Algebraic Meta-Theory of Processes with Data

Daniel Gebler
Department of Computer Science, VU University Amsterdam (VU),
De Boelelaan 1081a, NL-1081 HV Amsterdam, The Netherlands

Eugen-Ioan Goriac
Department of Theoretical Computer Science, Reykjavik University (RU),
Menntavegur 1, IS-101, Reykjavik, Iceland

Mohammad Reza Mousavi
Center for Research on Embedded Systems (CERES), Halmstad University
Kristian IV:s väg 3, SE-302 50, Halmstad, Sweden

There exists a rich literature of rule formats guaranteeing different algebraic properties for formalisms with a Structural Operational Semantics (SOS). Moreover, there exist a few approaches for automatically deriving axiomatizations characterizing strong bisimilarity of processes. To our knowledge, this literature has never been extended to the setting with data (e.g., to model storage and memory). We show how the rule formats for algebraic properties can be exploited in a generic manner in the setting with the data. Moreover, we introduce a new approach of deriving sound and ground-complete axiom schemata for a notion of bisimilarity with data, called stateless bisimilarity, based on intuitive auxiliary function symbols for handling the store component.

1 Introduction

Algebraic properties capture some key features of programming and specification constructs and can be used both as design principles (for the semantics of such constructs) as well as for verification of programs and specifications built using them. When given the semantics of a language, inferring properties such as commutativity, associativity and unit element, as well deriving sets of axioms for reasoning on the behavioral equivalence of two processes constitute the cornerstone of process algebras [8, 39] and play essential roles in several disciplines for behavioral modeling and analysis such as term rewriting [7] and model checking [10].

For formalisms with a Structural Operational Semantics (SOS), there exists a rich literature of meta-theorems guaranteeing key algebraic properties (commutativity [31], associativity [18], zero and unit elements [4], idempotence [1], and distributivity [4]) by means of restrictions on the syntactic shape of the transition rules. At the same time, for GSOS, a restricted yet expressive form of SOS specifications, one can obtain a sound and ground-complete axiomatization modulo strong bisimilarity [2]. Supporting some form of data (memory or store) is a missing aspect of these existing meta-theorems, which bars applicability to the semantics of numerous programming languages and formalism that do feature these aspects in different forms.

In this paper we provide a natural and generic link between the meta-theory of algebraic properties and axiomatizations, and SOS with data for which we consider that one data state models the whole memory. Namely, we move the data terms in SOS with data to the labels and instantiate them to closed terms; we call this process currying. Currying allows us to apply directly the existing rule formats for algebraic properties on the curried SOS specifications (which have process terms as states and triples of (datum, label, datum) as labels). We also present a new way of automatically deriving sound and ground-complete axiomatization schemas modulo strong bisimilarity for the curried systems. It turns out that
strong bisimilarity for the curried SOS specification coincides with the notion of stateless bisimilarity in the original SOS specifications with data. The latter notion is extensively studied in [33] and used, among others, in [13] [22] [11] [12]. (This notion, in fact, coincides with the notion of strong bisimilarity proposed for Modular SOS in [28] Section 4.1.) Hence, using the existing rule formats, we can obtain algebraic laws for SOS specification with data that are sound with respect to stateless bisimilarity, as well as other weaker notions of initially stateless bisimilarity and statebased bisimilarity, studied in [33].

Related work. SOS with data and store has been extensively used in specifying semantics of programming and specification languages, dating back to the original work of Plotkin [37] [38]. Since then, several pieces of work have been dedicated to providing a formalization for SOS specification frameworks allowing to include data and store and reason over it. The current paper builds upon the approach proposed in [33] (originally published as [30]).

The idea of moving data from the configurations (states) of operational semantics to labels is reminiscent of Modular SOS [28] [27], Enhanced SOS [20], the Tile Model [21], and context-dependent-behaviour framework of [17]. The idea has also been applied in instances of SOS specification, such as those reported in [9] [11] [34]. The present paper contributes to this body of knowledge by presenting a generic transformation from SOS specifications with data and store (as part of the configuration) to Transition System Specifications [24] [14]. The main purpose of this generic transformation is to enable exploiting the several existing rule formats defined on transition system specifications on the result of transformation and then, transform the results back to the original SOS specifications (with data and store in the configuration) using a meaningful and well-studied notion of bisimilarity with data. Our transformation is also inspired by the translation of SOS specifications of programming languages into rewriting logic, e.g., [26] [25].

Structure of the paper. The rest of this paper is organized as follows. In Section 2, we recall some basic definitions regarding SOS specifications and behavioral equivalences. In Section 3, we present the currying technique and formulate the theorem regarding the correspondence between strong and stateless bisimilarity. In Section 4, we show how to obtain sound and ground-complete axiomatizations modulo strong bisimilarity for the curried systems. We apply these results to Linda [16], a coordination language from the literature chosen as case study in Section 5 and show how key algebraic properties of the operators defined in the language semantics are derived. We also generate a sound and ground-complete axiomatization of stateless bisimilarity on Linda programs. We conclude the paper in Section 6 by summarizing the results and presenting some directions for future work.

2 Preliminaries

2.1 Transition System Specifications

We assume a multisorted signature $\Sigma$ with designated and distinct sorts $P$ and $D$ for processes and data, respectively. Moreover, we assume infinite and disjoint sets of process variables $V_P$ (typical members: $x_P, y_P, x_P, y_P, \ldots$) and data variables $V_D$ (typical members: $x_D, y_D, x_D, y_D, \ldots$), ranging over their respective sorts $P$ and $D$.

Process and data signatures, denoted respectively by $\Sigma_P \subseteq \Sigma$ and $\Sigma_D \subseteq \Sigma$, are sets of function symbols with fixed arities. We assume in the remainder that the function symbols in $\Sigma_D$ take only
parameters of the sort $\Sigma_D$, while those in $\Sigma_P$ can take parameters both from $\Sigma_P$ and $\Sigma_D$, as in practical specifications of systems with data process function symbols do take data terms as their parameters.

Terms are built using variables and function symbols by respecting their domains of definition. The sets of open process and data terms are denoted by $T(\Sigma_P)$ and $T(\Sigma_D)$, respectively. Disjointness of process and data variables is mostly for notational convenience. Function symbols from the process signature are typically denoted by $f_P$, $g_P$, $f_p$, and $g_p$. Process terms are typically denoted by $t_P$, $t'_P$, and $t_{P_i}$. Function symbols from the data signature are typically denoted by $f_D$, $f'_D$, and $f_{D_i}$, and data terms are typically denoted by $t_D$, $t'_D$, and $t_{D_i}$. The sets of closed process and data terms are denoted by $T(\Sigma_P)$ and $T(\Sigma_D)$, respectively. Closed process and data terms are typically denoted by $p$, $q$, $p'$, $p_i$, $p'_i$, and $d, e, d', d_i, d'_i$, respectively. We denote process and data substitutions by $\sigma$, $\sigma'$, and $\xi$, $\xi'$, respectively. We call substitutions $\sigma : V_P \to T(\Sigma_P)$ process substitutions and $\xi : V_D \to T(\Sigma_D)$ data substitutions. A substitution replaces a variable in an open term with another (possibly open) term. Notions of open and closed and the concept of substitution are lifted to formulae in the natural way.

**Definition 1** (Transition System Specification). Consider a signature $\Sigma$ and a set of labels $L$ (with typical members $l,l',l_0,\ldots$). A positive transition formula is a triple $(t,t',l)$, where $t,t' \in T(\Sigma)$ and $l \in L$, written $t \xrightarrow{l} t'$, with the intended meaning: process $t$ performs the action labeled as $l$ and becomes process $t'$. A transition rule is defined as a tuple $(H,\alpha)$, where $H$ is a set of formulae and $\alpha$ is a formula. The formulae from $H$ are called premises and the formula $\alpha$ is called the conclusion. A transition rule is mostly denoted by $\frac{H}{\alpha}$ and has the following generic shape:

$$
\frac{\{t_i \xrightarrow{l_{ij}} t_{ij} \mid i \in I, j \in J_i\}}{t \xrightarrow{l} t'}
$$

where $I, J_i$ are sets of indexes, $t,t',t_i,t_{ij} \in T(\Sigma)$, and $l_{ij} \in L$. A transition system specification (abbreviated TSS) is a tuple $(\Sigma, L, R)$ where $\Sigma$ is a signature, $L$ is a set of labels, and $R$ is a set of transition rules of the provided shape.

