Presenting Morphisms of Distributive Laws*

Bartek Klin[1] and Beata Nachyła[2]

1 **University of Warsaw, Poland**
`klin@mimuw.edu.pl`
2 **Institute of Computer Science, Polish Academy of Sciences, Poland**
`beatanachyla@gmail.com`

Abstract

A format for well-behaved translations between structural operational specifications is derived from a notion of distributive law morphism, previously studied by Power and Watanabe.

1998 ACM Subject Classification F.3.2 Semantics of Programming Languages – Operational semantics

Keywords and phrases coalgebra, bialgebra, distributive law, structural operational semantics

Digital Object Identifier 10.4230/LIPIcs.CALCO.2015.190

1 Introduction

Since [13], distributive laws of functors (or pointed functors, or monads) over other functors (or copointed functors, or comonads), and bialgebras for them, have been a useful categorical tool to study various kinds of structural operational semantics (SOS, [10, 1]). Several formats of well-behaved operational specifications can be understood as kinds of distributive laws, and some desired properties (mostly compositionality of behavioural equivalences) can be proved in terms of bialgebras. See [7] for a recent introduction to this topic.

One advantage of abstraction is that sometimes notions or results readily available at the abstract level, can be instantiated in the concrete setting in previously unforeseen ways. One such example are morphisms of distributive laws, studied by Power and Watanabe [11, 14] as abstract notions of well-behaved translations between operational specifications.

The issue of translating specifications have attracted limited attention in the SOS community so far; apart from a general notion of conservative extension [1], which can be seen as a simple embedding of one specification into another, only isolated examples of well-behaved translations have been studied [5], with no attempt at a general theory. This is unfortunate, as translating operational semantics from one language to another, and from one type of behaviour to another, is very useful in modular SOS development. When different parts of a language, or different aspects of its semantics, are specified separately, they must be combined somehow, for example via a family of translations.

In [11], distributive law morphisms were studied in the abstract, with a few concrete examples provided in [14] (see also [4] for a slightly different application). In this paper we pick up that line of work and provide a characterization of morphisms between GSOS specifications [3] understood as distributive laws. A morphism between laws is presented as a syntactic translation, together with a behavioural translation presented by inference rules similar to SOS, subject to a compatibility condition. Importantly, for finite specifications the condition is decidable. Decidability is a key property of any reasonable format for

* This work was supported by the Polish National Science Centre (NCN) grant 2012/07/E/ST6/03026.

6th International Conference on Algebra and Coalgebra in Computer Science (CALCO'15).
Editors: Lawrence S. Moss and Pawel Sobocinski; pp. 190–204

Leibniz International Proceedings in Informatics
Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

specifications or their translations, and it is the reason why we choose to work with GSOS rather than with more general, and mathematically more pleasing, distributive laws of monads over comonads [13, 9]. The latter do not admit a decidable presentation (see [8]).

This paper is only a small step in what should become a long-term research programme. We only study morphisms between GSOS specifications of labelled transition systems (LTSs), whereas the abstract framework of distributive laws and their morphisms covers different types of behaviour and even different underlying categories. A question of translating operational descriptions up to various behavioural equivalences, using e.g. ideas from [6], is not touched upon. A framework for defining diagrams of distributive law morphisms and combining operational descriptions by means of (co)limits of such diagrams, and many other related tasks, are left for future work. However, we believe that even a simple characterization of GSOS specification morphisms sheds some light on the strengths and limitations of the bialgebraic approach to SOS translation.

We begin the paper by studying a special case of GSOS, so called simple SOS specifications, recalled from [7] in Section 3. We work out this special case in detail: in Section 4 we define morphisms of simple distributive laws, a special case of definitions studied in [11, 14]. In Section 5, syntactic and behavioural translations are defined, and in Section 6 a compatibility condition is shown that is equivalent to the translations forming a morphism of distributive laws. In Section 7, the case of full GSOS is sketched briefly: while technically slightly more complicated, it does not introduce any new conceptual difficulties. Section 8 contains some illustrative examples.

2 Preliminaries

A *signature* is a set Σ of symbols, where each symbol $\mathtt{f} \in \Sigma$ has an associated finite *arity* $\sharp\mathtt{f} \in \mathbb{N}$. With any signature we associate an endofunctor $\Sigma X = \coprod_{\mathtt{f}\in\Sigma} X^{\sharp\mathtt{f}}$ on the category **Set** of sets and functions. We shall only consider endofunctors Σ that arise from signatures in this way. Algebras for the functor (in the categorical sense) are then exactly algebras for the signature (in the universal-algebraic sense). For any set X, the set $\Sigma^* X$ of Σ-terms with variables from X carries an obvious Σ-algebra structure; $\Sigma^* 0$, for 0 the empty set, is an initial Σ-algebra. The construction Σ^* is a monad on **Set**; it is the free monad over Σ.

All coalgebras [12] considered in this paper are for **Set**-functors of the form $BX = (\mathcal{P}_\omega X)^L$, where $\mathcal{P}_\omega$ is the finite powerset functor and L is a finite set of *labels*. As is well known, such coalgebras are finitely branching L-labeled transition systems (LTSs). B-coalgebra morphisms are functional bisimulations, i.e., functions whose graphs are bisimulation relations.

3 Simple distributive laws and SOS

A *simple distributive law* of an endofunctor Σ over an endofunctor B is a natural transformation $\lambda : \Sigma B \Longrightarrow B\Sigma$. A *$\lambda$-bialgebra* is then a Σ-algebra $g : \Sigma X \to X$ and a B-coalgebra $k : X \to BX$ with the same carrier X, such that the following diagram commutes:

$$\begin{array}{ccccc} \Sigma X & \xrightarrow{g} & X & \xrightarrow{k} & BX \\ {\scriptstyle \Sigma k}\downarrow & & & & \uparrow{\scriptstyle Bg} \\ \Sigma BX & & \xrightarrow[\lambda_X]{} & & B\Sigma X. \end{array}$$

Morphisms of bialgebras are functions between their carriers that are simultaneously algebra and coalgebra morphisms between the respective bialgebra components. Bialgebras for a

fixed distributive law form a category. Under our assumptions an initial λ-bialgebra always exists, and is of the form:

$$\Sigma\Sigma^*0 \xrightarrow{\cong} \Sigma^*0 \xrightarrow{k_\lambda} B\Sigma^*0\ , \tag{1}$$

where the algebraic component is an initial Σ-algebra, hence (by Lambek's Lemma) an isomorphism. The coalgebraic component k_λ of an initial λ-bialgebra is called the B-coalgebra induced by λ. For a detailed introduction to these notions see e.g. [7].

If Σ is a polynomial functor arising from a signature, and if $BX = (\mathcal{P}_\omega X)^L$ for some finite set L of labels, simple distributive laws may be presented by sets of inference rules. This has been known since [13] for the more general case of GSOS laws (see Section 7.1), and studied in detail in [2, 7]. We now briefly recall the main idea.

For any set X, an *X-literal* (or simply a *literal*, when no risk of confusion arises) is an expression $x \xrightarrow{a} y$ where $x, y \in X$ and $a \in L$. In such a literal, x is called the *source*, a the *label* and y the *target*.

Fix a countably infinite set of variables $\mathcal{V} \ni \mathtt{x}, \mathtt{y}, \mathtt{z}, \ldots$.

▶ **Definition 1.** A *simple SOS specification* (over Σ and L) is a finite set of *simple SOS rules*, i.e., expressions of the form

$$\frac{\left\{\mathtt{x}_{i_j} \xrightarrow{a_j} \mathtt{y}_j\right\}_{j=1..m} \quad \left\{\mathtt{x}_{i_k} \not\xrightarrow{b_k} \right\}_{k=1..l}}{\mathtt{f}(\mathtt{x}_1, \ldots, \mathtt{x}_{\sharp\mathtt{f}}) \xrightarrow{c} \mathtt{g}(\mathtt{z}_1, \ldots, \mathtt{z}_{\sharp\mathtt{g}})} \tag{2}$$

where $\mathtt{f}, \mathtt{g} \in \Sigma$ and $m, l \in \mathbb{N}$; all $i_j, i_k \in \{1, \ldots, \sharp\mathtt{f}\}$; all $\mathtt{x}_i, \mathtt{y}_j \in \mathcal{V}$ are pairwise distinct; $\mathtt{z}_1, \ldots, \mathtt{z}_{\sharp\mathtt{g}} \in \{\mathtt{y}_1, \ldots, \mathtt{y}_m\}$; and $a_j, b_k, c \in L$.

The $\mathcal{V}$-literals $\mathtt{x}_{i_j} \xrightarrow{a_j} \mathtt{y}_j$ above are called *positive premises*, and the expressions $\mathtt{x}_{i_k} \not\xrightarrow{b_k}$ *negative premises* of the rule. The $\Sigma\mathcal{V}$-literal below the inference line is called the *conclusion*. The *source* and *target* of the rule are, respectively, the source and the target of its conclusion. Note that variables from the source of a simple SOS rule do not appear in its target.

For any set X, we say that a rule R as in (2) is *triggered* by a set Φ of X-literals if there is a substitution $\sigma : \mathcal{V} \to X$ such that:

- for each positive premise $\mathtt{x}_i \xrightarrow{a} \mathtt{y}$ in R, the literal $\sigma(\mathtt{x}_i) \xrightarrow{a} \sigma(\mathtt{y})$ is in Φ,
- for each negative premise $\mathtt{x}_i \not\xrightarrow{b}$ in R, there is no literal of the form $\sigma(\mathtt{x}_i) \xrightarrow{b} y$ in Φ.

