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
ICE-TCS, School of Computer Science, Reykjavik University,
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. 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 data. Moreover, we introduce a new approach for 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. We do restrict, however, the axiomatization to the setting where the store component is only given in terms of constants.

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 behavioural equivalence of two processes constitute one of the cornerstones of process algebras [7, 39] and play essential roles in several disciplines for behavioural modeling and analysis such as term rewriting [6] and model checking [9].

For formalisms with a Structural Operational Semantics (SOS), there exists a rich literature on meta-theorems guaranteeing key algebraic properties (commutativity [32], associativity [17], zero and unit elements [4], idempotence [1], and distributivity [3]) by means of restrictions on the syntactic shape of the transition rules. At the same time, for GSOS [13], 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 formalisms 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

J. Borgström and B. Luttik (Eds.): Combined Workshop on Expressiveness in Concurrency and Structural Operational Semantics (EXPRESS/SOS 2013)
EPTCS 120, 2013, pp. 63–77, doi:10.4204/EPTCS.120.6

the form (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 for the setting in which the data component is characterized by constants. 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 [31] and used, among others, in [12, 20, 10, 11]. (This notion, in fact, coincides with the notion of strong bisimilarity proposed for Modular SOS in [27, 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 the weaker notions of initially stateless bisimilarity and statebased bisimilarity, studied in [31].

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 [36, 37]. Since then, several pieces of work have been dedicated to providing a formalization for SOS specification frameworks allowing one to include data and store and reason over it. The current paper builds upon the approach proposed in [31] (originally published as [29]).

The idea of moving data from the configurations (states) of operational semantics to labels is reminiscent of Modular SOS [26, 27], Enhanced SOS [18], the Tile Model [19], and context-dependent-behaviour framework of [16]. The idea has also been applied in instances of SOS specification, such as those reported in [8, 10, 33]. 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 [13, 22]. The main purpose of this generic transformation is to enable exploiting the several existing rule formats defined on transition system specifications on the results of the 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, see e.g., [24, 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 behavioural 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 those curried systems for which the domain of the data component is a finite set of constants. We apply the currying technique to Linda [15], 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 conclude the paper in Section 6, by summarizing the results and presenting some directions for future work.

2 Preliminaries

2.1 Transition Systems Specifications

We assume a multisorted signature Σ 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: $x_P, y_P, x_{P_i}, y_{P_i} \ldots$) and data variables V_D (typical members: $x_D, y_D, x_{D_i}, y_{D_i} \ldots$), 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 Σ_D take only parameters of the sort Σ_D, while those in Σ_P can take parameters both from Σ_P and Σ_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 $\mathbb{T}(\Sigma_P)$ and $\mathbb{T}(\Sigma_D)$, 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_i} and g_{P_i}. 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 σ, σ', and ξ, ξ', respectively. We call substitutions $\sigma : V_P \to \mathbb{T}(\Sigma_P)$ process substitutions and $\xi : V_D \to \mathbb{T}(\Sigma_D)$ 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 Σ and a set of labels L (with typical members $l, l', l_0, \ldots$). A* positive transition formula *is a triple (t, l, t'), where $t, t' \in \mathbb{T}(\Sigma)$ 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, α), where H is a set of formulae and α is a formula. The formulae from H are called* premises *and the formula α is called the* conclusion*. A transition rule is mostly denoted by $\dfrac{H}{\alpha}$ and has the following generic shape:*

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

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

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

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

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

1. *the root node is labelled by ϕ, and*

2. *if ψ 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 $\frac{\{\chi_i \mid i \in I\}}{\chi}$, a process substitution σ, and a data substitution ξ such that the application of these substitutions to χ gives the formula ψ, and for all $i \in I$, the application of the substitutions to χ_i gives the formula ψ_i.*

Note that by removing the data substitution ξ 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 ϕ from the TSS (with data) $\mathcal{T}$. Whenever $\mathcal{T}$ is known from the context, we will write ϕ directly instead of $\mathcal{T} \vdash \phi$.

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 [10, 11, 12, 20].

Definition 3 (Strong Bisimilarity [34])**.** *Consider a TSS $\mathcal{T} = (\Sigma_P, L, \mathcal{R})$. A relation $R \subseteq T(\Sigma_P) \times T(\Sigma_P)$ is a* strong bisimulation *if and only if it is symmetric and $\forall_{p,q}\ (p,q) \in R \Rightarrow (\forall_{l,p'}\ p \xrightarrow{l} p' \Rightarrow \exists_{q'}\ q \xrightarrow{l} q' \wedge (q,q') \in R)$. Two closed terms p and q are* strongly bisimilar*, denoted by $p \underline{\leftrightarrow}^{\mathcal{T}} q$ if there exists a strong bisimulation relation R such that $(p,q) \in R$.*

Definition 4 (Stateless Bisimilarity [31])**.** *Consider a TSS with data $\mathcal{T} = (\Sigma_P \cup \Sigma_D, L, \mathcal{R})$. A relation $R_{\text{sl}} \subseteq T(\Sigma_P) \times T(\Sigma_P)$ is a* stateless bisimulation *if and only if it is symmetric and $\forall_{p,q}\ (p,q) \in R_{\text{sl}} \Rightarrow \forall_{d,l,p',d'}\ (p,d) \xrightarrow{l} (p',d') \Rightarrow \exists_{q'}\ (q,d) \xrightarrow{l} (q',d') \wedge (p',q') \in R_{\text{sl}}$. Two closed process terms p and q are* stateless bisimilar*, denoted by $p \underline{\leftrightarrow}_{\text{sl}}^{\mathcal{T}} q$, if there exists a stateless bisimulation relation R_{sl} such that $(p,q) \in R_{\text{sl}}$.*

