Exploiting Algebraic Laws to Improve Mechanized Axiomatizations[⋆]

Luca Aceto[1], Eugen-Ioan Goriac[1], Anna Ingolfsdottir[1], Mohammad Reza Mousavi[2], and Michel A. Reniers[3]

[1] ICE-TCS, School of Computer Science, Reykjavik University, Menntavegur 1, IS-101, Reykjavik, Iceland
[2] Department of Computer Science, Eindhoven University of Technology, P.O. Box 513, NL-5600 MB Eindhoven, The Netherlands
[3] Department of Mechanical Engineering, Eindhoven University of Technology, P.O. Box 513, NL-5600 MB Eindhoven, The Netherlands

Abstract. In the field of structural operational semantics (SOS), there have been several proposals both for syntactic rule formats guaranteeing the validity of algebraic laws, and for algorithms for automatically generating ground-complete axiomatizations. However, there has been no synergy between these two types of results. This paper takes the first steps in marrying these two areas of research in the meta-theory of SOS and shows that taking algebraic laws into account in the mechanical generation of axiomatizations results in simpler axiomatizations. The proposed theory is applied to some examples from the literature, showing that, in these cases, the generated axiomatizations coincide with hand-crafted ones.

1 Introduction

Algebraic properties, such as commutativity, associativity and idempotence of binary operators, specify some natural properties of programming and specification constructs. These properties can either be validated using the semantics of the language with respect to a suitable notion of program equivalence, or they can be guaranteed a priori 'by design'. In particular, for languages equipped with a Structural Operational Semantics (SOS) [32,42,43], there are two closely related lines of work to achieve this goal: firstly, there is a rich body of syntactic rule formats that can guarantee the validity of certain algebraic properties; see [15,39] for recent surveys. Secondly, there are numerous results regarding the mechanical generation of ground-complete axiomatization of various behavioral equivalences and preorders for SOS language specifications in certain formats [3,5,18,22,30,46,47].

However, these two lines of research have evolved separately and no link has been established between the two types of results so far. In this paper, we take the first steps in marrying these two research areas and in using rule formats for algebraic properties (specifically, for commutativity) to enhance the process of automatic generation of

[⋆] The first three authors have been partially supported by the project 'Meta-theory of Algebraic Process Theories' (nr. 100014021) of the Icelandic Research Fund. Eugen-Ioan Goriac is also funded by the project 'Extending and Axiomatizing Structural Operational Semantics: Theory and Tools' (nr. 1102940061) of the Icelandic Research Fund.

axiomatizations for strong bisimilarity from GSOS language specifications [21,23]. In particular, we show that linking these two areas results both in fewer axioms and axioms that are more natural, i.e., look like hand-crafted ones.

Contribution and Related Work. A wealth of ground-completeness results have been presented in the literature on process calculi. (See, e.g., the classic references [20,27,33] and the survey paper [10] for further pointers to the literature.) A common proof strategy for establishing such ground-completeness results is to reduce the problem of axiomatizing the notion of behavioural equivalence under consideration over arbitrary closed terms to that of axiomatizing it over 'synchronization-tree terms'. This approach is also at the heart of the algorithm proposed in [5] for the automatic generation of finite, equational, ground-complete axiomatizations for bisimilarity over language specifications in the GSOS format. A variation on that algorithm for GSOS language specifications with termination has been presented in [18]; see [6] for an extension of the algorithm from [5] dealing with arbitrary predicates. In [46], Ulidowski has instead offered algorithms for the automatic generation of finite axiom systems for the testing preorder over de Simone process languages [45].

In this paper, we present a refinement of the algorithm from [5] that uses a rule format guaranteeing commutativity of certain operators to obtain ground-complete axiomatizations of bisimilarity that are closer to the hand-crafted ones than those produced by existing algorithms.

Our rule format for commutativity is a generalization of the rule format for commutativity presented in [37], which allows operators to have various sets of commutative arguments. Apart from being natural, such a generalization is useful in the automatic generation of ground-complete axiomatizations, as the developments in this study show.

Structure of the Paper. In Section 2 we recall some standard notions from process theory and the meta-theory of SOS. In Section 3 we present a generalized notion of commutativity and a corresponding rule format. In Section 4 we give an algorithm for the automatic generation of ground-complete axiomatizations for bisimilarity from GSOS language specifications. The algorithm uses information derived from the generalized commutativity format, where possible, to reduce the number of auxiliary operators and to produce axiom systems that are close to hand-crafted ones. In Section 5 we apply the algorithm to axiomatize the classic parallel composition operator and compare the generated axiomatization to earlier ones. Section 6 concludes the paper and suggests some avenues for future research.

2 Preliminaries

In this section we review, for the sake of completeness, some standard definitions from process theory and the meta-theory of SOS that will be used in the remainder of the paper. We refer the interested reader to [12,39] for further details.

2.1 Transition System Specifications

Definition 1 (Signature and terms). *We let V denote an infinite set of variables with typical members $x, x', x_i, y, y', y_i, \ldots$. A* signature *$\Sigma$ is a set of function symbols,* *each with a fixed arity. We call these symbols* operators *and usually represent them by $f, g, \ldots$. An operator with arity zero is called a* constant. *We define the set $\mathbb{T}(\Sigma)$ of* terms *over Σ (sometimes referred to as Σ-terms) as the smallest set satisfying the following constraints.*

- *A variable $x \in V$ is a term.*
- *If $f \in \Sigma$ has arity n and $t_1, \ldots, t_n$ are terms, then $f(t_1, \ldots, t_n)$ is a term.*

We use $s, t, t', t_i, u, \ldots$ to range over terms. We write $t_1 \equiv t_2$ if t_1 and t_2 are syntactically equal. The function $vars : \mathbb{T}(\Sigma) \rightarrow 2^V$ gives the set of variables appearing in a term. The set $\mathbb{C}(\Sigma)$ is the set of closed terms, *i.e., the set of all terms t such that $vars(t) = \emptyset$. We use $p, p', p_i, q, r \ldots$ to range over closed terms. A* substitution *σ is a function of type $V \rightarrow \mathbb{T}(\Sigma)$. We extend the domain of substitutions to terms homomorphically. If the range of a substitution lies in $\mathbb{C}(\Sigma)$, we say that it is a* closed substitution.

