The Microcosm Principle and Compositionality of GSOS-Based Component Calculi

Ichiro Hasuo*

University of Tokyo, Japan

Abstract. In the previous work by Jacobs, Sokolova and the author, synchronous parallel composition of coalgebras—yielding a coalgebra—and parallel composition of behaviors—yielding a behavior, where behaviors are identified with states of the final coalgebra—were observed to form an instance of the *microcosm principle*. The microcosm principle, a term by Baez and Dolan, refers to the general phenomenon of nested algebraic structures such as a monoid in a monoidal category. Suitable organization of these two levels of parallel composition led to a general *compositionality* theorem: the behavior of the composed system relies only on the behaviors of its constituent parts. In the current paper this framework is extended so that it accommodates any process operator—not restricted to parallel composition—whose meaning is specified by means of GSOS rules. This generalizes Turi and Plotkin's bialgebraic modeling of GSOS, by allowing a process operator to act as a connector between components as coalgebras.

1 Introduction

1.1 Structural Operational Semantics and Its Bialgebraic Modeling

Structural operational semantics (SOS) [19] is a well-developed mathematical tool for defining operational semantics of a programming language. It is based on *SOS rules* from which transitions between program terms are derived. SOS has been notably successful for *process calculi*: simple programming languages for concurrent processes. Various *syntactic formats*—syntactic restrictions on SOS rules—have been proposed to ensure good properties of SOS (see [1] for a survey); the *GSOS format* [6] is one of the most common among them.

In SOS for process calculi, dynamic behaviors of processes—such as $(a; b) \parallel \overline{b} \xrightarrow{a} (0; b) \parallel \overline{b}$—are derived by structural induction on the construction of process terms. In categorical terms, dynamics of processes (the former) are modeled by a *coalgebra* while the set of process terms forms an initial *algebra* that supports structural induction (the latter); see e.g. [14]. It is Turi and Plotkin's seminal work [21] that combines these two on a categorical level, resulting in a *bialgebraic modeling* of SOS.

* Thanks are due to Marcelo Fiore, Masahito Hasegawa, Bart Jacobs, Paul-André Melliès, Bartek Klin, John Power, Ana Sokolova and Sam Staton for helpful discussions. Helpful comments from the reviewers for the earlier versions of this paper are gratefully acknowledged. Supported by PRESTO Promotion Program, Japan Science and Technology Agency.

Basic bialgebraic modeling In the simplest setting in [21], a bialgebra is a carrier X equipped with both algebra and coalgebra structures: $\Sigma X \to X \to FX$. Typically, the functor Σ for the algebra part is $\Sigma = \coprod_{\sigma\in\Sigma}(\_)^{\mathsf{arity}(\sigma)} : \mathbf{Sets} \to \mathbf{Sets}$, a functor that represents a process algebra signature Σ; and the other functor is $F = (\mathcal{P}_\omega \_)^A : \mathbf{Sets} \to \mathbf{Sets}$, a functor for coalgebraic modeling of labeled transition systems (LTSs). Here $\mathcal{P}_\omega$ denotes the finite powerset functor, and A is the set of labels. The algebra and coalgebra structures are further subject to a certain compatibility condition via a natural transformation $\Sigma F \Rightarrow F\Sigma$; this natural transformation is what represents SOS rules.

The following two are almost only examples of such bialgebras in the literature.

- The one induced by the initial algebra $\Sigma I \to I$ (or, slightly more generally, a free algebra). Here I is the set of Σ-terms. The induced coalgebra $I \to FI$ is the transition structure between process terms, which is what is derived in the conventional SOS framework [19].
- The one induced by the final coalgebra $Z \to FZ$ (or, slightly more generally, a cofree coalgebra). Here Z is the set of "behaviors"—specifically bisimilarity classes of states of LTSs (see e.g. [11, §1.3]).[1] Existence of the induced algebra structure $\Sigma Z \to Z$ implies that the process operators are well-defined modulo bisimilarity, that is, *bisimilarity is a congruence*. As laid out shortly in §1.2, this induced algebraic structure is what is generalized in the current paper.

Bialgebraic modeling of GSOS rules It turns out, however, that only a very limited class of SOS rules can be represented by a natural transformation of the form $\Sigma F \Rightarrow F\Sigma$. Therefore in [21] a couple of extensions of the above basic scheme are proposed; the most notable among which is for the GSOS format. It is such that: a natural transformation representing GSOS rules is of the form $\Sigma F_\bullet \Rightarrow F\Sigma^*$. Here Σ^* is the *free monad* over Σ, with $\Sigma^* X$ being the set of Σ-terms that can contain elements of X as variables. The functor $F_\bullet$ is the *cofree copointed functor* over F, concretely: $F_\bullet X = X \times FX$. In [21] it is shown that any *GSOS specification*—a set of SOS rules compliant with the GSOS format—can be represented by a natural transformation of the form $\Sigma F_\bullet \Rightarrow F\Sigma^*$ and vice versa. See [15] for an introduction to the development.

The idea of bialgebraic modeling of SOS has been further pursued by many authors. See [15] and the references therein.

1.2 Parallel Composition of Coalgebras and the Microcosm Principle

In bialgebraic modeling, it is the elements of the carrier of a bialgebra that are combined using process operators. As described above, typical examples of such are: process terms (combined syntactically); and "behaviors," i.e. bisimilarity classes of LTS states (combined thanks to 'bisimilarity is a congruence').

However, the rise of *component calculi* as a foundation of component-based system design (see e.g. [4, 7]) poses a new challenge. In component-based design it is existing systems that are to be composed; and this lies out of the realm of bialgebraic modeling. Specifically, 'existing systems' do not always mean 'process terms': one may be

[1] "Bisimilarity" here is more precisely *behavioral equivalence*; they coincide for functors F that weakly preserve pullbacks (see e.g. [14]). This is the case with $F = (\mathcal{P}_\omega \_)^A$.

given two LTSs $\mathcal{S}_1$ and $\mathcal{S}_2$ that are generated from two process terms written in two different process calculi. 'Existing systems' do not necessarily mean their 'behaviors' either: given two LTSs $\mathcal{S}_1$ and $\mathcal{S}_2$, calculating their full behaviors is usually expensive. Therefore it is nice to be able to combine LTSs *as they are*—much like a product of two automata (e.g. in [4]), where one takes the product of two state spaces.

This idea of combining LTSs as they are, using a process operator whose meaning is specified by SOS rules, has been in the literature implicitly or explicitly (e.g. in [5, 8]). However this idea is often regarded as a "cosmetic" extension to SOS and its mathematical/categorical foundation has not been systematically pursued.

In our previous work [13] we formalized combination of LTSs as: a functor $\|$: $\mathbf{Coalg}(F)\times\mathbf{Coalg}(F) \to \mathbf{Coalg}(F)$ arising from a natural transformation $\mathsf{sync}_{X,Y}$: $FX \times FY \to F(X \times Y)$. If $F = (\mathcal{P}_\omega\_)^A$ this allows us to model synchronous parallel composition of LTSs (but hardly any other operator). There the carrier of the LTS $(X \xrightarrow{c} FX) \parallel (Y \xrightarrow{d} FY)$ is $X \times Y$. The natural transformation sync specifies a "synchronization mechanism," which represents a very limited class of SOS rules.

The operation $\|$ for composing LTSs yields a canonical operation $\|$ for composing "behaviors" like the one in the bialgebraic modeling (§1.1). Namely, behaviors are identified with elements of the final coalgebra $\zeta : Z \xrightarrow{\cong} FZ$ and $\|$ is induced by the coinduction diagram on the right.

$$\begin{array}{ccc} F(Z\times Z) & \xrightarrow{F\|} & FZ \\ {\scriptstyle\uparrow\zeta\|\zeta} & & {\scriptstyle\zeta\uparrow} \\ Z\times Z & \xrightarrow[\|]{} & Z \end{array}$$

Furthermore in [13], the two composition operators

$$\| : \mathbf{Coalg}(F) \times \mathbf{Coalg}(F) \longrightarrow \mathbf{Coalg}(F) \quad \text{and} \quad \| : Z \times Z \longrightarrow Z$$

with the latter being a coalgebra morphism $\|\colon \zeta \parallel \zeta \to \zeta$, are identified as an instance of so-called the *microcosm principle*. It is a term coined in [2], referring to the phenomenon that: a category $\mathbb{C}$ and its object $X \in \mathbb{C}$ have the "same" algebraic structures, with X's *inner* algebraic structure depending on $\mathbb{C}$'s *outer* one. A prototypical example is a monoid object X in a monoidal category $\mathbb{C}$: they both have a multiplication operator ($\mathsf{m} : X \otimes X \to X$ and $\otimes : \mathbb{C} \times \mathbb{C} \to \mathbb{C}$); and the definition of m uses $\otimes$ in it. In [13] we formalized what the microcosm principle means—especially what is meant by the "same" algebraic structures on different levels $\mathbb{C}$ and $X \in \mathbb{C}$—using Lawvere theories.