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 [5])**.** *A transition system specification over signature Σ 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*

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

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*

$$\textbf{(c')}\ \frac{H'}{f(x'_0, x'_1) \xrightarrow{l} t'}$$

and a bijective mapping (substitution) $\hbar$ on variables such that

- $\hbar(x'_0) = x_1$ *and* $\hbar(x'_1) = x_0$,
- $\hbar(t') \sim_{cc} t$ *and*
- $\hbar(h') \in H$, *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 [5])**.** *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 procedure for converting any GSOS language definition that disjointly extends the language for synchronization trees to a finite complete equational axiom system which characterizes strong bisimilarity over a disjoint extension of the original language. 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 [13]. For the sake of simplicity, we confine ourselves to the positive subset of the GSOS format; we expect the generalization to the full GSOS format to be straightforward.

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

$$\textbf{(g)}\ \frac{\{x_i \xrightarrow{l_{ij}} y_{ij} \mid i \in I, j \in J_i\}}{f(x_1, \ldots, x_n) \xrightarrow{l} C[\vec{x}, \vec{y}]},$$

where all variables are distinct, f is an operation symbol form Σ_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 Σ_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:

$$t ::= \mathbf{0} \mid l.t\ (\forall l \in L) \mid t + t.$$

We denote this signature by Σ_{BCCSP}. Intuitively, $\mathbf{0}$ 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 [21]:

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

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

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

$$x + y = y + x \qquad x + x = x$$
$$(x + y) + z = x + (y + z) \qquad x + \mathbf{0} = x\,.$$

Theorem 10 ([23])**.** E_{BCCSP} *is sound and ground-complete for bisimilarity on* $T(\Sigma_{\text{BCCSP}})$*. That is, it holds that* $E_{\text{BCCSP}} \vdash p = q$ *if, and only if,* $p \underline{\leftrightarrow}^{\text{BCCSP}} q$ *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* $G \sqsubseteq G'$*, 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 to transform 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 behaviour 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 the syntactic shape of (transformed) SOS rules.

Definition 12 (Currying and Label Closure)**.** *Consider the TSS with data* $\mathcal{T} = (\Sigma_P \cup \Sigma_D, L, \mathcal{R})$ *and transition rule* $\rho \in \mathcal{R}$ *of the shape* $\rho = \dfrac{\{(t_{P_i}, t_{D_i}) \xrightarrow{l_{ij}} (t_{P_{ij}}, t_{D_{ij}}) \mid i \in I, j \in J_i\}}{(t_P, t_D) \xrightarrow{l} (t'_P, t'_D)}$*.*

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

By $\rho^c_\xi = \dfrac{\{t_{P_i} \xrightarrow{(\xi(t_{D_i}), l_{ij}, \xi(t_{D_{ij}}))} t_{P_{ij}} \mid i \in I, j \in J_i\}}{t_P \xrightarrow{(\xi(t_D), l, \xi(t'_D))} t'_P}$ *we denote the* closed label version *of* ρ^c *with respect to the closed data substitution* ξ*. By* $cl(\rho^c)$ *we denote the set consisting of all closed label versions of* ρ^c*, i.e.* $cl(\rho^c) = \{\rho^c_\xi \mid \rho^c \in \mathcal{R}^c, \xi$ *is a closed data substitution*$\}$*. We further define* $cl(\mathcal{R}^c) = \{cl(\rho^c) \mid$

$\rho^c \in \mathcal{R}^c\}$ *and* $cl(L^c) = \{(\xi(t_D), l, \xi(t'_D)) \mid (t_D, l, t'_D) \in L^c, \xi$ *is a closed data substitution*$\}$. *The* closed label version *of* $\mathcal{T}^c$ *is* $cl(\mathcal{T}^c) = (\Sigma_P, cl(L^c), cl(\mathcal{R}^c))$.

Our goal is to reduce the notion of stateless bisimilarity between two closed process 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* $\mathcal{T} = (\Sigma, L, D)$ *with data, for each two closed process terms* $p, q \in T(\Sigma_P)$, $p \underline{\leftrightarrow}^{\mathcal{T}}_{\text{sl}} q$ *if, and only if,* $p \underline{\leftrightarrow}^{cl(\mathcal{T}^c)} q$.

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 adapt the method introduced in [2] by considering the set of more complex labels that integrate the data, as presented in Section 3.

It is important to note that we present the schema by considering that the signature for data terms, Σ_D, consists only of a finite set of constants. However, as we foresee a future extension to a setting with arbitrary data terms, we choose to use the notation for arbitrary data terms instead of the one for constants in some of the following definitions.

BCCSP is extended to a setting with data, BCCSP$_D$. This is done by adding to the signature for process terms Σ_{BCCSP} two auxiliary operators for handling the store, named *check* and *update*, obtaining a new signature, Σ_{BCCSP_D}. Terms over Σ_{BCCSP_D} are build according to the following grammar:

$$t_P ::= \mathbf{0} \mid l.t_P \quad \forall_{l \in L} \mid \textit{check}(t_D, t_P) \mid \textit{update}(t_D, t_P) \mid t_P + t_P.$$

Intuitively, operation $\textit{check}(t_D, t_P)$ makes sure that, before executing an initial action from t_P, the store has the value t_D, and $\textit{update}(t_D, t_P)$ changes the store value to t_D after executing an initial action of process t_P. The prefix operation does not affect the store. We directly provide the curried set of rules defining the semantics of BCCSP^c_D.