If that is the case, the rule R *infers* from Φ the ΣX-literal

$$\mathtt{f}(\sigma(\mathtt{x}_1), \ldots, \sigma(\mathtt{x}_{\sharp\mathtt{f}})) \xrightarrow{c} \mathtt{g}(\sigma(\mathtt{z}_1), \ldots, \sigma(\mathtt{z}_{\sharp\mathtt{g}})).$$

Note that a single rule may infer more than one ΣX-literal from the same set of X-literals, depending on the substitution σ used. Note also that the literal inferred from a rule, if any, depends only on how σ acts on the variables present in the rule. Moreover, the target of the literal and its transition label depend only on how σ acts on the variables $\mathtt{y}_j$.

By $\Lambda[\Phi]$ we will denote the set of literals inferred from Φ by rules from a specification Λ. Given a specification Λ, for any set X define a function $\lambda_X : \Sigma(\mathcal{P}_\omega X)^L \to (\mathcal{P}_\omega \Sigma X)^L$ by:

$$\lambda_X(\mathtt{f}(\gamma_1, \ldots, \gamma_{\sharp\mathtt{f}}))(c) = \left\{t \in \Sigma X \mid (\mathtt{f}(x_1, \ldots, x_{\sharp\mathtt{f}}) \xrightarrow{c} t) \in \Lambda[\Phi]\right\}, \text{ where} \tag{3}$$
$$\Phi = \{x_i \xrightarrow{a} y \mid 1 \le i \le \sharp\mathtt{f},\ a \in L,\ y \in \gamma_i(a)\}$$

for any $\mathtt{f} \in \Sigma$, a family of functions $\gamma_1, \ldots, \gamma_{\sharp f} : L \to \mathcal{P}_\omega X$, a $c \in L$, and a family of distinct $x_1, \ldots, x_{\sharp\mathtt{f}}$. It is not difficult to see that for a given $\mathtt{f}(\gamma_1, \ldots, \gamma_{\sharp\mathtt{f}})$, the value of λ_X does not depend on the choice of $x_1, \ldots, x_{\sharp\mathtt{f}}$, as long as they are distinct; this is thanks to the fact that variables from the source of a simple SOS rule do not appear in its target. Note that one could even choose some $x_i = x_j$ as long as $\gamma_i = \gamma_j$, without affecting the value of λ_X.

▶ **Example 2.** Consider a signature $\Sigma = \{\mathtt{f}, \mathtt{g}\}$ with $\sharp\mathtt{f} = 2$, $\sharp\mathtt{g} = 1$, giving rise to a functor $\Sigma X = X^2 + X$. Consider also a set of labels $L = \{a, b\}$, and let a specification Λ consist of the following three rules:

$$\frac{\mathtt{x} \xrightarrow{a} \mathtt{x}' \quad \mathtt{y} \xrightarrow{b} \mathtt{y}'}{\mathtt{f}(\mathtt{x},\mathtt{y}) \xrightarrow{a} \mathtt{f}(\mathtt{x}',\mathtt{y}')} \qquad \frac{\mathtt{x} \not\xrightarrow{b} \quad \mathtt{x} \xrightarrow{a} \mathtt{x}' \quad \mathtt{y} \xrightarrow{a} \mathtt{y}'}{\mathtt{f}(\mathtt{x},\mathtt{y}) \xrightarrow{a} \mathtt{g}(\mathtt{y}')} \qquad \frac{\mathtt{x} \xrightarrow{b} \mathtt{x}'}{\mathtt{g}(\mathtt{x}) \xrightarrow{b} \mathtt{f}(\mathtt{x}',\mathtt{x}')}$$

where $\mathtt{x}, \mathtt{y}, \mathtt{y}', \mathtt{y}' \in \mathcal{V}$. For $X = \{u, v, w\}$, consider the set of X-literals:

$$\Phi = \{u \xrightarrow{a} v,\ u \xrightarrow{a} w,\ v \xrightarrow{a} u,\ v \xrightarrow{b} w\}.$$

Then all three rules are triggered, and the following literals are inferred:

$$\Lambda[\Phi] = \{\mathtt{f}(u,v) \xrightarrow{a} \mathtt{f}(v,w),\ \mathtt{f}(u,v) \xrightarrow{a} \mathtt{f}(w,w),\ \mathtt{f}(v,v) \xrightarrow{a} \mathtt{f}(u,w),\ \mathtt{f}(u,u) \xrightarrow{a} \mathtt{g}(v),$$
$$\mathtt{f}(u,u) \xrightarrow{a} \mathtt{g}(w),\ \mathtt{f}(u,v) \xrightarrow{a} \mathtt{g}(u),\ \mathtt{g}(v) \xrightarrow{b} \mathtt{f}(w,w)\}.$$

Keeping Λ and X as above, choose $\mathtt{f} \in \Sigma$ and $a \in L$, and consider $\gamma_1, \gamma_2 : L \to P_\omega X$ defined by: $\gamma_1(a) = \{v, w\}$, $\gamma_1(b) = \emptyset$, $\gamma_2(a) = \{u\}$, $\gamma_2(b) = \{w\}$. For any $x_1 \neq x_2$, according to (3) this gives rise to:

$$\Phi = \{x_1 \xrightarrow{a} v,\ x_1 \xrightarrow{a} w,\ x_2 \xrightarrow{a} u,\ x_2 \xrightarrow{b} w\}, \text{ and}$$
$$\lambda_X(\mathtt{f}(\gamma_1, \gamma_2))(a) = \{\mathtt{f}(v,w),\ \mathtt{f}(w,w),\ \mathtt{g}(u)\};$$

note how the latter does not depend on the choice of x_1 and x_2.

The following theorem is a special case of a more general result concerning GSOS specifications, formulated first in [13] and proved in detail in [2]; we omit the proof here.

▶ **Theorem 3.** *For any specification Λ, the functions λ_X defined by* (3) *form a natural transformation $\lambda : \Sigma(\mathcal{P}_\omega -)^L \Longrightarrow (\mathcal{P}_\omega \Sigma -)^L$. Moreover, every natural transformation of this type arises this way from some simple SOS specification.*

Rule-based presentation of distributive laws suggests a notion of derivation; this can be defined in standard terms of SOS theory, see e.g. [1]. A labelled transition system derived from a simple SOS specification Λ coincides with the coalgebra induced by the corresponding distributive law λ, i.e., with the coalgebraic component of the initial λ-bialgebra (1).

4 Distributive law morphisms

▶ **Definition 4.** A *distributive law morphism* from $\lambda : \Sigma B \Longrightarrow B\Sigma$ to $\lambda' : \Sigma' B' \Longrightarrow B'\Sigma'$ consists of natural transformations $\alpha : \Sigma \Longrightarrow \Sigma'$ and $\theta : B' \Longrightarrow B$ such that the following diagram commutes:

$$\begin{array}{ccccc} \Sigma B' & \xRightarrow{\Sigma\theta} & \Sigma B & \xRightarrow{\lambda} & B\Sigma \\ & {\scriptstyle \alpha B'} \searrow & & & \swarrow {\scriptstyle B\alpha} \\ & & \Sigma' B' & \xRightarrow[\lambda']{} & B'\Sigma' \xRightarrow[\theta\Sigma']{} B\Sigma'. \end{array} \tag{4}$$

In [11, 14] distributive law morphisms were defined in a more general framework where Σ, B and Σ', B' operate on different categories connected by an additional functor. Although important for SOS specifications of systems other than LTSs, here we work in a simplified setting to illustrate the basic issues of presenting morphisms on a classical example.

In [11, 14] another definition of distributive law morphism was also considered, with two natural transformations $\alpha : \Sigma \Longrightarrow \Sigma'$, $\theta : B \Longrightarrow B'$ going in the same direction. We defer the issue of presenting such morphisms to a full version of this paper; their presentations are technically quite similar to the ones presented here, although they carry slightly different intuitions.

Any distributive law morphism as in Definition 4 induces a functor from the category of λ'-bialgebras to the category of λ-bialgebras (see also [4]), mapping every

$$\Sigma' X \xrightarrow{g} X \xrightarrow{k} B'X \qquad \text{to} \qquad \Sigma X \xrightarrow{\alpha_X} \Sigma' X \xrightarrow{g} X \xrightarrow{k} B'X \xrightarrow{\theta_X} BX.$$

Consider this functor applied to the initial λ'-bialgebra, and the unique morphism α_0^* from the initial λ-bialgebra to the result of that application:

$$\begin{array}{ccccc} \Sigma\Sigma^*0 & \xrightarrow{\cong} & \Sigma^*0 & \xrightarrow{k_\lambda} & BX \\ \downarrow{\scriptstyle \Sigma\alpha_0^*} & & \downarrow{\scriptstyle \alpha_0^*} & & \downarrow{\scriptstyle B\alpha_0^*} \\ \Sigma\Sigma'^*0 & \xrightarrow{\alpha_{\Sigma'^*0}} \Sigma'\Sigma'^*0 & \xrightarrow{\cong} \Sigma'^*0 & \xrightarrow{k_{\lambda'}} B'\Sigma'^*0 & \xrightarrow{\theta_{\Sigma'^*0}} B\Sigma'^*0 \end{array}$$

As an algebra morphism from an initial Σ-algebra, α_0^* is an inductively defined translation of Σ-terms to Σ'-terms according to α (hence the name). As a bialgebra morphism, it is also a B-coalgebra morphism.

For $BX = (\mathcal{P}_\omega X)^L$, coalgebra morphisms are functional bisimulations. As a result, if α and θ form a distributive law morphism from λ to λ' then the translation α_0^* of Σ-terms to Σ'-terms according to α, maps a term in the transition system k_λ induced by λ, to a bisimilar term in the system $k_{\lambda'}$ induced by λ', with the behaviour translated according to θ. A useful intuition to hold is that B' is somehow richer than B, and θ projects B'-behaviours to B-behaviours by ignoring some components. In the following sections we will provide examples of both α and θ that will illustrate these intuitions.