As an application we proved a *compositionality* result: for any coalgebras $X \xrightarrow{c} FX$ and $Y \xrightarrow{d} FY$ we have the top diagram on the right commute, where maps like $\mathsf{beh}(c)$ are by coinduction (the bottom diagram). This reads: the behavior of the composed system $c \parallel d$ can be computed from the behaviors of c and d, using $\|$. In particular—denoting bisimilarity by $\sim$—we have that $c \sim c'$ and $d \sim d'$ implies $c \parallel d \sim c' \parallel d'$, with regards to an appropriate choice of initial states.

$$\begin{array}{ccc} X\times Y & \xrightarrow{\mathsf{beh}(c\|d)} & Z \\ {\scriptstyle\mathsf{beh}(c)\times\mathsf{beh}(d)}\downarrow & \nearrow_{\|} & \\ Z\times Z & & \end{array}$$

$$\begin{array}{ccc} FX & \xrightarrow{F\,\mathsf{beh}(c)} & FZ \\ {\scriptstyle\uparrow c} & & {\scriptstyle\zeta\uparrow} \\ X & \xrightarrow[\mathsf{beh}(c)]{} & Z \end{array}$$

1.3 Microcosm Interpretation of Full GSOS Rules

	composing behaviors	composing both LTSs & behaviors
sync. $\|$	[21], $\Sigma F \Rightarrow F\Sigma$	[13]
full GSOS	[21], $\Sigma F_\bullet \Rightarrow F\Sigma^*$	current work

The state of art in [13] roughly corresponds to the basic bialgebraic modeling in §1.1 where an SOS specification is represented by a natural transformation $\Sigma F \Rightarrow F\Sigma$ (see the table). The current work extends [13] by accommodating any process operator whose meaning is specified by GSOS rules. Some GSOS rules are shown below; we assume that the set of labels is $N \cup \overline{N} \cup \{\tau\}$, consisting of names, conames and the internal action.

$$\frac{x \xrightarrow{a} x'}{x \parallel y \xrightarrow{a} x' \parallel y}\ (\|\mathrm{L}) \qquad \frac{y \xrightarrow{a} y'}{x \parallel y \xrightarrow{a} x \parallel y'}\ (\|\mathrm{R}) \qquad \frac{x \xrightarrow{a} x' \quad y \xrightarrow{\overline{a}} y'}{x \parallel y \xrightarrow{\tau} x' \parallel y'}\ (\|\textsc{Sync}) \qquad \frac{}{a \xrightarrow{a} 0}\ (\textsc{At})$$

$$\frac{x \xrightarrow{a} x'}{x; y \xrightarrow{a} x'; y}\ (;\mathrm{L}) \qquad \frac{x \overset{a}{\not\to}\ (\forall a \in A) \quad y \xrightarrow{b} y'}{x; y \xrightarrow{b} y'}\ (;\mathrm{R}) \qquad \frac{x \xrightarrow{a} x'}{!x \xrightarrow{a} x' \parallel !x}\ (!) \qquad \frac{x \xrightarrow{a} x'}{x^* \xrightarrow{a} x'; x^*}\ ((\_)^*)$$

We use these rules for constructing a new LTS from given LTSs. This means:

$$(\|\textsc{Sync}) \text{ is read as } \quad \frac{x \xrightarrow{a} x' \ \boxed{\text{in } \mathcal{S}} \qquad y \xrightarrow{\overline{a}} y' \ \boxed{\text{in } \mathcal{T}}}{x \parallel y \xrightarrow{\tau} x' \parallel y' \quad \boxed{\text{in } \mathcal{S} \parallel \mathcal{T}}}\ (\|\textsc{Sync}) \tag{1}$$

with an additional class of variables ($\mathcal{S}$ and $\mathcal{T}$) that tells in which LTS each transition takes place. The variables x, y here designate states of LTSs—unlike in the usual reading where they designate process terms. Different variables can designate states of distinct LTSs: in (1), x and x' are states of $\mathcal{S}$, while y and y' are of $\mathcal{T}$.

1.4 A Technical Challenge: State Spaces

Think of the new reading of the rules (;R) and (!). The challenge is: what is the state space of the sequential composition $\mathcal{S}; \mathcal{T}$? We can start with $X \times Y$—where X and Y are the state spaces of $\mathcal{S}$ and $\mathcal{T}$, respectively—denoting its element by $x; y$. According to the rule (;R), however, a state $x; y$ can "evolve" into y', which is no longer in $X \times Y$. So the answer seems to be $X \times Y + Y$. But how about the replication $!\mathcal{S}$? One would think of $X^+ = \coprod_{1 \le n} X^n$ or X^ω; both seem plausible.

In this paper we introduce a uniform, syntactic and modular way of constructing such a state space, so that it is compatible with the given set $\mathcal{R}$ of GSOS rules. We shall call it an *$\mathcal{R}$-state space*. The construction is syntactic in the sense that an $\mathcal{R}$-state space consists of rather simple sets like $X_1 \times \cdots \times X_m$, summed up over all the relevant algebraic terms (Def. 2.10). The construction is modular because the $\mathcal{R}$-state space for a composed term t can be calculated using the state spaces for t's subterms as building blocks ((4) later). Such modularity is an essential property of an "algebra."

As a technical tool in this construction we introduce the notion of *term lineage graph (TLG)*. A TLG is roughly a graph between two terms (thought of as parse trees) that keeps track of evolution of terms (like $x; y \mapsto y'$ in $(;\mathrm{R})$ above).

Organization of the paper In §2, after fixing notations for GSOS rules we describe the construction of $\mathcal{R}$-state spaces. In §3 we present a GSOS specification as a natural

transformation: this generalizes $\Sigma F_\bullet \Rightarrow F\Sigma^*$ in [21] (for composing different LTSs) as well as $\mathsf{sync}_{X,Y} : FX \times FY \to F(X \times Y)$ in [13] (for the full GSOS format). This results in the two interpretations of process operators in §4, acting on LTSs and on behaviors. We prove a compositionality result, and show that our framework indeed generalizes the GSOS fragment of [21]. In §5 we conclude and discuss related work.

2 State Space Compatible with GSOS rules

2.1 GSOS specification

We first fix a signature Σ of a process calculus. For each $n \in \mathbb{N}$ it determines the set Σ_n of *n-ary operators*.

Definition 2.1 (GSOS) A *GSOS rule* R (as in [21]) over a signature Σ is a syntactic expression of the following form.

$$\frac{\{x_i \xrightarrow{a} y_i^{a,j}\}_{i\in[1,m]}^{a\in A, j\in[1,N_i^a]} \quad \{x_i \overset{b}{\not\to}\}_{i\in[1,m]}^{b\in B_i}}{\sigma(x_1,\ldots,x_m) \xrightarrow{e} t} \tag{2}$$

Here A is a fixed set of *labels*; $x_i, y_i^{a,j}$ are distinct variables; N_i^a is a natural number that is 0 for almost every i and a; B_i is a (possibly infinite) subset of A; σ is an m-ary operator in Σ; and t is a Σ-term where only x_i and $y_i^{a,j}$ occur as variables. For a GSOS rule R like (2), the operator σ on the left in the conclusion shall be denoted by σ_R; the term t on the right is denoted by t_R.

A *GSOS specification* is a pair $(\Sigma, \mathcal{R})$ of a signature Σ and an *image-finite* set $\mathcal{R}$ of GSOS rules over Σ. Here image-finiteness means that there are only finitely many rules in $\mathcal{R}$, once $\sigma \in \Sigma$ and $c \in A$ are fixed.

Our another reading of (2)—in the sense of (1)—is as follows, deriving a transition in a new LTS $\sigma(\mathcal{S}_1,\ldots,\mathcal{S}_m)$.

$$\frac{\{x_i \xrightarrow{a} y_i^{a,j} \text{ in } \mathcal{S}_i\}_{i\in[1,m]}^{a\in A, j\in[1,N_i^a]} \quad \{x_i \overset{b}{\not\to} \text{ in } \mathcal{S}_i\}_{i\in[1,m]}^{b\in B_i}}{\sigma(x_1,\ldots,x_m) \xrightarrow{e} t \quad \text{in } \sigma(\mathcal{S}_1,\ldots,\mathcal{S}_m)}$$

Here $\mathcal{S}_1,\ldots,\mathcal{S}_m$ are a new class of variables that designate LTSs; variables x_i and $y_i^{a,j}$ with a subscript i therefore designate states of $\mathcal{S}_i$.

We will need a careful inspection of the structure of Σ-terms. We fix some notations.