We extend the shape of a transition rule to handle process terms paired with data terms in the following manner:

$$
\frac{\{(t_{P_i},t_{D_{ij}}) \xrightarrow{l_{ij}} (t_{P_i},t_{D_{ij}}) \mid i \in I, j \in J_i\}}{(t_P,t_{D_D}) \xrightarrow{l'} (t'_P,t'_D)}
$$

where $I, J_i$ are index sets, $t_P,t'_P,t_{P_i},t_{P_{i,j}} \in T(\Sigma_P)$, $t_D,t'_D,t_{D_i},t_{D_{ij}} \in T(\Sigma_D)$, and $l_{ij} \in L$. A transition system specification with data is a triple $T = (\Sigma_P \cup \Sigma_D, L, R)$ where $\Sigma_P$ and $\Sigma_D$ are process and data signatures respectively, $L$ is a set of labels, and $R$ is a set of transition rules handling pairs of process and data terms.

**Definition 2.** Let $T$ be a TSS with data. A proof of a formula $\phi$ from $T$ is an upwardly branching tree whose nodes are labelled by formulae such that

1. the root node is labelled by $\phi$, and
2. if $y$ is the label of a node $q$ and the set $\{\psi_i \mid i \in I\}$ is the set of labels of the nodes directly above $q$, then there exist a deduction rule $\frac{\chi_1 \mid i \in I}{\chi}$, a process substitution $\sigma$, and a data substitution...
$\xi$ such that the application of these substitutions to $\chi$ gives the formula $\psi$, and for all $i \in I$, the application of the substitutions to $\chi_i$ gives the formula $\psi_i$.

Note that by removing the data substitution $\xi$ from above we obtain the definition for proof of a formula from a standard TSS. The notation $\mathcal{T} \vdash \phi$ expresses that there exists a proof of the formula $\phi$ from the TSS (with data) $\mathcal{T}$. Whenever $\mathcal{T}$ is known from the context, we will write $\phi$ directly instead of $\mathcal{T} \vdash \phi$.

The induced LTS has as state space the term algebra and as transition rules all derivable axioms (closed transitions with empty premise).

2.2 Bisimilarity

In this paper we use two notions of equivalence over processes, one for standard transition system specifications and one for transition system specifications with data. Stateless bisimilarity is the natural counterpart of strong bisimilarity, used in different formalisms such as [13, 22, 11, 12].

Definition 3 (Strong Bisimilarity [35]). Consider a TSS $\mathcal{T} = (\Sigma_P, L, R)$. A relation $R \subseteq T(\Sigma_P) \times T(\Sigma_P)$ is a strong bisimulation if and only if it is symmetric and $\forall_{p,q}(p,q) \in R \Rightarrow \forall_{l,p'} p \xrightarrow{l} p' \Rightarrow \exists_{q',q} q \xrightarrow{l} q'$ $\land$ $(q,q') \in R$. Two closed terms $p$ and $q$ are strongly bisimilar, denoted by $p \leftrightarrow^T q$ if there exists a strong bisimulation relation $R$ such that $(p,q) \in R$.

Definition 4 (Stateless Bisimilarity [33]). Consider a TSS with data $\mathcal{T} = (\Sigma_P \cup \Sigma_D, L, R)$. A relation $R_{sl} \subseteq T(\Sigma_P) \times T(\Sigma_P)$ is a stateless bisimulation if and only if it is symmetric and $\forall_{p,q}(p,q) \in R_{sl} \Rightarrow \forall_{d,l,p',d'} (p,d) \xrightarrow{l} (p',d') \Rightarrow \exists_{q',q} (q,d) \xrightarrow{l} (q',d')$ $\land$ $(p',q') \in R_{sl}$. Two closed process terms $p$ and $q$ are stateless bisimilar, denoted by $p \leftrightarrow_{T}^T q$, if there exists a stateless bisimulation relation $R_{sl}$ such that $(p,q) \in R_{sl}$.

2.3 Rule Formats for Algebraic Properties

As already stated, the literature on rule formats guaranteeing algebraic properties is extensive. For the purpose of this paper we show the detailed line of reasoning only for the commutativity of binary operators, while, for readability, we refer to the corresponding papers and theorems for the other results in Section 5.

Definition 5 (Commutativity). Given a TSS and a binary process operator $f$ in its process signature, $f$ is called commutative w.r.t. $\sim$, if the following equation is sound w.r.t. $\sim$:

$$f(x_0, x_1) = f(x_1, x_0).$$

Definition 6 (Commutativity format [6]). A transition system specification over signature $\Sigma$ is in comm-form format with respect to a set of binary function symbols $COMM \subseteq \Sigma$ if all its $f$-defining transition rules with $f \in COMM$ have the following form

$$(c) \quad \frac{\{x_j \xrightarrow{l_{ij}} y_{ij} \mid i \in I\}}{f(x_0, x_1) \xrightarrow{l} t}$$

where $j \in \{0, 1\}$, $I$ is an arbitrary index set, and variables appearing in the source of the conclusion and target of the premises are all pairwise distinct. We denote the set of premises of $(c)$ by $H$ and the
conclusion by α. Moreover, for each such rule, there exist a transition rule \((c')\) of the following form in the transition system specification

\[
\begin{array}{c}
(c') \\
H' \\
f(x'_0, x'_1) \xrightarrow{l} t'
\end{array}
\]

and a bijective mapping (substitution) \(\overline{h}\) on variables such that

- \(\overline{h}(x'_0) = x_1\) and \(\overline{h}(x'_1) = x_0\),
- \(\overline{h}(t') \sim_{cc} t\) and
- \(\overline{h}(h') \in H\), for each \(h' \in H'\),

where \(\sim_{cc}\) means equality up to swapping of arguments of operators in COMM in any context. Transition rule \((c')\) is called the commutative mirror of \((c)\).

**Theorem 7** (Commutativity for comm-form [6]). If a transition system specification is in comm-form format with respect to a set of operators COMM, then all operators in COMM are commutative with respect to strong bisimilarity.

### 2.4 Sound and ground-complete axiomatizations

In this section we recall several key aspects presented in [2], where the authors provide a thorough procedure for converting any GSOS language definition that disjointly extends the language for trees to a finite complete equational axiom system which characterizes strong bisimilarity. It is important to note that we work with the GSOS format because it guarantees that bisimilarity is a congruence and that the transition relation is finitely branching [14].

**Definition 8** (Positive GSOS rule format). Consider a process signature \(\Sigma_P\). A positive GSOS rule \(\rho\) over \(\Sigma_P\) has the shape:

\[
\begin{array}{c}
\{x_i \xrightarrow{l_{ij}} y_{ij} \mid i \in I, j \in J_i\} \\
f(x_1, \ldots, x_n) \xrightarrow{l} C[\vec{x}, \vec{y}]
\end{array}
\]

where all variables are distinct, \(f\) is an operation symbol form \(\Sigma_P\) with arity \(n\), \(I \subseteq \{1, \ldots, n\}\), \(J_i\) finite for each \(i \in I\), \(l_{ij}\) and \(l\) are labels standing for actions ranging over a given set denoted by \(L\), and \(C[\vec{x}, \vec{y}]\) is a \(\Sigma_P\)-context with variables including at most the \(x_i\)'s and \(y_{ij}\)'s.

A finite tree term \(t\) is built according to the following grammar:

\[
t ::= \delta \mid l.t \ (\forall l \in L) \mid t + t.
\]

We denote this signature by \(\Sigma_{\text{BCCSP}}\). Intuitively, \(\delta\) represents a process that does not exhibit any behaviour, \(s + t\) is the nondeterministic choice between the behaviours of \(s\) and \(t\), while \(l.t\) is a process that first performs action \(l\) and behaves like \(t\) afterwards. The operational semantics that captures this intuition is given by the rules of BCCSP [23]:

\[
\begin{align*}
l.x & \xrightarrow{l} x \\
x & \xrightarrow{l} x' \\
y & \xrightarrow{l} y' \\
x + y & \xrightarrow{l} x'
\end{align*}
\]
Definition 9 (Axiom System). An axiom (or equation) system $E$ over a signature $\Sigma$ is a set of equalities of the form $t = t'$, where $t, t' \in T(\Sigma)$. An equality $t = t'$, for some $t, t' \in T(\Sigma)$, is derivable from $E$, denoted by $E \vdash t = t'$, if and only if it is in the smallest congruence relation over $\Sigma$-terms induced by the equalities in $E$.

We consider the axiom system $E_{\text{BCCSP}}$ which consists of the following axioms:

\[
\begin{align*}
x + y &= y + x & x + x &= x \\
(x + y) + z &= x + (y + z) & x + 0 &= x.
\end{align*}
\]

Theorem 10. $E_{\text{BCCSP}}$ is sound and ground-complete for bisimilarity on $T(\Sigma_{\text{BCCSP}})$. That is, it holds that $E_{\text{BCCSP}} \vdash p = q$ iff $p \equiv_{\text{BCCSP}} q$ for any two ground terms $p$ and $q \in T(\Sigma_{\text{BCCSP}})$.