Definition 2 (Transition System Specifications (TSS), formulae and transition systems). *A* transition system specification *$\mathcal{T}$ is a triple (Σ, L, D) where*

- *Σ is a signature.*
- *L is a set of labels. If $l \in L$ and $t, t' \in \mathbb{T}(\Sigma)$, we say that $t \xrightarrow{l} t'$ is a* positive formula *and $t \overset{l}{\nrightarrow}$ is a* negative formula. *A* formula *is either a positive formula or a negative one. A formula is typically denoted by ϕ, ψ, ϕ', ϕ_i, ….*
- *D is a set of* deduction rules, *i.e., pairs of the form (Φ, ϕ) where Φ is a set of formulae and ϕ is a positive formula. We call the formulae contained in Φ the* premises *of the rule and ϕ the* conclusion.

We say that a formula is closed *if all of its terms are closed. Substitutions are also extended to formulae and sets of formulae in the natural way.*

We often refer to a closed formula $t \xrightarrow{l} t'$ as a *transition* with t being its *source*, l its *label*, and t' its *target*. The notions of source and label are similarly defined for negative formulae. A deduction rule (Φ, ϕ) is typically written as $\frac{\Phi}{\phi}$.

2.2 GSOS Format

The GSOS format is a widely studied format of deduction rules in transition system specifications proposed by Bloom, Istrail and Meyer [21,23]. Transition system specifications whose rules are in the GSOS format enjoy many desirable properties, and several studies in the literature on the meta-theory of SOS have focused on them—see, for instance, [2,3,5,12,13,18]. Following [5], in this study we shall also focus on transition system specifications in the GSOS format, which we now proceed to define.

Definition 3 (GSOS Format [23]). *A deduction rule for an operator f of arity n is in the GSOS format if and only if it has the following form:*

$$\frac{\{x_i \xrightarrow{l_{ij}} y_{ij} \mid 1 \leq i \leq n, 1 \leq j \leq m_i\} \cup \{x_i \overset{l_{ik}}{\nrightarrow} \mid 1 \leq i \leq n, 1 \leq k \leq n_i\}}{f(\overrightarrow{x}) \xrightarrow{l} C[\overrightarrow{x}, \overrightarrow{y}]}$$

where the x_i's and the y_{ij}'s $(1 \leq i \leq n$ and $1 \leq j \leq m_i)$ are all distinct variables, m_i and n_i are natural numbers, $C[\overrightarrow{x}, \overrightarrow{y}]$ is a Σ-term with variables including at most the x_i's and the y_{ij}'s, and the l_{ij}'s and l are labels. If $m_i > 0$, for some i, then we say that the rule tests its i-th argument positively. Similarly, if $n_i > 0$ then we say that the rule tests its i-th argument negatively.

The above rule is said to be f-defining *and* l-emitting.

A TSS is in the GSOS format when it has a finite signature, a finite set of labels, a finite set of deduction rules and all its deduction rules are in the GSOS format. We shall sometimes refer to a TSS in the GSOS format as a GSOS system.

In addition to the syntactic restrictions on deduction rules, the GSOS format, as presented in [21,23], requires the signature to include a constant **0**, a collection of unary operators $a.\_$ $(a \in L)$ and a binary operator $\_+\_$. Intuitively, **0** represents a process that does not exhibit any behaviour, $s + t$ is the nondeterministic choice between the behaviours of s and t, while $a.t$ is a process that first performs action a and behaves like t afterwards. The standard deduction rules for these operations are given below:

$$\frac{}{a.x_1 \xrightarrow{a} x_1} \qquad \frac{x_1 \xrightarrow{a} x_1'}{x_1 + x_2 \xrightarrow{a} x_1'} \qquad \frac{x_2 \xrightarrow{a} x_2'}{x_1 + x_2 \xrightarrow{a} x_2'}\,.$$

In the remainder of this paper, following [21,23], we shall tacitly assume that each TSS in the GSOS format contains these operators with the rules given above. The import of this assumption is that, as is well known, within each TSS in the GSOS format it is possible to express each finite synchronization tree over L [35]. Following [29], the TSS containing the operators **0**, $a.\_$ $(a \in L)$ and $\_+\_$, with the above-given rules will be denoted by BCCSP.

Informally, the intent of a GSOS rule is as follows. Suppose that we are wondering whether $f(\overrightarrow{p})$ is capable of taking an l-step. We look at each f-defining and l-emitting rule in turn. We inspect each positive premise $x_i \xrightarrow{l_{ij}} y_{ij}$, checking if p_i is capable of taking an l_{ij}-step for each j and if so calling the l_{ij}-children q_{ij}. We also check the negative premises: if p_i is *in*capable of taking an l_{ik}-step for each k. If so, then the rule *fires* and $f(\overrightarrow{p}) \xrightarrow{l} C[\overrightarrow{p}, \overrightarrow{q}]$. This means that the transition relation $\longrightarrow$ associated with a TSS in the GSOS format is the one defined by the rules using structural induction over closed Σ-terms. This transition relation is the unique sound and supported transition relation [31]. Here *sound* means that whenever a closed substitution σ 'satisfies' the premises of a rule of the form given in Definition 3, then $\sigma(f(x_1, \ldots, x_l)) \xrightarrow{l} \sigma(C[\overrightarrow{x}, \overrightarrow{y}])$. On the other hand, *supported* means that any transition $p \xrightarrow{l} q$ can be obtained by instantiating the conclusion of a rule of the form given in Definition 3 with a substitution that satisfies its premises. We refer the interested reader to [21,23] for the precise definition of $\longrightarrow$ and much more information on GSOS languages. The above informal description of the transition relation associated with a TSS in the GSOS format suffices to follow the technical developments in the remainder of the paper.

The notion of disjoint extension of a TSS is defined as in [5].

Definition 4. *A GSOS system $\mathcal{T}'$ is a* disjoint extension *of a GSOS system $\mathcal{T}$, denoted by $\mathcal{T} \sqsubseteq \mathcal{T}'$, if the signature and rules of $\mathcal{T}'$ include those of $\mathcal{T}$, and $\mathcal{T}'$ introduces no new rules for operators in the signature of $\mathcal{T}$.*

In the light of our assumption, BCCSP $\sqsubseteq \mathcal{T}$ holds for each GSOS system $\mathcal{T}$. If $\mathcal{T}'$ disjointly extends $\mathcal{T}$ then $\mathcal{T}'$ introduces no new outgoing transitions for the closed terms of $\mathcal{T}$. (More general conservative extension results are discussed in, for instance, [28,38].)

2.3 Bisimilarity and Axiom Systems

To establish a link between the operational model and the algebraic properties, a notion of behavioural equivalence should be fixed. The notion of behavioural equivalence that we will use in this paper is the following, classic notion of bisimilarity [36,40].