Notation 2.2 We assume that a Σ-term t always comes with an explicit *context* of variables: $x_1,\ldots,x_m \vdash t$. Any occurrence of a variable in t must be that of $x_1, x_2, \ldots$ or x_m; each variable x_i can have multiple or no occurrences in t. For example, we distinguish two terms $x_1 \vdash x_1 \parallel x_1$ and $x_1, x_2 \vdash x_1 \parallel x_1$ because of different contexts. In the sequel, however, we suppress the context of a Σ-term when it is obvious.

Definition 2.3 (i_t, $|t|$ and t^ℓ) Given a Σ-term $x_1, \ldots, x_m \vdash t$, the number of *occurrences* of variables in t is denoted by $|t|$. Note that this is not necessarily the same as m. Then the term t induces an "indexing" function $\mathsf{i}_t : [1, |t|] \to [1, m]$ such that: the i-th occurrence of variables (counted from left to right) in t is that of $x_{\mathsf{i}_t(i)}$.

The term obtained from t, by replacing all the occurrences of variables by those of distinct variables $x_1, \ldots, x_{|t|}$ from left to right, is denoted by t^ℓ. This is the *linear term* induced by t. An easy consequence is: $t = t^\ell[x_{\mathsf{i}_t(1)}/x_1, \ldots, x_{\mathsf{i}_t(|t|)}/x_{|t|}]$.

For example, let t be the term $x_1, x_2 \vdash x_2 \parallel (x_1 \parallel !x_1)$. Then $|t| = 3$, $\mathsf{i}_t : 1 \mapsto 2, 2 \mapsto 1, 3 \mapsto 1$. The term t^ℓ is $x_1, x_2, x_3 \vdash x_1 \parallel (x_2 \parallel !x_3)$.

2.2 Requirements on State Spaces

Shortly we will introduce the notion of *term lineage graph (TLG)*. Since it is technically rather involved, we shall first lay out what we aim to achieve using TLGs.

We construct state spaces for LTSs like $\mathcal{S}_1 \parallel \mathcal{S}_2$, $\mathcal{S}_1; \mathcal{S}_2$, $!\mathcal{S}$, etc., deriving from given GSOS rules. Let $x_1, \ldots, x_m \vdash t$ be a Σ-term and $X_1, \ldots, X_m$ be sets, with the idea that X_i is the state space of the LTS $\mathcal{S}_i$. We shall define an *$\mathcal{R}$-state space* $(\!|t|\!)(X_1, \ldots, X_m)$. The following requirements are essential to to base the framework in [13] on it; these will be exploited in the technical development later in §3.

Requirements 2.4 1. The set $(\!|t|\!)(\vec{X})$ should accommodate *initial* states that are there prior to any evolution—such as $x_1 \parallel x_2$ in $\mathcal{S}_1 \parallel \mathcal{S}_2$ or $!x$ in $!\mathcal{S}$ (see §1.4). The set of such initial states are given by

$$|t|(X_1, \ldots, X_m) \;:=\; X_{\mathsf{i}_t(1)} \times \cdots \times X_{\mathsf{i}_t(|t|)} \;, \tag{3}$$

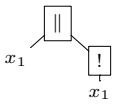
where i_t and $|t|$ are from Def. 2.3. Note that $|t|(\vec{X})$ need not be the same as $X_1 \times \cdots \times X_m$. Our definition (3) implies, for example, that we allow $x_1 \parallel x_1'$ (with $x_1, x_1' \in X_1$ and $x_1 \neq x_1'$) as an initial state in an LTS $\mathcal{S}_1 \parallel \mathcal{S}_1$.

2. The set $(\!|t|\!)(\vec{X})$ should also accommodate those states which would arise through "evolution" specified by GSOS rules (see §1.4).
3. *Functoriality*: the operation $(\!|t|\!)$ extends to a functor $\mathbf{Sets}^m \to \mathbf{Sets}$.
4. *Modularity*: the operation $(\!|\_|\!)$—applied to a term t and yielding a functor $(\!|t|\!) : \mathbf{Sets}^m \to \mathbf{Sets}$—is compatible with substitution of terms. That is,

$$(\!|\, t[t_i/x_i] \,|\!)\,(X_1, \ldots, X_m) \;\cong\; (\!|t|\!)\,\big(\,(\!|t_1|\!)(\vec{X}),\, \ldots,\, (\!|t_n|\!)(\vec{X})\,\big) \;. \tag{4}$$

2.3 Term Lineage Graph (TLG)

First we note that a Σ-term t can be identified with its *parse tree*. Its leaves are variables and 0-ary operators; and its internal nodes are operators with positive arities with a suitable branching degree. For example the term $x_1 \parallel !x_1$ is understood as on the right.



Definition 2.5 (Term lineage graph) Let s, t be Σ-terms. We require them to be in the same variable context (Notation 2.2): $x_1, \ldots, x_m \vdash s, t$. A *term lineage graph (TLG)* ρ from s to t, denoted by $\rho : s \Rightarrow t$, is an unlabeled directed graph (like in (5)) whose nodes are nodes of s and t (seen as parse trees), such that:

– any edge is from a node in the *domain term* s to a node in the *codomain term* t;
– each node in s has exactly one outgoing edge;
– the edges are *monotone*: assume that the origin of one edge is a descendant (in the parse tree s) of the origin of another edge. Then the target of the former is also a descendant of (or the same as) that of the latter;
– an edge from an operator symbol σ goes into a (not necessarily the same) operator symbol;
– an edge from a variable x_i in s goes into the same variable x_i in t.

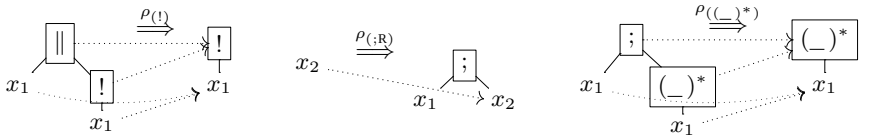
A TLG is sometimes denoted by $x_1, \ldots, x_m \vdash \rho : s \Rightarrow t$, making its context explicit.

Definition 2.6 (The TLG ρ_R) Let R be a GSOS rule of the form (2). It induces a *TLG* ρ_R defined as follows. Its type is $\rho_R : t_R[x_i/y_i^{a,j}] \Rightarrow \sigma_R(x_1, \ldots, x_m)$. Concretely

– each node in the domain term $t_R[x_i/y_i^{a,j}]$ that is labeled with an operator symbol is tied to the root node (labeled with σ_R) of the codomain term $\sigma_R(x_1, \ldots, x_m)$;
– each node in the domain term that is labeled with a variable x_i is tied to the unique occurrence of the same variable x_i in the codomain $\sigma_R(x_1, \ldots, x_m)$.

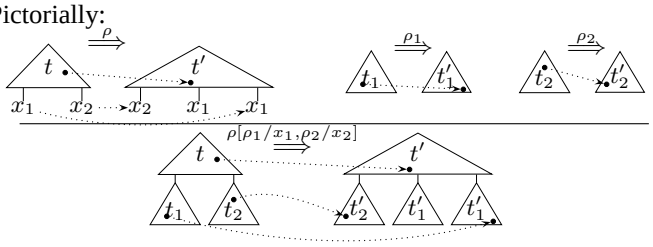
Intuitively, the substitution $y_i^{a,j} \mapsto x_i$ in the domain term $t_R[x_i/y_i^{a,j}]$ forces every occurrence of a state of $\mathcal{S}_i$ to be denoted by x_i.

Here are some examples of TLGs induced by GSOS rules. The solid lines represent the order in each parse tree; the dotted lines are the edges of the TLGs.



(5)

Definition 2.7 (Operations on TLGs) – (Identity) For each Σ-term s, the *identity TLG* $\mathrm{id}_s : s \Rightarrow s$ is the one in which each node in the domain term is tied to the same node of the codomain term.
– (Composition) Given two successive TLGs $\rho : t \Rightarrow s$ and $\rho' : s \Rightarrow u$, the *composition* $\rho' \bullet \rho : t \Rightarrow u$ is obtained by composing two successive edges of ρ and ρ', and then forgetting about the mediating term s.
– (Substitution) Let $x_1, \ldots, x_n \vdash \rho : t \Rightarrow t'$ be a TLG. Assume further that we have a TLG $x_1, \ldots, x_m \vdash \rho_i : t_i \Rightarrow t_i'$, for each $i \in [1, n]$.[2] We define the *substitution* $\rho[\rho_i/x_i]$, which is a TLG between substituted terms from $t[t_i/x_i]$ to $t'[t_i'/x_i]$, as follows. Pictorially:

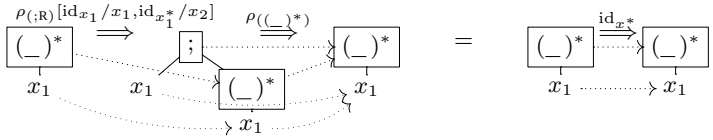


That is,

[2] Distinguish m and n. Note we have the same context for all of ρ_i.