Definition 11 (Disjoint extension). A GSOS system $G'$ is a disjoint extension of a GSOS system $G$, written $G \sqsubset G'$, if the signature and the rules of $G'$ include those of $G$, and $G'$ does not introduce new rules for operations in $G$.

In [2] it is elaborated how to obtain an axiomatization for a GSOS system $G$ that disjointly extends BCCSP. For technical reasons the procedure involves initially transforming $G$ into a new system $G'$ that conforms to a restricted version of the GSOS format, named smooth and distinctive. We avoid presenting this restricted format as the method proposed in Section 4 allows us to obtain the axiomatization without the need of transforming the initial system $G$.

3 Currying Data

We apply the process of currying [40] known from functional programming to factor out the data from the source and target of transitions and enrich the label to a triple capturing the data flow of the transition. This shows that for specifying behavior and data of dynamic systems, the data may be freely distributed over states (as part of the process terms) or system dynamics (action labels of the transition system), providing a natural correspondence between the notions of stateless bisimilarity and strong bisimilarity. An essential aspect of our approach is that the process of currying is a syntactic transformation defined on transition system specifications (and not a semantic transformation on transition systems); this allows us to apply meta-theorems from the meta-theory of SOS and obtain semantic results by considering syntactic shape of (transformed) SOS rules.

Definition 12 (Currying and Label Closure). Consider the TSS with data $T = (\Sigma_P \cup \Sigma_D, L, R)$ and transition rule $\rho \in R$ of the shape $\rho = \{(t_{P_i}, t_{D_{ij}}) \xrightarrow{l_{ij}} (t_{P_{ij}}, t_{D_{ij}}) \mid i \in I, j \in J_i\}$.

The curried version of $\rho$ is the rule $\rho^c = \{(t_{P_i}, t_{D_{ij}}) \xrightarrow{(t_{D_{ij}}, l_{ij}, t_{D_{ij}})} t_{P_{ij}} \mid i \in I, j \in J_i\}$. We further define $R^c = \{\rho^c \mid \rho \in R\}$ and $L^c = \{(t_{D_{ij}}, l, t'_{P_{ij}}) \mid l \in L, t_{D_{ij}}, t'_{P_{ij}} \in T(\Sigma_D)\}$. The curried version of $T$ is defined as $T^c = (\Sigma_P, L^c, R^c)$.

By $\rho^c_{\xi} = \{(t_{P_i}, t_{D_{ij}}, \xi(t_{D_{ij}})) \xrightarrow{\xi(l_{ij}), \xi(t_{D_{ij}})} t_{P_{ij}} \mid i \in I, j \in J_i\}$ we denote the closed label version of $\rho^c$ with respect to the closed data substitution $\xi$. By $\text{cl}(\rho^c)$ we denote the set consisting of all closed label versions of $\rho^c$, i.e. $\text{cl}(\rho^c) = \{\rho^c_{\xi} \mid \rho^c \in R^c, \xi \text{ is a closed data substitution}\}$. We further define $\text{cl}(R^c) = \{\text{cl}(\rho^c) \mid \rho^c \in R^c\}$. 


Our goal is to reduce the notion of stateless bisimilarity between two closed process terms with data terms to strong bisimilarity by means of currying the TSS with data and closing its labels. The following theorem states how this goal can be achieved.

**Theorem 13.** Given a TSS $T = (\Sigma, L, D)$ with data, for each two closed process terms $p, q \in T(\Sigma_P)$, $p \leftrightarrow_D^T q$ if and only if $p \leftrightarrow^{cl(T^c)} q$.

### 4 Axiomatizing GSOS with Data

In this section, we provide an axiomatization schema for reasoning about stateless bisimilarity. We find it easier to work directly with curried systems instead of systems with data because this allows us to adjust the method introduced in [2] by considering the set of more complex labels that integrate the data, as presented in Section 3.

Consider TSS with data $T = (\Sigma_P \cup \Sigma_D, L, R)$. For an operation $f \in \Sigma_P$, consider a rule $\rho \in R_f$ in the GSOS format extended with the data component. As introduced in Definition 12, $cl(\rho^c)$ – the curried and closed label version of $\rho$ – can be an infinite set of rules. As axiomatization process involves obtaining a summand of an axiom schema for every rule in $cl(\rho^c)$, we may end up with an infinite number of summands. For this reason we choose to operate symbolically with the rules in $cl(\rho^c)$, with means that we operate just with $\rho^c$, the curried version of the rule.

We now adapt the definition for the notion of strong bisimilarity to the new setting.

**Definition 14.** Consider a TSS $T = (\Sigma_P \cup \Sigma_D, L, R)$, which means that $T^c = (\Sigma_P, L^c, R^c)$. A relation $R \subseteq T(\Sigma_P) \times T(\Sigma_P)$ is a strong bisimulation if and only if it is symmetric and $\forall p, q(p, q) \in R \Rightarrow \forall s_D, l, s'_D, p'_l \exists t_D, t'_D, q, q' \in R \land \forall \xi (s_D, t_D, s'_D, t'_D, q, q') \in \Sigma_P, cl(\rho^c)$ – the curried and closed label version of $R$ – can be an infinite set of rules. As axiomatization process involves obtaining a summand of an axiom schema for every rule in $cl(\rho^c)$, we may end up with an infinite number of summands. For this reason we choose to operate symbolically with the rules in $cl(\rho^c)$, with means that we operate just with $\rho^c$, the curried version of the rule.

We consider the equation system $E_D$ for the data terms algebra $\Sigma_D$ is $\omega$-complete, which means that two open terms $s_D, t_D$ are provably equal ($E_D \models s_D = t_D$) whenever all the closed instantiations of this equality can be derived from $E_D, \forall \xi (s_D) = \xi (t_D)$ [5]. This assumption basically allows us to reduce Definition 14 to Definition 3.

BCCSP is extended to a setting with data, BCCSP$_D$. We do this by extending the signature for process terms $\Sigma_{BCCSP}$ to $\Sigma_{BCCSP_D}$ with two auxiliary operators for handling the store, named check and update. Terms over $\Sigma_{BCCSP_D}$ are built according to the following grammar:

$$t_p ::= \delta \mid l.t_p \mid \forall l \in L \mid (check(t_D, t_P) \mid update(t_D, t_P) \mid t_P + t_P).$$

Intuitively, operation $check(t_D, t_P)$ makes sure that before executing $t_P$ the store has the value $t_D$, and $update(t_D, t_P)$ changes the store value to $t_D$ after executing process $t_P$. The prefix operation does not affect the store. We directly provide the curried set of rules defining the semantics of BCCSP$_D$:

$$\frac{l.x_p \cdot (t_D, l, t_D)}{x_p} \quad \frac{(t_D, l, t_D)}{x_p} \quad \frac{x_p \cdot (l, x_p) \cdot (t_D, l, t_D)}{x'_p} \quad \frac{update(t_D, x_p) \cdot (s_D, l, t_D)}{x'_p} \quad \frac{check(t_D, x_p) \cdot (t_D, l, t_D)}{x'_p} \quad \frac{x_p \cdot (s_D, l, t_D)}{x'_p}$$
Lemma 16 (Towards ground completeness). The following theorem is proved in the standard fashion.

Theorem 15 (Soundness). For each two terms \( s, t \) in \( \mathcal{T}(\Sigma_{\text{BCCSP}}) \) it holds that if \( E_{\text{BCCSP}} \vdash s = t \) then \( s \sim_{\text{BCCSP}} t \).

Lemma 16 (Towards ground completeness). For each closed term \( p \), that if \( p \xrightarrow{(d,l,d')} p' \), then \( p = p + \text{update}(d', \text{check}(d,l,p')) \).

Proof. By induction on the number of symbols appearing in \( p \). We proceed with a case distinction on the head symbol of \( p \).

- Assume that \( p \) is \( \delta \), which is the base case of induction; this case is vacuous, because \( p \) cannot make any transition.