5 Syntactic and behavioural translations

To characterize morphisms of distributive laws in terms of rules, premises, literals etc., we first provide straightforward complete characterizations of natural transformations α and θ introduced in the previous section.

5.1 Syntactic translations

▶ **Definition 5.** For signatures Σ, Σ', a *syntactic translation* from Σ to Σ' consists of:

- a function $\alpha : \Sigma \to \Sigma'$ between the underlying sets of function symbols,
- for each $\mathtt{f} \in \Sigma$, a function $\alpha_{\mathtt{f}} : \{1, \ldots, \sharp\alpha(\mathtt{f})\} \to \{1, \ldots, \sharp\mathtt{f}\}$.

For any set X, a syntactic translation α determines a function $\alpha_X : \Sigma X \to \Sigma' X$ by:

$$\alpha_X(\mathtt{f}(x_1, \ldots, x_{\sharp f})) = \alpha(\mathtt{f})\big(x_{\alpha_{\mathtt{f}}(1)}, \ldots, x_{\alpha_{\mathtt{f}}(\sharp\alpha(\mathtt{f}))}\big). \tag{5}$$

We abuse the notation by denoting different entities by α, but this should not lead to any confusion.

▶ **Example 6.** For $\Sigma = \{\|\}$ with $\sharp(\|) = 2$, consider a syntactic translation from Σ to Σ that exchanges the arguments of $\|$, defined by $\alpha(\|) = \|$ and $\alpha_\|(1) = 2$, $\alpha_\|(2) = 1$. For any set X, this determines a function $\alpha_X : \Sigma X \to \Sigma X$ given by $\alpha_X(x \parallel y) = y \parallel x$ for $x, y \in X$.

Neither component of a syntactic translation is required to be an injective function. For example, consider $\Sigma = \{\mathtt{f}, \mathtt{g}\}$ and $\Sigma' = \{\mathtt{k}\}$ with $\sharp\mathtt{f} = \sharp\mathtt{g} = 1$ and $\sharp\mathtt{k} = 2$, and a syntactic translation from Σ to Σ' defined by: $\alpha(\mathtt{f}) = \alpha(\mathtt{g}) = \mathtt{k}$ and $\alpha_{\mathtt{f}}(1) = \alpha_{\mathtt{f}}(2) = \alpha_{\mathtt{g}}(1) = \alpha_{\mathtt{g}}(2) = 1$. This determines, for any X, a function $\alpha_X(\mathtt{f}(x)) = \alpha_X(\mathtt{g}(x)) = \mathtt{k}(x, x)$.

It is a standard exercise to prove that for any syntactic translation, the functions α_X defined by (5) form a natural transformation $\alpha : \Sigma \Longrightarrow \Sigma'$. Moreover, every natural transformation of this type arises this way from some syntactic translation.

5.2 Behavioural translations

Natural transformations $\theta : (\mathcal{P}_\omega -)^{L'} \Longrightarrow (\mathcal{P}_\omega -)^{L}$ are almost a special case of simple distributive laws $\lambda : \Sigma(\mathcal{P}_\omega -)^L \Longrightarrow (\mathcal{P}_\omega \Sigma -)^L$ considered in Section 3, for $\Sigma = \mathrm{Id}$; the only difference is the use of two sets of transition labels L, L'. Accordingly, specifications of such transformations look almost like degenerated cases of simple SOS specifications.

▶ **Definition 7.** A *behavioural translation* Θ (from L' to L) is a set of *behavioural rules*, i.e., expressions of the form

$$\frac{\left\{\mathtt{x} \xrightarrow{a_j} \mathtt{y}_j\right\}_{j=1..m} \quad \left\{\mathtt{x} \not\xrightarrow{b_k}\right\}_{k=1..l}}{\mathtt{x} \xrightarrow{c} \mathtt{y}} \tag{6}$$

where $m, l \in \mathbb{N}$; $\mathtt{x}$ and all $\mathtt{y}_j \in \mathcal{V}$ are all distinct; $\mathtt{y} \in \{\mathtt{y}_1, \ldots, \mathtt{y}_m\}$; $a_j, b_k \in L'$ and $c \in L$.

For any set X, rules are triggered by sets of X-literals just as in the case of SOS specifications; the only difference is that they infer X-literals rather than ΣX-literals, and that the triggering literals use labels from L' rather than L. For a behavioural translation Θ, $\Theta[\Phi]$ will denote the set of literals inferred from Φ by rules from Θ.

A behavioural translation Θ induces a function $\theta_X : (\mathcal{P}_\omega X)^{L'} \Longrightarrow (\mathcal{P}_\omega X)^L$ by:

$$\theta_X(\gamma)(c) = \{y \in X \mid (x \xrightarrow{c} y) \in \Theta[\Phi]\}, \text{ where } \Phi = \{x \xrightarrow{a} y \mid a \in L',\ y \in \gamma(a)\} \tag{7}$$

for a function $\gamma : L' \to \mathcal{P}_\omega X$, a $c \in L$, and any $x \in X$. As in (3), the value of θ_X does not depend on the choice of x.

▶ **Example 8.** Consider a totally ordered set of labels $L = \{a_1, a_2, \ldots, a_n\}$, with the intuition that a_i has a higher "priority" than a_j, for $i < j$. Consider the following behavioural translation from L to L:

$$\frac{\left\{\mathtt{x} \not\xrightarrow{a_j}\right\}_{j<i} \quad \mathtt{x} \xrightarrow{a_i} \mathtt{y}}{\mathtt{x} \xrightarrow{a_i} \mathtt{y}} \quad \text{for } i = 1, \ldots, n.$$

Intuitively, this translation selects transitions with labels of the highest available priority. According to (7), this defines:

$$\theta_X(\gamma)(a_i) = \begin{cases} \gamma(a_i) & \text{if } \gamma(a_j) = \emptyset \text{ for all } j < i \\ \emptyset & \text{otherwise.} \end{cases}$$

▶ **Theorem 9.** *For any behavioural translation from L' to L, the functions θ_X defined in (7) form a natural transformation $\theta : (\mathcal{P}_\omega -)^{L'} \Longrightarrow (\mathcal{P}_\omega -)^L$. Moreover, every natural transformation of this type arises this way from some behavioural translation.*

Proof. This is essentially a special case of Theorem 3; the distinction between sets of labels L and L' does not change the proof in any essential way. ◀

Some behavioural translations arise from functions between transition labels. On one hand, a function $l : L \to L'$ gives rise to a natural transformation $\theta : (\mathcal{P}_\omega -)^{L'} \Longrightarrow (\mathcal{P}_\omega -)^{L}$:

$$\theta_X(\gamma)(c) = \gamma(l(c)) \text{ for } \gamma : L' \to \mathcal{P}_\omega X \text{ and } c \in L,$$

specified by rules $\dfrac{\mathtt{x} \xrightarrow{l(c)} \mathtt{y}}{\mathtt{x} \xrightarrow{c} \mathtt{y}}$ for $c \in L$. On the other hand, a function $k : L' \to L$ determines a transformation θ of the same type by:

$$\theta_X(\gamma)(c) = \bigcup\{\gamma(a) \mid k(a) = c\} \text{ for } \gamma : L' \to \mathcal{P}_\omega X \text{ and } c \in L,$$

specified by rules $\dfrac{\mathtt{x} \xrightarrow{a} \mathtt{y}}{\mathtt{x} \xrightarrow{k(a)} \mathtt{y}}$ for $a \in L'$.

6 Compatible translations

A morphism from an SOS specification Λ over syntax Σ and transition labels L, to a specification Λ' over syntax Σ' and transition labels L', should be presented by a syntactic translation from Σ to Σ' and a behavioural translation from L' to L, specified as in Sections 5.1-5.2, subject to a condition abstractly presented as the diagram (4). We shall now present that condition in elementary terms.

▶ **Definition 10.** A syntactic translation α together with a behavioural translation Θ are *compatible translations* from Λ to Λ' if for any set Φ of $\mathcal{V}$-literals with transition labels from L', a term $\mathtt{s} \in \Sigma\mathcal{V}$ and a label $c \in L$:

- for every $\mathtt{r} \in \Sigma\mathcal{V}$ such that $\mathtt{s} \xrightarrow{c} \mathtt{r}$ is in $\Lambda[\Theta[\Phi]]$, the literal $\alpha_\mathcal{V}(\mathtt{s}) \xrightarrow{c} \alpha_\mathcal{V}(\mathtt{r})$ is in $\Theta[\Lambda'[\Phi]]$, and
- for every $\mathtt{t} \in \Sigma'\mathcal{V}$ such that $\alpha_\mathcal{V}(\mathtt{s}) \xrightarrow{c} \mathtt{t}$ is in $\Theta[\Lambda'[\Phi]]$, there is some $\mathtt{r} \in \Sigma\mathcal{V}$ such that: $\alpha_\mathcal{V}(\mathtt{r}) = \mathtt{t}$ and the literal $\mathtt{s} \xrightarrow{c} \mathtt{r}$ is in $\Lambda[\Theta[\Phi]]$.

This bisimulation-like condition can be more succinctly written as:

$$\{\alpha_\mathcal{V}(\mathtt{r}) \in \Sigma'\mathcal{V} \mid (\mathtt{s} \xrightarrow{c} \mathtt{r}) \in \Lambda[\Theta[\Phi]]\} = \{\mathtt{t} \in \Sigma'\mathcal{V} \mid (\alpha_\mathcal{V}(\mathtt{s}) \xrightarrow{c} \mathtt{t}) \in \Theta[\Lambda'[\Phi]]\}. \tag{8}$$

Before we prove that compatible translations are equivalent to distributive law morphisms, let us elaborate the sets $\Lambda[\Theta[\Phi]]$ and $\Theta[\Lambda'[\Phi]]$, which arise from different ways of combining SOS rules with behavioural rules. In the following, a $\mathcal{V}$-*instance* of a rule (either an SOS one, or one of a behavioural translation) will mean a rule translated along some renaming function $\sigma : \mathcal{V} \to \mathcal{V}$, not necessarily bijective.