- Each node in the (upper) t-part of the domain term $t[t_i/x_i]$ has the same outgoing edge as in ρ, into the t'-part of the codomain term.
- As for a node in the (lower) t_i-part of the domain term: basically the outgoing edge is the same as the corresponding one in $\rho_i : t_i \Rightarrow t'_i$. However, like t'_1 in the above example, there can be multiple occurrences of t'_i in the codomain term; we must decide which occurrence the edge points to. There we use the information in $\rho : t \Rightarrow t'$: we follow the edge in ρ that starts from the corresponding occurrence of a variable.

Let us now illustrate the operations on TLGs. Using three TLGs $\rho_{(;\mathrm{R})} : x_2 \Rightarrow x_1; x_2$, $\mathrm{id}_{x_1} : x_1 \Rightarrow x_1$ and $\mathrm{id}_{x_1^*} : x_1^* \Rightarrow x_1^*$, we can form the substitution $\rho_{(;\mathrm{R})}[\mathrm{id}_{x_1}/x_1, \mathrm{id}_{x_1^*}/x_2]$ that is shown below on the left. The equality below asserts that, if we pre-compose $\rho_{((\_)^*)}$ to that substitution, then it is the same as the identity TLG.



(6)

Finally, a given set $\mathcal{R}$ of GSOS rules determines a class of TLGs that are relevant to it.

Definition 2.8 ($\mathcal{R}$-TLG) Let $(\Sigma, \mathcal{R})$ be a GSOS specification (Def. 2.1). The class of *$\mathcal{R}$-TLGs* is a subclass of TLGs between Σ-terms, defined inductively as follows: 1) for each GSOS rule $R \in \mathcal{R}$, the induced TLG ρ_R (Def. 2.6) is an $\mathcal{R}$-TLG; 2) the class of $\mathcal{R}$-TLGs is closed under identities, composition, and substitution (Def. 2.7).

2.4 $\mathcal{R}$-State Space

We define an *$\mathcal{R}$-state space* operator $(\!|t|\!)$. Let us fix a GSOS specification $(\Sigma, \mathcal{R})$.

Definition 2.9 (Initial state space) Given a Σ-term $x_1, \ldots, x_m \vdash t$, we define the *initial state space functor* $|t| : \mathbf{Sets}^m \to \mathbf{Sets}$ by (3) in §2.2. Its functoriality is obvious. We are overriding the notation $|t|$ (cf. Def. 2.3); this will not cause any confusion.

Definition 2.10 ($\mathcal{R}$-state space) Let t be a Σ-term. We define the *$\mathcal{R}$-state space functor* $(\!|t|\!) : \mathbf{Sets}^m \to \mathbf{Sets}$ by:

$$(\!|t|\!)(X_1, \ldots, X_m) \;:=\; \coprod \big\{\, |s|(X_1, \ldots, X_m) \mid \rho : s \Rightarrow t,\ \mathcal{R}\text{-TLG} \,\big\} \ . \tag{7}$$

Here $|s|$ is the initial state space functor (Def. 2.9); the sum $\coprod$ is taken over all the $\mathcal{R}$-TLGs ρ with a codomain term t. We have one summand for each ρ, not for each s.

Let us now calculate some $\mathcal{R}$-state spaces. We take as $\mathcal{R}$ the set of rules presented in §1.3; we take the corresponding signature $\Sigma = \{\|, !, ;, (\_)^*\} \cup \{a \mid a \in A\}$.

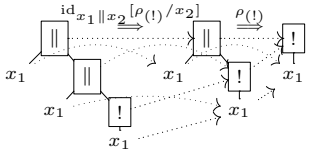
For a rule $R \in \mathcal{R}$ whose *principal operator* is $\|$—namely $(\|\mathrm{L})$, $(\|\mathrm{R})$ and $(\|\textsc{Sync})$—the induced TLG ρ_R coincides with the identity (Def. 2.7). It follows that any $\mathcal{R}$-TLG of

the form $\cdot \Rightarrow x_1 \parallel x_2$ is the identity TLG. Hence by (7) we have $(\!|x_1 \parallel x_2|\!)(X_1, X_2) = |x_1 \parallel x_2|(X_1, X_2) = X_1 \times X_2$.

For the operator $;$, an $\mathcal{R}$-TLG whose codomain term is $x_1, x_2 \vdash x_1; x_2$ is either the identity or $\rho_{(;\mathrm{R})}$ (see (5)). Hence $(\!|x_1; x_2|\!)(X_1, X_2) = |x_1; x_2|(X_1, X_2) + |x_2|(X_1, X_2) = X_1 \times X_2 + X_2$. Here the term x_2 inside the expression $|x_2|$ is $x_1, x_2 \vdash x_2$, to be precise about its context.

Regarding $!$, we have countably many $\mathcal{R}$-TLGs whose codomain is $!x_1$:

$\mathrm{id} : \quad !x_1 \Rightarrow !x_1$,
$\rho_{(!)} : \quad x_1 \parallel !x_1 \Rightarrow !x_1$,
$\rho_{(!)} \bullet (\mathrm{id}_{x_1 \parallel x_2}[\rho_{(!)}/x_2]) : \quad x_1 \parallel (x_1 \parallel !x_1) \Rightarrow !x_1$, etc.



The third one is depicted above. Therefore we have:

$$(\!|!x_1|\!)(X_1) \;=\; \coprod_{i \geq 0} \Big| \underbrace{x_1 \parallel (\cdots \parallel (x_1}_{i} \;\; \parallel !x_1) \cdots) \Big| (X_1) \;=\; \coprod_{i \geq 0} X_1^{i+1} \;=\; X_1^+ \;.$$

Regarding $(\_)^*$, from the equality (6) it follows that the situation is similar to $!$ above. Namely, we have one $\mathcal{R}$-TLG for each of the following types: $x_1^* \Rightarrow x_1^*$, $x_1; x_1^* \Rightarrow x_1^*$, $x_1; (x_1; x_1^*) \Rightarrow x_1^*$, etc. Hence we have $(\!|x_1^*|\!)(X_1) = X_1^+$.

We now verify that Requirements 2.4 are indeed fulfilled. The item 3 is obvious. The proof is in Appendix A.1.

Proposition 2.11 *1. There is a canonical embedding $(\nu_t)_{\vec{X}} : |t|(\vec{X}) \to (\!|t|\!)(\vec{X})$. This extends to a natural transformation $\nu_t : |t| \Rightarrow (\!|t|\!) : \mathbf{Sets}^m \to \mathbf{Sets}$.*

2. Given an $\mathcal{R}$-TLG $\rho : s \Rightarrow t$, it induces a canonical map $(\!|\rho|\!)_{\vec{X}} : (\!|s|\!)(\vec{X}) \to (\!|t|\!)(\vec{X})$. It also extends to a natural transformation $(\!|\rho|\!) : (\!|s|\!) \Rightarrow (\!|t|\!)$.

4. The operation $(\!|\_|\!)$ is compatible with substitution: for Σ-terms $x_1, \ldots, x_n \vdash t$ and $x_1, \ldots, x_m \vdash t_i$ for each $i \in [1, n]$, we have a canonical isomorphism (4). □

Using the embedding $(\nu_t)_{\vec{X}}$ in Prop. 2.11.1, we can also embed the set $X_1 \times \cdots \times X_m$ in the $\mathcal{R}$-state space $(\!|t|\!)(X_1, \ldots, X_m)$. This is by the following $\theta_{X_1, \ldots, X_m}$:

$$\theta_{\vec{X}} \;:=\; \big(X_1 \times \cdots \times X_m \overset{(\psi_t)_{X_1,\ldots,X_m}}{\longrightarrow} X_{\mathrm{i}_t(1)} \times \cdots \times X_{\mathrm{i}_t(|t|)} = |t|(\vec{X}) \overset{(\nu_t)_{\vec{X}}}{\longrightarrow} (\!|t|\!)(\vec{X}) \big) \;, \quad (8)$$

where $(\psi_t)_{\vec{X}}$ is a function such that $(x_1, \ldots, x_m) \mapsto (x_{\mathrm{i}_t(1)}, \ldots, x_{\mathrm{i}_t(|t|)})$ (cf. Def. 2.3). It rearranges arguments $x_1, \ldots, x_m$ according to the occurrences of variables in t.

Remark 2.12 Our choice of state spaces (Def. 2.10) is in fact one out of a spectrum. The smallest extreme in the spectrum is obtained by: singling out the "reachable" part of our $\mathcal{R}$-state space and then quotienting it out by the bisimilarity. Although this yields a smaller state space, calculating such is expensive. Moreover it does not satisfy Requirements 2.4.4, breaking modularity/algebraicity of the whole framework.