- Assume that \( p \) is of the form \( l.p_0 \); then, we have that \( d' = d \) and \( p' = p_0 \), because \( p \xrightarrow{(d,l,d)} p_0 \), for each arbitrary closed data term \( d \), is the only type of transition that \( p \) affords. Hence, we obtain the following equational derivation:

\[
\begin{align*}
p & \overset{\text{def.}}{=} l.p_0 & \text{axiom (lc)} & \Rightarrow & l.p_0 + \text{update}(d, \text{check}(d,l,p_0)) & \overset{\text{def.}}{=} p_0 = p'.' \\
& \Rightarrow & p + \text{update}(d, \text{check}(d,l,p')) & \overset{d' = d}{=} & p + \text{update}(d', \text{check}(d,l,p'))
\end{align*}
\]

- Assume that \( p \) is of the form \( \text{check}(d'',p'') \); then, we have that \( d'' = d \) and \( p'' \xrightarrow{(d,l,d')} p' \).

\[
\begin{align*}
p & \overset{\text{def.}}{=} \text{check}(d'',p'') & \text{ind. hyp.} & \Rightarrow & \text{check}(d'',p'' + \text{check}(d'', \text{update}(d', l,p'))) & \overset{\text{axiom (nc)}}{=} \\
& \Rightarrow & \text{check}(d'' + \text{check}(d'', \text{update}(d', l,p'))) & \overset{\text{def.}}{=} p \\
& \Rightarrow & \text{check}(d'' + \text{check}(d'', \text{update}(d', l,p'))) & \overset{\text{axiom (cc)}}{=} \\
& \Rightarrow & \text{check}(d'' \text{update}(d', l,p')) & \overset{d'' = d}{=} \\
& \Rightarrow & \text{check}(d, \text{update}(d', l,p')) & \overset{\text{axiom (cu)}}{=} & p + \text{update}(d, \text{check}(d', l,p'))
\end{align*}
\]

- Assume that \( p \) is of the form \( \text{update}(d'',p'') \); then, we have that \( d'' = d' \) and \( p'' \xrightarrow{(d,l,d'')} p' \).
Definition 21 (Axiomatization schema). We consider that a process term is in head normal form (h.n.f.) if it is of the form $\sigma(\vec{p},\vec{q})$, where $\vec{p}$ is of the form $\vec{x}_i \cdot y_{ij} \cdot i \in I, j \in J_i$.

Lemma 18. For each two closed terms $p$ and $q$, if $p \leftrightarrow q$, then $p = q$.

Remark 19. We consider that a process term is of the form $p_0 + p_1$, also assume without loss of generality that the transition of $p$ is due to $p_0$, i.e., $p_0 \cdot (d,l,d') \rightarrow p'$.

Theorem 17 (Completeness). For each two closed terms $p$ and $q$, if $p \leftrightarrow q$, then $p = q$.

Proof. In order to prove this theorem, it suffices to show that Lemma 18 holds.

Lemma 18 holds.

Proof. We do this by an induction on the number of symbols appearing in $p + q$, and using Lemma 16. The minimum number of symbols is 3, namely in the case of $\delta + \delta$.

Definition 20 (Head Normal Form). Let $\Sigma P$ be a signature such that $\Sigma_{BCCSP_D} \subseteq \Sigma P$. A term $t$ in $T(\Sigma P)$ is in head normal form (for short, h.n.f.) if $t = \sum_{i \in I} \text{update}(t_{D_i}', \text{check}(t_{D_i}, l, t_{P_i}))$, where for every $i \in I$, $t_{D_i}, t_{D_i}' \in T(\Sigma D)$, $t_{P_i} \in T(\Sigma D)$, $l_i \in L$. The empty sum ($I = \emptyset$) is denoted by the deadlock constant $\delta$.

When given a signature $\Sigma P$ that includes $\Sigma_{BCCSP_D}$, the purpose of an axiomatization for a term $t \in T(\Sigma P)$ is to derive another term $t'$ such that $t \leftrightarrow t'$ and $t' \in T(\Sigma_{BCCSP_D})$.

Definition 21 (Axiomatization schema). Consider a TSS $T^c = (\Sigma P, L^c, R^c)$ such that $BCCSP_D \subseteq T^c$.

By $E_{T^c}$ we denote the axiom system that extends $E_{BCCSP_D}$ with the following axiom scheme for every operation $f$ in $T$, parameterized over the vector of closed process terms $\vec{p}$:

$$ f(\vec{p}) = \sum \left\{ \text{update}(t_{D}, \text{check}(t_{D}, l, C[\vec{p}, \vec{q}])) \mid \rho^c = \frac{H}{f(\vec{x})} (t_{D}, t_{D}'), C[\vec{x}, \vec{y}] \in R^c \text{ and } \cdot \vec{p} = \sigma(\vec{x}) \cdot \vec{q} = \sigma(\vec{y}) \cdot \sqrt[\vee](\vec{p}, \rho^c) \right\}, $$

where $\sqrt[\vee]$ is defined as $\sqrt[\vee](\vec{p}, \rho^c) = \bigwedge_{p_k \in \vec{p}} \sqrt[\vee](p_k, k, \rho^c)$,

and $\sqrt[\vee]'(p_k, k, \{ x_i \rightarrow (t_{D_i}, l_{D_i}, t_{D_i}') \rightarrow y_{ij} \mid i \in I, j \in J_i \}) = f(\vec{x}) (t_{D_i}, t_{D_i}', C[\vec{x}, \vec{y}])$,

if $k \in I$ then $\forall j \in J_k \exists p', p'' \in T(\Sigma P) E_{BCCSP_D} + p_k = \text{update}(t_{D_k}, \text{check}(t_{D_k}, l_{k_j}, p')) + p''$. 

Assume that $p$ is of the form $p_0 + p_1$, also assume without loss of generality that the transition of $p$ is due to $p_0$, i.e., $p_0 \cdot (d,l,d') \rightarrow p'$. 

Proof.

In order to prove this theorem, it suffices to show that Lemma 18 holds.

Lemma 18 holds.
Intuitively, the axiom transforms \( f(\vec{p}) \) into a sum of closed terms covering all its execution possibilities. In order to obtain them we iterate through the set of \( f \)-defining rules and check if \( \vec{p} \) satisfies their hypotheses by means of \( \checkmark \). \( \checkmark \) makes sure that, for a given rule, every component of \( \vec{p} \) is a term with enough action prefixed summands that satisfy the hypotheses associated to that component. Note that the axiomatization is built in such a way that it always derives terms in head normal form. Also note that the sum on the right hand side is finite because of our initial assumption that the transition relation is finitely branching.

The reason why we conceived the axiomatization in this manner is of practical nature. Our past experience shows that this type of schemas may bring terms to their normal form faster than finite axiomatizations. Aside this, we do not need to transform the initial system, as presented in [2].

**Theorem 22.** Consider a TSS \( T^c = (\Sigma_P, L^c, R^c) \) such that \( \text{BCCSP}_D^c \sqsubseteq T^c \). \( E_{T^c} \) is sound and ground-complete for strong bisimilarity on \( T(\Sigma_P) \).

**Proof.** It is easy to see that, because of the head normal form of the right hand side of every axiom, the completeness of the axiom schema reduces to the completeness proof for bisimilarity on \( T(\Sigma_{\text{BCCSP}_D^c}) \).

In order to prove the soundness, we denote, for brevity, the right hand side of the schema in Definition 21 by \( \text{RHS} \). For a fixed data substitution \( \xi \) we need to show that \( \xi(f(\vec{p})) \cong^T \xi(\text{RHS}) \).

Let us first prove that if \( f(\vec{p}) \) performs a transition then it can be matched by \( \text{RHS} \). Consider a rule \( \rho^* \subseteq \rho^c \) that can be applied for \( f(\vec{p}) \): \( \rho^c \xi = \{ x_i \xrightarrow{\xi(t_{D_i}), l_i, \xi(t_{D_{ij}})} y_{ij} \mid i \in I, j \in J_i \} \). Then it holds that \( \xi(f(\vec{p})) \xrightarrow{\xi(t_{D_i}), l_i, \xi(t_{D_{ij}})} \xi(C[\vec{x}, \vec{y}]) \) and at the same time all of the rule’s premises are met. This means that \( p_i = \sum_{j \in J_i} (\xi(t_{D_i}), l_i, \xi(t_{D_{ij}})) \cdot p_{ij} + p' \) for some \( p' \) and \( p_{ij} \)'s. It is easy to see that all the conditions for \( \checkmark \) are met, so \( (\xi(t_{D_i}), l, \xi(t_{D'})), \xi(C[\vec{p}, \vec{q}]) \) is a summand of \( \text{RHS} \), and therefore it holds that \( \xi(f(\vec{p})) \xrightarrow{\xi(t_{D_i}), l_i, \xi(t_{D_{ij}})} \xi(C[\vec{p}, \vec{q}]) \), which matches the transition from \( f(\vec{p}) \).

The proof for the fact that \( f(\vec{p}) \) can match any of the transitions of \( \text{RHS} \) is similar.

## 5 Case Study: The Coordination Language Linda

In what follows we present the semantics, properties and axiomatization of a core prototypical language.

