Bialgebraic Operational Semantics and Modal Logic
(extended abstract)

Bartek Klin

University of Edinburgh, Warsaw University
bklin@inf.ed.ac.uk

Abstract

A novel, general approach is proposed to proving the compositionality of process equivalences on languages defined by Structural Operational Semantics (SOS). The approach, based on modal logic, is inspired by the simple observation that if the set of formulas satisfied by a process can be derived from the corresponding sets for its subprocesses, then the logical equivalence is a congruence. Striving for generality, SOS rules are modeled categorically as bialgebraic distributive laws for some notions of process syntax and behaviour, and modal logics are modeled via coalgebraic polyadic modal logic. Compositionality is proved by providing a suitable notion of behaviour for the logic together with a dual distributive law, reflecting the one modeling the SOS specification. Concretely, the dual laws may appear as SOS-like rules where logical formulas play the role of processes, and their behaviour models logical decomposition over process syntax. The approach can be used either to proving compositionality for specific languages or for defining SOS congruence formats.

Introduction

Structural Operational Semantics (SOS) [25, 1] is one of the most successful frameworks for the formal description of programming languages and process calculi. There, the behaviour of programs or processes is described by means of transition relations, also called *labeled transition systems* (LTSs), induced by inference rules following the syntactic structure of processes. For example, rules:

$$\frac{x \xrightarrow{a} x'}{x||y \xrightarrow{a} x'||y} \qquad \frac{y \xrightarrow{a} y'}{x||y \xrightarrow{a} x||y'} \qquad (1)$$

define the behaviour of a binary parallel composition operator $||$ without communication. In particular, the rule on the left says that if a process can do a transition labelled with a, then the same process put in parallel with any other process can do a similar transition. One could also enrich states and/or transitions in SOS specifications with environments, stores, probabilities, time durations etc., to induce other, more sophisticated kinds of transition systems. The intuitive appeal of SOS and, importantly, its inherent support for modeling nondeterministic behaviour, makes it a natural framework for the formal description of process algebras (see [5] for many examples).

For reasoning about processes a suitable notion of *process equivalence* is needed. Various equivalences on LTSs have been proposed (see [12] for a survey). Bisimilarity is the most widely studied, but other equivalences such as trace equivalence or testing equivalence have also been considered. Several equivalences have also been proposed for probabilistic, timed and other kinds of transition systems, including their respective notions of bisimilarity.

To support inductive reasoning, it is important for the chosen process equivalence to be *compositional*; indeed, it is useful to know that if a part of a process is replaced by an equivalent part then the resulting process will be equivalent to the original one. Compositionality proofs for specific languages can be quite lengthy, therefore in the literature many *congruence formats* have been proposed. Such a format is a syntactic restriction on SOS specifications that guarantees a specific equivalence to be compositional on the induced transition system. The most popular format is GSOS [7], which guarantees the compositionality of bisimilarity, but formats for other equivalences and/or kinds of transition systems have also been studied (see [1, 14]).

The task of finding a reasonably permissive congruence format for a given equivalence is usually quite demanding, therefore it would be desirable to have a general framework for the derivation of formats as well as for proving compositionality for specific languages. To be sufficiently general, such a framework should be parametrized by the process equivalence and by the kind of transition system. It is the purpose of this paper to provide such a framework.

Our approach is based on the categorical framework of *bialgebraic semantics* [30], where process syntax is modeled via algebras, and transition systems are viewed as coal-

gebras. For example, LTSs are coalgebras for the functor $(\mathcal{P}-)^A$ on the category **Set** of sets and functions, where $\mathcal{P}$ is the powerset functor and A a set of labels, and other kinds of transition systems are coalgebras for other functors, called *behaviour* functors in this context. Coalgebras also provide a general and abstract notion of bisimilarity (for more information on the coalgebraic theory of systems, see [26]). As it turns out, SOS specifications in the GSOS format are essentially *distributive laws* of syntax functors over $(\mathcal{P}-)^A$. Moreover, the process of inducing an LTS with a syntactic structure on processes from SOS rules is a special case of an abstract construction, where distributive laws of syntax over behaviour induce *bialgebras*, i.e., coalgebras with algebraic structures on their carriers. Also the fact that GSOS is a congruence format for bisimilarity can be proved at the level of distributive laws. This makes bialgebraic semantics a general framework for deriving congruence formats for bisimilarities, parametrized by the kind of transition systems; it was used to this purpose in [4, 9, 16] for probabilistic, timed and name-passing systems. In this paper, the framework is further parametrized by the notion of process equivalence.

Typically, process equivalences are characterized by *modal logics*. For example, two processes in an LTS are bisimilar if and only if they satisfy the same formulas in Hennessy-Milner logic [15], and fragments of that logic characterize other interesting equivalences on LTSs. Recently [19] we have proposed a categorical generalization of modal logics for coalgebras in arbitrary categories. There, the syntax of a logic is modeled via algebras for an endofunctor, and its semantics via a suitable natural transformation connecting the logic syntax with the process behaviour.

The main contribution of this paper is a combination of the coalgebraic perspective on modal logic taken in [19] with the bialgebraic approach to SOS from [30]. Roughly speaking, to merge a logic and its semantics with a distributive law representing an SOS specification, one should provide a suitable notion of behaviour for the logic, and define a "dual", logical distributive law, where formulas play the role of processes, in a way that reflects the SOS specification. One might think of the logical behaviour as a way to decompose logical formulas over the syntax of processes. Our main result says that if such a logical distributive law exists, then the equivalence characterized by the logic is compositional on the transition system induced by the SOS specification.

For some kinds of logical behaviours, logical distributive laws can be presented as SOS-like inference rules where formulas act for processes, logical operators (modalities) for syntactic constructs, and logical inference operators for transitions. For example, rules:

$$\frac{\phi \dashv \psi \| \sigma}{\langle a \rangle \phi \dashv \langle a \rangle \psi \| \sigma} \qquad \frac{\phi \dashv \psi \| \sigma}{\langle a \rangle \phi \dashv \psi \| \langle a \rangle \sigma} \qquad (2)$$

are used to define a logical distributive law reflecting (1). In particular, the rule on the left says that if a formula ϕ holds for every process of the form $x \| y$ such that ψ holds for x and σ holds for y, then the formula $\langle a \rangle \phi$ holds for every process of the form $z \| w$ such that $\langle a \rangle \psi$ holds for z and σ holds for w. Since $\langle a \rangle \phi$ means that a process can do an a-transition to a process for which ϕ holds, this corresponds to the left rule in (1).

The framework proposed here can be seen as a very general "meta-congruence format", parametrized both by the notion of process equivalence and by the kind of transition system. It can be used directly to prove compositionality for specific languages and equivalences. Obviously it is hard to expect that such a general approach would be as easy to use as syntactic congruence formats designed for specific equivalences, and indeed finding the right logical distributive law and presenting it in a readable form is not always easy. However, our framework can also be used to derive specialized formats by proving that suitable distributive laws exist for a whole class of SOS specifications. The direct application to specific languages can be then left to problematic cases that do not fit in any known format.

The structure of the paper is as follows. The basics of classical SOS and congruence formats are presented in §1. In §2 the bialgebraic approach of [30] is recalled, followed by a brief description of our approach to coalgebraic modal logic [19] in §3. In §4, the main technical result of the paper is obtained by merging the two approaches, and it is illustrated in §5 on some simple examples. Finally, §6 sketches some related and future work. Some familiarity with basic category theory is expected; [2, 22] are good references.