$$\frac{}{l.x_P \xrightarrow{(x_D,l,x_D)} x_P} \qquad \frac{x_P \xrightarrow{(x_D,l,x'_D)} x'_P}{\textit{check}(x_D, x_P) \xrightarrow{(x_D,l,x'_D)} x'_P} \qquad \frac{x_P \xrightarrow{(x_D,l,x'_D)} x'_P}{\textit{update}(y_D, x_P) \xrightarrow{(x_D,l,y_D)} x'_P}$$

$$\frac{x_P \xrightarrow{(x_D,l,x'_D)} x'_P}{x_P + y_P \xrightarrow{(x_D,l,x'_D)} x'_P} \qquad \frac{y_P \xrightarrow{(x_D,l,x'_D)} y'_P}{x_P + y_P \xrightarrow{(x_D,l,x'_D)} y'_P}.$$

Definition 3 can easily be adapted to the setting of SOS systems with data that are curried.

Definition 14. *Consider a TSS* $\mathcal{T} = (\Sigma_P \cup \Sigma_D, L, \mathcal{R})$, *which means that* $\mathcal{T}^c = (\Sigma_P, L^c, \mathcal{R}^c)$. *A relation* $R \subseteq T(\Sigma_P) \times T(\Sigma_P)$ *is a* strong bisimulation *if and only if it is symmetric and* $\forall_{p,q}\ (p,q) \in R \Rightarrow \forall_{d,l,d',p'}\ p \xrightarrow{(d,l,d')} p' \Rightarrow \exists_{q'}\ q \xrightarrow{(d,l,d')} q' \wedge (q,q') \in R$. *Two closed terms* p *and* q *are* strongly bisimilar, *denoted by* $p \underline{\leftrightarrow}^{\mathcal{T}^c} q$ *if there exists a strong bisimulation relation* R *such that* $(p,q) \in R$.

The axiomatization $E_{\mathrm{BCCSP}^c_D}$ of strong bisimilarity over BCCSP^c_D, which is to be proven sound and ground-complete in the remainder of this section, is given below:

$$
\begin{array}{lcll}
x_P + y_P & = & y_P + x_P & \text{(n-comm)} \\
x_P + (y_P + z_P) & = & (x_P + y_P) + z_P & \text{(n-assoc)} \\
x_P + x_P & = & x_P & \text{(n-idem)} \\
x_P + \mathbf{0} & = & x_P & \text{(n-zero)} \\
\mathit{check}(x_D, x_P + y_P) & = & \mathit{check}(x_D, x_P) + \mathit{check}(x_D, y_P) & \text{(nc)} \\
\mathit{update}(x_D, x_P + y_P) & = & \mathit{update}(x_D, x_P) + \mathit{update}(x_D, y_P) & \text{(nu)} \\
\mathit{check}(x_D, \mathit{update}(y_D, x_P)) & = & \mathit{update}(y_D, \mathit{check}(x_D, x_P)) & \text{(cu)} \\
\mathit{update}(x_D, \mathit{update}(y_D, x_P)) & = & \mathit{update}(x_D, x_P) & \text{(uu)} \\
\mathit{check}(d, \mathit{check}(d, x_P)) & = & x_P \quad (\forall d \in \Sigma_D) & \text{(cc)} \\
\mathit{check}(d, \mathit{check}(d', x_P)) & = & \mathbf{0} \quad (\forall d, d' \in \Sigma_D, d \neq d') & \text{(cc')} \\
l.x_P & = & \sum_{d \in \Sigma_D} \mathit{update}(d, \mathit{check}(d, l.x_P)) & \text{(lc)}
\end{array}
$$

Recall that Σ_D is a finite set of constants, and, therefore, the right hand side of axiom (lc) has a finite number of summands.

The following theorem is proved in the standard fashion.

Theorem 15 (Soundness)**.** *For each two terms s, t in $\mathbb{T}(\Sigma_{\mathrm{BCCSP}^c_D})$ it holds that if $E_{\mathrm{BCCSP}^c_D} \vdash s = t$ then $s \leftrightarrow^{\mathrm{BCCSP}^c_D} t$.*

We now introduce the concept of terms in *head normal form*, which is essential for proving the completeness of axiom systems.

Definition 16 (Head Normal Form)**.** *Let Σ_P be a signature such that $\Sigma_{\mathrm{BCCSP}^c_D} \subseteq \Sigma_P$. A term t in $\mathbb{T}(\Sigma_P)$ is in* head normal form *(for short, h.n.f.) if*

$$t = \sum_{i \in I} \mathit{update}(t'_{Di}, \mathit{check}(t_{Di}, l_i.t_{Pi})),$$

where, for every $i \in I$, $t_{Di}, t'_{Di} \in \mathbb{T}(\Sigma_D)$, $t_{Pi} \in \mathbb{T}(\Sigma_D)$, $l_i \in L$. The empty sum $(I = \emptyset)$ is denoted by the deadlock constant $\mathbf{0}$*.*

Lemma 17 (Head Normalization)**.** *For any term p in $T(\Sigma_{\mathrm{BCCSP}^c_D})$, there exists p' in $T(\Sigma_{\mathrm{BCCSP}^c_D})$ in h.n.f. such that $E_{\mathrm{BCCSP}^c_D} \vdash p = p'$.*

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

Base case

- $p = \mathbf{0}$; this case is vacuous, because p is already in h.n.f.