For a Λ and Θ as above, a $\Lambda\Theta$-*derivation* consists of:

- an $\mathcal{V}$-instance R of a rule in Λ,
- for each positive premise of R, a $\mathcal{V}$-instance of a rule in Θ with that premise as the conclusion.

A derivation is naturally presented as a simple tree-like structure built of rule instances. It has three types of premises: (a) positive premises of the Θ-rules, (b) negative premises of the Θ-rules, and (c) negative premises of the Λ-rule. Note that transition labels in premises of type (a) and (b) come from L', and of type (c) – from L.

We say that such a derivation is *triggered* by a set Φ of $\mathcal{V}$-literals with labels from L' if:

- every Θ-rule in the derivation is triggered by Φ, and
- for every premise $\mathtt{x} \not\xrightarrow{a}$ of type (c), there is no instance of a Θ-rule triggered by Φ and with conclusion of the form $\mathtt{x} \xrightarrow{a} \mathtt{y}$ for any $\mathtt{y} \in \mathcal{V}$.

In that case we say that the $\Sigma\mathcal{V}$-literal (with a transition label from L) that is the conclusion of the derivation, is $\Lambda\Theta$-*inferred* from Φ. It is straightforward to check that $\Lambda[\Theta[\Phi]]$ is the set of all literals $\Lambda\Theta$-inferred from Φ.

▶ **Example 11.** For $\Sigma = \{\otimes\}$ with $\sharp(\otimes) = 2$, and for $L = \{a_1, \ldots, a_n\}$, consider a specification Λ with rules:

$$\frac{\mathtt{x} \xrightarrow{a_i} \mathtt{x}' \quad \mathtt{y} \xrightarrow{a_i} \mathtt{y}'}{\mathtt{x} \otimes \mathtt{y} \xrightarrow{a_i} \mathtt{x}' \otimes \mathtt{y}'} \qquad \text{for } i = 1, \ldots, n.$$

For the behavioural translation Θ from Example 8, $\Lambda\Theta$-derivations are of the form:

$$\frac{\frac{\{\mathtt{w} \xrightarrow{a_j} \not\}_{j<i} \quad \mathtt{w} \xrightarrow{a_i} \mathtt{w}'}{\mathtt{w} \xrightarrow{a_i} \mathtt{w}'} \quad \frac{\{\mathtt{z} \xrightarrow{a_j} \not\}_{j<i} \quad \mathtt{z} \xrightarrow{a_i} \mathtt{z}'}{\mathtt{z} \xrightarrow{a_i} \mathtt{z}'}}{\mathtt{w} \otimes \mathtt{z} \xrightarrow{a_i} \mathtt{w}' \otimes \mathtt{z}'}$$

for any (not necessarily distinct) $\mathtt{w}, \mathtt{w}', \mathtt{z}, \mathtt{z}' \in \mathcal{V}$ and $a_i \in L$. A set Φ of $\mathcal{V}$-literals triggers the derivation if and only if $\{\mathtt{w} \xrightarrow{a_i} \mathtt{w}',\ \mathtt{z} \xrightarrow{a_i} \mathtt{z}'\} \subseteq \Phi$ and Φ contains no literals with source $\mathtt{w}$ or $\mathtt{z}$ and label a_j for $j < i$.

On the other hand, a $\Theta\Lambda'$-*derivation* consists of:

- an $\Sigma'\mathcal{V}$-instance R of a rule in Θ,
- for each positive premise of R, a $\mathcal{V}$-instance of a rule in Λ' with that premise as the conclusion.

Such a derivation has three types of premises: (a) positive premises of the Λ'-rules, (b) negative premises of the Λ'-rules, and (c) negative premises of the Θ-rule. Note that transition labels in all these premises come from L'. However, while sources of premises of type (a) and (b) come from $\mathcal{V}$, sources of premises of type (c) come from $\Sigma'\mathcal{V}$. Such a derivation is *triggered* by a set Φ of $\mathcal{V}$-literals with transition labels from L', if:

- every Λ'-rule in the derivation is triggered by Φ, and
- for every premise $\mathtt{f}(\mathtt{x}_1, \ldots, \mathtt{x}_{\sharp\mathtt{f}}) \not\xrightarrow{a}$ of type (c), there is no instance of a Λ'-rule triggered by Φ and with conclusion of the form $\mathtt{f}(\mathtt{x}_1, \ldots, \mathtt{x}_{\sharp\mathtt{f}}) \xrightarrow{a} \mathtt{t}$ for any $\mathtt{t} \in \Sigma'\mathcal{V}$.

In that case we say that the $\Sigma'\mathcal{V}$-literal that is the conclusion of the derivation, is $\Theta\Lambda'$-*inferred* from Φ. As before, it is easy to check that the set of all such literals is equal to $\Theta[\Lambda'[\Phi]]$.

▶ **Example 12.** For Σ, L, $\Lambda' = \Lambda$ and Θ as in Example 11, $\Theta\Lambda$-derivations are of the form:

$$\frac{\{\mathtt{w} \otimes \mathtt{z} \xrightarrow{a_j} \not\}_{j<i} \quad \frac{\mathtt{w} \xrightarrow{a_i} \mathtt{w}' \quad \mathtt{z} \xrightarrow{a_i} \mathtt{z}'}{\mathtt{w} \otimes \mathtt{z} \xrightarrow{a_i} \mathtt{w}' \otimes \mathtt{z}'}}{\mathtt{w} \otimes \mathtt{z} \xrightarrow{a_i} \mathtt{w}' \otimes \mathtt{z}'}$$

for any (not necessarily distinct) $\mathtt{w}, \mathtt{w}', \mathtt{z}, \mathtt{z}' \in \mathcal{V}$ and $a_i \in L$. A set Φ of $\mathcal{V}$-literals triggers the derivation if and only if $\{\mathtt{w} \xrightarrow{a_i} \mathtt{w}',\ \mathtt{z} \xrightarrow{a_i} \mathtt{z}'\} \subseteq \Phi$ and Λ infers from Φ no literals with source $\mathtt{w} \otimes \mathtt{z}$ and label a_j for $j < i$.

For example, $\Phi = \{\mathtt{w} \xrightarrow{a_2} \mathtt{w}',\ \mathtt{z} \xrightarrow{a_1} \mathtt{z}', \mathtt{z} \xrightarrow{a_2} \mathtt{z}'\}$ triggers the above derivation (for $i = 2$), but it does not trigger the $\Theta\Lambda$-derivation in Example 11. Indeed, it is easy to check that $\mathtt{w} \otimes \mathtt{z} \xrightarrow{a_2} \mathtt{w}' \otimes \mathtt{z}$ is $\Lambda\Theta$-inferred but not $\Theta\Lambda$-inferred from Φ so, according to Definition 10, the identity syntactic translation from Σ to itself, together with Θ from Example 8, do not form a compatible translation from Λ to itself.

We are now ready for our main characterization of distributive law morphisms:

▶ **Theorem 13.** *Translations α and Θ are compatible from Λ to Λ' if and only if α and the corresponding θ form a morphism of the corresponding distributive laws from λ to λ'.*

Proof. The diagram in Definition 4 states that two composite natural transformations from $\Sigma B'$ to $B\Sigma'$ are equal. It is not difficult to see that one can equivalently ask for their components at $\mathcal{V}$ to be equal. Indeed, in general, for any natural transformations $\phi, \psi : F \Longrightarrow G$ between functors on **Set**, if F is finitary and G preserves monomorphisms then for any infinite set $\mathcal{V}$, if $\phi_{\mathcal{V}} = \psi_{\mathcal{V}}$ then $\phi = \psi$. All functors considered in this paper are finitary and preserve monomorphisms, therefore we can replace the diagram in Definition 4 by its component at $\mathcal{V}$:

$$\begin{array}{ccccc} \Sigma B'\mathcal{V} & \xrightarrow{\Sigma\theta_{\mathcal{V}}} & \Sigma B\mathcal{V} & \xrightarrow{\lambda_{\mathcal{V}}} & B\Sigma\mathcal{V} \\ & \searrow^{\alpha_{B'\mathcal{V}}} & & & \searrow^{B\alpha_{\mathcal{V}}} \\ & & \Sigma' B'\mathcal{V} & \xrightarrow[\lambda'_{\mathcal{V}}]{} & B'\Sigma'\mathcal{V} \xrightarrow[\theta_{\Sigma'\mathcal{V}}]{} B\Sigma'\mathcal{V} \end{array} \tag{9}$$

and ask for it to commute. To this end, consider an arbitrary $\mathbf{A} = \mathtt{f}(\gamma_1, \ldots, \gamma_n) \in \Sigma B'\mathcal{V}$ where $n = \sharp\mathtt{f}$, and denote:

$$\begin{array}{lll} \mathbf{B} = \Sigma\theta_{\mathcal{V}}(\mathbf{A}) \in \Sigma B\mathcal{V}, & \mathbf{C} = \lambda_{\mathcal{V}}(\mathbf{B}) \in B\Sigma\mathcal{V}, & \mathbf{D} = B\alpha_{\mathcal{V}}(\mathbf{C}) \in B\Sigma'\mathcal{V}, \\ \mathbf{E} = \alpha_{B'\mathcal{V}}(\mathbf{A}) \in \Sigma' B'\mathcal{V}, & \mathbf{F} = \lambda'_{\mathcal{V}}(\mathbf{E}) \in B'\Sigma'\mathcal{V}, & \mathbf{G} = \theta_{\Sigma'\mathcal{V}}(\mathbf{F}) \in B\Sigma'\mathcal{V}. \end{array}$$