Definition 5 (Bisimilarity [40]). *Let $\mathcal{T}$ be a GSOS system with signature Σ. A relation $R \subseteq \mathbb{C}(\Sigma) \times \mathbb{C}(\Sigma)$ is a* bisimulation *if and only if R is symmetric and, for all $p_0, p_1, p_0' \in \mathbb{C}(\Sigma)$ and $l \in L$,*

$$(p_0 \mathrel{R} p_1 \wedge p_0 \xrightarrow{l} p_0') \Rightarrow \exists p_1' \in \mathbb{C}(\Sigma).\ (p_1 \xrightarrow{l} p_1' \wedge p_0' \mathrel{R} p_1').$$

Two terms $p_0, p_1 \in \mathbb{C}(\Sigma)$ are called bisimilar, *denoted by $\mathcal{T} \vdash p_0 \underline{\leftrightarrow} p_1$ (or simply by $p_0 \underline{\leftrightarrow} p_1$ when $\mathcal{T}$ is clear from the context), when there exists a bisimulation R such that $p_0 \mathrel{R} p_1$.*

It is well known that $\underline{\leftrightarrow}$ is an equivalence relation over $\mathbb{C}(\Sigma)$. Any equivalence relation $\sim$ over closed terms in a TSS $\mathcal{T}$ is extended to open terms in the standard fashion, i.e., for all $t_0, t_1 \in \mathbb{T}(\Sigma)$, the equation $t_0 = t_1$ holds over $\mathcal{T}$ modulo $\sim$ (sometimes abbreviated to $t_0 \sim t_1$) if, and only if, $\mathcal{T} \vdash \sigma(t_0) \sim \sigma(t_1)$ for each closed substitution σ.

Remark 1. If $\mathcal{T}'$ is a disjoint extension of $\mathcal{T}$, then two closed terms over the signature of $\mathcal{T}$ are bisimilar in $\mathcal{T}$ if and only if they are bisimilar in $\mathcal{T}'$.

Definition 6. *Let Σ be a signature. An equivalence relation $\sim$ over Σ-terms is a* congruence *if, for all $f \in \Sigma$ and closed terms $p_1, \ldots, p_n, q_1, \ldots, q_n$, where n is the arity of f, if $p_i \sim q_i$ for each $i \in \{1, \ldots, n\}$ then $f(p_1, \ldots, p_n) \sim f(q_1, \ldots, q_n)$.*

Remark 2. Let Σ be a signature and let $\sim$ be a congruence. It is easy to see that, for all $f \in \Sigma$ and terms $t_1, \ldots, t_n, u_1, \ldots, u_n$, where n is the arity of f, if $t_i \sim u_i$ for each $i \in \{1, \ldots, n\}$ then $f(t_1, \ldots, t_n) \sim f(u_1, \ldots, u_n)$.

The following result is well known [21].

Proposition 1. *$\underline{\leftrightarrow}$ is a congruence for any TSS in GSOS format.*

Ideally, a notion of behavioural congruence should coincide with the equational theory generated by some 'finitely presentable' set of axioms describing the desired algebraic properties of the operators in a language. One side of this coincidence is captured by the *soundness* requirement, which states that all the (closed) equalities that are derivable from the axiom system using the rules of equational logic are indeed valid with respect to the chosen notion of behavioural equivalence. The other side of the coincidence, called *ground completeness*, states that all the valid behavioural equivalences over *closed* terms are derivable from the axiom system. These concepts are formalized in what follows.

Definition 7 (Axiom System). *An* axiom 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*.*

In the context of a fixed TSS $\mathcal{T}$*, an axiom system* E *(over the same signature) is* sound *with respect to a congruence relation* $\sim$ *if and only if for all* $t, t' \in \mathbb{T}(\Sigma)$*, if* $E \vdash t = t'$*, then it holds that* $\mathcal{T} \vdash t \sim t'$*. The axiom system* E *is* ground complete *if the implication holds in the opposite direction whenever* t *and* t' *are closed terms.*

3 Commutativity Format

Commutativity is an essential property specifying that the order of arguments of an operator is immaterial. In the setting of process algebras, commutativity is defined with respect to a notion of behavioural equivalence over terms. In this section, we first present a generalized notion of commutativity that allows n-ary operators to have various sets of commutative arguments and then slightly adapt the commutativity rule format proposed in [37] to the extended setting. Moreover, we give some auxiliary definitions that will be used in the axiomatization procedure proposed in the next section.

In order to motivate the generalized notion of commutativity we present below, consider, by way of example, the ternary operator f defined by the rules below, where a ranges over the collection of action labels L.

$$\frac{x \xrightarrow{a} x'}{f(x,y,z) \xrightarrow{a} f(x',y,z)} \quad \frac{y \xrightarrow{a} y'}{f(x,y,z) \xrightarrow{a} f(x,y',z)}$$
$$\frac{x \xrightarrow{a} x' \quad z \xrightarrow{a} z'}{f(x,y,z) \xrightarrow{a} f(x',y,z')} \quad \frac{y \xrightarrow{a} y' \quad z \xrightarrow{a} z'}{f(x,y,z) \xrightarrow{a} f(x,y',z')}.$$

It is not hard to show that the operator f is commutative in its first two arguments modulo bisimilarity, irrespective of the other operators in the TSS under consideration—that is, for all closed terms p, q, r,

$$f(p,q,r) \leftrightarrow f(q,p,r).$$

On the other hand, the third argument is not commutative with respect to the other two. For example, we have that

$$f(a.\mathbf{0},\mathbf{0},\mathbf{0}) \not\leftrightarrow f(\mathbf{0},\mathbf{0},a.\mathbf{0})$$

because $f(a.\mathbf{0}, \mathbf{0}, \mathbf{0}) \xrightarrow{a} f(\mathbf{0}, \mathbf{0}, \mathbf{0})$, but $f(\mathbf{0}, \mathbf{0}, a.\mathbf{0})$ has no outgoing transitions.

The commutativity format presented in [37] can only deal with operators that are commutative for each pair of arguments and, unlike the format that we present below, is therefore unable to detect that f is commutative in its first two arguments.

In what follows, we shall often use $[n]$, $n \geq 0$, to stand for the set $\{1, \ldots, n\}$. Note that [0] is just the empty set.