Inductive step cases

- p is of the shape $l.p'$; then
  $p \overset{\text{def. } p}{=} l.p' \overset{\text{(lc)}}{=} \sum_{d \in T(\Sigma_D)} \mathit{update}(d, \mathit{check}(d, l.p'))$, which is in h.n.f.
- p is of the shape $\mathit{check}(d'', p'')$; then
  $p \overset{\text{def. } p}{=} \mathit{check}(d'', p'') \overset{\text{ind. hyp.}}{=}$
  $\mathit{check}(d'', \sum_{i \in I} \mathit{update}(d'_i, \mathit{check}(d_i, l_i.p'_i))) \overset{\text{(nc)}}{=} \sum_{i \in I} \mathit{check}(d'', \mathit{update}(d'_i, \mathit{check}(d_i, l_i.p'_i))) \overset{\text{(cu)}}{=}$
  $\sum_{i \in I} \mathit{update}(d'_i, \mathit{check}(d'', \mathit{check}(d_i, l_i.p'_i))) \overset{\text{(cc,cc')}}{=} \sum_{i \in I, d_i = d''} \mathit{update}(d'_i, \mathit{check}(d_i, l_i.p'_i))$,
  which is in h.n.f.

- p is of the form $\mathit{update}(d'',p'')$; then
  $p \overset{\text{def. } p}{=} \mathit{update}(d'',p'') \overset{\text{ind. hyp.}}{=} \mathit{update}(d'',\sum_{i\in I}\mathit{update}(d'_i,\mathit{check}(d_i,l_i.p'_i))) \overset{\text{(nu)}}{=}$
  $\sum_{i\in I}\mathit{update}(d'',\mathit{update}(d'_i,\mathit{check}(d_i,l_i.p'_i))) \overset{\text{(uu)}}{=} \sum_{i\in I}\mathit{update}(d'',\mathit{check}(d_i,l_i.p'_i))$,
  which is in h.n.f.
- p is of the form p_0+p_1; then
  $p \overset{\text{def. } p}{=} p_0+p_1 \overset{\text{ind. hyp.}}{=} \sum_{i\in I}\mathit{check}(d'',\mathit{update}(d'_i,\mathit{check}(d_i,l_i.p'_i))) +$
  $\sum_{j\in J}\mathit{check}(d'',\mathit{update}(d'_j,\mathit{check}(d_j,l_j.p'_j))) =$
  $\sum_{k\in I\cup J}\mathit{check}(d'',\mathit{update}(d'_k,\mathit{check}(d_k,l_k.p'_k)))$, which is in h.n.f.

□

Theorem 18 (Ground-completeness)**.** *For each two closed terms $p,q \in T(\Sigma_{\mathrm{BCCSP}^c_D})$, it holds that if $p \underline{\leftrightarrow}^{\mathrm{BCCSP}^c_D} q$, then $E_{\mathrm{BCCSP}^c_D} \vdash p = q$.*

Proof. We assume, by Lemma 17 that p,q are in h.n.f., define the function *height* as follows:

$$\mathit{height}(p) = \begin{cases} 0 & \text{if } p = \mathbf{0} \\ 1+\max(\mathit{height}(p_1),\mathit{height}(p_2)) & \text{if } p = p_1+p_2 \\ 1+\mathit{height}(p') & \text{if } p = \mathit{update}(d',\mathit{check}(d,l.p')), \end{cases}$$

and prove the property by induction on $M = \max(\mathit{height}(p),\mathit{height}(q))$.

Base case ($M=0$) This case is vacuous, because $p=q=\mathbf{0}$, so $E_{\mathrm{BCCSP}^c_D} \vdash p=q$.

Inductive step case ($M>0$) We prove $E_{\mathrm{BCCSP}^c_D} \vdash p = q+p$ by arguing that every summand of q is provably equal to a summand of p. Let $\mathit{update}(d',\mathit{check}(d,l.q'))$ be a summand of q. By applying the rules defining BCCSP^c_D, we derive $q \xrightarrow{(d,l,d')} q'$. As $q \underline{\leftrightarrow}^{\mathrm{BCCSP}^c_D} p$ holds, it has to be the case that $p \xrightarrow{(d,l,d')} p'$ and $q' \underline{\leftrightarrow}^{\mathrm{BCCSP}^c_D} p'$ hold. As $\max(\mathit{height}(q'),\mathit{height}(p')) < M$, from the inductive hypothesis it results that $E_{\mathrm{BCCSP}^c_D} \vdash q' = p'$, hence $\mathit{update}(d',\mathit{check}(d,l.q'))$ is provably equal to $\mathit{update}(d',\mathit{check}(d,l.p'))$, which is a summand of p.

It follows, by symmetry, that $E_{\mathrm{BCCSP}^c_D} \vdash q = p+q$ holds, which ultimately leads to the fact that $E_{\mathrm{BCCSP}^c_D} \vdash p = q$ holds. □

Consider a TSS with data $\mathcal{T} = (\Sigma_P \cup \Sigma_D, L, \mathcal{R})$. For an operation $f \in \Sigma_P$, we denote by $\mathcal{R}_f$ the set of all rules defining f. All the rules in $\mathcal{R}_f$ are in the GSOS format extended with the data component. For the simplicity of presenting the axiomatization schema, we assume that f only has process terms as arguments, baring in mind that adding data terms is trivial.

When given a signature Σ_P that includes $\Sigma_{\mathrm{BCCSP}^c_D}$, the purpose of an axiomatization for a term $p \in T(\Sigma_P)$ is to derive another term p' such that $p \underline{\leftrightarrow}^{\mathcal{T}^c} p'$ and $p' \in T(\Sigma_{\mathrm{BCCSP}^c_D})$.