The biggest extreme is to take, as the state space $(\!|t|\!)(X_1, \ldots, X_m)$, the whole set of Σ-terms with all states $x_i \in X_i$ as variables. This satisfies Requirements 2.4 so we could develop the theory on top of it. However, we claim it to be our finding that we

can trim down this big state space into the one in Def. 2.10. In particular, our refined definition yields $(\!|\, \|\, |\!)(X_1, X_2) = X_1 \times X_2$ which matches with intuition as well as with the usual definitions of product of automata in e.g. [4].

In-between is the "Kleene-style" composition of automata. The classic result on regular language and DFA/NFA has recently seen its vast generalization in coalgebraic terms; see e.g. [20]. In this approach the Kleene star $(\_)^*$ preserves finiteness of state spaces, whereas our approach yields the state space X^* that is inevitably infinite. However the definition of Kleene-style composition calls for different ingenuity for each operator; it is not clear if that can be done uniformly for any operation defined in GSOS.

We further note that our choice goes well with *first-order representation* of LTSs. (Variants of) the latter are now widely used (e.g. CARML in [3]) because it enables BDD-based representation and symbolic model checking [9]. In first-order representation, a state is represented by an assignment of values to variables such as $[\texttt{agent1} \mapsto \texttt{critical}, \texttt{agent2} \mapsto \texttt{noncritical}]$; hence a state space is the *product* of the value domains for the variables. Combining such state spaces by means of products and coproducts (7) could be done in a programming language with advanced type constructors.

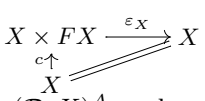
3 Categorical GSOS Specification

We introduced appropriate state spaces for $\mathcal{S}_1 \parallel \mathcal{S}_2, \mathcal{S}_1; \mathcal{S}_2$, etc. in §2; we wish to derive transition structures on those state spaces, making them LTSs. In concrete terms it is via derivations using GSOS rules, with their reading like (1). In categorical terms, GSOS rules are first translated into a natural transformation ξ_t (for each term t, this section); which gives rise to transition structures (§4).

3.1 Copointed Functors and Copointed Coalgebras

We will need the following notion of copointedness—a technical but standard one in the field (see e.g. [17, 21]). For the functor $F = (\mathcal{P}_\omega \_)^A : \mathbf{Sets} \to \mathbf{Sets}$, we set a functor $F_\bullet : \mathbf{Sets} \to \mathbf{Sets}$ by $F_\bullet X := X \times FX = X \times (\mathcal{P}_\omega X)^A$. We denote the first projection $F_\bullet X (= X \times FX) \to X$ by ε_X.

An $F_\bullet$-coalgebra $c : X \to F_\bullet X$ is said to be *copointed* if the diagram on the right commutes. Intuitively, the additional component X in $F_\bullet X = X \times FX$ records the "original" state before a transition; indeed for a copointed $F_\bullet$-coalgebra $c : X \to X \times (\mathcal{P}_\omega X)^A$ we have $c(x) = (\, x,\, \lambda a.\, \{x' \mid x \xrightarrow{a} x'\}\,)$. The next result is standard and easy; see [17].



Lemma 3.1 *Let us denote the category of F-coalgebras by $\mathbf{Coalg}(F)$; and the category of copointed $F_\bullet$-coalgebras by $\mathbf{Coalg}_\bullet(F_\bullet)$. The two categories are isomorphic: $\mathbf{Coalg}(F) \cong \mathbf{Coalg}_\bullet(F_\bullet)$. In particular, $\mathbf{Coalg}_\bullet(F_\bullet)$ has a final object (a* final copointed $F_\bullet$-coalgebra*) that corresponds to a final object in $\mathbf{Coalg}(F)$.* □

Due to this result, in the sequel we identify an LTS with a copointed coalgebra for the functor $F_\bullet X = X \times (\mathcal{P}_\omega X)^A$. Bisimilarity in LTSs can then be captured by the final copointed $F_\bullet$-coalgebra—which we denote by $\zeta_\bullet : Z \to F_\bullet Z$. Note that the coalgebra $\zeta_\bullet$ has the same carrier set Z as the final F-coalgebra; this is also a standard fact.

3.2 GSOS Rules, Categorically

We shall translate a set $\mathcal{R}$ of GSOS rules into functions of the following type:

$$(\xi_t)_{\vec{X}} \;:\; (\!|t|\!)\big(F_\bullet X_1,\, \ldots,\, F_\bullet X_m\big) \longrightarrow F_\bullet\big((\!|t|\!)(X_1,\ldots,X_m)\big) \;, \tag{9}$$

defined for each Σ-term $x_1,\ldots,x_m \vdash t$ and sets $X_1,\ldots,X_m$. Here $F = (\mathcal{P}_\omega\_)^A$, and m is the length of t's context $x_1,\ldots,x_m \vdash t$. These functions $(\xi_t)_{\vec{X}}$ form a natural transformation ξ_t, defined for each term t. The difference from the natural transformation $\Sigma F_\bullet X \Rightarrow F\Sigma^* X$ in [21] is that ours $(\xi_t)_{\vec{X}}$ is required to be natural in multiple sets $X_1,\ldots,X_m$. This is because we deal with state spaces of multiple LTSs.

Towards the natural transformation ξ_t in (9) we proceed step by step: we first derive from a GSOS rule R a natural transformation $\xi^{(1)}_R$, from which we derive $\xi^{(2)}_R, \xi^{(3)}_\sigma, \ldots$, finally reaching ξ_t of the desired form.

Step 1 Let $R \in \mathcal{R}$ be a GSOS rule in (2). It induces the following function $(\xi^{(1)}_R)_{X_1,\ldots,X_m}$:

$$\begin{array}{ll}
& |\sigma_R|\,(F_\bullet X_1, \cdots, F_\bullet X_m) \overset{(\xi^{(1)}_R)_{\vec{X}}}{\longrightarrow} F\big(\,|t_R[x_i/y_i^{a,j}]|\,(X_1,\ldots,X_m)\big) \;, \\
\text{i.e.} & \prod_{i\in[1,m]}\big(X_i \times (\mathcal{P}_\omega X_i)^{A}\big) \overset{(\xi^{(1)}_R)_{\vec{X}}}{\longrightarrow} \Big(\mathcal{P}_\omega\Big(\,|\,t_R[x_i/y_i^{a,j}]\,|(\vec{X})\Big)\Big)^A \\
\text{by} & \big((\xi^{(1)}_R)_{\vec{X}}(\mathtt{x}_1,\varphi_1,\ldots,\mathtt{x}_m,\varphi_m)\big)(e) \\
& = \left\{\, t_R[\mathtt{x}_i/x_i, \mathtt{y}_i^{a,j}/y_i^{a,j}] \;\middle|\; \begin{array}{l} \forall i.\, \forall b \in B_i.\;\; \varphi_i(b) = \emptyset, \text{ and} \\ \forall i.\, \forall a.\, \forall j \in [1, N_i^a].\;\; \mathtt{y}_i^{a,j} \in \varphi_i(a). \end{array} \right\} \;.
\end{array}$$

Here $\mathtt{x}_i$ and $\mathtt{y}_i^{a,j}$ denote elements of X_i; φ_i belongs to $(\mathcal{P}_\omega X_i)^A$; and $e \in A$ is a label. Note that $|\sigma_R|(F_\bullet X_1,\cdots,F_\bullet X_m) = F_\bullet X_1 \times \cdots \times F_\bullet X_m$ (Def. 2.9). The expression $t_R[\mathtt{x}_i/x_i, \mathtt{y}_i^{a,j}/y_i^{a,j}]$ denotes the obvious element of the set $|\,t_R[x_i/y_i^{a,j}]\,|(X_1,\ldots,X_m)$. To be precise, the latter set is of the form $X_{k_1} \times \cdots \times X_{k_{|t_R|}}$, with each component X_{k_l} coming from an occurrence of either x_i or $y_i^{a,j}$ in t_R. If X_{k_l} is from x_i then the l-th component of $t_R[\mathtt{x}_i/x_i, \mathtt{y}_i^{a,j}/y_i^{a,j}]$ is $\mathtt{x}_i$; if X_{k_l} is from $y_i^{a,j}$ then it is $\mathtt{y}_i^{a,j}$.

Step 2 We exploit properties of $(\!|\_|\!)$ (Prop. 2.11.1–2) to obtain the following two successive functions. Here ρ_R is the $\mathcal{R}$-TLG induced by the GSOS rule R (Def. 2.6).

$$\begin{array}{c}
(\nu_{t_R[x_i/y_i^{a,j}]})_{\vec{X}} : |\,t_R[x_i/y_i^{a,j}]\,|(\vec{X}) \longrightarrow (\!|\,t_R[x_i/y_i^{a,j}]\,|\!)(\vec{X}) \;, \\
(\!|\rho_R|\!)_{\vec{X}} : (\!|\,t_R[x_i/y_i^{a,j}]\,|\!)(\vec{X}) \longrightarrow (\!|\sigma_R|\!)(\vec{X}) \;.
\end{array}$$

The functions are natural in $X_1,\ldots,X_m$. We compose these arrows, apply the functor F, and then post-compose it to $(\xi^{(1)}_R)_{\vec{X}}$ in Step 1. The outcome is a natural transformation $(\xi^{(2)}_R)_{X_1,\ldots,X_m} \;:\; |\sigma_R|\,(F_\bullet X_1,\cdots,F_\bullet X_m) \longrightarrow F\big((\!|\sigma_R|\!)\,(X_1,\ldots,X_m)\big)$.