Our goal is to show that α and Θ are compatible if and only if $\mathbf{D} = \mathbf{G}$, for any $\mathbf{A}$. To this end, we unfold the definition of θ according to (7) to obtain:

$$\mathbf{B} = \mathtt{f}(\delta_1, \ldots, \delta_n) \qquad \text{where } \delta_i(b) = \{\mathtt{y} \in X \mid (\mathtt{x}_i \xrightarrow{b} \mathtt{y}) \in \Theta[\Phi_i]\} \text{ for any } b \in L, \\ \Phi_i = \{\mathtt{x}_i \xrightarrow{a} \mathtt{y} \mid a \in L',\ \mathtt{y} \in \gamma_i(a)\}$$

where for each $i = 1..n$ a distinct variable $\mathtt{x}_i \in \mathcal{V}$ is chosen. Further, we unfold the definition of λ using the same variables $\mathtt{x}_1, \ldots, \mathtt{x}_n$ according to (3), to get:

$$\mathbf{C}(c) = \{\mathtt{r} \in \Sigma\mathcal{V} \mid (\mathtt{f}(\mathtt{x}_1, \ldots, \mathtt{x}_n) \xrightarrow{c} \mathtt{r}) \in \Lambda[\Upsilon]\} \\ \text{where } \Upsilon = \{\mathtt{x}_i \xrightarrow{b} \mathtt{y} \mid 1 \le i \le n,\ b \in L,\ \mathtt{y} \in \delta_i(b)\} = \bigcup_{i=1}^{n} \Theta[\Phi_i] = \Theta\left[\bigcup_{i=1}^{n} \Phi_i\right]$$

for any $c \in L$. The last equality holds by definition of Θ, since literals in distinct Φ_i have distinct sources, and all premises in any single Θ-rule have the same source. Denoting $\Phi = \bigcup_{i=1}^{n} \Phi_i$, we further obtain

$$\mathbf{D}(c) = \{\alpha_{\mathcal{V}}(\mathtt{r}) \in \Sigma'\mathcal{V} \mid (\mathtt{f}(\mathtt{x}_1, \ldots, \mathtt{x}_n) \xrightarrow{c} \mathtt{r}) \in \Lambda[\Theta[\Phi]]\}$$

for any $c \in L$. For the other side of the diagram, put $\mathtt{g} = \alpha(\mathtt{f})$ and $m = \sharp\mathtt{g}$; then $\mathbf{E} = \mathtt{g}(\gamma_{\alpha_\mathtt{f}(1)}, \ldots, \gamma_{\alpha_\mathtt{f}(m)})$. Further, unfold according to (3) the definition of λ' using variables $\mathtt{x}_{\alpha_\mathtt{f}(1)}, \ldots, \mathtt{x}_{\alpha_\mathtt{f}(m)}$ where each $\mathtt{x}_i$ was chosen above, to obtain:

$$\mathbf{F}(b) = \{\mathtt{t} \in \Sigma'\mathcal{V} \mid (\mathtt{g}(\mathtt{x}_{\alpha_\mathtt{f}(1)}, \ldots, \mathtt{x}_{\alpha_\mathtt{f}(m)}) \xrightarrow{b} \mathtt{t}) \in \Lambda'[\Psi]\} \\ \text{where } \Psi = \{\mathtt{x}_{\alpha_\mathtt{f}(i)} \xrightarrow{a} \mathtt{y} \mid 1 \le i \le m,\ a \in L',\ \mathtt{y} \in \gamma_{\alpha_\mathtt{f}(i)}(a)\}$$

for any $b \in L'$. Observe that $\Psi \subseteq \Phi$, and Ψ coincides with Φ when restricted to literals whose sources are among $\mathtt{x}_{\alpha_\mathtt{f}(1)}, \ldots, \mathtt{x}_{\alpha_\mathtt{f}(m)}$ (indeed, $\Psi = \Phi$ if $\alpha_\mathtt{f} : m \to n$ is surjective). This implies that $\Lambda'[\Psi]$ coincides with $\Lambda'[\Phi]$ when restricted to literals with source $\mathtt{g}(\mathtt{x}_{\alpha_\mathtt{f}(1)}, \ldots, \mathtt{x}_{\alpha_\mathtt{f}(m)})$). As a result, we may write:

$$\mathbf{F}(b) = \{\mathtt{t} \in \Sigma'\mathcal{V} \mid (\mathtt{g}(\mathtt{x}_{\alpha_\mathtt{f}(1)}, \ldots, \mathtt{x}_{\alpha_\mathtt{f}(m)}) \xrightarrow{b} \mathtt{t}) \in \Lambda'[\Phi]\}.$$

Finally, we unfold the definition of θ according to (7), using $\mathtt{g}(\mathtt{x}_{\alpha_\mathtt{f}(1)}, \ldots, \mathtt{x}_{\alpha_\mathtt{f}(m)}) \in \Sigma'\mathcal{V}$ as the variable, to obtain:

$$\mathbf{G}(c) = \{\mathtt{t} \in \Sigma'\mathcal{V} \mid (\mathtt{g}(\mathtt{x}_{\alpha_\mathtt{f}(1)}, \ldots, \mathtt{x}_{\alpha_\mathtt{f}(m)}) \xrightarrow{c} \mathtt{t}) \in \Theta[\Xi]\}$$
$$\text{where } \Xi = \{(\mathtt{g}(\mathtt{x}_{\alpha_\mathtt{f}(1)}, \ldots, \mathtt{x}_{\alpha_\mathtt{f}(m)}) \xrightarrow{b} \mathtt{t}) \mid b \in L',\ \mathtt{t} \in \mathbf{F}(b)\} = \Lambda'[\Phi].$$

Putting it all together, the diagram (9) commutes if and only if, for every $\mathtt{f}(\gamma_1, \ldots, \gamma_n) \in \Sigma B\mathcal{V}$ and every $c \in L$:

$$\{\alpha_\mathcal{V}(\mathtt{r}) \in \Sigma'\mathcal{V} \mid (\mathtt{f}(\mathtt{x}_1, \ldots, \mathtt{x}_n) \xrightarrow{c} \mathtt{r}) \in \Lambda[\Theta[\Phi]]\} = \{\mathtt{t} \in \Sigma'\mathcal{V} \mid (\mathtt{g}(\mathtt{x}_{\alpha_\mathtt{f}(1)}, \ldots, \mathtt{x}_{\alpha_\mathtt{f}(m)}) \xrightarrow{c} \mathtt{t}) \in \Theta[\Lambda'[\Phi]]\} \quad (10)$$

$$\text{where } \Phi = \{\mathtt{x}_i \xrightarrow{a} \mathtt{y} \mid 1 \leq i \leq n,\ a \in L',\ \mathtt{y} \in \gamma_i(a)\}. \quad (11)$$

It is easy to see that (10) is implied by the condition (8) of the definition of compatible translation, therefore if α and Θ form a compatible translation from Λ to Λ' then the diagram (9) commutes.

For the implication from (10) to (8), consider any $\mathtt{s} = \mathtt{f}(\mathtt{x}_1, \ldots, \mathtt{x}_n) \in \Sigma\mathcal{V}$, any $c \in L$ and any set Ψ of $\mathcal{V}$-literals. Define $\gamma_i(a) = \{y \in \mathcal{V} \mid (\mathtt{x}_i \xrightarrow{a} y) \in \Psi\}$ for $1 \leq i \leq n$ and $a \in L'$, and define Φ from the γ_i as in (11). Clearly $\Phi \subseteq \Psi$; moreover, Φ and Ψ coincide on literals whose sources are among the $\mathtt{x}_i$. As a result:

- $\Lambda[\Theta[\Phi]]$ and $\Lambda[\Theta[\Psi]]$ coincide on literals with source $\mathtt{f}(\mathtt{x}_1, \ldots, \mathtt{x}_n)$, and
- $\Theta[\Lambda'[\Phi]]$ and $\Theta[\Lambda'[\Psi]]$ coincide on literals with source $\mathtt{g}(\mathtt{x}_{\alpha_\mathtt{f}(1)}, \ldots, \mathtt{x}_{\alpha_\mathtt{f}(m)})$.

The implication from (10) to (8) follows immediately. ◀

▶ Remark. It is important to note that the defining property of compatible translations, Definition 10, is decidable for given Λ, Λ', α and Θ. First, for a fixed finite set Φ of literals it is possible to compute $\Lambda[\Theta[\Phi]]$ and $\Theta[\Lambda'[\Phi]]$ by checking all combinations of Λ-, Λ'- and Θ-rules; each rule can have infinitely many $\mathcal{V}$-instances, but one only needs to consider those instances where all variables are present in Φ, which only leaves finitely many cases to check.

Moreover, one may restrict attention to finitely many sets Φ, all of them finite. Indeed, if a literal with source $\mathtt{s}$ is $\Theta\Lambda$- or $\Lambda\Theta$-inferred from Φ, then the derivation only depends on $\mathcal{V}$-literals with sources that are present in $\mathtt{s}$; this gives a bound on the number of different source variables in Φ's worth considering. The number of literals in Φ with a particular source variable can be bound by the number of premises with the same source in a $\Theta\Lambda$- or $\Lambda\Theta$-derivation; this gives a computable bound on the size of Φ worth checking. Finally, the condition of Definition 10 is invariant with respect to bijective renaming of variables. Altogether, this gives an effective procedure for checking whether given translations form a valid distributive law morphism.