Definition 19 (Axiomatization schema)**.** *Consider a TSS $\mathcal{T}^c = (\Sigma_P, L^c, \mathcal{R}^c)$ such that $\mathrm{BCCSP}^c_D \sqsubseteq \mathcal{T}^c$. By $E_{\mathcal{T}^c}$ we denote the axiom system that extends $E_{\mathrm{BCCSP}^c_D}$ with the following axiom schema for every operation f in $\mathcal{T}$, parameterized over the vector of closed process terms $\vec{p}$ in h.n.f.:*

$$f(\vec{p}) = \sum \left\{ \mathit{update}(d',\mathit{check}(d,l.C[\vec{p},\vec{y_P}])) \;\middle|\; \rho = \frac{H}{f(\vec{p}) \xrightarrow{(d,l,d')} C[\vec{p},\vec{q}]} \in cl(\mathcal{R}^c_f) \text{ and } \checkmark(\vec{p},\rho) \right\},$$

where $\checkmark$ *is defined as* $\checkmark(\vec{p},\rho) = \bigwedge_{p_k\in\vec{p}} \checkmark'(p_k,k,\rho)$,

and $\checkmark'\left(p_k, k, \dfrac{\{x_{P_i} \xrightarrow{(d_i,l_{ij},d'_{ij})} y_{P_{ij}} \mid i\in I, j\in J_i\}}{f(\vec{p}) \xrightarrow{(d,l,d')} C[\vec{p},\vec{q}]}\right) =$

$$\text{if } k\in I \text{ then } \forall_{j\in J_k}\ \exists_{p',p''}\ E_{\text{BCCSP}^c_D} \vdash p_k = \textit{update}(d'_{kj}, \textit{check}(d_k, l_{kj}.p')) + p''.$$

Intuitively, the axiom transforms $f(\vec{p})$ into a sum of closed terms covering all its execution possibilities. We iterate, in order to obtain them, through the set of f-defining rules and check if $\vec{p}$ satisfies their hypotheses by using the meta-operation $\checkmark$. $\checkmark$ makes sure that, for a given rule, every component of $\vec{p}$ is a term with enough action prefixed summands satisfying 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 signature for data is a finite set of constants.

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 20. *Consider a TSS* $\mathcal{T}^c = (\Sigma_P, L^c, \mathcal{R}^c)$ *such that* $\text{BCCSP}^c_D \sqsubseteq \mathcal{T}^c$. $E_{\mathcal{T}^c}$ *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 $T(\Sigma_{\text{BCCSP}^c_D})$.

In order to prove the soundness, we denote, for brevity, the right hand side of the schema in Definition 19 by *RHS*.

Let us first prove that if $f(\vec{p})$ performs a transition then it can be matched by *RHS*. Consider a rule $\rho \in cl(\mathcal{R}^c_f)$ that can be applied for $f(\vec{p})$: $\rho = \dfrac{\{x_i \xrightarrow{(d_i,l_{ij},d_{ij})} y_{ij} \mid i\in I, j\in J_i\}}{f(\vec{x}) \xrightarrow{(d,l,d')} C[\vec{x},\vec{y}]}$. Then $f(\vec{p}) \xrightarrow{(d,l_{ij},d')} C[\vec{p},\vec{q}]$ holds and, at the same time, all of the rule's premises are met. This means that p_i is of the form $\sum_{j\in J_i} \textit{update}(d_{ij}, \textit{check}(d_i, l_{ij}.p_{ij})) + p'$ for some p' and p_{ij}'s. It is easy to see that all the conditions for $\checkmark$ are met, so $(d,l,d').C[\vec{p},\vec{q}]$ is a summand of *RHS*, and therefore it holds that $\textit{RHS} \xrightarrow{(d_i,l_{ij},d_{ij})} C[\vec{p},\vec{q}]$, witch matches the transition from $f(\vec{p})$.

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

We end this section with the remark that the problem of extending the axiomatization schema to the setting with arbitrary data terms is still open. The most promising solution we have thought of involves using the infinite alternative quantification operation from [38]. This operation would us to define and express head normal forms as (potentially) infinite sums, parameterized over data variables.

5 Case Study: The Coordination Language Linda

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

The provided specification defines a structural operational semantics for the coordination language Linda; the specification is taken from [31] and is a slight adaptation of the original semantics presented in [14] (by removing structural congruences and introducing a terminating process ϵ). Process constants (atomic process terms) in this language are ϵ (for terminating process), *ask*(u) and *nask*(u) (for checking existence and absence of tuple u in the shared data space, respectively), *tell*(u) (for adding tuple u to the

space) and *get*(u) (for taking tuple u from the space). Process composition operators in this language include nondeterministic choice ($+$), sequential composition ($;$) and parallel composition ($\|$). 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, x_D) where p is a process term in the above syntax and x_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 $\xrightarrow{\downarrow}$ 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.