1 SOS and congruence formats

We begin by recalling the classical framework of SOS [1]. A *labelled transition system* (LTS) $(X, A, \longrightarrow)$ is a set $X \ni x, y, \ldots$ of *processes*, a set $A \ni a, b, \ldots$ of *labels*, and a *transition relation* $\longrightarrow \subseteq X \times A \times X$, typically written $x \xrightarrow{a} y$ for $(x, a, y) \in \longrightarrow$; y is then an a-*successor* of x. An LTS is *image finite* if each process has only finitely many successors for each label, and $x \not\longrightarrow$ means that x has no successors.

Various equivalences are defined on processes in an LTS; usually they are characterized by modal logics. For example, on image finite LTSs, finitary *Hennessy-Milner logic* (HML) [15], with syntax:

$$\phi ::= \top \mid \neg\phi \mid \phi \wedge \phi \mid \langle a \rangle \phi$$

where $a \in A$, and with semantics defined on a given LTS by:

$$x \models \langle a \rangle \phi \iff \exists y \in X. \; x \xrightarrow{a} y, \; y \models \phi$$

and by the standard interpretation of propositional connectives, characterizes strong bisimilarity [23]. Fragments of HML have also been considered; see [12] for a survey. For example, the fragment without conjunction or negation characterizes *trace equivalence* on arbitrary LTSs, and the same fragment extended with a constant $\varnothing$ with semantics:

$$x \models \varnothing \iff x \not\longrightarrow$$

characterizes *completed trace equivalence*.

In the context of SOS, processes are closed terms over some algebraic signature, i.e., a set $\Sigma \ni \mathtt{f}, \mathtt{g}, \ldots$ of *operation symbols* with an *arity* function $ar : \Sigma \to \mathbb{N}$, and the transition relation is induced from a set of inference rules. Assuming a fixed set $\Xi \ni \mathtt{x}, \mathtt{y}, \ldots$ of variables, a *positive* (resp. *negative*) *literal* over Σ is an expression of the form $\mathtt{t} \xrightarrow{a} \mathtt{s}$ (resp. $\mathtt{t} \not\xrightarrow{a}$), where $\mathtt{t}$ and $\mathtt{s}$ are terms over Σ with variables from Ξ, and an *inference rule* is an expression $\frac{H}{c}$, where H is a set of literals, called *premises*, and c is a positive literal called the *conclusion*.

Considered in this generality, inference rules do not guarantee the compositionality of any nontrivial process equivalence. Indeed, it is not even clear that they meaningfully induce an LTS. For these reasons, various restricted formats of SOS specifications have been proposed that guarantee these and other desirable properties. The most widely studied format is that of GSOS [7], where only rules of the following form are allowed:

$$\frac{\{\mathtt{x}_i \xrightarrow{a_{ij}} \mathtt{y}_{ij}\}_{1 \le j \le m_i}^{1 \le i \le n} \quad \{\mathtt{x}_i \not\xrightarrow{b_{ik}}\}_{1 \le k \le l_i}^{1 \le i \le n}}{\mathtt{f}(\mathtt{x}_1, \ldots, \mathtt{x}_n) \xrightarrow{c} \mathtt{t}}$$

where $n = ar(\mathtt{f})$, $m_i, l_i \in \mathbb{N}$, $a_{ij}, b_{ik}, c \in A$, $\mathtt{x}_i$ and $\mathtt{y}_{ij}$ are all distinct and no other variables occur in $\mathtt{t}$. A GSOS specification is *image finite* if it contains only finitely many rules for each $\mathtt{f} \in \Sigma$ and $c \in A$. Image finite GSOS specifications induce image finite LTSs in an obvious way, and bisimilarity is guaranteed to be a congruence. Another well-known format is de Simone format for trace equivalence. A considerably more complex format for completed trace equivalence was suggested in [20, 17]. For a detailed study of various congruence formats and their properties, see [1, 14].

2 Bialgebraic operational semantics

LTSs can be seen as functions $h : X \to (\mathcal{P}X)^A$ along the correspondence $y \in h(x)(a) \iff x \xrightarrow{a} y$. In category theory, such functions are called *coalgebras* for the functor $(\mathcal{P}-)^A$ on **Set**. Similarly, image finite LTSs

ar $(\mathcal{P}_\omega -)^A$-coalgebras, where $\mathcal{P}_\omega$ is the finite powerset functor. Several other kinds of transition systems are B-coalgebras for other endofunctors B, called *behaviour functors* in this context, on **Set** or on other categories (see e.g. [26, 10]). A coalgebra morphism from $h : X \to BX$ to $g : Y \to BY$ is a function $f : X \to Y$ such that $g \circ f = Bf \circ h$.

Dually, syntax is traditionally modeled with algebras [13]. An algebraic signature Σ corresponds to a functor $\Sigma X = \coprod_{\mathtt{f} \in \Sigma} X^{ar(\mathtt{f})}$ on **Set**, in the sense that algebras for the signature are exactly Σ-algebras for the functor, i.e, functions $g : \Sigma X \to X$. An algebra morphism from $g : \Sigma X \to X$ to $h : \Sigma Y \to Y$ is a function $f : X \to Y$ such that $f \circ g = h \circ \Sigma f$. Given an algebra $g : \Sigma X \to X$, (kernels of) algebra morphisms from g are called *congruences* on g. For Σ's corresponding to algebraic signatures, they correspond to congruences in the sense of universal algebra.

In LTSs induced from SOS descriptions, processes are closed terms over some signatures. Abstractly, such coalgebras are induced by *distributive laws*, i.e., natural transformations like

$$\lambda : \Sigma B \Longrightarrow B\Sigma \qquad (3)$$

for functors Σ, B on the same category $\mathcal{C}$. Indeed, assuming Σ has an initial algebra $a : \Sigma P \to P$, a B-coalgebra structure on P is defined as the unique algebra morphism:

$$\begin{array}{ccc}
P & \xleftarrow{\quad a \quad} & \Sigma P \\
\scriptstyle h_\lambda \downarrow & & \downarrow \scriptstyle \Sigma h_\lambda \\
BP & \xleftarrow[Ba]{} B\Sigma P \xleftarrow[\lambda_P]{} & \Sigma BP.
\end{array} \qquad (4)$$

The pair (a, h_λ) is then an (initial) λ-*bialgebra*. In a very similar fashion (see [30] for details), such bialgebras are induced also by more general types of distributive laws like

$$\lambda : \Sigma(\mathrm{Id} \times B) \Longrightarrow BT_\Sigma, \text{ or} \qquad (5)$$
$$\lambda : \Sigma D_B \Longrightarrow B(\mathrm{Id} + \Sigma), \qquad (6)$$

where T_Σ is the free monad over Σ and D_B is the cofree comonad over B (assuming they exist), or even more generally by distributive laws of the monad T_Σ over the comonad D_B, i.e., by natural transformations

$$\lambda : T_\Sigma D_B \Longrightarrow D_B T_\Sigma \qquad (7)$$

subject to a few axioms. Laws of type (3), (5) or (6) uniquely induce laws of type (7) by appropriate "recursion theorems" [30].