Step 3 The natural transformation $\xi^{(2)}_R$ has been defined for each rule $R \in \mathcal{R}$; we shall take their "union" to define $\xi^{(3)}_\sigma$, now for each operator $\sigma \in \Sigma$. Specifically, for each operator $\sigma \in \Sigma$ and $e \in A$, let $\mathcal{R}_{\sigma,e}$ denote the collection of those rules $R \in \mathcal{R}$ whose

conclusion is of the form $\sigma(x_1,\ldots,x_m) \xrightarrow{e} \cdot$. By the image finiteness assumption (Def. 2.1) the set $\mathcal{R}_{\sigma,e}$ is finite. We define a function

$$\begin{array}{l}(\xi^{(3)}_\sigma)_{\vec{X}} \;:\; |\sigma|\,(F_\bullet X_1,\cdots,F_\bullet X_m) \longrightarrow F\big(\,(\!|\sigma|\!)\,(X_1,\ldots,X_m)\,\big)\\ \text{by } \big(\,\xi^{(3)}_\sigma(\mathtt{x}_1,\varphi_1,\ldots,\mathtt{x}_m,\varphi_m)\,\big)(e)\\ \quad := \bigcup_{R\in\mathcal{R}_{\sigma,e}}\big(\,\xi^{(2)}_R(\mathtt{x}_1,\varphi_1,\ldots,\mathtt{x}_m,\varphi_m)\,\big)(e)\;\;.\end{array}$$

Recall that $FX = (\mathcal{P}_\omega X)^A$ and $F_\bullet X = X \times (\mathcal{P}_\omega X)^A$; the union is in $\mathcal{P}_\omega((\!|\sigma|\!)(\vec{X}))$.

Step 4 We shall extend F in the codomain of $\xi^{(3)}_\sigma$ to $F_\bullet$, so that we obtain $(\xi^{(4)}_\sigma)_{\vec{X}} : |\sigma|(\overrightarrow{F_\bullet X}) \to F_\bullet((\!|\sigma|\!)(\vec{X}))$. What we need is a function of the type $|\sigma|(\overrightarrow{F_\bullet X}) \to (\!|\sigma|\!)(\vec{X})$; then we can tuple it with $\xi^{(3)}_\sigma$ and obtain $\xi^{(4)}_\sigma$. Such a function is given by

$$|\sigma|(\overrightarrow{F_\bullet X}) \overset{|\sigma|(\varepsilon\vec{X})}{\longrightarrow} |\sigma|(\vec{X}) \overset{(\nu_\sigma)_{\vec{X}}}{\longrightarrow} (\!|\sigma|\!)(\vec{X})\;\;.$$

Here $\varepsilon_{X_i} : F_\bullet X_i \to X_i$ is the first projection (§3.1); ν_σ is from Prop. 2.11.1. We used the functoriality of the operation $|\sigma|$ (Def. 2.9).

We shall further extend this definition of $\xi^{(4)}_\sigma$ to $(\xi^{(4)}_t)_{\vec{X}} : |t|(\overrightarrow{F_\bullet X}) \to F_\bullet((\!|t|\!)(\vec{X}))$, now defined for each Σ-term t. This is by induction on the formation of Σ-terms.

If t is a variable x_i (i.e. $x_1,\ldots,x_m \vdash x_i$), we have $|x_i|(\overrightarrow{F_\bullet X}) = F_\bullet X_i$ (Def. 2.9) and $(\!|x_i|\!)(\vec{X}) = X_i$ (Def. 2.10, the only $\mathcal{R}$-TLG into x_i is the identity). Hence the function $(\xi^{(4)}_{x_i})_{\vec{X}}$ we are defining is of the type $F_\bullet X_i \to F_\bullet X_i$; it is defined to be the identity. If t is a composed term $\sigma(s_1,\ldots,s_n)$, we define $(\xi^{(4)}_t)_{\vec{X}}$ by the composite:

$$\begin{array}{l}|\sigma(s_1,\ldots,s_n)|(F_\bullet X_1,\ldots,F_\bullet X_m)\\ \qquad\overset{\cong}{\longrightarrow}\\ \qquad\overset{|\sigma|\left(\,(\xi^{(4)}_{s_1})_{\vec{X}},\;\ldots,\;(\xi^{(4)}_{s_n})_{\vec{X}}\,\right)}{\longrightarrow} |\sigma|\big(\,|s_1|(\overrightarrow{F_\bullet X}),\;\ldots,\;|s_n|(\overrightarrow{F_\bullet X})\,\big)\\ \qquad\overset{(\xi^{(4)}_\sigma)_{(\!|s_1|\!)(\vec{X}),\ldots,(\!|s_n|\!)(\vec{X})}}{\longrightarrow} |\sigma|\big(\,F_\bullet((\!|s_1|\!)(\vec{X})),\;\ldots,\;F_\bullet((\!|s_n|\!)(\vec{X}))\,\big)\\ \qquad\overset{}{\longrightarrow} F_\bullet\big(\,(\!|\sigma|\!)\big(\,(\!|s_1|\!)(\vec{X}),\;\ldots,\;(\!|s_n|\!)(\vec{X})\,\big)\,\big)\\ \qquad\overset{\cong}{\longrightarrow} F_\bullet\big(\,(\!|\sigma(s_1,\ldots,s_n)|\!)(X_1,\ldots,X_m)\,\big)\;\;.\end{array}$$

Here the first isomorphism is a variant of (4), for $|\_|$ instead of $(\!|\_|\!)$, which we readily obtain. The second one uses functoriality of $|\sigma|$ applied to functions $(\xi^{(4)}_{s_i})_{\vec{X}}$; the latter are available by the induction hypothesis. The third function is what we defined in the first half of the current Step 4; the last isomorphism is by Prop. 2.11.4.

Step 5 Finally we shall obtain the goal (9) of the current §3.2, by extending $|t|$ into $(\!|t|\!)$ in the domain of the previous $(\xi^{(4)}_t)_{\vec{X}} : |t|(\overrightarrow{F_\bullet X}) \to F_\bullet((\!|t|\!)(\vec{X}))$. By Def. 2.10 of $(\!|t|\!)$, such an extension can be done through finding, for each $\mathcal{R}$-TLG $\rho : s \Rightarrow t$, a function of the type $|s|(\overrightarrow{F_\bullet X}) \to F_\bullet((\!|t|\!)(\vec{X}))$. The following composite does this job.

$$|s|(F_\bullet X_1,\ldots,F_\bullet X_m) \overset{(\xi^{(4)}_s)_{\vec{X}}}{\longrightarrow} F_\bullet((\!|s|\!)(\vec{X})) \overset{F_\bullet((\!|\rho|\!)_{\vec{X}})}{\longrightarrow} F_\bullet((\!|t|\!)(\vec{X}))$$

Here $(\!|\rho|\!)$ is from Prop. 2.11.2. We take the cotuple of these functions (i.e. definition by cases) to obtain $(\xi_t)_{\vec{X}}$.

4 The Microcosm Interpretation of GSOS Rules

Let $(\Sigma, \mathcal{R})$ be a GSOS specification. We shall define its *microcosm interpretation*. It consists of, for each Σ-term $x_1, \ldots, x_m \vdash t$,

$$\begin{array}{l} \text{the } \textit{outer} \text{ interpretation } [\![t]\!] \ : \ \big(\mathbf{Coalg}_\bullet(F_\bullet)\big)^m \to \mathbf{Coalg}_\bullet(F_\bullet) \ ; \quad \text{and} \\ \text{the } \textit{inner} \text{ interpretation } \ [t] \ : \ Z^m \to Z \ . \end{array}$$

Recall (§3.1) that we identify an LTS with $c \in \mathbf{Coalg}_\bullet(F_\bullet)$. Hence the outer interpretation is a "process operator" t acting on LTSs. Therefore the two interpretations combined constitute an instance of the microcosm principle; see §1.2.

Definition 4.1 Let $(\Sigma, \mathcal{R})$ be a GSOS specification, and t be a Σ-term.