$$(1)\ \frac{}{(\epsilon, x_D)\downarrow} \qquad (1_c)\ \frac{}{\epsilon\downarrow}$$

$$(2)\ \frac{}{(\mathit{ask}(u), x_D \cup \{u\}) \rightarrow (\epsilon, x_D \cup \{u\})} \qquad (2_c)\ \frac{}{\mathit{ask}(u) \xrightarrow{(d\cup\{u\},-,d\cup\{u\})} \epsilon}$$

$$(3)\ \frac{}{(\mathit{tell}(u), x_D) \rightarrow (\epsilon, x_D \cup \{u\})} \qquad (3_c)\ \frac{}{\mathit{tell}(u) \xrightarrow{(d,-,d\cup\{u\})} \epsilon}$$

$$(4)\ \frac{}{(\mathit{get}(u), x_D \cup \{u\}) \rightarrow (\epsilon, x_D)} \qquad (4_c)\ \frac{}{\mathit{get}(u) \xrightarrow{(d\cup\{u\},-,d)} \epsilon}$$

$$(5)\ \frac{}{(\mathit{nask}(u), x_D) \rightarrow (\epsilon, x_D)}[u \notin x_D] \qquad (5_c)\ \frac{}{\mathit{nask}(u) \xrightarrow{(d,-,d)} \epsilon}[u \notin d]$$

$$(6)\ \frac{(x_P, x_D)\downarrow}{(x_P + y_P, x_D)\downarrow} \quad (7)\ \frac{(y_P, x_D)\downarrow}{(x_P + y_P, x_D)\downarrow} \qquad (6_c)\ \frac{x_P\downarrow}{x_P + y_P\downarrow} \quad (7_c)\ \frac{y\downarrow}{x_P + y_P\downarrow}$$

$$(8)\ \frac{(x_P, x_D) \rightarrow (x'_P, x_D{}')}{(x_P + y_P, x_D) \rightarrow (x'_P, x_D{}')} \qquad (8_c)\ \frac{x_P \xrightarrow{(d,-,d')} x'_P}{x_P + y_P \xrightarrow{(d,-,d')} x'_P}$$

$$(9)\ \frac{(y_P, x_D) \rightarrow (y'_P, x_D{}')}{(x_P + y_P, x_D) \rightarrow (y'_P, x_D{}')} \qquad (9_c)\ \frac{y_P \xrightarrow{(d,-,d')} y'_P}{x_P + y_P \xrightarrow{(d,-,d')} y'_P}$$

$$(10)\ \frac{(x_P, x_D) \rightarrow (x'_P, x_D{}')}{(x_P\ ;\ y_P, x_D) \rightarrow (x'_P\ ;\ y_P, x_D{}')} \qquad (10_c)\ \frac{x_P \xrightarrow{(d,-,d')} x'_P}{x_P\ ;\ y_P \xrightarrow{(d,-,d')} x'_P\ ;\ y_P}$$

$$(11)\ \frac{(x_P, x_D)\downarrow\ (y_P, x_D) \rightarrow (y'_P, x_D{}')}{(x_P\ ;\ y_P, x_D) \rightarrow (y'_P, x_D{}')} \qquad (11_c)\ \frac{x_P\downarrow \quad y_P \xrightarrow{(d,-,d')} y'_P}{x_P\ ;\ y_P \xrightarrow{(d,-,d')} y'_P}$$

$$(12)\ \frac{(x_P, x_D)\downarrow\ (y_P, x_D)\downarrow}{(x_P\ ;\ y_P, x_D)\downarrow} \qquad (12_c)\ \frac{x_P\downarrow\ y_P\downarrow}{x_P\ ;\ y_P\downarrow}$$

$$(13)\ \frac{(x_P, x_D) \rightarrow (x'_P, x_D{}')}{(x_P \parallel y_P, x_D) \rightarrow (x' \parallel y, x_D{}')} \qquad (13_c)\ \frac{x_P \xrightarrow{(d,-,d')} x'_P}{x_P \,||\, y_P \xrightarrow{(d,-,d')} x'_P \,||\, y_P}$$

$$(14)\ \frac{(y_P, x_D) \to (y_P', x_D')}{(x_P \parallel y_P, x_D) \to (x_P \parallel y_P', x_D')} \qquad (14_c)\ \frac{y_P \xrightarrow{(d,-,d')} y_P'}{x_P \mid\mid y_P \xrightarrow{(d,-,d')} x_P \mid\mid y_P'}$$

$$(15)\ \frac{(x_P, x_D)\downarrow\ (y_P, x_D)\downarrow}{(x_P \parallel y_P, x_D)\downarrow} \qquad (15_c)\ \frac{x_P\downarrow\ y_P\downarrow}{x_P \parallel y_P\downarrow}$$

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 [28] 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 commutativity 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 statebased bisimilarity for all $d \in T(\Sigma_D)$.)

Following a similar line of reasoning, we get that $x \mid\mid y = y \mid\mid x$ 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 [17]; there, it is shown that this semantics conforms to the ASSOC-DE SIMONE format introduced in [17] 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 [17] 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 [17]. 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 ϵ is unit element for ; is proved similarly as in [4], Example 10.

Property	Axiom	Meta-Theorem
Associativity for ;	$x\,;(y\,;z) = (x\,;y)\,;z$	Theorem 1 of [17]
Associativity for $+$	$x+(y+z) = (x+y)+z$	Theorem 1 of [17]
Associativity for $\mid\mid$	$x \mid\mid (y \mid\mid z) = (x \mid\mid y) \mid\mid z$	Theorem 1 of [17]
Idempotence for $+$	$x+x = x$	Theorem 42 of [1]
Unit element for ;	$\epsilon\,;x = x$	Theorem 3 of [4]
Distributivity of $+$ over ;	$(x+y)\,;z = (x\,;y)+(x\,;z)$	Theorem 3 of [3]

We currently cannot derive an axiomatization for Linda because its semantics involves arbitrary data terms, as opposed to a finite number of constants.

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 [30] 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 [31]. 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. Furthermore, it is of interest to check how our technique can be applied to quantitative systems where non-functional aspects like probabilistic choice or stochastic timing is encapsulated as data. We also plan to investigate the possibility of automatically deriving axiom schemas for systems whose data component is given as arbitrary terms, instead of just constants.