As was observed in [30], proved in detail in [4], and explained on simple examples in [29, 17, 18], for $B = (\mathcal{P}_\omega -)^A$ and for Σ on **Set** corresponding to an algebraic signature, laws of type (5) correspond to image finite GSOS

specifications, and h_λ is the LTS induced by the corresponding specification; hence (5) is called *abstract GSOS*. In [4, 16, 9], it was shown how this abstract treatment specializes to useful formats for probabilistic, timed or name-passing systems for other choices of B and/or of the underlying category.

Similarly, distributive laws (6) for $B = (\mathcal{P}_\omega -)^A$ correspond to SOS specifications in the *safe ntree format* [30], allowing rules of the form

$$\frac{\{\mathbf{z}_i \xrightarrow{a_i} \mathbf{y}_i\}_{i \in I} \quad \{\mathbf{w}_j \xrightarrow{b_j} \!\!\!\!\!\!/\; \}_{j \in J}}{\mathbf{f}(\mathbf{x}_1, \ldots, \mathbf{x}_n) \xrightarrow{c} \mathbf{t}}$$

where $\mathbf{x}_i$ and $\mathbf{y}_i$ are all distinct and are the only variables occurring in the rule, I and J are countable sets, the graph of positive premises is subject to a well-foundedness condition, $\mathbf{t}$ is either a variable or a term built of a single operation symbol and variables, and the entire specification is again subject to an image finiteness condition. Unlike GSOS, this allows *lookahead* in premises, i.e., rules such as

$$\frac{\mathbf{x} \xrightarrow{a} \mathbf{y} \quad \mathbf{y} \xrightarrow{b} \mathbf{z}}{\mathbf{f}(\mathbf{x}) \xrightarrow{c} \mathbf{g}(\mathbf{z})}$$

are allowed. Distributive laws (6) will be called *abstract safe ntree*.

Laws of type (3), called *abstract toy SOS* in the following, are special cases of both abstract GSOS and abstract safe ntree. Concrete SOS formats defined by (3) are quite restrictive, and not many interesting specifications conform to them. However, the simplicity of abstract toy SOS makes it useful for the presentation of abstract results, since they are usually straightforward to generalize to other types of distributive laws. This includes also the general (7); however, that type of laws has not been yet understood concretely as a syntactic format of SOS specifications.

3 Coalgebraic modal logic

To study HML and other modal logics at the level of generality of distributive laws, we will use the recent approach of [19], inspired by earlier results of [21, 24, 27]. Assume a category $\mathcal{C}$ of structures of processes, and a category $\mathcal{D}$ of structures of logical formulas, connected by an adjunction $F \dashv G^{op} : \mathcal{C} \to \mathcal{D}^{op}$. This means that a bijection $\mathcal{C}(X, G\Phi) \cong \mathcal{D}(\Phi, FX)$ holds for any $X \in \mathcal{C}$, $\Phi \in \mathcal{D}$; slightly abusing the notation, we will denote both directions of this bijection by $-^\flat$. To avoid notational clutter, all *op*-notation for functors and natural transformations is omitted in the following; formally, we see F and G as contravariant functors between $\mathcal{C}$ and $\mathcal{D}$, and compose them with (covariant) functors on $\mathcal{C}$ or $\mathcal{D}$ in the obvious way. In all concrete examples considered in this paper, $\mathcal{C} = \mathcal{D} = \mathbf{Set}$ and $F = G = 2^-$, where $2 = \{\mathtt{tt}, \mathtt{ff}\}$.

Functors F and G provide the infrastructure for linking processes and formulas. Note that GF is a monad on $\mathcal{C}$; denote its unit by η. For any $f : \Phi \to FX$ in $\mathcal{D}$, one has $f^\flat = Gf \circ \eta_X$. Also FG is a monad on $\mathcal{D}$, with the unit denoted by ϵ.

Assuming a functor B on $\mathcal{C}$, a (*coalgebraic polyadic modal*) *logic* for B-coalgebras is a functor L on $\mathcal{D}$ (the *syntax*) together with a connection between L and B, i.e., a natural transformation $\rho : LF \implies FB$ (the *semantics*). Such a ρ determines the adjoint connection $\rho^\star = GL\epsilon \circ G\rho G \circ \eta_{BG} : BG \implies GL$; it is not difficult to see that the correspondence between ρ and $\rho^\star$ is bijective.

If L has an initial algebra $a : L\Phi \to \Phi$, then for any coalgebra $h : X \to BX$ the interpretation $\llbracket \_ \rrbracket_h : \Phi \to FX$ is defined as the unique algebra morphism:

$$
\begin{array}{ccc}
\Phi & \xleftarrow{\quad a \quad} & L\Phi \\
{\scriptstyle \llbracket \_ \rrbracket_h} \Big\downarrow & & \Big\downarrow {\scriptstyle L\llbracket \_ \rrbracket_h} \\
FX \xleftarrow{Fh} FBX & \xleftarrow{\rho_X} & LFX,
\end{array} \qquad (8)
$$

and the transpose $\llbracket \_ \rrbracket_h^\flat : X \to G\Phi$ represents the logical equivalence associated with (L, ρ).

Example 1. The logic for completed trace equivalence on finitely branching LTSs, i.e., on B-coalgebras for $B = (\mathcal{P}_\omega -)^A$ on **Set**, is defined by syntax:

$$L\Phi = \{\top\} + \{\varnothing\} + A \times \Phi$$

on **Set**, with semantics $\rho_X : L2^X \to 2^{BX}$ defined by cases:

$$
\begin{array}{rcl}
\rho_X(\top)(\beta) & = & \mathtt{tt} \quad \text{always} \\
\rho_X(\varnothing)(\beta) & = & \mathtt{tt} \iff \forall a \in A.\ \beta(a) = \emptyset \\
\rho_X(\langle a \rangle \phi)(\beta) & = & \mathtt{tt} \iff \exists y \in \beta(a).\ \phi(y) = \mathtt{tt}.
\end{array}
$$

It is easy to see how L corresponds to the syntax of the logic for completed trace equivalence and ρ to its semantics. Indeed, for any B-coalgebra h, the map $\llbracket \_ \rrbracket_h$ defined by (8) is the usual semantics of the logic for completed traces, and the kernel of $\llbracket \_ \rrbracket_h^\flat$ is completed trace equivalence on h.

4 Logical distributive laws

A logic (L, ρ) for B-coalgebras lifts B to an endofunctor B^ρ on the category $(\mathcal{D} \downarrow F)$, i.e., the slice category of the contravariant adjunction of F and G. Objects of $(\mathcal{D} \downarrow F)$ are triples (X, r, Φ) where $X \in \mathcal{C}$, $\Phi \in \mathcal{D}$ and $r : \Phi \to FX$ in $\mathcal{D}$, and a morphism $(f, g) : (X, r, \Phi) \to (Y, s, \Psi)$ is a pair of maps $f : X \to Y$, $g : \Psi \to \Phi$ such that $Ff \circ s = r \circ g$. The functor B^ρ on $(\mathcal{D} \downarrow F)$ is defined by:

$$
\begin{array}{rcl}
B^\rho(X, r, \Phi) & = & (BX, \rho_X \circ Lr, L\Phi) \\
B^\rho(f, g) & = & (Bf, Lg)
\end{array}
$$

and a B^ρ-coalgebra is a B-coalgebra together with an L-algebra interpreted in it according to ρ.