Definition 8 (Generalized Commutativity). *Given a set I, a family $\prod_I$ of non-empty subsets of I is called a* partition *of I when $\bigcup \prod_I = I$ and, for each two distinct $J, K \in \prod_I$, it holds that $J \cap K = \emptyset$.*

*Let Σ be a signature. Assume that $f \in \Sigma$ is an n-ary operator, $\prod_{[n]}$ is a partition of $[n]$ and $\sim$ is an equivalence relation over $\mathbb{C}(\Sigma)$. The operator f is called $\prod_{[n]}$-*commutative with respect to $\sim$ *when, for each $K \in \prod_{[n]}$ and each two $j, k \in K$ such that $j < k$, the following equation is sound with respect to $\sim$:*

$$f(x_1, \ldots, x_n) = f(x_1, \ldots, x_{j-1}, x_k, x_{j+1}, \ldots, x_{k-1}, x_j, x_{k+1}, \ldots, x_n),$$

where $x_1, \ldots, x_n$ is a sequence of variables.

Note that the traditional notion of commutativity for binary operators can be recovered using Definition 8 in terms of $\{\{1, 2\}\}$-commutativity. Moreover, the notion of commutativity for n-ary operators from [37] corresponds to $\{\{1, \ldots, n\}\}$-commutativity. Any n-ary operator is $1_{[n]}$-commutative with respect to any equivalence relation $\sim$, where $1_{[n]} = \{\{1\}, \ldots, \{n\}\}$ is the discrete partition of $[n]$.

From this point onward, whenever a signature Σ is provided, we also assume that every function symbol $f \in \Sigma$ of arity n has an associated fixed partition of its set of arguments $[n]$ denoted by $\prod_f$. We denote the indexed set of all these partitions by $\prod^{\Sigma} = \{\prod_f\}_{f \in \Sigma}$.

Definition 9. *Let $\prod_1$ and $\prod_2$ be partitions of some set I. We say that $\prod_1$ is* at least as fine as *$\prod_2$ if and only if for each $K_1 \in \prod_1$ there is some $K_2 \in \prod_2$ such that $K_1 \subseteq K_2$. A family of partitions $\prod_1^{\Sigma}$ is at least as fine as $\prod_2^{\Sigma}$ if and only if $\prod_{1f}$ is at least as fine as $\prod_{2f}$, for each $f \in \Sigma$. In other words, this holds if, for each $f \in \Sigma$, the equivalence relation associated with $\prod_{1f}$ is included in the equivalence relation associated with $\prod_{2f}$.*

Definition 10. *Assume $\Sigma_1 \subseteq \Sigma_2$. Let $\prod^{\Sigma_1}$ be a family of partitions. The* extension of $\prod^{\Sigma_1}$ to Σ_2 *is obtained by taking $\prod_f$ to be the discrete partition over $[n]$ for each $f \in \Sigma_2 \setminus \Sigma_1$, where n is the arity of f.*

Our aim is to define a restriction of the GSOS rule format that guarantees the notion of generalized commutativity defined above for any behavioural equivalence that is coarser than bisimilarity. To this end, we begin by extending the notion of commutative congruence introduced in [37] to the context of this generalized notion of commutativity.

Definition 11 (Commutative Congruence). *Consider a signature Σ and a set of partitions $\prod^{\Sigma}$. The commutative congruence relation $\sim_{cc}$ (with respect to $\prod^{\Sigma}$) is the least relation over $\mathbb{T}(\Sigma)$ satisfying the following requirements:*

1. *$\sim_{cc}$ is reflexive and transitive;*
2. *$\sim_{cc}$ is a congruence;*
3. *for all $f \in \Sigma$, $K \in \prod_f$, $j, k \in K$ with $j < k$, and $t_1, \ldots, t_n \in \mathbb{T}(\Sigma)$, it holds that*

$$f(t_1, \ldots, t_n) \sim_{cc} f(t_1, \ldots, t_{j-1}, t_k, t_{j+1}, \ldots, t_{k-1}, t_j, t_{k+1}, \ldots, t_n),$$

where n is the arity of f.

Lemma 1. *$\sim_{cc}$ is an equivalence relation over $\mathbb{T}(\Sigma)$. Moreover, for all terms t, u, if $t \sim_{cc} u$ then $vars(t) = vars(u)$.*

Proof. The relation $\sim_{cc}$ is reflexive and transitive by definition. The fact that it is also symmetric can be shown easily by induction on the definition of $\sim_{cc}$. The second claim also follows by a straightforward induction on the definition of $\sim_{cc}$. □

Lemma 2. *Let Σ be a signature and let $t, t' \in \mathbb{T}(\Sigma)$. Let σ and σ' be two substitutions. Assume that $t \sim_{cc} t'$ (with respect to some $\prod^{\Sigma}$) and that $\sigma(x) \sim_{cc} \sigma'(x)$, for each $x \in vars(t)$. Then $\sigma(t) \sim_{cc} \sigma'(t')$.*

Proof. The claim can be shown by induction on the definition of $\sim_{cc}$. In the case that $t \sim_{cc} t'$ because $t \equiv t'$, one proceeds by a further induction on the structure of t. The details of the proof are standard and we therefore omit them. □

The following two lemmas can both be shown by induction on the definition of $\sim_{cc}$.

Lemma 3. *Let Σ be a signature. Assume that $\prod_1^{\Sigma}$ and $\prod_2^{\Sigma}$ are two families of partitions over Σ, and that $\prod_1^{\Sigma}$ is at least as fine as $\prod_2^{\Sigma}$. Then the commutative congruence associated with $\prod_1^{\Sigma}$ is included in the commutative congruence associated with $\prod_2^{\Sigma}$.*

Lemma 4. *Assume $\Sigma_1 \subseteq \Sigma_2$. Let $\prod^{\Sigma_1}$ be a family of partitions. Then the commutative congruence associated with $\prod^{\Sigma_1}$ over Σ_1-terms is included in the commutative congruence associated with the extension of $\prod_1^{\Sigma_1}$ to Σ_2.*

We are now ready to present a syntactic restriction on the GSOS format that guarantees commutativity with respect to a set of partitions $\prod^{\Sigma}$ modulo any notion of behavioural equivalence that includes strong bisimilarity. Unlike the format for $\{[n]\}$-commutativity given in [37], the format offered below applies to generalized commutativity, in the sense of Definition 8, and is defined for TSSs whose rules can have negative premises. On the other hand, unlike ours, the format introduced in [37] applies to rules whose positive premises need not have variables as their sources and targets. This is a straightforward generalization of our format, but is not relevant for the purpose of this paper.