Acknowledgements. We thank Luca Aceto, Peter Mosses, and Michel Reniers for their valuable comments on earlier versions of the paper. Eugen-Ioan Goriac is funded by the project 'Extending and Axiomatizing Structural Operational Semantics: Theory and Tools' (nr. 1102940061) of the Icelandic Research Fund.

References

[1] Luca Aceto, Arnar Birgisson, Anna Ingólfsdóttir, Mohammad Reza Mousavi & Michel A. Reniers (2012): *Rule formats for determinism and idempotence*. *Science of Computer Programming* 77(7–8), pp. 889–907, doi:10.1016/j.scico.2010.04.002.

[2] Luca Aceto, Bard Bloom & Frits W. Vaandrager (1994): *Turning SOS rules into equations*. *Information and Computation* 111, pp. 1–52, doi:10.1006/inco.1994.1040.

[3] Luca Aceto, Matteo Cimini, Anna Ingólfsdóttir, Mohammad Reza Mousavi & Michel A. Reniers (2011): *Rule Formats for Distributivity*. In Adrian Horia Dediu, Shunsuke Inenaga & Carlos Martín-Vide, editors: *Language and Automata Theory and Applications - 5th International Conference, LATA 2011, Tarragona, Spain, May 26–31, 2011. Proceedings*, *Lecture Notes in Computer Science* 6638, Springer, pp. 80–91, doi:10.1007/978-3-642-21254-3_5.

[4] Luca Aceto, Matteo Cimini, Anna Ingólfsdóttir, Mohammad Reza Mousavi & Michel A. Reniers (2011): *SOS rule formats for zero and unit elements*. *Theoretical Computer Science* 412(28), pp. 3045–3071, doi:10.1016/j.tcs.2011.01.024.

[5] Luca Aceto, Anna Ingolfsdottir, MohammadReza Mousavi & Michel A. Reniers (2009): *Algebraic Properties for Free!* *Bulletin of the European Association for Theoretical Computer Science (BEATCS)* 99, pp. 81–104.

[6] Franz Baader & Tobias Nipkow (1999): *Term Rewriting and All That*. Cambridge University Press.

[7] J.C.M. (Jos) Baeten, Twan Basten & Michel A. Reniers (2010): *Process Algebra: Equational Theories of Communicating Processes*. Cambridge Tracts in Theoretical Computer Science, Cambrdige University Press.

[8] Jos C. M. Baeten & Jan A. Bergstra (1997): *Process Algebra with Propositional Signals*. *Theoretical Computer Science (TCS)* 177(2), pp. 381–405, doi:10.1016/S0304-3975(96)00253-8.

[9] Christel Baier & Joost-Pieter Katoen (2008): *Principles of Model Checking*. MIT Press.

[10] D. A. van Beek, Ka Lok Man, Michel A. Reniers, J. E. Rooda & Ramon R. H. Schiffelers (2006): *Syntax and consistent equation semantics of hybrid Chi*. *J. Log. Algebr. Program.* 68(1-2), pp. 129–210, doi:10.1016/j.jlap.2005.10.005.

[11] D. A. van Beek, Michel A. Reniers, Ramon R. H. Schiffelers & J. E. Rooda (2007): *Foundations of a Compositional Interchange Format for Hybrid Systems*. In Alberto Bemporad, Antonio Bicchi & Giorgio C. Buttazzo, editors: *Proceedings of the 10th International Workshop on Hybrid Systems: Computation and Control (HSCC'07), Lecture Notes in Computer Science* 4416, Springer, pp. 587–600, doi:10.1007/978-3-540-71493-4_45.

[12] Jan A Bergstra & A. (Kees) Middelburg (2007): *Synchronous cooperation for explicit multi-threading*. *Acta Informatica* 44, pp. 525–569, doi:10.1007/s00236-007-0057-9.

[13] Bard Bloom, Sorin Istrail & Albert R. Meyer (1995): *Bisimulation can't be traced*. *J. ACM* 42, pp. 232–268, doi:10.1145/200836.200876.

[14] Antonio Brogi & Jean-Marie Jacquet (1998): *On the Expressiveness of Linda-like Concurrent Languages*. *Electr. Notes Theor. Comput. Sci.* 16(2), pp. 75–96, doi:10.1016/S1571-0661(04)00118-5.

[15] Nicholas Carriero & David Gelernter (1989): *Linda in Context*. *Communications of the ACM* 32(4), pp. 444–459, doi:10.1145/63334.63337.

[16] Robert J. Colvin & Ian J. Hayes (2011): *Structural Operational Semantics through Context-Dependent Behaviour*. *Journal of Logic and Algebraic Programming* 80(7), pp. 392–426, doi:10.1016/j.jlap.2011.05.001.

[17] Sjoerd Cranen, Mohammad Reza Mousavi & Michel A. Reniers (2008): *A Rule Format for Associativity*. In Franck van Breugel & Marsha Chechik, editors: *Proceedings of the 19th International Conference on Concurrency Theory (CONCUR'08), Lecture Notes in Computer Science* 5201, Springer-Verlag, pp. 447–461, doi:10.1007/978-3-540-85361-9_35.

[18] Pierpaolo Degano & Corrado Priami (2001): *Enhanced operational semantics*. *ACM Computing Surveys* 33(2), pp. 135–176, doi:10.1145/384192.384194.

[19] Fabio Gadducci & Ugo Montanari (2000): *The Tile Model*. In Gordon D. Plotkin, Colin Stirling & Mads Tofte, editors: *Proof, Language and Interaction: Essays in Honour of Robin Milner*, MIT Press, Boston, MA, USA, 2000, pp. 133–166.