The above suggests that, in a sense, coalgebraic modal logic is a special case of the study of coalgebras, and to combine it with the bialgebraic approach to SOS one should interpret the latter in $(\mathcal{D} \downarrow F)$. To simplify the presentation, we do it first for abstract toy SOS (3), and then show without proof how the approach applies to the more general types of distributive laws (5) and (7).
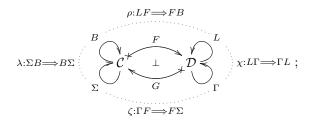
4.1 Abstract toy SOS

Assume that a syntax functor Σ on $\mathcal{C}$ is lifted to a functor Σ^ζ with a functor Γ on $\mathcal{D}$ and a transformation $\zeta : \Gamma F \Longrightarrow F\Sigma$, just as B is lifted to B^ρ with L and ρ. Some calculation shows that a distributive law $(\lambda, \chi) : \Sigma^\zeta B^\rho \Longrightarrow B^\rho \Sigma^\zeta$ is a pair of laws in $\mathcal{C}$ and $\mathcal{D}$:

$$\lambda : \Sigma B \Longrightarrow B\Sigma \qquad \chi : L\Gamma \Longrightarrow \Gamma L$$

such that the hexagon

$$
\begin{array}{ccc}
L\Gamma F \xRightarrow{L\zeta} LF\Sigma \xRightarrow{\rho\Sigma} FB\Sigma \\
\chi F \Big\Downarrow \qquad\qquad \Big\Downarrow F\lambda \\
\Gamma LF \underset{\Gamma\rho}{\Longrightarrow} \Gamma FB \underset{\zeta B}{\Longrightarrow} F\Sigma B
\end{array}
\qquad (9)
$$

commutes. The following informal picture shows categories, functors and natural transformations involved in this distributive law:

the contravariance of F and G is marked with crossed arrow tails. Σ, B and λ model a language syntax, behaviour and an SOS specification, as described in §2. L and ρ model a modal logic for B-coalgebras, as described in §3. The following theorem says that if the remaining ingredients Γ, ζ and χ can be found, then the logical equivalence induced by the logic on the transition system h_λ induced from the SOS specification is a congruence.

Theorem 2. *Under the above notation, for given Σ, B, λ, L and ρ, if Σ and L have initial algebras and if some Γ, ζ and χ exist such that (9) holds, then $[\![-]\!]^\flat_{h_\lambda}$ is a congruence, i.e., a Σ-algebra morphism from the initial Σ-algebra.*

Proof. Initial algebras $a_\Sigma : \Sigma P \to P$ and $a_L : L\Phi \to \Phi$ induce initial λ- and χ-bialgebras as in (4):
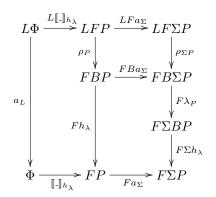
$$
\begin{array}{ccc}
\Sigma P \xrightarrow{\Sigma h_\lambda} \Sigma BP & & L\Gamma\Phi \xleftarrow{Lh_\chi} L\Phi \\
\Big\downarrow a_\Sigma \quad \Big\downarrow \lambda_P & & \chi_\Phi \Big\downarrow \quad \Big\downarrow a_L \\
\quad\quad B\Sigma P & & \Gamma L\Phi \\
\quad\quad \Big\downarrow Ba_\Sigma & & \Big\downarrow \Gamma a_L \\
P \dashrightarrow BP & & \Gamma\Phi \xleftarrow{\quad} \Phi. \\
\quad h_\lambda & & \quad h_\chi
\end{array}
\qquad (10)
$$

Then $[\![-]\!]_{h_\lambda}$ is a "twisted coalgebra morphism" as below:

$$
\begin{array}{ccc}
\Gamma FP \xrightarrow{\zeta_P} F\Sigma P \xleftarrow{Fa_\Sigma} FP \\
\Gamma[\![-]\!]_{h_\lambda} \Big\uparrow \qquad\qquad \Big\uparrow [\![-]\!]_{h_\lambda} \\
\Gamma\Phi \xleftarrow{\quad h_\chi \quad} \Phi.
\end{array}
\qquad (11)
$$

This is proved by L-induction, as both sides of this diagram are algebra morphisms from the initial L-algebra to $F(\lambda_P \circ \Sigma h_\lambda) \circ \rho_{\Sigma P} : LF\Sigma P \to F\Sigma P$. Indeed, in the diagram

the left part is (8), the upper right part commutes by the naturality of ρ, and the lower right part is the left diagram in (10) mapped along F. On the other hand, in the diagram

the left part is the diagram on the right in (10), the upper middle part commutes by the naturality of χ, the lower middle part is (8) mapped along Γ, the upper right part is (9), and the lower right part commutes by the naturality of ζ.

Mapped along G, (11) is the upper right part of the following diagram, where the upper left part commutes by the naturality of η, the lower left part by general properties of adjunctions, and the lower right part is the naturality of $\zeta^\star$:

$$
\begin{array}{ccccc}
P & \xrightarrow{\ \eta_P\ } & GFP & \xrightarrow{\ G[\![\_]\!]_{h_\lambda}\ } & G\Phi \\
\uparrow{\scriptstyle a_\Sigma} & & \uparrow{\scriptstyle GFa_\Sigma} & & \uparrow{\scriptstyle Gh_\chi} \\
 & & GF\Sigma P & & \\
 & \nearrow{\scriptstyle \eta_{\Sigma P}} & \downarrow{\scriptstyle G\zeta_P} & & \\
\Sigma P & & G\Gamma FP & \xrightarrow{\ G\Gamma[\![\_]\!]_{h_\lambda}\ } & G\Gamma\Phi \\
 & \searrow{\scriptstyle \Sigma\eta_P} & \uparrow{\scriptstyle \zeta^\star_{FP}} & & \uparrow{\scriptstyle \zeta^\star_\Phi} \\
 & & \Sigma GFP & \xrightarrow[\ \Sigma G[\![\_]\!]_{h_\lambda}\ ]{} & \Sigma G\Phi
\end{array}
$$

Thus $[\![\_]\!]^\flat_{h_\lambda} = G[\![\_]\!]_{h_\lambda} \circ \eta_P$ is a Σ-algebra morphism from a_Σ. $\qquad\square$

4.2 Abstract GSOS and further

To generalize the framework of §4.1 to distributive laws λ of type (5), some technicalities are necessary. Assume both $\mathcal{C}$ and $\mathcal{D}$ have products and coproducts. A connection $\rho : LF \Longrightarrow FB$ induces a connection between the cofree copointed functor over B and the free pointed functor over L:

$$\rho^+ : (\mathrm{Id} + L)F \Longrightarrow F(\mathrm{Id} \times B).$$

To define it, define its adjoint

$$\rho^{+\star} : (\mathrm{Id} \times B)G \Longrightarrow G(\mathrm{Id} + L) = G \times GL$$

by $\rho^{+\star} = \mathrm{id} \times \rho^\star$.

Further, assume that Σ freely generates a monad T_Σ on $\mathcal{C}$, i.e., that $T_\Sigma X$ is the carrier of an initial $(X + \Sigma-)$-algebra, and that Γ cofreely generates a comonad D_Γ on $\mathcal{D}$, i.e., that $D_\Gamma\Phi$ is the carrier of a final $(\Phi \times \Gamma-)$-coalgebra. Then $\zeta : \Gamma F \Longrightarrow F\Sigma$ induces a connection $\zeta^\sharp : D_\Gamma F \Longrightarrow FT_\Sigma$. To define it, define its adjoint $\zeta^{\sharp\star} : T_\Sigma G \Longrightarrow GD_\Gamma$ from $\zeta^\star : \Sigma G \Longrightarrow G\Gamma$ pointwise, by a straightforward induction in $\mathcal{C}$.