Definition 12 (Comm-GSOS). *A transition system specification over signature Σ is in the comm-GSOS format with respect to a set of partitions $\prod^{\Sigma}$ if it is in the GSOS format and for each f-defining deduction rule with $f \in \Sigma$ of the following form*

$$(d)\ \frac{H}{f(x_1, \ldots, x_n) \xrightarrow{l} t},$$

each $K \in \prod_f$ and for all $j, k \in K$ with $j < k$, there exist a deduction rule of the following form in the transition system specification

$$(d')\ \frac{H'}{f(x'_1, \ldots, x'_n) \xrightarrow{l} t'}$$

and a bijective mapping $\hbar$ over variables such that

- *$\hbar(x'_i) = x_i$ for each $i \in [n]$ such that $i \neq j$ and $i \neq k$,*
- *$\hbar(x'_j) = x_k$ and $\hbar(x'_k) = x_j$,*
- *$\hbar(t') \sim_{cc} t$, and*
- *$\hbar(H') = H$.*

Deduction rule d' is called a commutative mirror *of d (with respect to j, k and $\prod^{\Sigma}$).*

Informally, the role of the bijection $\hbar$ in the definition above is to account for the swapping of variables in the source of the conclusion and a possible bijective renaming of variables. Thus, the above format requires that, when $f \in \Sigma$, for each f-defining rule and for each pair (j, k) of arguments for which f is supposed to be commutative, as specified by $\prod_f$, there exists a commutative mirror that enables the 'same transitions up to the commutative congruence $\sim_{cc}$ associated with $\prod^{\Sigma}$' when the jth and kth arguments of f are swapped. This is the essence of the proof of the following theorem, which states the correctness of the syntactic comm-GSOS format.

Theorem 1. *If a transition system specification is in the comm-GSOS format with respect to a set of partitions $\prod^{\Sigma}$, then each operator $f \in \Sigma$ is $\prod_f$-commutative with respect to any notion of behavioural equivalence that includes bisimilarity.*

Proof. The restriction to closed terms of the commutative congruence relation with respect to $\prod^{\Sigma}$ is a bisimulation. The details of the proof may be found in Appendix A. □

Remark 3. Theorem 1 would still hold if the last constraint on the bijection $\hbar$ in Definition 12 were relaxed to $\hbar(H') \subseteq H$.

Example 1. Consider the ternary operator f we used earlier to motivate the notion of generalized commutativity. For ease of reference, we recall that the rules for f are

$$\frac{x \xrightarrow{a} x'}{f(x,y,z) \xrightarrow{a} f(x',y,z)} \quad \frac{y \xrightarrow{a} y'}{f(x,y,z) \xrightarrow{a} f(x,y',z)}$$
$$\frac{x \xrightarrow{a} x' \quad z \xrightarrow{a} z'}{f(x,y,z) \xrightarrow{a} f(x',y,z')} \quad \frac{y \xrightarrow{a} y' \quad z \xrightarrow{a} z'}{f(x,y,z) \xrightarrow{a} f(x,y',z')},$$

where a ranges over L.

Any transition system specification including the operator f given by the above rules is in the comm-GSOS format with respect to any set of partitions $\prod^{\Sigma}$ such that $\prod_f = \{\{1,2\},\{3\}\}$. Indeed, the a-emitting rules in the first row are one the commutative mirror of the other with respect to $\prod^{\Sigma}$, and so are those in the second row. By way of example, consider the rule $\dfrac{y \xrightarrow{a} y' \;\; z \xrightarrow{a} z'}{f(x,y,z) \xrightarrow{a} f(x,y',z')}$ and take $j = 1$ and $k = 2$ in Definition 12. To see that the rule $\dfrac{x \xrightarrow{a} x' \;\; z \xrightarrow{a} z'}{f(x,y,z) \xrightarrow{a} f(x',y,z')}$ is a commutative mirror of the other with respect to $\prod^{\Sigma}$, take the bijection $\hbar$ over variables such that $\hbar(x) = y$, $\hbar(y) = x$, $\hbar(x') = y'$, $\hbar(y') = x'$ and that is the identity function on all the other variables. Then $\hbar(\{x \xrightarrow{a} x',\ z \xrightarrow{a} z'\}) = \{y \xrightarrow{a} y',\ z \xrightarrow{a} z'\}$ and $\hbar(f(x',y,z')) = f(y',x,z') \sim_{cc} f(x,y',z')$. The constraints in Definition 12 are vacuously satisfied when we take $K = \{3\}$.

Therefore, by Theorem 1, we recover the fact that f is commutative in its first two arguments.

Example 2 (Parallel Composition). A frequently occurring commutative operator is parallel composition. It appears in, amongst others, ACP [20], CCS [35], and CSP [34,44]. Here we discuss parallel composition with communication in the style of ACP [20], of which the others are special cases. The rules for this operator are listed below. In those rules, a, b, c range over L and $\gamma : L \times L \hookrightarrow L$ is a partial communication function.

$$(\mathrm{p}_1)\ \frac{x \xrightarrow{a} x'}{x \,||\, y \xrightarrow{a} x' \,||\, y} \qquad (\mathrm{p}_2)\ \frac{y \xrightarrow{a} y'}{x \,||\, y \xrightarrow{a} x \,||\, y'} \qquad (\mathrm{p}_3)\ \frac{x \xrightarrow{a} x' \quad y \xrightarrow{b} y'}{x \,||\, y \xrightarrow{c} x' \,||\, y'} \quad \gamma(a,b) = c$$

If the partial synchronization function γ is commutative, then any GSOS system including the operator $||$ given by the above rules is in the comm-GSOS format with respect to any set of partitions $\prod^{\Sigma}$ such that $\prod_{||} = \{\{1,2\}\}$. Hence it follows from Theorem 1 that $||$ is $\{\{1,2\}\}$-commutative.

Example 3 (Timed Alternative Composition). Below, we consider the timing rules for a discrete-time version of the alternative composition operator [17]. This operator is commutative and its defining rules involve negative premises. (In the rules below, 1 denotes the discrete time transition.)

$$(\mathrm{at}_1)\ \frac{x \xrightarrow{1} x' \quad y \overset{1}{\not\rightarrow}}{x \otimes y \xrightarrow{1} x'} \qquad (\mathrm{at}_2)\ \frac{x \overset{1}{\not\rightarrow} \quad y \xrightarrow{1} y'}{x \otimes y \xrightarrow{1} y'} \qquad (\mathrm{at}_3)\ \frac{x \xrightarrow{1} x' \quad y \xrightarrow{1} y'}{x \otimes y \xrightarrow{1} x' \otimes y'}$$

Any TSS including the operator $\otimes$ given by the above rules is in the comm-GSOS format with respect to any set of partitions $\prod^{\Sigma}$ such that $\prod_{\otimes} = \{\{1,2\}\}$. Indeed, with respect to such $\prod^{\Sigma}$, each of the rules (at_1) and (at_2) is the commutative mirror of the other, and rule (at_3) is its own commutative mirror. From Theorem 1, it follows that $\otimes$ is $\{\{1,2\}\}$-commutative. (By including the standard action transitions for alternative composition, such as those given for the binary operator $+$, the same result can be obtained.)