7 Extensions

The framework of simple SOS specifications is very restrictive, and covers very few interesting examples of operational semantics. Right from the beginning [13], the distributive law approach to SOS was designed to cover far more general classes of specifications. So far in this paper we only treated simple SOS, to explain the general idea of presenting distributive law morphisms in a relatively basic setting. In this section we sketch two extensions of that basic setting: to GSOS specifications, and to extended syntactic translations. Fortunately, as we shall see, these extensions require little effort and everything works essentially as before.

7.1 GSOS specifications

A *GSOS law* of Σ over B is a natural transformation $\lambda : \Sigma(\mathrm{Id} \times B) \Longrightarrow B\Sigma^*$, where Σ^* is the (underlying functor of) the free monad over Σ. Refer to [13] to see a notion of bialgebra for such laws, or [9] to see how they are equivalent to distributive laws of the copointed functor $\mathrm{Id} \times B$ over the monad Σ^*. Those results are crucial for the abstract theory of GSOS laws, but not necessary for understanding of our elementary development.

The following, introduced in [3], is a generalization of Definition 1:

▶ **Definition 14.** A *GSOS specification* Λ is finite set of *GSOS rules*, i.e., expressions of the form

$$\frac{\left\{\mathtt{x}_{i_j} \xrightarrow{a_j} \mathtt{y}_j\right\}_{j=1..m} \quad \left\{\mathtt{x}_{i_k} \not\xrightarrow{b_k}\right\}_{k=1..l}}{\mathtt{f}(\mathtt{x}_1, \ldots, \mathtt{x}_{\sharp\mathtt{f}}) \xrightarrow{c} \mathtt{t}} \tag{12}$$

where: $\mathtt{f} \in \Sigma$; $m, l \in \mathbb{N}$; all $i_j, i_k \in \{1, \ldots, \sharp\mathtt{f}\}$; all $\mathtt{x}_i, \mathtt{y}_j \in \mathcal{V}$ are pairwise distinct, $a_j, b_k, c \in L$; and $\mathtt{t}$ is a Σ-term whose all variables come from the $\mathtt{x}_i$ and $\mathtt{y}_j$.

GSOS rules generalize simple SOS rules in that their targets $\mathtt{t}$ are arbitrary terms rather than single operations from Σ, and in that their source variables $\mathtt{x}_i$ are allowed in the target.

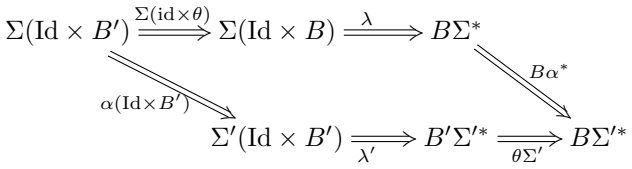
The notions of triggering rules and inferred literals $\Lambda[\Phi]$ are as for simple SOS specifications, except that now targets of inferred literals are arbitrary Σ-terms. A GSOS specification Λ determines a function $\lambda_X : \Sigma(X \times (\mathcal{P}_\omega X)^L) \to (\mathcal{P}_\omega \Sigma^* X)^L$ as in (3):

$$\lambda_X(\mathtt{f}(x_1, \gamma_1, \ldots, x_{\sharp\mathtt{f}}, \gamma_{\sharp\mathtt{f}}))(c) = \left\{t \in \Sigma^* X \mid (\mathtt{f}(x_1, \ldots, x_{\sharp\mathtt{f}}) \xrightarrow{c} t) \in \Lambda[\Phi]\right\}, \text{ where}$$
$$\Phi = \{x_i \xrightarrow{a} y \mid 1 \le i \le \sharp\mathtt{f},\ a \in L,\ y \in \gamma_i(a)\},$$

except this definition is actually slightly simpler than (3), since the values $x_1, \ldots, x_{\sharp\mathtt{f}}$ need not be chosen arbitrarily, as they are provided in the argument of λ_X.

A generalization of Theorem 3, which was actually the result stated in [13] and proved in [2], shows that GSOS specifications correspond to GSOS laws just as simple SOS specifications correspond to simple distributive laws.

By analogy to Definition 4, a morphism of GSOS laws is a pair of natural transformations $\alpha : \Sigma \Longrightarrow \Sigma'$ and $\theta : B' \Longrightarrow B$ such that



commutes, where $\alpha^* : \Sigma^* \Longrightarrow \Sigma'^*$ is the obvious inductive extension of α to all Σ-terms.

Definition 10 of compatible translations carries over to GSOS specifications almost verbatim, except that $\alpha_{\mathcal{V}}(\mathtt{r})$ needs to be replaced by $\alpha^*_{\mathcal{V}}(\mathtt{r})$, as now $\mathtt{r} \in \Sigma^*\mathcal{V}$ is an arbitrary term. Stated succinctly by analogy to (8), the compatibility condition becomes:

$$\{\alpha^*_{\mathcal{V}}(\mathtt{r}) \in \Sigma'^*\mathcal{V} \mid (\mathtt{s} \xrightarrow{c} \mathtt{r}) \in \Lambda[\Theta[\Phi]]\} = \{\mathtt{t} \in \Sigma'^*\mathcal{V} \mid (\alpha_{\mathcal{V}}(\mathtt{s}) \xrightarrow{c} \mathtt{t}) \in \Theta[\Lambda'[\Phi]]\}$$

for all $\mathtt{s} \in \Sigma\mathcal{V}$, $c \in L$ and Φ a set of $\mathcal{V}$-literals. Theorem 13 still holds with these changes, with a completely analogous proof.

7.2 Generalized syntactic translations

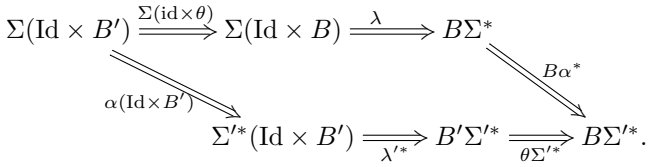
In practical examples of translations between operational specifications, one often wishes to interpret an operation from the source signature not as a single operation, but as a complex term over the target signature. A natural way to model such situations is to consider an extended definition of a syntactic translation as a natural transformation $\alpha : \Sigma \Longrightarrow \Sigma'^*$. Such a transformation can be presented much the same as in Section 5.1, by a function from Σ to Σ'-terms (over some fixed set of variables), together with a function $\alpha_{\mathtt{f}}$ for each $\mathtt{f} \in \Sigma$ as in Definition 5, where arity of complex terms is defined inductively in an obvious way.

▶ **Example 15.** Consider signatures $\Sigma = \{\|\}$ and $\Sigma' = \{\lfloor, +\}$, where all operators have arity 2 (so that $\Sigma X = X^2$ and $\Sigma' X = X^2 + X^2$). A translation that maps every term $\mathtt{x} \parallel y \in \Sigma X$ to $(x\lfloor y) + (y\lfloor x) \in \Sigma'^* X$ is formally defined by a function that maps the symbol $\|$ to a term $(1\lfloor 2) + (3\lfloor 4) \in \Sigma'^* \mathbb{N}$ together with a mapping $\alpha_{\|}(1) = \alpha_{\|}(4) = 1,\ \alpha_{\|}(2) = \alpha_{\|}(3) = 2$.

▶ **Example 16.** Consider signatures $\Sigma = \{\mathtt{p}\}$ with $\sharp\mathtt{p} = 3$ and $\Sigma' = \{\|\}$ with $\sharp(\|) = 2$. Two extended syntactic translations from Σ to Σ' come to mind, given by:

$$\alpha_X(\mathtt{p}(x,y,z)) = (x \parallel y) \parallel z \qquad \text{or} \qquad \alpha'_X(\mathtt{p}(x,y,z)) = x \parallel (y \parallel z), \qquad \text{for } x,y,z \in X.$$

Generalized syntactic translations make sense already in connection to morphisms of simple distributive laws, but they are more naturally considered in the context of GSOS laws. A GSOS law $\lambda : \Sigma(\mathrm{Id} \times B) \Longrightarrow B\Sigma^*$ extends, by induction on Σ-terms, to a natural transformation $\lambda^* : \Sigma^*(\mathrm{Id} \times B) \Longrightarrow B\Sigma^*$ (see [9] for a detailed study of this and related issues). It is natural to redefine a morphism between GSOS laws λ and λ' as an (extended) syntactic translation $\alpha : \Sigma \Longrightarrow \Sigma'^*$ and a natural transformation $\theta : B' \Longrightarrow B$ as before, such that



commutes, where $\alpha^* : \Sigma^* \Longrightarrow \Sigma'^*$ is the inductive extension of α to all Σ-terms.

A corresponding notion of compatible translations is straightforward to define, but a little less so than in Section 7.1. For a GSOS specification Λ, one defines Λ^*-*derivations* as well-formed trees built of instances of rules from Λ. For example, for Λ as in Example 11,

$$\frac{\dfrac{\mathtt{x} \xrightarrow{a_i} \mathtt{x}' \quad \mathtt{y} \xrightarrow{a_i} \mathtt{y}'}{\mathtt{x} \otimes \mathtt{y} \xrightarrow{a_i} \mathtt{x}' \otimes \mathtt{y}'} \qquad \mathtt{z} \xrightarrow{a_i} \mathtt{z}'}{(\mathtt{x} \otimes \mathtt{y}) \otimes \mathtt{z} \xrightarrow{a_i} (\mathtt{x}' \otimes \mathtt{y}') \otimes \mathtt{z}'}$$

is a valid derivation triggered by $\Phi = \{\mathtt{x} \xrightarrow{a_i} \mathtt{x}',\ \mathtt{y} \xrightarrow{a_i} \mathtt{y}',\ \mathtt{z} \xrightarrow{a_i} \mathtt{z}'\}$, and it infers the literal in the conclusion; the (usually infinite) set of all inferred literals is denoted by $\Lambda^*[\Phi]$.