Theorem 2 can be generalized to distributive laws $\lambda : \Sigma(\mathrm{Id} \times B) \Longrightarrow BT_\Sigma$ as follows: for any $\rho : LF \Longrightarrow FB$, if Γ on $\mathcal{D}$, $\zeta : \Gamma F \Longrightarrow F\Sigma$ and $\chi : LD_\Gamma \Longrightarrow \Gamma(\mathrm{Id} + L)$ exist such that

$$
\begin{array}{ccccc}
LD_\Gamma F & \xRightarrow{\ L\zeta^\sharp\ } & LFT_\Sigma & \xRightarrow{\ \rho T_\Sigma\ } & FBT_\Sigma \\
{\scriptstyle \chi F}\Big\Downarrow & & & & \Big\Downarrow{\scriptstyle F\lambda} \\
\Gamma(\mathrm{Id}+L)F & \xRightarrow[\ \Gamma\rho^+\ ]{} & \Gamma F(\mathrm{Id}\times B) & \xRightarrow[\ \zeta(\mathrm{Id}\times B)\ ]{} & F\Sigma(\mathrm{Id}\times B)
\end{array}
$$

$$(12)$$

(compare with (9)) commutes, then $[\![\_]\!]^\flat_{h_\lambda}$ is a Σ-algebra morphism from the initial Σ-algebra. The proof of this proceeds as for Theorem 2. Note that while λ is generalized to abstract GSOS, the logical distributive law χ needs to be generalized to abstract safe ntree.

Further, the theorem can be generalized to type (7). This time, for any distributive law λ of the free monad T_Σ over the cofree comonad D_B, and for any $\rho : LF \Longrightarrow FB$, one requires that Γ on $\mathcal{D}$, $\zeta : \Gamma F \Longrightarrow F\Sigma$ and a distributive law χ of the free monad T_L over the cofree comonad D_Γ exist such that

$$
\begin{array}{ccccc}
T_L D_\Gamma F & \xRightarrow{\ T_L\zeta^\sharp\ } & T_L FT_\Sigma & \xRightarrow{\ \rho^\sharp T_\Sigma\ } & FD_B T_\Sigma \\
{\scriptstyle \chi F}\Big\Downarrow & & & & \Big\Downarrow{\scriptstyle F\lambda} \\
D_\Gamma T_L F & \xRightarrow[\ D_\Gamma\rho^\sharp\ ]{} & D_\Gamma FD_B & \xRightarrow[\ \zeta^\sharp D_B\ ]{} & FT_\Sigma D_B
\end{array}
\qquad (13)
$$

commutes, where $\rho^\sharp$ is defined from ρ by analogy to $\zeta^\sharp$ above. This result is of little practical importance now, as the type (7) has not yet been understood as a concrete syntactic SOS format. However, its proof is entirely analogous to that in §4.1, and the more specific results concerning abstract toy SOS and abstract GSOS above can be obtained from it via recursion theorems such as those used in [30].

5 Examples

In this section the framework developed in §4 is illustrated on three simple examples, aimed at explaining the workings of logical distributive laws rather than at exploring the full scope of our approach. First, a very simple example of an SOS specification in the abstract toy SOS format is explained in detail. The two remaining examples are described rather more concisely due to lack of space: first, trace equivalence is proved compositional for a subset of CCS; then, completed trace equivalence is proved compositional for a language with binary Kleene star, which does not conform to any previously known congruence format for completed traces.

In all examples in this section $\mathcal{C} = \mathcal{D} = \mathbf{Set}$, $F = G = 2^-$ and $B = (\mathcal{P}_\omega-)^A$, for a fixed set A of labels. For simplicity we assume A to be finite, although all examples work with little change for an infinite A.

5.1 A toy SOS specification

Consider a tiny language with synchronous product and no sequential composition, with syntax and semantics de-

scribed by:

$$t ::= \mathtt{nil} \mid a \mid t \otimes t$$

$$\frac{}{a \xrightarrow{\;a\;} \mathtt{nil}} \qquad \frac{\mathtt{x} \xrightarrow{\;a\;} \mathtt{x}' \quad \mathtt{y} \xrightarrow{\;a\;} \mathtt{y}'}{\mathtt{x} \otimes \mathtt{y} \xrightarrow{\;a\;} \mathtt{x}' \otimes \mathtt{y}'} \qquad (14)$$

where a ranges over A. Categorically, the syntax is modeled by the functor

$$\Sigma X = \{\mathtt{nil}\} + A + X \times X$$

on **Set**, and the rules specify a distributive law $\lambda : \Sigma B \implies B\Sigma$. More specifically, for any set X, λ is defined by cases:

$$\lambda_X(\mathtt{nil})(b) = \emptyset$$
$$\lambda_X(a)(b) = \begin{cases} \{\mathtt{nil}\} & \text{if } a = b \\ \emptyset & \text{otherwise} \end{cases}$$
$$\lambda_X(\beta_1 \otimes \beta_2)(b) = \{\, x \otimes y \mid \ x \in \beta_1(b), y \in \beta_2(b) \,\}$$

where $a, b \in A$ and $\beta_1, \beta_2 \in BX = (\mathcal{P}_\omega X)^A$. Clearly λ is natural in X.

We will apply the framework of §4 to prove that trace equivalence is compositional for this language. The compositionality result is hardly interesting in itself (and indeed easy to prove without any advanced techniques), but it should be useful to explain our approach on such a very simple instance of abstract toy SOS.

The logic for trace equivalence on B-coalgebras (image finite LTSs) is defined as in Example 1, but with syntax restricted to

$$L\Phi = \{\top\} + A \times \Phi.$$

For the required compositionality result, Theorem 2 requires a functor Γ on **Set** and transformations $\zeta : \Gamma 2^- \to 2^{\Sigma^-}$ and $\chi : L\Gamma \implies \Gamma L$ such that (9) commutes.

To illustrate the role of Γ and explain the process of finding ζ and χ, we begin with a very simple and natural (although, as we shall see, wrong) choice, where $\Gamma = \Sigma$ and ζ is defined as follows:

$$\zeta_X(\mathtt{nil})(t) = \mathtt{tt} \iff t = \mathtt{nil}$$
$$\zeta_X(a)(t) = \mathtt{tt} \iff t = a$$
$$\zeta_X(\phi_1 \otimes \phi_2)(t) = \mathtt{tt} \iff t = x_1 \otimes x_2, \phi_i(x_i) = \mathtt{tt}$$

along the lines of §3. Constructors $\mathtt{nil}$, a and $\otimes$ used here will be called *spatial modalities*, as opposed to *behavioural* modalities $\top$ and $\langle a \rangle$ used in the definition of L. Intuitively, formulas built from spatial modalities can check the structure of Σ-terms.