[20] Vashti Galpin, Luca Bortolussi & Jane Hillston (2013): *HYPE: Hybrid modelling by composition of flows*. *Formal Asp. Comput.* 25(4), pp. 503–541, doi:10.1007/s00165-011-0189-0.

[21] R.J. van Glabbeek (2001): *The Linear Time - Branching Time Spectrum I. The Semantics of Concrete, Sequential Processes*. In A. Ponse S.A. Smolka J.A. Bergstra, editor: *Handbook of Process Algebra*, Elsevier, pp. 3–99, doi:10.1007/3-540-57208-2_6.

[22] Jan Friso Groote & Frits W. Vaandrager (1992): *Structured Operational Semantics and Bisimulation As a Congruence*. *Information and Computation* 100(2), pp. 202–260, doi:10.1016/0890-5401(92)90013-6.

[23] Matthew Hennessy & Robin Milner (1985): *Algebraic laws for nondeterminism and concurrency*. *J. ACM* 32(1), pp. 137–161, doi:10.1145/2455.2460.

[24] Narciso Martí-Oliet & José Meseguer (2002): *Rewriting Logic as a Logical and Semantic Framework*. In Dov M. Gabbay & Franz Guenthner, editors: *Handbook of Philosophical Logic*, 9, Kluwer Academic Publishers, 2002, pp. 1–87, doi:10.1007/978-94-017-0464-9_1.

[25] José Meseguer & Christiano Braga (2004): *Modular Rewriting Semantics of Programming Languages*. In Charles Rattray, Savi Maharaj & Carron Shankland, editors: *Proceedings of the 10th International Conference on Algebraic Methodology and Software Technology (AMAST'04)*, *Lecture Notes in Computer Science* 3116, Springer-Verlag, Berlin, Germany, 2004, pp. 364–378, doi:10.1007/978-3-540-27815-3_29.

[26] Peter D. Mosses (2004): *Exploiting Labels in Structural Operational Semantics*. *Fundam. Inform.* 60(1-4), pp. 17–31.

[27] Peter D. Mosses (2004): *Modular structural operational semantics*. *J. Log. Algebr. Program.* 60-61, pp. 195–228, doi:10.1016/j.jlap.2004.03.008.

[28] Peter D. Mosses & Mark J. New (2009): *Implicit Propagation in Structural Operational Semantics*. *Electr. Notes Theor. Comput. Sci.* 229(4), pp. 49–66, doi:10.1016/j.entcs.2009.07.073.

[29] Mohammad Reza Mousavi, Michel Reniers & Jan Friso Groote (2004): *Congruence for SOS with Data*. In: *Proceedings of the 19th Annual IEEE Symposium on Logic in Computer Science (LICS'04)*, IEEE Computer Society Press, Los Alamitos, CA, USA, 2004, pp. 302–313, doi:10.1109/LICS.2004.1319625.

[30] Mohammad Reza Mousavi, Michel A. Reniers & Jan Friso Groote (2004): *Congruence for SOS with Data*. In: *LICS*, pp. 303–312, doi:10.1109/LICS.2004.1319625.

[31] Mohammad Reza Mousavi, Michel A. Reniers & Jan Friso Groote (2005): *Notions of Bisimulation and Congruence Formats for SOS with Data*. *Information and Computation* 200(1), pp. 107–147, doi:10.1016/j.ic.2005.03.002.

[32] Mohammad Reza Mousavi, Michel A. Reniers & Jan Friso Groote (2005): *A syntactic commutativity format for SOS*. *Inf. Process. Lett.* 93(5), pp. 217–223, doi:10.1016/j.ipl.2004.11.007.

[33] Scott Owens (2008): *A Sound Semantics for OCamllight*. In Sophia Drossopoulou, editor: *ESOP*, *Lecture Notes in Computer Science* 4960, Springer, pp. 1–15, doi:10.1007/978-3-540-78739-6_1.

[34] David Michael Ritchie Park (1981): *Concurrency and Automata on Infinite Sequences*. In Peter Deussen, editor: *Theoretical Computer Science*, *Lecture Notes in Computer Science* 104, Springer, pp. 167–183, doi:10.1007/BFb0017309.

[35] Gordon D. Plotkin (1981): *A structural approach to operational semantics*. Technical Report DAIMI FN-19, Computer Science Department, Aarhus University, Aarhus, Denmark.

[36] Gordon D. Plotkin (2004): *The origins of structural operational semantics*. *Journal of Logic and Algebraic Programming (JLAP)* 60, pp. 3–15, doi:10.1016/j.jlap.2004.03.009.

[37] Gordon D. Plotkin (2004): *A structural approach to operational semantics*. *Journal of Logic and Algebraic Progamming (JLAP)* 60, pp. 17–139. This article first appeared as [35].

[38] Michel A. Reniers, Jan Friso Groote, Mark B. van der Zwaag & Jos van Wamel (2002): *Completeness of Timed μCRL*. *Fundamenta Informaticae* 50(3-4), pp. 361–402.

[39] A. W. (Bill) Roscoe (2010): *Understanding Concurrent Systems*. Springer, doi:10.1007/978-1-84882-258-0.

[40] Christopher Strachey (2000): *Fundamental Concepts in Programming Languages*. *Higher-Order and Symbolic Computation* 13, pp. 11–49, doi:10.1023/A:1010000313106.

[41] Daniele Turi & Gordon D. Plotkin (1997): *Towards a Mathematical Operational Semantics*. In: *LICS*, IEEE Computer Society, pp. 280–291, doi:10.1109/LICS.1997.614955.