2012.
26. D. Sangiorgi and D. Walker. *The Pi-Calculus - a theory of mobile processes*. Cambridge University Press, 2001.
27. D. Turi and G. Plotkin. Towards a mathematical operational semantics. In *LICS*, pages 280–291. IEEE Computer Society, 1997.