One might now attempt to define a distributive law $\chi : L\Gamma \implies \Gamma L$ such that (9) commutes. Since both L and Γ are polynomial functors, such a law can be defined by cases, separately for each combination of spatial and behavioural

modalities. Then (9) can also be proved by cases. For example, consider the following partial definition of χ:

$$\chi_\Phi(\langle a \rangle(\phi_1 \otimes \phi_2)) = (\langle a \rangle \phi_1) \otimes (\langle a \rangle \phi_2).$$

Note that both the argument on the left side and the right side of this equation have a simple intuitive meaning: the former says "the process can do an a-step to a process of the form $y_1 \otimes y_2$ such that ϕ_1 holds for y_1 and ϕ_2 holds for y_2", and the latter says "the process is of the form $x_1 \otimes x_2$, x_1 can do an a-step to a process for which ϕ_1 holds, and x_2 can do an a-step to a process for which ϕ_2 holds". A quick look on (14) should convince anyone that these conditions are equivalent; formally, the corresponding case of (9) commutes. Indeed, to check the equation

$$2^{\lambda_X}(\rho_{\Sigma X}(L\zeta_X(\langle a \rangle(\phi_1 \otimes \phi_2))))(t) = \\ \zeta_{BX}(\Gamma\rho_X(\chi_{2^X}(\langle a \rangle(\phi_1 \otimes \phi_2))))(t)$$

for a given X, $a \in A$, $\phi_1, \phi_2 \in 2^X$ and $t \in \Sigma BX$, it is enough to unfold the definitions of ρ, ζ and χ to obtain the equivalent condition:

$$\exists r \in (\lambda_X(t))(a). \, r = y_1 \otimes y_2, \phi_i(x_i) = \mathtt{tt} \iff \\ t = \beta_1 \otimes \beta_2, \exists y_i \in \beta_i(a). \, \phi_i(y_i) = \mathtt{tt}$$

which follows directly from the definition of λ.

Unfortunately, other cases of χ are harder to define. Already the simple behavioural modality $\top$ is problematic: for $\chi_\Phi(\top)$ one would like an element of $\Gamma L\Phi$ that would represent the always true condition. This is, however, impossible with our initial choice of Γ: every test in $\Gamma L\Phi$ imposes some syntactic condition on the tested process. A simple attempt to overcome this problem would be to add a single constant, always true modality $\mathbb{T}$ to Γ, with ζ extended by:

$$\zeta_X(\mathbb{T})(t) = \mathtt{tt} \text{ always.}$$

Then, however, it becomes unclear what $\chi_\Phi(\langle a \rangle \mathbb{T})$ should be. To formalize the condition "the process can do an a-step" as an element of $\Gamma L\Phi$, one would like to write something like:

$$\chi_\Phi(\langle a \rangle \mathbb{T}) = a \lor (\langle a \rangle \top) \otimes (\langle a \rangle \top);$$

this is, however, forbidden as Γ does not allow one to write anything like logical disjunction in spatial modalities.

A solution to this problem is to extend Γ more substantially by closing spatial modalities under finite disjunctions. This amounts to taking $\Gamma = \mathcal{P}_\omega \Sigma$ with $\zeta : \Gamma 2^- \implies 2^{\Sigma^-}$ defined by:

$$\zeta_X(\gamma)(\mathtt{nil}) = \mathtt{tt} \iff \mathtt{nil} \in \gamma$$
$$\zeta_X(\gamma)(a) = \mathtt{tt} \iff a \in \gamma$$
$$\zeta_X(\gamma)(x_1 \otimes x_2) = \mathtt{tt} \iff \exists \phi_1 \otimes \phi_2 \in \gamma. \, \phi_i(x_i) = \mathtt{tt}.$$

One could then define χ with a set of equations as before, writing for example

$$\chi_\Phi(\top) = \mathtt{nil} \vee (\top \otimes \top) \vee \bigvee_{a \in A} a$$
$$\chi_\Phi(\langle a \rangle (\mathtt{nil} \vee (\phi_1 \otimes \phi_2))) = a \vee (\langle a \rangle \phi_1 \otimes \langle a \rangle \phi_2)$$

and so forth. However, the theory of GSOS in §2 provides another, more elegant method of presenting such distributive laws: inference rules. Note that the functor $\mathcal{P}_\omega \Sigma$ is rather similar to $B = (\mathcal{P}_\omega -)^A$, and it is reasonable to expect that χ could be presented in a manner similar to GSOS rules. Two differences between $\mathcal{P}_\omega \Sigma$-coalgebras and B-coalgebras are that in the former there is no labelled component in transitions, and successors are Σ-terms rather than simple elements. This suggests that instead of literals like $\mathtt{x} \xrightarrow{a} \mathtt{y}$, in rules for χ one should use literals such as $\mathtt{x} \longrightarrow \mathtt{y} \otimes \mathtt{z}$. To distinguish the logical rules from the SOS ones, we will use variables like ϕ, ψ, instead of $\mathtt{x}, \mathtt{y}$, and we will replace the sign $\longrightarrow$ with $\dashv$. Now the following rules:

$$\overline{\top \dashv \mathtt{nil}} \qquad \overline{\top \dashv a} \qquad \overline{\top \dashv \top \otimes \top}$$

$$\frac{\phi \dashv \mathtt{nil}}{\langle a \rangle \phi \dashv a} \qquad \frac{\phi \dashv \psi \otimes \sigma}{\langle a \rangle \phi \dashv ((\langle a \rangle \psi) \otimes (\langle a \rangle \sigma))}$$

define a distributive law $\chi : L\Gamma \implies \Gamma L$ just as (14) defined λ above. Note that behavioural modalities $\top$, $\langle a \rangle$ play the role of *syntax* here, and spatial modalities $\mathtt{nil}$, a and $\otimes$ are a part of the behaviour. The sign $\dashv$ might be read "is guaranteed by"; this is justified by the definition of ζ.

It turns out that for this χ the condition (9) holds. This is formally proved by a rather tedious calculation; however, it is not difficult to convince oneself, looking at (14), that e.g., if a property ϕ holds for all processes $x \otimes y$ such that ψ holds for x and σ holds for y, then $\langle a \rangle \phi$ holds for all processes $z \otimes w$ such that $\langle a \rangle \psi$ holds for z and $\langle a \rangle \sigma$ holds for w, which means that the last rule above is correct.

5.2 A step towards GSOS

For a slightly more complex, but still very simple example, consider the subset of CCS [23] with action prefixing, nondeterministic choice and parallel composition, with syntax and semantics defined by:

$$t ::= \mathtt{nil} \mid a.t \mid t + t \mid t || t$$

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

$$\frac{\mathtt{x} \xrightarrow{a} \mathtt{x}'}{\mathtt{x} || \mathtt{y} \xrightarrow{a} \mathtt{x}' || \mathtt{y}} \qquad \frac{\mathtt{y} \xrightarrow{a} \mathtt{y}'}{\mathtt{x} || \mathtt{y} \xrightarrow{a} \mathtt{x} || \mathtt{y}'} \qquad \frac{\mathtt{x} \xrightarrow{a} \mathtt{x}' \quad \mathtt{y} \xrightarrow{\bar{a}} \mathtt{y}}{\mathtt{x} || \mathtt{y} \xrightarrow{\tau} \mathtt{x}' || \mathtt{y}'}$$

where a ranges over A, assuming that $A = A_0 + \{\bar{\alpha} \mid \alpha \in A_0\} + \{\tau\}$ for some set A_0, and $\bar{\alpha}$ means α. The functor on **Set** corresponding to the above syntax is

$$\Sigma X = \{\mathtt{nil}\} + A \times X + X \times X + X \times X,$$

and it is not difficult to see how the rules define a distributive law similarly as in §5.1, but of the more complex form:

$$\lambda : \Sigma(\mathrm{Id} \times B) \implies B(\mathrm{Id} + \Sigma). \qquad (15)$$

Note that λ is not a special case of abstract toy SOS (3), but it lies in the intersection of abstract GSOS (5) and abstract safe ntree (6). Indeed, the above rules are both in GSOS and safe ntree formats.