1. We define t's *outer interpretation* $[\![t]\!] : (\mathbf{Coalg}_\bullet(F_\bullet))^m \to \mathbf{Coalg}_\bullet(F_\bullet)$ as follows, using functoriality of $(\!|t|\!)$. The function $(\xi_t)_{\vec{X}}$ was introduced in §3.2 as "categorical GSOS rules." Functoriality of thus defined $[\![t]\!]$ is easy.

$$\left(\begin{array}{c} F_\bullet X_1 \\ c_1 \uparrow \\ X_1 \end{array} , \ldots , \begin{array}{c} F_\bullet X_m \\ c_m \uparrow \\ X_m \end{array} \right) \quad \overset{[\![t]\!]}{\longmapsto} \quad \begin{array}{c} F_\bullet\big((\!|t|\!)(X_1, \ldots, X_m) \big) \\ \uparrow (\xi_t)_{X_1,\ldots,X_m} \\ (\!|t|\!)(F_\bullet X_1, \ldots, F_\bullet X_m) \\ \uparrow (\!|t|\!)(c_1, \ldots, c_m) \\ (\!|t|\!)(X_1, \ldots, X_m) \end{array}$$

   The copointedness (§3.1) of the resulting $F_\bullet$-coalgebra is easy, too. It has $(\!|t|\!)(\vec{X})$—the $\mathcal{R}$-state space from §2—as a carrier.

2. We define t's *inner interpretation* $[t] : Z^m \to Z$ as follows. Once we have the outer interpretation $[\![t]\!]$, we can apply it to the m-tuple $(\zeta_\bullet, \ldots, \zeta_\bullet)$ of the final copointed coalgebra $\zeta_\bullet : Z \to F_\bullet Z$. Then the resulting coalgebra $[\![t]\!](\vec{\zeta_\bullet})$ induces a unique "behavior" map into the final one, as on the right. This "behavior" map is almost what we want; we precompose $\theta_{Z,\ldots,Z} : Z^m \to (\!|t|\!)(\vec{Z})$ (from (8)) to it and obtain $[t]$. That is, $[t] := \mathsf{beh}([\![t]\!])(\vec{\zeta_\bullet}) \circ \theta_{\vec{Z}} : Z^m \to Z$.

$$\begin{array}{ccc} F_\bullet((\!|t|\!)(\vec{Z})) & \dashrightarrow & F_\bullet Z \\ {\scriptstyle [\![t]\!](\vec{\zeta_\bullet})}\uparrow & & \zeta_\bullet \uparrow {\scriptstyle \text{final}} \\ (\!|t|\!)(\vec{Z}) & \underset{\mathsf{beh}([\![t]\!](\vec{\zeta_\bullet}))}{\dashrightarrow} & Z \end{array}$$

Proposition 4.2 (Modularity) *Assume we have the same Σ-terms t and t_i as in Prop. 2.11.4. The operations $[\![\_]\!]$ and $[\_]$ are compatible with substitution: given LTSs $c_1, \ldots, c_m \in \mathbf{Coalg}_\bullet(F_\bullet)$ and "behaviors" $z_1, \ldots, z_m \in Z$, we have*

$$\begin{array}{l} [\![\, t[t_i/x_i] \,]\!]\,(c_1, \ldots, c_m) \cong [\![t]\!]\,\big([\![t_1]\!](\vec{c}), \ \ldots, \ [\![t_n]\!](\vec{c}) \big) \ ; \\ [\, t[t_i/x_i] \,]\,(z_1, \ldots, z_m) = [t]\,\big([t_1](\vec{z}), \ \ldots, \ [t_n](\vec{z}) \big) \ . \end{array} \qquad \square$$

We present our main result on compositionality. It relates the outer and inner interpretations. The latter arose from the former via finality (Def. 4.1); the following result follows straightforward from finality, too.

Theorem 4.3 (Compositionality) *Given a Σ-term $x_1, \ldots, x_m \vdash t$ and LTSs $c_1, \ldots, c_m \in \mathbf{Coalg}_\bullet(F_\bullet)$, we have the following diagram commute.*

$$\begin{array}{ccc} X_1 \times \cdots \times X_m & \xrightarrow{\mathsf{beh}(c_1) \times \cdots \times \mathsf{beh}(c_m)} & Z \times \cdots \times Z \\ {\scriptstyle (\theta_t)_{X_1,\ldots,X_m}}\downarrow & & \downarrow{\scriptstyle [t]} \\ (\!|t|\!)(X_1, \ldots, X_m) & \xrightarrow[\mathsf{beh}([\![t]\!](c_1, \ldots, c_m))]{} & Z \end{array}$$

Here $(\theta_t)_{X_1,\ldots,X_m}$ is the "bookkeeping" function from (8); Z is the carrier of the final coalgebra (§3.1). Putting it equationally: for any state $x_i \in X_i$ of each LTS we have

$$\mathsf{beh}\big(\, [\![t]\!](c_1, \ldots, c_m)\,\big)\ (\,\theta_t(x_1, \ldots, x_m)\,) \;=\; [t]\ \big(\,\mathsf{beh}(c_1)(x_1), \ldots,\ \mathsf{beh}(c_m)(x_m)\,\big)\ .$$

That is: the behavior of a composed system $[\![t]\!](\vec{c})$ can be computed from the behaviors $\mathsf{beh}(c_i)$ of its constituent parts, using the inner operator $[t]$. □

Let $\hat{\zeta} : \Sigma^* Z \to Z$ be the (Eilenberg-Moore) Σ^*-algebra on Z induced by a GSOS specification $(\Sigma, \mathcal{R})$, due to [21, Cor. 7.2]. Here Σ^* is the free monad induced by the signature Σ. The following result—claiming that our framework is indeed an extension of (the GSOS fragment of) [21]—holds because our ξ_t in §3, when suitably restricted, coincides with the categorical GSOS in [21].

Theorem 4.4 *Let $x_1, \ldots, x_m \vdash t$ be a Σ-term; let κ_t be the corresponding coprojection $Z^m \hookrightarrow \Sigma^* Z$. For these, the diagram on the right commutes.* □

$$\begin{array}{ccc} \Sigma^* Z & \xrightarrow{\hat{\zeta}} & Z \\ {\scriptstyle \kappa_t}\uparrow & \nearrow_{[t]} & \\ Z^m & & \end{array}$$

5 Conclusions and Future Work

We have extended our previous work [13] so that any process operator specified by GSOS rules can now be interpreted as a component connector that combines LTSs as components. This outer interpretation gives rise to a canonical inner interpretation that coincides with what is derived by the bialgebraic modeling of SOS [21].

Our framework is categorical, hence comes with great potential generality. This includes application to systems other than LTS—like bialgebraic modeling applied to weighted systems [16]—which we wish to pursue. In particular we believe our generic construction of $\mathcal{R}$-state spaces will carry over.

Regular languages and automata seem to be the first computer science example of the microcosm principle. Our current framework fails to include it. One difficulty is: the outer operators (specified in GSOS-like rules) are naturally defined on *NFAs*, while the inner operators is on regular languages that form the final *DFA*. Use of coalgebraic techniques such as trace semantics [12] is being investigated.

The notion of TLG and operations on them (§2) indicates strong relevance of rewriting logic [18] and the theory of *generalized operads/combinatorial species/clones* recently pursued by many authors, including [10]. We are especially interested in the latter, but not only because of TLGs. Roughly we can call their theory *universal algebra in varying contexts*. Here a "context" can be a monoidal one (where variables

are not to be deleted, duplicated or swapped), a symmetric monoidal one (where swapping is allowed), a Cartesian one (where all three are allowed), and so on. In fact such a context can be thought of as algebraic structure itself; hence the theory may be also called *universal (algebra in algebra)*. The microcosm principle then offers a degenerate example of such, where we have the same structure on the two levels.