Lemma 5. *Assume that a TSS $\mathcal{T}$ is in the* comm-GSOS *format with respect to a family of partitions $\prod_1^{\Sigma}$, and $\prod_1^{\Sigma}$ is at least as fine as $\prod_2^{\Sigma}$. Then $\mathcal{T}$ is also in the* comm-GSOS *format with respect to $\prod_2^{\Sigma}$.*

Proof. The claim follows from Lemma 3. □

4 Mechanized Axiomatization

In this section, our goal is to generate a disjoint extension of the original TSS and a finite axiom system that is sound and complete for bisimilarity over it. This finite axiom system may then also be used for equationally establishing bisimilarity over closed terms from the original TSS. We present a technique for the automatic generation of ground-complete axiomatizations of bisimilarity over TSSs in the comm-GSOS format, which is derived from the one introduced in [5]. Our approach improves upon the one in [5] by making use of the rule format for generalized commutativity we introduced in the previous section. We start by axiomatizing a rather restrictive subset of 'good' operators in Section 4.1. Then we turn 'bad' operators into good ones by means of auxiliary operators. In both of these steps, we exploit commutativity information, where possible, in order to reduce the number of generated axioms, as well as the number of generated auxiliary operators.

4.1 Axiomatizing Good Operators

The approach offered in [5] relies on the fact that the signature includes the three operators whose semantics has been presented in Section 2.2, namely the deadlock constant, the prefixing operator and the nondeterministic choice operator. (Recall that, in keeping with [21,23], we assume that these operators are present in any TSS in the GSOS format.) The aim of the axiomatization procedure is then to generate an axiom system that can rewrite any closed term p into a term p' in head normal form such that $p \leftrightarrow p'$. (We call an axiom system with this property *head normalizing*.)

Definition 13. *A term t is in* head normal form *if it has the form $a_1.t_1 + \cdots + a_n.t_n$ for some $n \geq 0$, some set of actions $\{a_i \mid i \in [n]\}$ and set of terms $\{t_i \mid i \in [n]\}$. If $n = 0$ then $a_1.t_1 + \cdots + a_n.t_n$ stands for* **0**.

For 'semantically well founded' terms (see [5, Definition 5.1 on page 28]), rewriting into head normal form can be used to prove that each closed term is equal to a closed term over the signature for BCCSP. This leads to a ground-complete axiomatization of bisimilarity, since BCCSP is finitely axiomatized modulo bisimilarity by the following axiom system E_{BCCSP} from [33]:

$$\begin{aligned} x + y &= y + x & \qquad x + x &= x \\ (x + y) + z &= x + (y + z) & \qquad x + \mathbf{0} &= x. \end{aligned}$$

Proposition 2 (Hennessy and Milner [33]). *The axiom system E_{BCCSP} is sound and ground complete for bisimilarity over BCCSP.*

To start with, we focus on the case of closed terms built using only *good* operators, which we now proceed to define.

Definition 14 (Smooth operator). *Consider an n-ary operator f.*

1. *A* smooth GSOS transition rule *is of the form*

$$\frac{\{x_i \xrightarrow{a_i} y_i \mid i \in I\} \cup \{x_i \overset{b_{ij}}{\not\rightarrow} \mid i \in J, 1 \leq j \leq n_i\}}{f(x_1, \ldots, x_n) \xrightarrow{c} C[\overrightarrow{x}, \overrightarrow{y}]}$$

   where
   (a) I and J are disjoint subsets of $[n]$ such that $I \cup J = [n]$;
   (b) $C[\overrightarrow{x}, \overrightarrow{y}]$ can only include the variables x_i ($i \in [n] \setminus I$) and y_i ($i \in I$).
2. *An operator f of a TSS in the GSOS format is* smooth *if all its rules are smooth.*

Definition 15 (Distinctive operator). *An n-ary operator f of a TSS in the GSOS format is* distinctive *if it is smooth and every two distinct f-defining rules are of the form*

$$\frac{\{x_i \xrightarrow{a_i} y_i \mid i \in I\} \cup \{x_i \overset{b_{ij}}{\not\rightarrow} \mid i \in J, 1 \leq j \leq n_i\}}{f(x_1, \ldots, x_n) \xrightarrow{a} C[\overrightarrow{x}, \overrightarrow{y}]} \quad \frac{\{x'_i \xrightarrow{a'_i} y'_i \mid i \in I'\} \cup \{x'_i \overset{b'_{ij}}{\not\rightarrow} \mid i \in J', 1 \leq j \leq n'_i\}}{f(x'_1, \ldots, x'_n) \xrightarrow{a'} C'[\overrightarrow{x'}, \overrightarrow{y'}]},$$

with $I = I'$ and $a_i \neq a'_i$ for some $i \in I$.

If f is smooth and distinctive, then each rule for f tests the same set of arguments I positively. Thus, we can associate this set I with the operator f. For such an operator f, the set J is equal to $[n] \setminus I$ in each rule for f.

Remark 4. The ternary operator f from Example 1 and the parallel composition operator from Example 2 are smooth but not distinctive. On the other hand, the classic communication merge operator [16,19], given by the rules

$$\frac{x \xrightarrow{a} x' \quad y \xrightarrow{b} y'}{x \mid y \xrightarrow{c} x' \,||\, y'} \qquad (\gamma(a, b) = c),$$

is smooth and distinctive. Moreover, assuming that γ is commutative, any TSS whose signature Σ includes $||$ and $|$ with the previously given rules is in the the comm-GSOS format with respect to any set of partitions $\prod^{\Sigma}$ such that $\prod_{|} = \prod_{||} = \{\{1, 2\}\}$.

Definition 16 (Discarding and Good Operators). *A smooth GSOS rule of the form given in Definition 14 is* discarding *if none of the variables x_i with $i \in J$ and $n_i > 0$ occurs in $C[\overrightarrow{x}, \overrightarrow{y}]$. A smooth operator is discarding if so are all the rules for it. A smooth operator is* good *[24] if it is both distinctive and discarding.*

In the remainder of this subsection, we assume that the GSOS system $\mathcal{T}$ has signature Σ and is in the comm-GSOS format with respect to a set of partitions $\prod^{\Sigma}$. Let $f \in \Sigma$ be a good operator that is not in the signature for BCCSP, and let n be its arity. Our goal is to generate an axiom system that can be used to turn any term of the form $f(t_1, \ldots, t_n)$, where the t_i's are in head normal form, into a head normal form. In the generation of the axiom system, we will exploit the commutativity information that is provided by the partition $\prod_f$ and therefore we assume that $n \geq 2$. (If f is either a constant or a unary operator, then it will be axiomatized exactly as in [5], since commutativity information is immaterial.) Let $I_f \subseteq [n]$ be the set of arguments that are tested positively by f and let J_f be the complement of I_f.