To prove the compositionality of trace equivalence for this language, consider L and ρ as in §5.1. A version of Theorem 2 for laws as in (15) (not stated in §4, but a straightforward simplification of that from §4.2) requires a functor $\Gamma, \zeta : \Gamma 2^- \implies 2^{\Sigma-}$ and $\chi : L(\mathrm{Id} \times \Gamma) \implies \Gamma(\mathrm{Id} + L)$ such that a corresponding version of (9) holds. Take $\Gamma = \mathcal{P}_\omega \Sigma$ and define ζ in analogy with §5.1, and let χ be defined by the following rules:

$$\overline{\top \dashv \mathtt{nil}} \qquad \overline{\top \dashv a.\top} \qquad \overline{\top \dashv \top + \top} \qquad \overline{\top \dashv \top || \top}$$

$$\overline{\langle a \rangle \phi \dashv a.\phi} \qquad \overline{\langle a \rangle \phi \dashv \langle a \rangle \phi + \top} \qquad \overline{\langle a \rangle \phi \dashv \top + \langle a \rangle \phi}$$

$$\frac{\phi \dashv \psi || \sigma}{\langle a \rangle \phi \dashv \langle a \rangle \psi || \sigma} \qquad \frac{\phi \dashv \psi || \sigma}{\langle a \rangle \phi \dashv \psi || \langle a \rangle \sigma} \qquad \frac{\phi \vdash \psi || \sigma}{\langle \tau \rangle \phi \dashv \langle a \rangle \psi || \langle \bar{a} \rangle \sigma}$$

Another tedious calculation shows that the corresponding version of (9) holds. Again, rather that perform the formal calculation it is easier to convince oneself that the logical rules reflect the SOS rules of our language.

Note that χ is not of the abstract toy SOS type $L\Gamma \implies \Gamma L$. For example, in the first rule for $\langle a \rangle$ above, the variable ϕ from the left side of the conclusion is used on the right side of the conclusion (hence at least $L(\mathrm{Id} \times \Gamma)$ is needed in the domain of χ), and it is not put under any behavioural modality (hence at least $\Gamma(\mathrm{Id} + L)$ is needed in the codomain of χ). This corresponds to the reason why λ is not of the type $\Sigma B \implies B\Sigma$: look at the first SOS rule for a.

5.3 GSOS: Kleene star and completed trace equivalence

Consider a language with sequential composition and (binary) Kleene star, with syntax defined by the following grammar:

$$t ::= \mathtt{nil} \mid a \mid t;t \mid t * t$$

and semantics by the rules:

$$\frac{}{a \xrightarrow{a} \mathtt{nil}} \qquad \frac{\mathtt{x} \xrightarrow{a} \mathtt{x}'}{\mathtt{x};\mathtt{y} \xrightarrow{a} \mathtt{x}';\mathtt{y}} \qquad \frac{\{\mathtt{x} \xslashed{b} \}_{b \in A} \quad \mathtt{y} \xrightarrow{a} \mathtt{y}'}{\mathtt{x};\mathtt{y} \xrightarrow{a} \mathtt{y}'}$$

$$\frac{\mathtt{x} \xrightarrow{a} \mathtt{x}'}{\mathtt{x} * \mathtt{y} \xrightarrow{a} \mathtt{x}';(\mathtt{x} * \mathtt{y})} \qquad \frac{\mathtt{y} \xrightarrow{a} \mathtt{y}'}{\mathtt{x} * \mathtt{y} \xrightarrow{a} \mathtt{y}'}$$

where a ranges over A. The syntax is modeled by

$$\Sigma X = \{\mathtt{nil}\} + A + X \times X + X \times X$$

on **Set**. The distributive law defined by the rules is not in the form of (15), since a complex term is used in the conclusion of the first rule for $*$. However, since the rules are in the GSOS format, the law is of the form (5).

Consider the logic for completed trace equivalence with L and ρ as in Example 1. According to §4.2, to prove that completed trace equivalence is compositional one needs to find Γ, ζ and $\chi : LD_\Gamma \Longrightarrow \Gamma(\mathrm{Id} + L)$ such that (12) holds.

A tempting choice is $\Gamma = \mathcal{P}_\omega \Sigma$ which worked so well in §5.1 and §5.2, with an analogous definition of ζ interpreting the finite powerset construction as finite disjunctions of spatial modalities. In this example however, this choice does not work. Indeed, for a logical rule for the modality $\langle a \rangle$ that would reflect the first SOS rule for $*$, one is tempted to write something like:

$$\frac{\phi \dashv \psi;\sigma \quad \sigma \dashv \kappa * \theta}{\langle a \rangle \phi \dashv (\langle a \rangle \psi \wedge \kappa) * \theta}$$

since the variable x is indirectly duplicated in the conclusion of the SOS rule. This is, however, forbidden: the structure of L and Γ does not allow any use of conjunctions. An obvious solution is to extend Γ and allow finite conjunctions on the left side of the spatial modality $*$. Formally, consider

$$\Gamma \Phi = \mathcal{P}_\omega (1 + A + \Phi \times \Phi + \mathcal{P}_\omega \Phi \times \Phi)$$

with ζ defined by analogy to §5.1, with one difference:

$$\zeta(\gamma)(x * y) = \mathtt{tt}$$
$$\Updownarrow$$
$$\exists (\delta * \phi) \in \gamma.(\phi(y) = \mathtt{tt} \wedge \forall \psi \in \delta.\ \psi(x) = \mathtt{tt})$$

where the universal quantifier justifies the understanding of the inner powerset in Γ as conjunction. Now χ can be defined by the following rules:

$$\frac{}{\top \dashv \mathtt{nil}} \qquad \frac{}{\top \dashv a} \qquad \frac{}{\top \dashv \top;\top} \qquad \frac{}{\top \dashv \top * \top}$$

$$\frac{}{\varnothing \dashv \mathtt{nil}} \qquad \frac{}{\varnothing \dashv \varnothing;\varnothing} \qquad \frac{}{\varnothing \dashv \varnothing * \varnothing}$$

$$\frac{\phi \dashv \mathtt{nil}}{\langle a \rangle \phi \dashv a} \qquad \frac{\phi \dashv \psi;\sigma}{\langle a \rangle \phi \dashv (\langle a \rangle \psi);\sigma} \qquad \frac{}{\langle a \rangle \phi \dashv \varnothing;\langle a \rangle \phi}$$

$$\frac{\phi \dashv \psi;\sigma \quad \sigma \dashv K * \theta}{\langle a \rangle \phi \dashv (\langle a \rangle \psi \wedge K) * \theta} \qquad \frac{}{\langle a \rangle \phi \dashv \top * \langle a \rangle \phi}$$

where a ranges over A and K is a special variable denoting an arbitrary finite conjunction of formulas. The use of such variables in rules is justified by the structure of Γ, a little more complex than in the examples before. Again, a rather tedious calculation shows that (12) holds for this choice of χ, hence completed trace equivalence is compositional for our language. Here the calculation is a bit more complex than in previous examples, as it involves the derived ρ^+ and the inductively defined $\zeta^\sharp$.