The provided specification defines a structural operational semantics for the coordination language Linda; the specification is taken from [33] and is a slight adaptation of the original semantics presented in [15] (by removing structural congruences and introducing a terminating process \( \epsilon \)). Process constants (atomic process terms) in this language are \( \epsilon \) (for terminating process), \( \text{ask}(u) \) and \( \text{nask}(u) \) (for checking existence and absence of tuple \( u \) in the shared data space, respectively), \( \text{tell}(u) \) (for adding tuple \( u \) to the space) and \( \text{get}(u) \) (for taking tuple \( u \) from the space). Process composition operators in this language include nondeterministic choice (\( + \)), sequential composition (\( ; \)) and parallel composition (\( \parallel \)). The data signature of this language consists of a constant \( \{ \} \) for the empty multiset and a class of unary function symbols \( \cup \{u\} \), for all tuples \( u \), denoting the union of a multiset with a singleton multiset containing tuple \( u \). The operational state of a Linda program is denoted by \( (p, t_D) \) where \( p \) is a process term in the above syntax and \( t_D \) is a multiset modeling the shared data space.

The transition system specification defines one relation \( \rightarrow \) and one predicate \( \downarrow \). Note that \( \rightarrow \) is unlabeled, unlike the other relations considered so far. Without making it explicit, we tacitly consider
the termination predicate $\downarrow$ as a binary transition relation $\triangleright$ with the pair $(x_P, x_D)$, where $x_P$ and $x_D$ are fresh yet arbitrary process and data variables, respectively.

Below we provide a table consisting of both the original and the curried and closed label versions of the semantics of Linda on the left and, respectively, on the right.

\[
\begin{array}{ll}
(1) & (\epsilon, t_D) \downarrow \\
(2) & (\text{ask}(u), t_D \cup \{u\}) \rightarrow (\epsilon, t_D \cup \{u\}) \\
(3) & (\text{tell}(u), t_D) \rightarrow (\epsilon, t_D \cup \{u\}) \\
(4) & (\text{get}(u), t_D \cup \{u\}) \rightarrow (\epsilon, t_D) \\
(5) & (\text{nask}(u), t_D) \rightarrow (\epsilon, t_D)[u \notin t_D] \\
(6) & (x_P, t_D) \downarrow (x_P + y_P, t_D) \downarrow \\
(7) & (y_P, t_D) \downarrow (x_P + y_P, t_D) \downarrow \\
(8) & (x_P, t_D) \rightarrow (x'_P, t'_D) \\
(9) & (x_P, y_P, t_D) \rightarrow (y'_P, t'_D) \\
(10) & (x_P, y_P, t_D) \rightarrow (x'_P, y'_P, t_D') \\
(11) & (x_P, y_P, t_D) \rightarrow (y'_P, t'_D') \\
(12) & (x_P, y_P, t_D) \rightarrow (y'_P, t'_D') \\
(13) & (x_P, y_P, t_D) \rightarrow (x'_P, y'_P, t'_D') \\
(14) & (y_P, t_D) \rightarrow (y'_P, t'_D') \\
(15) & (x_P, y_P, t_D) \rightarrow (x'_P, y'_P, t'_D')
\end{array}
\]

\[
\begin{array}{ll}
(1c) & \epsilon \downarrow \\
(2c) & \text{ask}(u) \downarrow (d, \ldots, d \cup \{u\}) \rightarrow \epsilon \\
(3c) & \text{tell}(u) \downarrow (d, \ldots, d \cup \{u\}) \rightarrow \epsilon \\
(4c) & \text{get}(u) \downarrow (d, \ldots, d \cup \{u\}) \rightarrow \epsilon \\
(5c) & \text{nask}(u) \downarrow (d, \ldots, d \cup \{u\}) \rightarrow \epsilon \\
(6c) & x_P \downarrow x_P + y_P \downarrow \\
(7c) & y_P \downarrow x_P + y_P \downarrow \\
(8c) & x_P \rightarrow (d, \ldots, d) \rightarrow x'_P \\
(9c) & y_P \rightarrow (d, \ldots, d) \rightarrow y'_P \\
(10c) & x_P \rightarrow (d, \ldots, d) \rightarrow x'_P \\
(11c) & y_P \rightarrow (d, \ldots, d) \rightarrow y'_P \\
(12c) & x_P \rightarrow (d, \ldots, d) \rightarrow x'_P \\
(13c) & y_P \rightarrow (d, \ldots, d) \rightarrow y'_P \\
(14c) & x_P \rightarrow (d, \ldots, d) \rightarrow x'_P \\
(15c) & y_P \rightarrow (d, \ldots, d) \rightarrow y'_P
\end{array}
\]

In the curried SOS rules, $d$ and $d'$ are arbitrary closed data terms, i.e., each transition rule given in the curried semantics represents a (possibly infinite) number of rules for each and every particular $d, d' \in T(\Sigma_D)$. It is worth noting that by using the I-MSOS framework \[29\] we can present the curried system...
without explicit labels at all as they are propagated implicitly between the premises and conclusion.

Consider transition rules \((6_c), (7_c), (8_c), \) and \((9_c)\): they are the only \(+\)-defining rules and they fit in the Comm-form format of Definition \(6\). It follows from Theorem \(7\) that the equation \(x + y = y + x\) is sound with respect to strong bisimilarity in the curried semantics. Subsequently, following Theorem \(13\) we have that the previously given equation is sound with respect to stateless bisimilarity in the original semantics. (Moreover, we have that \((x_0 + x_1, d) = (x_1 + x_2, d)\) is sound with respect to state-based bisimilarity for all \(d \in T(\Sigma_D)\).)

Following a similar line of reasoning, we get that \(x \parallel y = y \parallel x\) is sound with respect to stateless bisimilarity in the original semantics.

In addition, we derived the following axioms for the semantics of Linda, using the meta-theorems stated in the third column of the table. The semantics of sequential composition in Linda is identical to the sequential composition (without data) studied in Example 9 of \([18]\); there, it is shown that this semantics conforms to the ASSOC-DE SIMONE format introduced in \([18]\) and hence, associativity of sequential composition follows immediately. Also semantics of nondeterministic choice falls within the scope of the ASSOC-DE SIMONE format (with the proposed coding of predicates), and hence, associativity of nondeterministic choice follows (note that in \([18]\) nondeterministic choice without termination rules is treated in Example 1; moreover, termination rules in the semantics of parallel composition are discussed in Section 4.3 and shown to be safe for associativity). Following a similar line of reasoning associativity of parallel composition follows from the conformance of its rules to the ASSOC-DE SIMONE format of \([18]\). Idempotence for \(+\) can be obtained, because rules \((6_c), (7_c)\) and \((8_c), (9_c)\) are choice rules \([1]\) Definition 40\) and the family of rules \((6_c)\) to \((9_c)\) for all data terms \(d\) and \(d'\) ensure that the curried specification is in idempotence format with respect to the binary operator \(+\). The fact that \(\epsilon\) is unit element for \(\parallel\) is proved similarly as in \([3]\), Example 10.

<table>
<thead>
<tr>
<th>Property</th>
<th>Axiom</th>
<th>Meta-Theorem</th>
</tr>
</thead>
<tbody>
<tr>
<td>Associativity for (\parallel)</td>
<td>(x \parallel (y \parallel z) = (x \parallel y) \parallel z)</td>
<td>Theorem 1 of ([18])</td>
</tr>
<tr>
<td>Associativity for (+)</td>
<td>(x + (y + z) = (x + y) + z)</td>
<td>Theorem 1 of ([18])</td>
</tr>
<tr>
<td>Associativity for (\parallel)</td>
<td>(x \parallel (y \parallel z) = (x \parallel y) \parallel z)</td>
<td>Theorem 1 of ([18])</td>
</tr>
<tr>
<td>Idempotence for (+)</td>
<td>(x + x = x)</td>
<td>Theorem 42 of ([1])</td>
</tr>
<tr>
<td>Unit element for (\parallel)</td>
<td>(\epsilon \parallel x = x)</td>
<td>Theorem 3 of ([4])</td>
</tr>
<tr>
<td>Distributivity of (+) over (\parallel)</td>
<td>((x + y) \parallel z = (x \parallel y) + (x \parallel z))</td>
<td>Theorem 3 of ([4])</td>
</tr>
</tbody>
</table>

Let us now discuss the axiomatization process form Section 4 applied for Linda.

It is important to note, first of all, that the operation \(\text{ask}\) cannot be axiomatized. This is due to its semantics being given by a set of curried rules with a side condition \(\text{ask}(u) : \text{CHECK}(t_D) \rightarrow \text{UPDATE}(u, l, t_D)\), which our method cannot currently handle.

Secondly, we need to ensure that the rule system is a disjoint extension of \(\text{BCCSP}^\parallel_D\). This means that the system also includes the semantics for the operations \(\text{check}\) and \(\text{update}\). At the same time, the termination predicate satisfiability is encoded by means of transitions. We introduce, therefore, the action \(\downarrow\) with the purpose that whenever \(\downarrow\cdot x\) is a summand within a given term \(t\) it means that \(t\) terminates. \(\epsilon\) thus becomes syntactic sugar for \(\downarrow\cdot \delta\).