Assume that $\prod_f = \{K_1, \ldots, K_\ell\}$. Since $\mathcal{T}$ is in the comm-GSOS format with respect to $\prod^{\Sigma}$, and f is smooth and distinctive, it is not hard to see that, for each $h \in [\ell]$,

$$K_h \subseteq I_f \text{ or } K_h \subseteq J_f.$$

Indeed exactly one of the above inclusions holds. Let $\prod_f^+ = \{K \mid K \in \prod_f \text{ and } K \subseteq I_f\}$ and $\prod_f^- = \{K \mid K \in \prod_f \text{ and } K \subseteq J_f\}$. We use K_f^+ (respectively, K_f^-) to denote a subset of I_f (respectively, J_f) that results by choosing exactly one representative element for each $K \in \prod_f^+$ (respectively, $K \in \prod_f^-$).

Example 4. Consider the communication merge operator $|$ from Remark 4. We already remarked that, when the communication function γ is commutative, the rules for $|$ are in the the comm-GSOS format with respect to any set of partitions $\prod^{\Sigma}$ such that $\prod_| = \prod_{||} = \{\{1, 2\}\}$. For the operator $|$, we may take $K_|^+ = \{1\}$. Since the rules for $|$ have no negative premises, $K_|^-$ is empty.

Definition 17. *Let f be a good n-ary operator with the properties given previously, and let K_f^+ and K_f^- be defined as above. We associate with f the finite axiom system E_f consisting of the following equations.*

1. Distributivity laws*: For each $i \in K_f^+$, we have the equation:*

$$f(x_1, \ldots, x_i + x_i', \ldots, x_n) = f(x_1, \ldots, x_i, \ldots, x_n) + f(x_1, \ldots, x_i', \ldots, x_n).$$

2. Peeling laws*: For each rule for f of the form given in Definition 14, each $k \in K_f^-$ with $n_k > 0$ and each $a \notin \{b_{kj} \mid 1 \leq j \leq n_k\}$, we have the equation:*

$$f(P_1, \ldots, P_n) = f(Q_1, \ldots, Q_n),$$

where

$$P_i \equiv \begin{cases} a_i.y_i & i \in I \\ a.x_k' + x_k'' & i = k \\ x_i & \text{otherwise} \end{cases} \quad \text{and} \quad Q_i \equiv \begin{cases} a_i.y_i & i \in I \\ x_k'' & i = k \\ x_i & \text{otherwise}. \end{cases}$$

3. Action laws*: For each rule for f of the form given in Definition 14, we have the equation:*

$$f(P_1, \ldots, P_n) = c.C[\overrightarrow{P}, \overrightarrow{y}],$$

where

$$P_i \equiv \begin{cases} a_i.y_i & i \in I \\ \mathbf{0} & i \in J \textit{ and } n_i > 0 \\ x_i & \textit{otherwise.} \end{cases}$$

4. Inaction laws*: For each $i \in K_f^+$, we have the equation*

$$f(x_1, \ldots, x_{i-1}, \mathbf{0}, x_{i+1}, \ldots, x_n) = \mathbf{0}.$$

Suppose that, for each $i \in [n]$, term P_i is of the form $a.z_i$ when $i \in I_f$, and of the form $a.z_i + z_i'$ or z_i when $j \in J_f$. Suppose further that, for each rule for f of the form given in Definition 14, there exists some $i \in [n]$ such that one of the following holds:
– *$i \in I_f$ and ($P_i \equiv a.z_i$, for some $a \neq a_i$),*
– *$i \in J_f$ and ($P_i \equiv b_{ij}.z_i + z_i'$, for some $1 \leq j \leq n_i$).*
Then we have the equation

$$f(P_1, \ldots, P_n) = \mathbf{0}\ .$$

5. Commutativity law*: For each equivalence class $K \in \prod_f$ and each two $i, j \in K$ such that $i < j$, we have the equation:*

$$f(x_1, \ldots, x_i, \ldots, x_j, \ldots, x_n) = f(x_1, \ldots, x_j, \ldots, x_i, \ldots, x_n).$$

The axiom system E_f (a rewrite system, when the axioms are oriented from left to right), in combination with the axioms for BCCSP, can be used to turn any term of the form $f(t_1, \ldots, t_n)$, where the t_i's are in head normal form, into a head normal form.

Theorem 2. *Consider a TSS $\mathcal{T}$ in GSOS format. Let Σ_g be a collection of good operators of $\mathcal{T}$. Let E_{Σ_g} be the finite axiom system that consists of the axioms in E_{BCCSP} and the axioms in E_f, for each $f \in \Sigma_g$. The following statements hold for any GSOS system $\mathcal{T}'$ such that $\mathcal{T} \sqsubseteq \mathcal{T}'$:*

1. *E_{Σ_g} is sound modulo bisimilarity over $\mathbb{T}(\Sigma')$, where Σ' is the signature of $\mathcal{T}'$.*
2. *E_{Σ_g} is head normalizing for $\mathbb{C}(\Sigma_g)$.*

Proof. The two statements can be shown following the lines of the corresponding theorem in [5]. See also the proof of Theorem 5.8 in [1]. □

Example 5. For the communication merge operator $|$, taking $K_|^+ = \{1\}$ as in Example 4, Definition 17 yields the following axiom system $E_|$:

$$\begin{array}{ll} (x + y) \mid z = (x \mid z) + (y \mid z) & a.x \mid b.y = c.(x \,||\, y) \quad \text{if } \gamma(a, b) = c \\ \mathbf{0} \mid y = \mathbf{0} & a.x \mid b.y = \mathbf{0} \quad \text{if } \gamma(a, b) \text{ is undefined.} \end{array}$$

These are exactly the equations describing the interplay between the operator $|$ and the BCCSP operators given in Table 7.1 on page 204 of [16].

4.2 Turning Bad into Good

In order to handle arbitrary GSOS operators, one needs two additional procedures: one for transforming non-smooth operators into smooth and discarding (but not necessarily distinctive) operators, and one for expressing smooth, discarding and non-distinctive operators in terms of good operators. We adopt the same approach for the first procedure as the one presented in Lemma 4.13 in [5]. (See also Proposition 5.13 in [1].) On the other hand, for the second procedure, we improve on the algorithm derived from Lemma 4.10 in [5] and Proposition 5.9 in [1].