Note that the logical rules above are not in the abstract GSOS format, as the first rule in the last row involves lookahead. However, they are in the abstract safe ntree format. This corresponds to the fact that SOS rules for our language are not in safe ntree format, as the first rule for $*$ has a complex term in the conclusion; however, they are in the GSOS format. Note how the lookahead in the logical rule stems from the complex conclusion in the SOS rule.

Note also that the behavioural modality $\varnothing$ is necessarily used in a logical rule for the modality $\langle a \rangle$. This suggests that there is no logical distributive law for the logic for traces for our language. Indeed, trace equivalence is not compositional for the sequential composition operator ;.

In all examples so far, a wrong choice of Γ was corrected by extending it with additional constructs. However, Γ can also be overextended. For example, if the choice above was extended to allow finite conjunctions on the left side of the spatial modality ;:

$$\Gamma \Phi = \mathcal{P}_\omega (1 + A + \mathcal{P}_\omega \Phi \times \Phi + \mathcal{P}_\omega \Phi \times \Phi),$$

then one would have to replace the middle logical rule in the third row above with a rule like:

$$\frac{\phi \dashv K;\sigma}{\langle a \rangle \phi \dashv (\langle a \rangle K);\sigma}.$$

This is, however, forbidden, since the logical conjunction represented by K is a part of a spatial modality and it cannot occur under the behavioural modality $\langle a \rangle$ in the conclusion of the rule. This shows that Γ should be chosen with care, and there might be no easy and general way to choose it.

6 Related and future work

The approach presented here is a refined and extended version of the framework of *test suites* [17], also aimed at deriving congruence formats. There, distributive laws are defined between fibred functors derived from notions of process equivalences, in total categories of certain fibrations. When $\mathcal{C} = \mathcal{D} = \mathbf{Set}$ and $F = G = \mathcal{V}^-$ for some set $\mathcal{V}$ of truth values, the present framework coincides with that of **Set**; note that $(\mathcal{D} \downarrow F)$ is always fibred over $\mathcal{C}$. However, in other cases the present approach provides a better treatment of equivalences. Moreover, it provides a clear connection to modal logic, and the presentation of distributive laws as SOS-like rules makes the framework easier to apply.

Some existing work on specific SOS formats and their properties can be rephrased in terms of the present framework. For example, the technique of frozen/liquid positions [6] used to derive formats for decorated trace equivalences corresponds exactly to extending the functors Γ with finite conjunctions as in in §5.3. More interestingly, the SOS-like presentation of logical distributive laws suggests a connection to compositional proof systems as in [28] and to techniques for modal logic decomposition as in [11]. Also, the notion of spatial modality used here seems to be related to spatial logics for process calculi as in [3, 8]. The precise nature of these connections needs to be studied.

Several other problems are left open. Importantly, some guidelines for finding the right logical behaviour Γ are much needed instead of informal guessing used in §5. It is also unfortunate that the crucial compositionality condition (9) is defined only abstractly, and it is not properly explained at the level of SOS and logical rules. Such a concrete explanation would make the entire framework much easier to use. Also, a treatment of quantitative logics, where metric spaces are used instead of equivalences, is missing. Last but not least, more examples involving various equivalences and kinds of transition systems need to be developed; examples of systems based on categories other than **Set** include name-passing systems interpreted in presheaf categories [9, 19].

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] J. Adámek, H. Herrlich, and G. E. Strecker. *Abstract and Concrete Categories*. Wiley-Interscience, 1990.

[3] H. R. Andersen, C. Stirling, and G. Winskel. A compositional proof system for the modal μ-calculus. In *Procs. LICS'94*, pages 144–153. IEEE Computer Society Press, 1994.

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

[5] J. A. Bergstra, A. Ponse, and S. Smolka. *Handbook of Process Algebra*. Elsevier, 2002.

[6] B. Bloom, W. J. Fokkink, and R. J. van Glabbeek. Precongruence formats for decorated trace semantics. *ACM Transactions on Computational Logic*, 5:26–78, 2004.

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

[8] L. Caires and L. Cardelli. A spatial logic for concurrency (part I). *Information and Computation*, 186:194–235, 2003.

[9] M. Fiore and S. Staton. A congruence rule format for name-passing process calculi from mathematical structural operational semantics. In *Proc. LICS'06*, pages 49–58. IEEE Computer Society Press, 2006.

[10] M. P. Fiore and D. Turi. Semantics of name and value passing. In *Proc. LICS'01*, pages 93–104. IEEE Computer Society Press, 2001.

[11] W. J. Fokkink, R. J. van Glabbeek, and P. de Wind. Compositionality of Hennessy-Milner logic through structural operational semantics. In *Proc. FCT'03*, volume 2751 of *LNCS*, pages 412–422. Springer, 2003.

[12] R. J. v. Glabbeek. The linear time – branching time spectrum I. In J. A. Bergstra, A. Ponse, and S. Smolka, editors, *Handbook of Process Algebra*. Elsevier, 1999.

[13] J. A. Goguen, J. W. Thatcher, et al. Initial algebra semantics and continuous algebras. *Journal of the ACM*, 24:68–95, 1977.

[14] J. F. Groote, M. Mousavi, and M. A. Reniers. A hierarchy of SOS rule formats. In *Proc. SOS'05*, 2005, pages 3–25. Elsevier, 2005.

[15] M. Hennessy and R. Milner. Algebraic laws for nondeterminism and concurrency. *Journal of the ACM*, 32:137–161, 1985.

[16] M. Kick. Rule formats for timed processes. In *Proc. CMCIM'02*, volume 68 of *ENTCS*, pages 12–31. Elsevier, 2002.

[17] B. Klin. From bialgebraic semantics to congruence formats. In *Proc. SOS 2004*, volume 128 of *ENTCS*, pages 3–37. Elsevier, 2005.

[18] B. Klin. Bialgebraic methods in structural operational semantics. In *Proc. SOS 2006*, ENTCS, 2007. To appear.

[19] B. Klin. Coalgebraic modal logic beyond sets. In *Proc. MFPS 2007*, volume 173 of *ENTCS*, pages 177–201, 2007.

[20] B. Klin and P. Sobocinski. Syntactic formats for free: An abstract approach to process equivalence. In *Proc. CONCUR 2003*, volume 2761 of *LNCS*, pages 72–86. Springer, 2003.

[21] A. Kurz. Coalgebras and their logics. *ACM SIGACT News*, 37, 2006.

[22] S. Mac Lane. *Categories for the Working Mathematician*. Springer, second edition, 1998.

[23] R. Milner. *Communication and Concurrency*. Prentice Hall, 1988.

[24] D. Pavlovic, M. Mislove, and J. B. Worrell. Testing semantics: connecting processes and process logics. In *Proc. AMAST'05*, volume 4019 of *LNCS*, pages 308–322. Springer, 2005.

[25] G. D. Plotkin. A structural approach to operational semantics. DAIMI Report FN-19, Computer Science Department, Aarhus University, 1981.

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

[27] L. Schröder. Expressivity of coalgebraic modal logic: the limits and beyond. In *Proc. FOSSACS'05*, volume 3441 of *LNCS*, pages 470–484, 2005.

[28] A. Simpson. Sequent calculi for process verification: Hennessy-Milner logic for an arbitrary GSOS. *Journal of Logic and Algebraic Programming*, 60–61:287–322, 2004.

[29] D. Turi. Categorical modeling of structural operational rules: case studies. In *Proc. CTCS'97*, volume 1290 of *LNCS*, pages 127–146. Springer, 1997.

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