The transition system specification is adapted so that it has only the relation \(\rightarrow\). This time \(\rightarrow\) is labeled with triples of the form \((t_D, l, t_D')\), where \(l\) is either the termination action \(\downarrow\) or the unlabeled action \(-\).

\[
\begin{align*}
\text{ask}(u) & \xrightarrow{(t_D \cup \{u\}, \neg\cdot t_D \cup \{u\})} \downarrow \cdot \delta \\
\text{tell}(u) & \xrightarrow{(t_D, \neg\cdot t_D \cup \{u\})} \downarrow \cdot \delta \\
\text{get}(u) & \xrightarrow{(t_D \cup \{u\}, \neg\cdot t_D)} \downarrow \cdot \delta
\end{align*}
\]
The axiomatization schema for the new operations follows naturally by applying Definition [21]:

\[
\begin{align*}
\text{ask}(u) &= \text{update}(t_D \cup \{u\}, \text{check}(t_D \cup \{u\}, - \downarrow \delta)) \\
\text{tell}(u) &= \text{update}(t_D \cup \{u\}, \text{check}(t_D, - \downarrow \delta)) \\
\text{get}(u) &= \text{update}(t_D, \text{check}(t_D \cup \{u\}, - \downarrow \delta))
\end{align*}
\]

\[
\begin{align*}
x; y &= \sum x = \text{update}(t'_{D'}, \text{check}(t_D, -x')) + x'' \\
&\sum x = \text{update}(s'_{D'}, \text{check}(s_D, -y')) + x'' \\
&\sum y = \text{update}(t'_{D'}, \text{check}(t_D, -y')) + y''
\end{align*}
\]

\[
\begin{align*}
x \parallel y &= \sum x = \text{update}(t'_{D'}, \text{check}(t_D, -x')) + x'' \\
&\sum y = \text{update}(t'_{D'}, \text{check}(t_D, -y')) + y'' \\
&\sum y = \text{update}(t'_{D'}, \text{check}(t_D, -y')) + y''
\end{align*}
\]

6 Conclusions

In this paper, we have proposed a generic technique for extending the meta-theory of algebraic properties to SOS with data, memory or store. In a nutshell, the presented technique allows for focusing on the structure of the process (program) part in SOS rules and ignoring the data terms in order to obtain algebraic properties, as well as, a sound and ground complete set of equations w.r.t. stateless bisimilarity. We have demonstrated the applicability of our method by means of the well known coordination language Linda.

It is also worth noting that one can check whether a system is in the process-tyft format presented in [32] in order to infer that stateless bisimilarity is a congruence, and if this is the case, then strong bisimilarity over the curried system is also a congruence. Our results are applicable to a large body of existing operators in the literature and make it possible to dispense with several lengthy and laborious soundness proofs in the future.

Our approach can be used to derive algebraic properties that are sound with respect to weaker notions of bisimilarity with data, such as initially stateless and statebased bisimilarity [33]. We do expect to obtain stronger results, e.g., for zero element with respect to statebased bisimilarities, by scrutinizing data dependencies particular to these weaker notions. We would like to study coalgebraic definitions of the notions of bisimilarity with data (following the approach of [41]) and develop a framework for SOS with data using the bialgebraic approach.

The presented technique allows for TSS to factor out state information to the level of state dynamics (action labels) for settings where states are independent pairs of processes and data. It would be very interesting to investigate under which conditions the currying could be applied in less restrictive settings, e.g., where processes and process data are intertwined in one single term.
Acknowledgements. We thank Luca Aceto, Peter Mosses, and Michel Reniers for their valuable comments on earlier versions of the paper.

References

Concurrency Theory (CONCUR’08), Lecture Notes in Computer Science 5201, Springer-Verlag, Berlin, Germany, Toronto, Canada, pp. 447–461.


A Proof of Theorem 13

Given a TSS $T = (\Sigma, L, D)$ with data, for each two closed process terms $p, q \in T(\Sigma_P)$, $p \leftrightarrow_T q$ if and only if $p \leftrightarrow_{\text{data}} T q$ if and only if $p \leftrightarrow_{\text{data}} p'$.

Proof. Before we proceed with the proof of the theorem, we state and prove the following auxiliary lemma.

Lemma 23. For each two closed process terms $p, p' \in T(\Sigma_P)$, each two closed data terms $d, d' \in T(\Sigma_D)$ and each label $l \in L$, it holds that $T \vdash (p, d) \rightarrow (p', d')$ if and only if $T^c \vdash p \rightarrow (d, d') \rightarrow p'$.

Proof. We split the bi-implication into two implications and prove them below:

$\Rightarrow$ By induction on the depth of the proof for $T \vdash (p, d) \rightarrow (p', d')$. Since the induction basis is a special case of the induction step (in which the last transition rule in the proof has no premises), we dispense with stating the induction basis separately. Assume that the last transition rule in the proof is

$$\rho = \frac{(t_{P_i}, t_{D}) \xrightarrow{l_{ij}} (t_{P_{ij}}, t_{D_{ij}}) \mid i \in I, j \in J_i}{(t_p, t_D) \xrightarrow{l} (t'_p, t'_D)}.$$

Then there exists a closed process substitution $\sigma$ and a closed data substitution $\xi$ such that $\sigma(t_P) = p, \sigma(t'_P) = p', \xi(t_D) = d, \xi(t'_D) = d'$ and moreover $T \vdash (\sigma(t_P), \xi(t_D)) \xrightarrow{l_{ij}} (\sigma(t_{P_{ij}}), \xi(t_{D_{ij}}))$ with a shallower proof, for each $i \in I$ and $j \in J_i$.

It follows from Definition 12 that there exists a transition rule

$$\rho^c_{\xi} = \frac{\{t_{P_i} \xrightarrow{(\xi(t_{D}), l_{ij}, \xi(t_{D_{ij}}))} t_{P_{ij}} \mid i \in I, j \in J_i\}}{t_p \xrightarrow{(\xi(t_D), l, \xi(t'_D))} t'_p},$$

in the transition rules of $T^c$. The induction hypothesis applies to the premises of $\rho$ under $\sigma$, and hence, we obtain $T^c \vdash \sigma(t_P) \xrightarrow{(\xi(t_{D}), l, \xi(t_{D_{ij}}))} \sigma(t_{P_{ij}})$ for every $i \in I, j \in J_i$. We therefore infer that $T^c \vdash \sigma(t_P) \xrightarrow{(\xi(t_D), l, \xi(t'_D))} \sigma(t'_P)$, hence $T^c \vdash p \rightarrow (d, d') \rightarrow p'$.

$\Leftarrow$ By induction on the depth of the proof for $T^c \vdash p \rightarrow (d, d') \rightarrow p'$. We dispense with stating the induction basis separately in this case too. Assume that the last transition rule in the proof is

$$\rho' = \frac{\{t_{P_i} \xrightarrow{(d, l_{ij}, d_{ij})} t_{P_{ij}} \mid i \in I, j \in J_i\}}{t_p \xrightarrow{(d, l, d')} t'_p},$$

there exists a closed process substitution $\sigma$ such that $\sigma(t_P) = p, \sigma(t'_P) = p'$ and, moreover, it holds that $T^c \vdash \sigma(t_P) \xrightarrow{(d, l_{ij}, d_{ij})} \sigma(t_{P_{ij}})$ with a shallower proof, for each $i \in I$ and $j \in J_i$.

It follows from Definition 12 that there exists a transition rule

$$\rho = \frac{(t_{P_i}, t_{D}) \xrightarrow{l_{ij}} (t_{P_{ij}}, t_{D_{ij}}) \mid i \in I, j \in J_i}{(t_p, t_D) \xrightarrow{l} (t'_p, t'_D)}.$$
in the transition rules of \( \mathcal{T} \) and a data substitution \( \xi \) such that \( \rho' = \rho'_\xi \), and hence, \( \xi(t_D) = d \), \( \xi(t'_D) = d' \), \( \xi(t_{D_i}) = d_i \), \( \xi(t_{D_{ij}}) = d_{ij} \). The induction hypothesis applies to the premises of \( \rho' \), and hence, we obtain \( \mathcal{T} \vdash (\sigma(t_P), \xi(t_{D_i})) \xrightarrow{1} (\sigma(t_{P_{ij}}), \xi(t_{D_{ij}})) \) for every \( i \in I, j \in J_i \). We therefore infer that \( \mathcal{T} \vdash (\sigma(t_P), \xi(t_D)) \xrightarrow{L} (\sigma(t'_P), \xi(t'_D)) \), hence \( \mathcal{T} \vdash (p, d) \xrightarrow{L} (p', d') \).