Compatible translations are then defined similarly as in Definition 10, with Λ'-derivations replaced by Λ'^*-derivations. By analogy to (8), the compatibility condition becomes:

$$\{\alpha^*_{\mathcal{V}}(\mathtt{r}) \in \Sigma'^*\mathcal{V} \mid (\mathtt{s} \xrightarrow{c} \mathtt{r}) \in \Lambda[\Theta[\Phi]]\} = \{\mathtt{t} \in \Sigma'^*\mathcal{V} \mid (\alpha_{\mathcal{V}}(\mathtt{s}) \xrightarrow{c} \mathtt{t}) \in \Theta[\Lambda'^*[\Phi]]\} \tag{13}$$

for all $\mathtt{s} \in \Sigma\mathcal{V}$, $c \in L$ and Φ a set of $\mathcal{V}$-literals. The corresponding version of Theorem 13 still holds; the proof is slightly more complex technically due to the presence of additional induction on Σ-terms, but no essentially new aspects arise in it.

▶ Remark. Although even for a finite Φ the set $\Theta[\Lambda'^*[\Phi]]$ will often be infinite, it will still be finite when restricted to literals with the source $\alpha_{\mathcal{V}}(\mathtt{s})$, for any fixed $\mathtt{s} \in \Sigma\mathcal{V}$. As a result, the above compatibility condition remains decidable.

8 Examples

▶ **Example 17** (Conservative extension). Consider a signature Σ' and its subsignature Σ (i.e., $\Sigma \subseteq \Sigma'$ and the arities of Σ-symbols are matched in Σ'), and two GSOS specifications Λ and Λ' over Σ and Σ' respectively, with the same set L of transition labels. We say that Λ' is a *conservative extension* of Λ if $\Lambda \subseteq \Lambda'$ and if no rule from $\Lambda' \setminus \Lambda$ has its source from Σ.

Note that this notion is more restrictive than usual definitions of conservative extension considered in SOS theory [1]. There, global properties of specifications play a role; for example, a new rule with a source from Σ may be allowed in Λ' as long as it it has some positive premise whose label cannot possibly be matched by a conclusion of a Λ-rule.

Consider a trivial inclusion syntactic translation $\alpha : \Sigma \Longrightarrow \Sigma'$, and the identity behavioural translation Θ. It is easy to see that α and Θ form a compatible translation from Λ to Λ'. Indeed, for any $\mathtt{s} = \mathtt{f}(\mathtt{x}_1, \ldots, \mathtt{x}_n) \in \Sigma\mathcal{V}$ and $c \in L$, there is a correspondence between $\Lambda\Theta$- and $\Theta\Lambda'$-derivations:

$$\frac{\left\{\dfrac{\mathtt{x}_{i_j} \xrightarrow{a_j} \mathtt{y}_j}{\mathtt{x}_{i_j} \xrightarrow{a_j} \mathtt{y}_j}\right\}_{j=1..m} \quad \left\{\mathtt{x}_{i_k} \not\xrightarrow{b_k}\right\}_{k=1..l}}{\mathtt{f}(\mathtt{x}_1, \ldots, \mathtt{x}_n) \xrightarrow{c} \mathtt{t}} \quad \text{vs.} \quad \frac{\left\{\mathtt{x}_{i_j} \xrightarrow{a_j} \mathtt{y}_j\right\}_{j=1..m} \quad \left\{\mathtt{x}_{i_k} \not\xrightarrow{b_k}\right\}_{k=1..l}}{\mathtt{f}(\mathtt{x}_1, \ldots, \mathtt{x}_n) \xrightarrow{c} \mathtt{t}} \tag{14}$$

that are triggered by the same sets Φ and infer the same conclusions, for any rule from Λ as in (12). No other $\Theta\Lambda'$-derivation for $\mathtt{f}$ is possible, by definition of conservative extension.

▶ **Example 18** (Nservative coextension). Dually, consider a signature Σ, two sets of labels $L \subseteq L'$, and two GSOS specifications Λ and Λ' over Σ and over labels L and L', respectively. We say that Λ' is, for lack of a better name, an *nservative coextension* of Λ if $\Lambda \subseteq \Lambda'$ and if the conclusion of each rule from $\Lambda' \setminus \Lambda$ has a transition label from $L' \setminus L$.

Intuitively, just as a conservative extension (Example 17) defines behaviour for a new part of syntax while leaving the behaviour of the old syntax intact, an nservative coextension defines new aspects of behaviour (i.e., $L' \setminus L$-transitions) while leaving old aspects intact.

Consider the identity syntactic translation on Σ and a trivial behavioural translation Θ from L' to L specified by rules $\dfrac{\mathtt{x} \xrightarrow{c} \mathtt{y}}{\mathtt{x} \xrightarrow{c} \mathtt{y}}$ for $c \in L$. Again, it is easy to see that α and θ are a compatible translation from Λ to Λ'. Indeed, for any $\mathtt{s} = \mathtt{f}(\mathtt{x}_1, \ldots, \mathtt{x}_n) \in \Sigma\mathcal{V}$ and $c \in L$, a correspondence between $\Lambda\Theta$- and $\Theta\Lambda'$-derivations is exactly the same as in (14); again, no other $\Theta\Lambda'$-derivation is possible since $c \in L$ and there are no new rules in Λ' with c as the conclusion label.

In the next example, neither α nor θ is identity.

▶ **Example 19.** For $\Sigma = \{!\}$ with $\sharp(!) = 2$ and $L = \{a_1, \ldots, a_n\}$, consider Λ with rules:

$$\frac{\mathtt{x} \xrightarrow{a_i} \mathtt{x}' \quad (\mathtt{y} \not\xrightarrow{a_j})_{j<i}}{\mathtt{x}!\mathtt{y} \xrightarrow{a_i} \mathtt{x}'!\mathtt{y}} \qquad \frac{(\mathtt{x} \not\xrightarrow{a_j})_{j<i} \quad \mathtt{y} \xrightarrow{a_i} \mathtt{y}'}{\mathtt{x}!\mathtt{y} \xrightarrow{a_i} \mathtt{x}!\mathtt{y}'} \quad \text{for } i = 1, \ldots, n. \tag{15}$$

Moreover, let $\Sigma' = \{\|\}$ with $\sharp(\|) = 2$ and, for the same L, let Λ' consist of rules:

$$\frac{\mathtt{x} \xrightarrow{a_i} \mathtt{x}'}{\mathtt{x} \parallel \mathtt{y} \xrightarrow{a_i} \mathtt{x}' \parallel \mathtt{y}} \qquad \frac{\mathtt{y} \xrightarrow{a_i} \mathtt{y}'}{\mathtt{x} \parallel \mathtt{y} \xrightarrow{a_i} \mathtt{x} \parallel \mathtt{y}'} \quad \text{for } i = 1, \ldots, n. \tag{16}$$

Consider a syntactic translation from Σ to Σ' defined by $\alpha_X(x!y) = x \parallel y$, and the behavioural translation Θ from Example 8. Then $\Lambda\Theta$-derivations are of the form:

$$\frac{\frac{\{\mathtt{x} \overset{a_j}{\not\longrightarrow}\}_{j<i} \quad \mathtt{x} \xrightarrow{a_i} \mathtt{x}'}{\mathtt{x} \xrightarrow{a_i} \mathtt{x}'} \quad \{\mathtt{y} \overset{a_j}{\not\longrightarrow}\}_{j<i}}{\mathtt{x!y} \xrightarrow{a_i} \mathtt{x}'!\mathtt{y}} \qquad \frac{\{\mathtt{x} \overset{a_j}{\not\longrightarrow}\}_{j<i} \quad \frac{\{\mathtt{y} \overset{a_j}{\not\longrightarrow}\}_{j<i} \quad \mathtt{y} \xrightarrow{a_i} \mathtt{y}'}{\mathtt{y} \xrightarrow{a_i} \mathtt{y}'}}{\mathtt{x!y} \xrightarrow{a_i} \mathtt{x!y}'}$$

and $\Theta\Lambda'$-derivations are of the form:

$$\frac{\frac{\mathtt{x} \xrightarrow{a_i} \mathtt{x}'}{\mathtt{x} \parallel \mathtt{y} \xrightarrow{a_i} \mathtt{x}' \parallel \mathtt{y}} \quad \{\mathtt{x} \parallel \mathtt{y} \overset{a_j}{\not\longrightarrow}\}_{j<i}}{\mathtt{x} \parallel \mathtt{y} \xrightarrow{a_i} \mathtt{x}' \parallel \mathtt{y}} \qquad \frac{\{\mathtt{x} \parallel \mathtt{y} \overset{a_j}{\not\longrightarrow}\}_{j<i} \quad \frac{\mathtt{y} \xrightarrow{a_i} \mathtt{y}'}{\mathtt{x} \parallel \mathtt{y} \xrightarrow{a_i} \mathtt{x} \parallel \mathtt{y}'}}{\mathtt{x} \parallel \mathtt{y} \xrightarrow{a_i} \mathtt{x} \parallel \mathtt{y}'}$$

It is easy to see that for any Φ these derivations infer the same literals up to α, since a negative premise $\mathtt{x} \parallel \mathtt{y} \overset{a_j}{\not\longrightarrow}$ holds for Φ if and only if both premises $\mathtt{x} \overset{a_j}{\not\longrightarrow}$ and $\mathtt{y} \overset{a_j}{\not\longrightarrow}$ hold. As a result, α with Θ form a morphism from Λ to Λ'.