References

1. Aceto, L., Fokkink, W., Verhoef, C.: Structural operational semantics. In: Bergstra, J., Ponse, A., Smolka, S. (eds.) Handbook of Process Algebra, pp. 197–292. Elsevier (2001)
2. Baez, J.C., Dolan, J.: Higher dimensional algebra III: n-categories and the algebra of opetopes. Adv. Math 135, 145–206 (1998)
3. Baier, C., Blechmann, T., Klein, J., Klüppelholz, S.: A uniform framework for modeling and verifying components and connectors. In: Field, J., Vasconcelos, V.T. (eds.) COORDINATION. Lect. Notes Comp. Sci., vol. 5521, pp. 247–267. Springer (2009)
4. Baier, C., Sirjani, M., Arbab, F., Rutten, J.J.M.M.: Modeling component connectors in reo by constraint automata. Science of Comput. Progr. 61(2), 75–113 (2006)
5. Bliudze, S., Krob, D.: Modelling of complex systems: Systems as dataflow machines. Fundam. Inform. 91(2), 251–274 (2009)
6. Bloom, B., Istrail, S., Meyer, A.R.: Bisimulation can't be traced. Journ. ACM 42(1), 232–268 (1995)
7. Bonsangue, M.M., Clarke, D., Silva, A.: Automata for context-dependent connectors. In: Field, J., Vasconcelos, V.T. (eds.) COORDINATION. Lect. Notes Comp. Sci., vol. 5521, pp. 184–203. Springer (2009)
8. Bruni, R., Lanese, I., Montanari, U.: A basic algebra of stateless connectors. Theor. Comp. Sci. 366(1–2), 98–120 (2006)
9. Clarke, E., Grumberg, O., Peled, D.: Model Checking. MIT Press (1999)
10. Curien, P.L.: Operads, clones, and distributive laws (2008), preprint, available online
11. Hasuo, I.: Tracing Anonymity with Coalgebras. Ph.D. thesis, Radboud Univ. Nijmegen (2008)
12. Hasuo, I., Jacobs, B., Sokolova, A.: Generic trace semantics via coinduction. Logical Methods in Comp. Sci. 3(4:11) (2007)
13. Hasuo, I., Jacobs, B., Sokolova, A.: The microcosm principle and concurrency in coalgebra. In: Foundations of Software Science and Computation Structures. Lect. Notes Comp. Sci., vol. 4962, pp. 246–260. Springer-Verlag (2008)
14. Jacobs, B.: Introduction to coalgebra. Towards mathematics of states and observations (2005), Draft of a book, `www.cs.ru.nl/B.Jacobs/PAPERS`
15. Klin, B.: Bialgebraic methods and modal logic in structural operational semantics. Inf. & Comp. 207(2), 237–257 (2009)
16. Klin, B.: Structural operational semantics for weighted transition systems. In: Palsberg, J. (ed.) Semantics and Algebraic Specification: Essays Dedicated to Peter D. Mosses on the Occasion of His 60th Birthday, Lect. Notes Comp. Sci., vol. 5700, pp. 121–139. Springer-Verlag (2009)
17. Lenisa, M., Power, J., Watanabe, H.: Category theory for operational semantics. Theor. Comp. Sci. 327(1–2), 135–154 (2004)
18. Martí-Oliet, N., Meseguer, J.: Rewriting logic: roadmap and bibliography. Theor. Comp. Sci. 285(2), 121–154 (2002)
19. Plotkin, G.D.: A structural approach to operational semantics (1981), report DAIMI FN-19, Aarhus Univ.

20. Silva, A., Bonchi, F., Bonsangue, M.M., Rutten, J.J.M.M.: Quantitative kleene coalgebras. Inf. & Comp. 209(5), 822–849 (2011)
21. Turi, D., Plotkin, G.: Towards a mathematical operational semantics. In: Logic in Computer Science. pp. 280–291. IEEE, Computer Science Press (1997)

A Appendix

A.1 Omitted Proofs

Proof of Prop. 2.11 For the item 1, by (7) the set $(\!|t|\!)(\vec{X})$ always has a summand $|t|(\vec{X})$ that comes from the identity TLG $\mathrm{id} : t \Rightarrow t$. We set the embedding $(\nu_t)_{\vec{X}}$ so that its domain $|t|(\vec{X})$ gets identified with this summand. For the item 2, given an $\mathcal{R}$-TLG $\rho' : s' \Rightarrow s$ (which induces a summand $|s'|(\vec{X})$ of $(\!|s|\!)(\vec{X})$), we obtain another $\mathcal{R}$-TLG $\rho \bullet \rho' : s' \Rightarrow t$ which then induces a summand $|s'|(\vec{X})$ of $(\!|t|\!)(\vec{X})$. We define the map $(\!|\rho|\!)_{\vec{X}} : (\!|s|\!)(\vec{X}) \to (\!|t|\!)(\vec{X})$ to identify these two summands.

The item 4 is the main technical lemma whose proof calls for a careful inspection of TLGs. By Def. 2.8 of $\mathcal{R}$-TLGs, it is not hard to see that the following map is bijective.

$$\begin{aligned}
&\coprod_{\rho' : t' \Rightarrow t} \prod_{j=1}^{|t'|} \left\{ \ \cdot \xRightarrow{\rho_j} t_{\mathsf{i}_{t'}(j)} \ \right\} \quad \xrightarrow{\cong} \quad \left\{ \ \cdot \xRightarrow{\rho} t[t_1/x_1, \ldots, t_n/x_n] \ \right\} \ , \\
&\big(\rho' : t' \Rightarrow t,\ (\rho_j : t'_j \Rightarrow t_{\mathsf{i}_{t'}(j)})_j \big) \\
&\quad \longmapsto \quad \begin{pmatrix} (t')^\ell[t'_j/x_j]_{j \in [1,|t'|]} & \xRightarrow{\mathrm{id}_{(t')^\ell[\rho_j/x_j]}} & (t')^\ell[t_{\mathsf{i}_{t'}(j)}/x_j]_{j \in [1,|t'|]} \\ \overset{(\dagger)}{=} t'[t_i/x_i]_{i \in [1,n]} & \xRightarrow{\rho[\mathrm{id}_{t_i}/x_i]} & t[t_i/x_i]_{i \in [1,n]} \end{pmatrix}
\end{aligned} \tag{10}$$

Here the notations $|t'|$, $\mathsf{i}_{t'}$ and $(t')^\ell$ are from Def. 2.3; the equality $(\dagger)$ holds because we have

$$\begin{aligned}
t'[t_i/x_i] &= \big((t')^\ell[x_{\mathsf{i}_{t'}(j)}/x_j] \big)[t_i/x_i] && \text{by def. of } (t')^\ell \\
&= (t')^\ell[t_{\mathsf{i}_{t'}(j)}/x_j] && \text{by merging two substitutions.}
\end{aligned}$$

Intuitively, the above bijection (10) breaks up a TLG ρ on the RHS—its codomain is a composed term $t[t_i/x_i]$—into a family of TLGs that are into the components of $t[t_i/x_i]$. This bijection is crucial in the following equational reasoning which proves modularity of $(\!|\_|\!)$.

$$\begin{aligned}
(\!|\, t[t_i/x_i] \,|\!)\,(X_1, \ldots, X_m) \ &= \coprod_{u \xRightarrow{\rho} t[t_i/x_i]} |u|(X_1, \ldots, X_m) \\
&\cong \coprod_{t' \xRightarrow{\rho'} t} \coprod_{t'_1 \xRightarrow{\rho_1} t_{\mathsf{i}_{t'}(1)}} \cdots \coprod_{t'_{|t'|} \xRightarrow{\rho_{|t'|}} t_{\mathsf{i}_{t'}(|t'|)}} \big|\, (t')^\ell[t'_j/x_j] \,\big|(\vec{X}) && \text{using (10)} \\
&\cong \coprod_{t' \xRightarrow{\rho'} t} \coprod_{t'_1 \xRightarrow{\rho_1} t_{\mathsf{i}_{t'}(1)}} \cdots \coprod_{t'_{|t'|} \xRightarrow{\rho_{|t'|}} t_{\mathsf{i}_{t'}(|t'|)}} |(t')^\ell| \Big(|t'_1|(\vec{X}), \ldots, |t'_{|t'|}|(\vec{X}) \Big) \\
&\cong \coprod_{t' \xRightarrow{\rho'} t} \coprod_{t'_1 \xRightarrow{\rho_1} t_{\mathsf{i}_{t'}(1)}} \cdots \coprod_{t'_{|t'|} \xRightarrow{\rho_{|t'|}} t_{\mathsf{i}_{t'}(|t'|)}} |t'_1|(\vec{X}) \times \cdots \times |t'_{|t'|}|(\vec{X}) && (\ddagger) \\
&\cong \coprod_{t' \xRightarrow{\rho'} t} \Bigg[\Big(\coprod_{t'_1 \xRightarrow{\rho_1} t_{\mathsf{i}_{t'}(1)}} |t'_1|(\vec{X}) \Big) \times \cdots \times \Big(\coprod_{t'_{|t'|} \xRightarrow{\rho_{|t'|}} t_{\mathsf{i}_{t'}(|t'|)}} |t'_{|t'|}|(\vec{X}) \Big) \Bigg] && (*)
\end{aligned}$$

$$
\begin{aligned}
&\cong \coprod_{t' \overset{p'}{\Rightarrow} t} \left((\!| t_{\mathrm{i}_{t'}(1)} |\!)(\vec{X}) \times \cdots \times (\!| t_{\mathrm{i}_{t'}(|t'|)} |\!)(\vec{X}) \right) && \text{Def. 2.10 of } (\!|\_|\!) \\
&\cong \coprod_{t' \overset{p'}{\Rightarrow} t} |t'| \Big((\!| t_1 |\!)(\vec{X}), \ldots, (\!| t_n |\!)(\vec{X}) \Big) && \text{Def. 2.9 of } |t'| \\
&= (\!| t |\!) \left((\!| t_1 |\!)(\vec{X}), \ldots, (\!| t_n |\!)(\vec{X}) \right) \ .
\end{aligned}
$$

Here the equality (‡) holds by Def. 2.9 and linearity of $(t')^{\ell}$ (Def. 2.3); $(*)$ holds since $\times$ distributes over $\coprod$ in $\mathbf{Sets}$. $\square$