Now we are ready to state the proof of Theorem 13. Let us first prove that \( p \xleftrightarrow{T^c} q \) if and only if \( p \xleftrightarrow{T} q \). Again we split the bi-implication into two implications and prove them separately below.

\[ \Rightarrow \] Since \( p \xleftrightarrow{T} q \), there exists a stateless bisimulation relation \( R \) w.r.t. \( \mathcal{T} \) such that \( (p, q) \in R \). We claim that \( R \) is also a strong bisimulation relation w.r.t. \( \mathcal{T}^c \). \( R \) is symmetric because it is a stateless bisimulation relation and hence, it remains to show that the following transfer condition holds:

If \( \mathcal{T}^c \vdash p \xrightarrow{\langle d, l, d' \rangle} p' \), then \( \mathcal{T}^c \vdash q \xrightarrow{\langle d, l, d' \rangle} q' \) for some \( q' \) such that \( (p', q') \in R \).

It follows from \( \mathcal{T}^c \vdash p \xrightarrow{\langle d, l, d' \rangle} p' \) and the right-to-left implication in Lemma 23 that \( \mathcal{T} \vdash (p, d) \xrightarrow{L} (p', d') \).

Since \( p \xleftrightarrow{T} q \), we have that \( \mathcal{T} \vdash (q, d) \xrightarrow{L} (q', d') \), for some \( q' \) such that \( (p', q') \in R \). It follows from the latter transition and the left-to-right implication in Lemma 23 that \( \mathcal{T}^c \vdash q \xrightarrow{\langle d, l, d' \rangle} q' \) and we already had that \( (p', q') \in R \), which concludes the proof of the left-to-right implication.

\[ \Leftarrow \] Symmetric to above. This implication is similar to the informal procedure of turning Modular SOS specifications into SOS specifications in [28, Section 3.9]; there it is mentioned that formalizing the transformation and proving a formal proof of correspondence between the original and the transformed specification is left for future work.

It follows trivially that \( p \xleftrightarrow{T^c} q \) if and only if \( p \xleftrightarrow{cl(T^c)} q \), by identifying the appropriate closed data substitution used in the label closure process.

\[ \square \]

**B The Hybrid Process Algebra HyPA**

In [19], a process algebra is presented for the description of hybrid systems, i.e., systems with both discrete events and continuous change of variables. The process signature of HyPA consists of the following process constants and functions:

- process constants: \( \delta, \epsilon, (a)_{a \in A}, (c)_{c \in C} \);
- unary process functions: \( (d \gg \cdot)_{d \in D}, (\partial H (\cdot))_{H \subseteq A} \);
- binary process functions: \( \oplus, \odot, \odot, \triangleright, \triangleright, \|, \|, \|, \|, \|, \|, \| \).

We refrain from giving further information about the intended meaning of the sets \( A, C, \) and \( D \), and the meaning of the process constants and functions as these are irrelevant to currying. The data state consists of mappings from model variables to values, denoted by \( Val \). The data signature is not made explicit.

The transition system specification defines the following predicate and relations:

- a ‘termination’-predicate \( \checkmark \);
- a family of ‘action-transition’ relations \( (\cdot \xrightarrow{\delta} \cdot)_{i \in A \times Val} \);
- a family of ‘flow-transition’ relations \( (\cdot \xrightarrow{\sigma} \cdot)_{\sigma \in T \rightarrow Val} \).
Also, the meaning of the set $T$ is irrelevant for our purposes. The ‘termination’-predicate is again tacitly considered as transition to a pair of fresh variables. Note that the curried TSS introduces for every predicate a family of predicates for each data term.

The transition rules are given below. Note that in the following semantics, each deduction rule with multiple transition formulae in the conclusion is actually a rule schema representing a separate deduction rule for each and every formula in the conclusion.

\[(1) \frac{(\epsilon, \nu)}{\sqrt{\mathcal{L}}}, \quad (2) \frac{(a, \nu) a' \nu}{\sqrt{\mathcal{L}}}, \quad (3) \frac{(c, \nu) \sigma (c, \nu(t)) [\nu, \sigma] \models \epsilon\{\nu} [\text{dom}(\sigma) = [0, l]],\]

\[(4) \frac{(x, \nu') \sqrt{\mathcal{L}}}{(d \gg x, \nu) \sqrt{\mathcal{L}}} [\nu, \nu'] \models \epsilon\{\nu}, \quad (5) \frac{(x, \nu') \vdash (y, \nu'')}{(d \gg x, \nu) \vdash (y, \nu'')} [\nu, \nu'] \models \epsilon\{\nu},\]

\[(6) \frac{(x_0, \nu) \sqrt{\mathcal{L}}}{(x_0 \ominus x_1, \nu) \sqrt{\mathcal{L}}}, \quad (7) \frac{(x, \nu) \vdash (y, \nu')}{(x_0 \ominus x_1, \nu) \vdash (y, \nu')},\]

\[(8) \frac{(x_0, \nu) \sqrt{\mathcal{L}} (y_0, \nu) \sqrt{\mathcal{L}}}{(x_0 \odot y_0, \nu) \sqrt{\mathcal{L}}}, \quad (9) \frac{(x_0, \nu) \vdash (y, \nu')}{(x_0 \odot x_1, \nu) \vdash (y, \nu')},\]

\[(10) \frac{(x_0, \nu) \sqrt{\mathcal{L}} (x_1, \nu) \vdash (y, \nu')}{(x_0 \odot x_1, \nu) \vdash (y, \nu')}, \quad (11) \frac{(x_0, \nu) \vdash (y, \nu')}{(x_0 \gg x_1, \nu) \vdash (y, \nu')},\]

\[(12) \frac{(x_0, \nu) \vdash (y, \nu')}{(x_0 \gg x_1, \nu) \vdash (y, \nu')}, \quad (13) \frac{(x_1, \nu) \vdash (y, \nu')}{(x_0 \gg x_1, \nu) \vdash (y, \nu')},\]

\[(14) \frac{(x_0, \nu) \vdash (y, \nu')}{(x_0 \gg x_1, \nu) \vdash (y, \nu')}, \quad (15) \frac{(x_0, \nu) \sqrt{\mathcal{L}} (x_1, \nu) \sqrt{\mathcal{L}}}{(x_0 \parallel x_1, \nu) \sqrt{\mathcal{L}}},\]

\[(16) \frac{(x_0, \nu) \vdash (y, \nu')}{(x_0 \parallel x_1, \nu) \vdash (y, \nu')}, \quad (17) \frac{(x_0, \nu) \sigma (y, \nu') (x_1, \nu) \vdash (y, \nu')}{(x_0 \parallel x_1, \nu) \vdash (y, \nu')},\]

\[(18) \frac{(x_0, \nu) \sigma (y, \nu') (x_1, \nu) \vdash (y, \nu')}{(x_0 \parallel x_1, \nu) \vdash (y, \nu')}, \quad (19) \frac{(x_0, \nu) \sigma (y, \nu') (x_1, \nu) \vdash (y, \nu')}{(x_0 \parallel x_1, \nu) \vdash (y, \nu')} [a'' = a \gamma a'],\]

\[(20) \frac{(x_0, \nu) \sigma (y, \nu') (x_1, \nu) \vdash (y, \nu')}{(x_0 \parallel x_1, \nu) \vdash (y, \nu')}, \quad (21) \frac{(x_0, \nu) \vdash (y, \nu')}{(x_0 \parallel x_1, \nu) \vdash (y, \nu')} [a'' = a \gamma a'],\]

\[(22) \frac{(x_0, \nu) \sigma (y, \nu') (x_1, \nu) \vdash (y, \nu')}{(x_0 \parallel x_1, \nu) \vdash (y, \nu')}, \quad (23) \frac{(x_0, \nu) \vdash (y, \nu')}{(x_0 \parallel x_1, \nu) \vdash (y, \nu')} [a'' = a \gamma a'],\]

\[(24) \frac{(x_0, \nu) \sigma (y, \nu') (x_1, \nu) \vdash (y, \nu')}{(x_0 \parallel x_1, \nu) \vdash (y, \nu')}, \quad (25) \frac{(x_0, \nu) \vdash (y, \nu')}{(x_0 \parallel x_1, \nu) \vdash (y, \nu')} [a'' = a \gamma a'],\]

\[(26) \frac{(x_0, \nu) \sigma (y, \nu') (x_1, \nu) \vdash (y, \nu')}{(x_0 \parallel x_1, \nu) \vdash (y, \nu')}, \quad (27) \frac{(x_0, \nu) \vdash (y, \nu')}{(x_0 \parallel x_1, \nu) \vdash (y, \nu')} [a'' = a \gamma a'].\]
The curried version of the semantics of HyPA is given below.