Note that the identity syntactic translation together with Θ does *not* give a morphism from Λ' to itself, for reasons similar to Example 12. Indeed, $\Lambda'\Theta$-derivations are of the form:

$$\frac{\frac{\{\mathtt{x} \overset{a_j}{\not\longrightarrow}\}_{j<i} \quad \mathtt{x} \xrightarrow{a_i} \mathtt{x}'}{\mathtt{x} \xrightarrow{a_i} \mathtt{x}'}}{\mathtt{x} \parallel \mathtt{y} \xrightarrow{a_i} \mathtt{x}' \parallel \mathtt{y}} \qquad \frac{\frac{\{\mathtt{y} \overset{a_j}{\not\longrightarrow}\}_{j<i} \quad \mathtt{y} \xrightarrow{a_i} \mathtt{y}'}{\mathtt{y} \xrightarrow{a_i} \mathtt{y}'}}{\mathtt{x} \parallel \mathtt{y} \xrightarrow{a_i} \mathtt{x} \parallel \mathtt{y}'}$$

and for $\Phi = \{\mathtt{x} \xrightarrow{a_2} \mathtt{x}',\ \mathtt{y} \xrightarrow{a_1} \mathtt{y}'\}$ they infer the literal $\mathtt{x} \parallel \mathtt{y} \xrightarrow{a_2} \mathtt{x}' \parallel \mathtt{y}$, whereas $\Theta\Lambda'$-literals above do not.

The following example was considered also in [14].

▶ **Example 20.** Consider Σ and Σ' from Example 15, over the same set of labels L. Let Λ consist of rules:

$$\frac{\mathtt{x} \xrightarrow{a} \mathtt{x}'}{\mathtt{x} \parallel \mathtt{y} \xrightarrow{a} \mathtt{x}' \parallel \mathtt{y}} \qquad \frac{\mathtt{y} \xrightarrow{a} \mathtt{y}'}{\mathtt{x} \parallel \mathtt{y} \xrightarrow{a} \mathtt{y}' \parallel \mathtt{x}} \quad \text{for } a \in L, \tag{17}$$

and let Λ' be the GSOS specification:

$$\frac{\mathtt{x} \xrightarrow{a} \mathtt{x}'}{\mathtt{x} + \mathtt{y} \xrightarrow{a} \mathtt{x}'} \qquad \frac{\mathtt{y} \xrightarrow{a} \mathtt{y}'}{\mathtt{x} + \mathtt{y} \xrightarrow{a} \mathtt{y}'} \qquad \frac{\mathtt{x} \xrightarrow{a} \mathtt{x}'}{\mathtt{x} \lfloor \mathtt{y} \xrightarrow{a} (\mathtt{x}' \lfloor \mathtt{y}) + (\mathtt{y} \lfloor \mathtt{x}')} \quad \text{for } a \in L.$$

Pick a syntactic translation is as in Example 15, and let Θ be the identity behavioural translation. $\Lambda\Theta$-derivations are very simple:

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

$\Theta\Lambda'^*$-derivations are more interesting; the only ones for the term $\alpha_{\mathcal{V}}(\mathtt{x} \parallel \mathtt{y}) = (\mathtt{x} \lfloor \mathtt{y}) + (\mathtt{y} \lfloor \mathtt{x})$ are:

$$\frac{\frac{\frac{\mathtt{x} \xrightarrow{a} \mathtt{x}'}{\mathtt{x} \lfloor \mathtt{y} \xrightarrow{a} (\mathtt{x}' \lfloor \mathtt{y}) + (\mathtt{y} \lfloor \mathtt{x}')}}{(\mathtt{x} \lfloor \mathtt{y}) + (\mathtt{y} \lfloor \mathtt{x}) \xrightarrow{a} (\mathtt{x}' \lfloor \mathtt{y}) + (\mathtt{y} \lfloor \mathtt{x}')}}{(\mathtt{x} \lfloor \mathtt{y}) + (\mathtt{y} \lfloor \mathtt{x}) \xrightarrow{a} (\mathtt{x}' \lfloor \mathtt{y}) + (\mathtt{y} \lfloor \mathtt{x}')} \qquad \frac{\frac{\frac{\mathtt{y} \xrightarrow{a} \mathtt{y}'}{\mathtt{y} \lfloor \mathtt{x} \xrightarrow{a} (\mathtt{y}' \lfloor \mathtt{x}) + (\mathtt{x} \lfloor \mathtt{y}')}}{(\mathtt{x} \lfloor \mathtt{y}) + (\mathtt{y} \lfloor \mathtt{x}) \xrightarrow{a} (\mathtt{y}' \lfloor \mathtt{x}) + (\mathtt{x} \lfloor \mathtt{y}')}}{(\mathtt{x} \lfloor \mathtt{y}) + (\mathtt{y} \lfloor \mathtt{x}) \xrightarrow{a} (\mathtt{y}' \lfloor \mathtt{x}) + (\mathtt{x} \lfloor \mathtt{y}')}$$

It is easy to see that the condition (13) is satisfied, therefore α and Θ form a morphism from Λ to Λ'. Note the slight difference between the targets of rules (16) and (17). There seems to be no morphism from (16) to Λ', which suggests that the notion of distributive law morphism could perhaps be relaxed in a useful way. We leave this for future work.

▶ **Example 21.** Consider Σ, Σ' and α from Example 16. Let Λ over Σ consist of rules:

$$\frac{\mathtt{x} \xrightarrow{a} \mathtt{x}'}{\mathtt{p}(\mathtt{x},\mathtt{y},\mathtt{z}) \xrightarrow{a} \mathtt{p}(\mathtt{x}',\mathtt{y},\mathtt{z})} \qquad \frac{\mathtt{y} \xrightarrow{a} \mathtt{y}'}{\mathtt{p}(\mathtt{x},\mathtt{y},\mathtt{z}) \xrightarrow{a} \mathtt{p}(\mathtt{x},\mathtt{y}',\mathtt{z})} \qquad \frac{\mathtt{z} \xrightarrow{a} \mathtt{z}'}{\mathtt{p}(\mathtt{x},\mathtt{y},\mathtt{z}) \xrightarrow{a} \mathtt{p}(\mathtt{x},\mathtt{y},\mathtt{z}')} \qquad \text{for } a \in L,$$

and let Λ' over Σ' be defined by rules as in (16). For the identity Θ, there is an easy correspondence between $\Lambda\Theta$-derivations and $\Theta\Lambda'$-derivations, for example:

$$\frac{\dfrac{\dfrac{\mathtt{x} \xrightarrow{a} \mathtt{x}'}{\mathtt{x} \xrightarrow{a} \mathtt{x}'}}{\mathtt{p}(\mathtt{x},\mathtt{y},\mathtt{z}) \xrightarrow{a} \mathtt{p}(\mathtt{x}',\mathtt{y},\mathtt{z})}}{} \quad \text{vs.} \quad \frac{\dfrac{\dfrac{\mathtt{x} \xrightarrow{a} \mathtt{x}'}{\mathtt{x} \parallel \mathtt{y} \xrightarrow{a} \mathtt{x}' \parallel \mathtt{y}}}{(\mathtt{x} \parallel \mathtt{y}) \parallel \mathtt{z} \xrightarrow{a} (\mathtt{x}' \parallel \mathtt{y}) \parallel \mathtt{z}}}{(\mathtt{x} \parallel \mathtt{y}) \parallel \mathtt{z} \xrightarrow{a} (\mathtt{x}' \parallel \mathtt{y}) \parallel \mathtt{z}}$$

which shows that α with Θ form a morphism from Λ and Λ'. By analogy, α' with Example 16 forms a similar morphism with Θ. This proves that the equation $(x \parallel y) \parallel z = x \parallel (y \parallel z)$ holds up to bisimilarity in the transition system induced by Λ'. This suggests a connection to quotients of distributive laws studied in [4]; we leave this for future work.

References

1 L. Aceto, W. J. Fokkink, and C. Verhoef. Structural operational semantics. In J. A. Bergstra, A. Ponse, and S. Smolka, editors, *Handbook of Process Algebra*, pages 197–292. Elsevier, 2002.

2 F. Bartels. *On Generalised Coinduction and Probabilistic Specification Formats.* PhD dissertation, CWI, Amsterdam, 2004.

3 B. Bloom, S. Istrail, and A. Meyer. Bisimulation can't be traced. *Journal of the ACM*, 42:232–268, 1995.

4 M. Bonsangue, H. H. Hansen, A. Kurz, and J. Rot. Presenting distributive laws. In *Procs. CALCO'13*, volume 8089 of *LNCS*, pages 95–109, 2013.

5 M. Hennessy, W. Li, and G. D. Plotkin. A first attempt at translating CSP into CCS. In *Proc. Second International Conference on Distributed Systems*, pages 105–115, 1981.

6 B. Klin. Bialgebraic methods and modal logic in structural operational semantics. *Information and Computation*, 207:237–257, 2009.

7 B. Klin. Bialgebras for structural operational semantics: An introduction. *Theoretical Computer Science*, 412(38):5043–5069, 2011. CMCS Tenth Anniversary Meeting.

8 B. Klin and B. Nachyła. Distributive laws and decidable properties of SOS specifications. In *Procs. EXPRESS/SOS'14*, volume 160 of *ENTCS*, pages 79–93, 2014.

9 M. Lenisa, J. Power, and H. Watanabe. Category theory for operational semantics. *Theoretical Computer Science*, 327(1-2):135–154, 2004.

10 G. D. Plotkin. A structural approach to operational semantics. *Journal of Logic and Algebraic Programming*, 60-61:17–139, 2004.

11 J. Power and H. Watanabe. Combining a monad and a comonad. *Theor. Comput. Sci.*, 280:137–162, 2002.

12 J. J. M. M. Rutten. Universal coalgebra: a theory of systems. *Theoretical Computer Science*, 249:3–80, 2000.

13 D. Turi and G. D. Plotkin. Towards a mathematical operational semantics. In *Proc. LICS'97*, pages 280–291. IEEE Computer Society Press, 1997.

14 H. Watanabe. Well-behaved translations between structural operational semantics. *ENTCS*, 65, 2002.