\[
\begin{align*}
(1c) & \quad \epsilon \vdash \nu, \\
(2c) & \quad \frac{(x, \nu) \overset{a, \nu'}{\rightarrow} (y, \nu'')}{(\partial_H(x), \nu) \overset{a, \nu'}{\rightarrow} (\partial_H(y), \nu'')} \quad [a \notin H], \\
(3c) & \quad \frac{[(\nu, \sigma) \models t \sigma]}{c} \quad c \quad [(\nu, \sigma) \models t \sigma][\text{dom}(\sigma) = [0, t]], \\
(4c) & \quad \frac{x \vdash \nu'}{d \gg x \vdash \nu}, \\
(5c) & \quad \frac{x \vdash (\nu', \nu'')}{y \vdash (\nu', \nu'')} \quad [\nu, \nu' \models \nu'], \\
(6c) & \quad \frac{x_0 \vdash \nu \quad x_1 \vdash \nu}{x_0 \uplus x_1 \vdash \nu}, \\
(7c) & \quad \frac{x_0 \vdash (\nu, \nu') \quad x_1 \vdash (\nu, \nu')}{y \vdash (\nu, \nu')}, \\
(8c) & \quad \frac{x_0 \vdash \nu \quad y_0 \vdash \nu}{x_0 \circ y_0 \vdash \nu}, \\
(9c) & \quad \frac{x_0 \vdash (\nu, \nu')}{y \vdash (\nu, \nu')}, \\
(10c) & \quad \frac{x_0 \vdash (\nu, \nu')}{y \vdash (\nu, \nu')}.
\end{align*}
\]
On HyPA process terms, in [19], a notion of robust bisimilarity is defined that, for HyPA, coincides with our definition of stateless bisimilarity.

Nondeterministic choice and sequential composition have the same semantics in HyPA as in Linda and hence their algebraic properties follow from an identical line of reasoning. Commutativity of parallel composition follows from the fact the commutative mirror of each rule derived from the rule schemata (15c), (16c), (17c), (18c), and (19c) is represented by the same rule schema. Also for parallel composition, all deduction rules but (16c) and (17c) are similar to Linda (modulo renaming of labels); deduction rules (16c) and (17c) are, respectively, communication and left-choice + testing rules in the ASSOC DE SIMONE format of [18] (with the addition of testing operators) and their combination trivially satisfies the constraints of this format (the antecedents of all constraints of this format are false) and hence, associativity of parallel composition follows from Theorem 2 of [18]. Idempotence for $\oplus$ is given because rules (6c) and (7c) are choice rules [1, Definition 40] and the family of rules (6c), (7c) for all data terms $d$ and $d'$ ensure that the curried specification is in idempotence format with respect to the binary operator $\oplus$. Finding zero and unit elements for the operations $\oplus$, $\odot$, $\triangleright$ and $\triangleright$ is similar to the examples presented in [3].

We derived the following axioms for the semantics of HyPA, using the meta-theorems stated in the third column of the table.
<table>
<thead>
<tr>
<th>Property</th>
<th>Axiom</th>
<th>Meta-Theorem</th>
</tr>
</thead>
<tbody>
<tr>
<td>Commutativity for $\oplus$</td>
<td>$x_0 \oplus x_1 = x_1 \oplus x_0$</td>
<td>Theorem 7</td>
</tr>
<tr>
<td>Commutativity for $||$</td>
<td>$x_0 | x_1 = x_1 | x_0$</td>
<td>Theorem 7</td>
</tr>
<tr>
<td>Associativity for $\oplus$</td>
<td>$x_0 \oplus (x_1 \oplus x_2) = (x_0 \oplus x_1) \oplus x_2$</td>
<td>Theorem 1 of 18</td>
</tr>
<tr>
<td>Associativity for $\odot$</td>
<td>$x_0 \odot (x_1 \odot x_2) = (x_0 \odot x_1) \odot x_2$</td>
<td>Theorem 1 of 18</td>
</tr>
<tr>
<td>Associativity for $|$</td>
<td>$x_0 | (x_1 | x_2) = (x_0 | x_1) | x_2$</td>
<td>Theorem 2 of 18</td>
</tr>
<tr>
<td>Idempotence for $\oplus$</td>
<td>$x_0 \oplus x_0 = x_0$</td>
<td>Theorem 42 of 11</td>
</tr>
<tr>
<td>Unit element for $\oplus$</td>
<td>$\delta \oplus x = x$</td>
<td>Theorem 3 of 3</td>
</tr>
<tr>
<td>Unit element for $\odot$</td>
<td>$\epsilon \odot x = x$</td>
<td>Theorem 3 of 3</td>
</tr>
<tr>
<td>Unit element for $\triangleright$</td>
<td>$\delta \triangleright x = x \triangleright \delta = x$</td>
<td>Theorem 3 of 3</td>
</tr>
<tr>
<td>Unit element for $\triangleright$</td>
<td>$x \triangleright \delta = x$</td>
<td>Theorem 3 of 3</td>
</tr>
<tr>
<td>Zero element for $\odot$</td>
<td>$\delta \odot x = \delta$</td>
<td>Theorem 5 of 3</td>
</tr>
<tr>
<td>Zero element for $\triangleright$</td>
<td>$\delta \triangleright x = \delta \triangleright \epsilon &gt; x = \epsilon$</td>
<td>Theorem 5 of 3</td>
</tr>
<tr>
<td>Distributivity of $\odot$ over $\oplus$</td>
<td>$x_0 \odot (x_1 \oplus x_2) = x_0 \odot x_1 \oplus x_0 \odot x_2$</td>
<td>Theorem 3 of 4</td>
</tr>
<tr>
<td>Distributivity of $\triangleright$ over $\oplus$</td>
<td>$(x_0 \oplus x_1) \triangleright x_2 = x_0 \triangleright x_2 \oplus x_1 \triangleright x_2$</td>
<td>Theorem 3 of 4</td>
</tr>
<tr>
<td>Distributivity of $\triangleright$ over $\odot$</td>
<td>$(x_0 \odot x_1) \triangleright x_2 = x_0 \triangleright x_2 \odot x_1 \triangleright x_2$</td>
<td>Theorem 3 of 4</td>
</tr>
<tr>
<td>Distributivity of $\triangleright$ over $\triangleright$</td>
<td>$\partial_H (x_0 \triangleright x_1) = \partial_H (x_0) \triangleright \partial_H (x_1)$</td>
<td>Theorem 3 of 4</td>
</tr>
<tr>
<td>Distributivity of $\triangleright$ over $\odot$</td>
<td>$\partial_H (x_0 \odot x_1) = \partial_H (x_0) \odot \partial_H (x_1)$</td>
<td>Theorem 3 of 4</td>
</tr>
<tr>
<td>Distributivity of $\triangleright$ over $\triangleright$</td>
<td>$\partial_H (x_0 \triangleright x_1) = \partial_H (x_0) \triangleright \partial_H (x_1)$</td>
<td>Theorem 3 of 4</td>
</tr>
<tr>
<td>Distributivity of $\triangleright$ over $\odot$</td>
<td>$\partial_H (x_0 \odot x_1) = \partial_H (x_0) \odot \partial_H (x_1)$</td>
<td>Theorem 3 of 4</td>
</tr>
<tr>
<td>Distributivity of $\triangleright$ over $\triangleright$</td>
<td>$\partial_H (x_0 \triangleright x_1) = \partial_H (x_0) \triangleright \partial_H (x_1)$</td>
<td>Theorem 3 of 4</td>
</tr>
<tr>
<td>Distributivity of $\triangleright$ over $\odot$</td>
<td>$\partial_H (x_0 \odot x_1) = \partial_H (x_0) \odot \partial_H (x_1)$</td>
<td>Theorem 3 of 4</td>
</tr>
<tr>
<td>Distributivity of $\triangleright$ over $\triangleright$</td>
<td>$\partial_H (x_0 \triangleright x_1) = \partial_H (x_0) \triangleright \partial_H (x_1)$</td>
<td>Theorem 3 of 4</td>
</tr>
<tr>
<td>Distributivity of $\triangleright$ over $\odot$</td>
<td>$\partial_H (x_0 \odot x_1) = \partial_H (x_0) \odot \partial_H (x_1)$</td>
<td>Theorem 3 of 4</td>
</tr>
<tr>
<td>Distributivity of $\triangleright$ over $\triangleright$</td>
<td>$\partial_H (x_0 \triangleright x_1) = \partial_H (x_0) \triangleright \partial_H (x_1)$</td>
<td>Theorem 3 of 4</td>
</tr>
</tbody>
</table>