The step from smooth, discarding and non-distinctive operators to good ones involves the synthesis of several new operators. We now show how to improve this transformation, as presented in the aforementioned references, by reducing the number of the generated auxiliary operators, making use of the ideas underlying the generalized commutativity format presented in Section 3.

Making Smooth and Discarding Operators Distinctive. Consider a TSS $\mathcal{T}$ with signature Σ in the comm-GSOS format with respect to a set of partitions $\prod^{\Sigma}$. Let $f \in \Sigma$ be a smooth and discarding, but not distinctive operator, and let n be its arity. We will now show how to express f in terms of good operators. We start with partition the set of f-defining rules into sets $R_1, \ldots, R_m$, $m > 1$, such that f is distinctive when its rules are restricted to those in R_i for each $i \in [m]$. Note that all the rules in each R_i test the same arguments positively. If $\prod_f$ is the discrete partition over $[n]$ then one proceeds by axiomatizing f as in the version of the original algorithm based on the so-called peeling laws presented in [5]. Indeed, in that case, f has no pair of commutative arguments. Suppose therefore that $\prod_f$ is not the discrete partition, and take some $K \in \prod_f$ of maximum cardinality.

Remark 5. Any non-singleton K would do in what follows. However, picking a K of maximum cardinality will reduce the number of auxiliary operators that is generated by the procedure outlined below.

Our aim now is to define when two sets of rules for f are 'essentially the same up to the commutative arguments in K' and to use this information in order to synthesize enough good operators for expressing f up to bisimilarity.

Definition 18.

– *Let* d *and* d' *be two* f*-defining and* l*-emitting rules. We say that* d' *is a* commutative mirror *of* d *with respect to* K *and* $\prod^{\Sigma}$ *if the constraints in Definition 12 are met for some* $j, k \in K$ *with* $j < k$.

– *We use* $\overset{K}{\smile}$ *to denote the reflexive and transitive closure of the relation 'is a commutative mirror with respect to* K*'.*

– *Let* R *and* R' *be two sets of* f*-defining rules. We write* $R \overset{K}{\smile} R'$ *if, and only if,*
  - *for each* $\mathrm{d} \in R$ *there is some* $\mathrm{d}' \in R'$ *such that* $\mathrm{d} \overset{K}{\smile} \mathrm{d}'$*, and*
  - *for each* $\mathrm{d}' \in R'$ *there is some* $\mathrm{d} \in R$ *such that* $\mathrm{d} \overset{K}{\smile} \mathrm{d}'$*.*

Example 6. Consider the ternary operator f defined by the rules on page 6. That operator is smooth and discarding, but not distinctive. Collecting all the rules that test the same arguments positively in the same set, we obtain the following four sets of rules:

– R_1 contains all the rules of the form $\dfrac{x \xrightarrow{a} x'}{f(x,y,z) \xrightarrow{a} f(x',y,z)} \quad (a \in L)$.
– R_2 contains all the rules of the form $\dfrac{y \xrightarrow{a} y'}{f(x,y,z) \xrightarrow{a} f(x,y',z)} \quad (a \in L)$.
– R_3 contains all the rules of the form $\dfrac{x \xrightarrow{a} x',\ z \xrightarrow{a} z'}{f(x,y,z) \xrightarrow{a} f(x',y,z')} \quad (a \in L)$.
– R_4 contains all the rules of the form $\dfrac{y \xrightarrow{a} y',\ z \xrightarrow{a} z'}{f(x,y,z) \xrightarrow{a} f(x,y',z')} \quad (a \in L)$.

We have already seen in Example 1 that any GSOS system including the operator f is in the comm-GSOS format with respect to any set of partitions $\prod^{\Sigma}$ such that $\prod_f = \{\{1,2\},\{3\}\}$. Take $K = \{1,2\}$.

It is not hard to see that $R_1 \overset{K}{\smile} R_2$ and $R_3 \overset{K}{\smile} R_4$ hold. Indeed, as we observed in Example 1, each a-emitting rule in R_1 (respectively, R_3) is the commutative mirror of the a-emitting rule in R_2 (respectively, R_4) with respect to K, and vice versa.

Lemma 6. *$\overset{K}{\smile}$ is an equivalence relation over f-defining rules and over sets of f-defining rules.*

The following notion will be useful in characterizing the relation $\overset{K}{\smile}$ over the collection of f-defining rules.

Definition 19. *Let $n > 0$ and let $K \subseteq [n]$. A bijection $\pi : [n] \to [n]$ is a K-*permutation *if it is the identity function over $[n] \setminus K$.*

Lemma 7. *Let d and d' be two f-defining and l-emitting rules. Suppose that $d = \dfrac{H}{f(x_1,\ldots,x_n) \xrightarrow{l} t}$ and $d' = \dfrac{H'}{f(x_1,\ldots,x_n) \xrightarrow{l} t'}$. Then $d \overset{K}{\smile} d'$ if, and only if there are some K-permutation π and some bijection $\hbar$ over variables such that*

– *$\hbar(x_i) = x_{\pi(i)}$, for each $i \in [n]$,*
– *$\hbar(t') \sim_{cc} t$, and*
– *$\hbar(H') = H$.*

Proof. By induction on the definition of $\overset{K}{\smile}$. □

Recall that $\{R_1,\ldots,R_m\}$, $m > 1$, is a partition of the set of f-defining rules such that f is distinctive when its rules are restricted to those in R_i for each $i \in [m]$. Consider $\{R_1,\ldots,R_m\}/\overset{K}{\smile}$, the quotient of the set $\{R_1,\ldots,R_m\}$ with respect to the equivalence relation $\overset{K}{\smile}$. Let $\rho_1,\ldots,\rho_\ell$ be representatives of its equivalence classes. For example, in the case of the operator considered in Example 6 above, one could pick R_1 and R_4, say, as representatives of the two equivalence classes with respect to $\overset{\{1,2\}}{\smile}$.

We proceed by adding to the signature Σ fresh n-ary operator symbols $f_1, \ldots, f_\ell$. The rules for the operator f_i are obtained by simply turning those in ρ_i into f_i-defining ones. Let $\mathcal{T}'$ be the resulting disjoint extension of $\mathcal{T}$.

Following [5], we now need to generate an axiom that expresses f in terms of $f_1, \ldots, f_\ell$.

The equation relating f to the f_i's, $i \in [\ell]$, can now be stated as follows:

$$f(x_1, \ldots, x_n) = \sum_{i=1}^{\ell} \sum \{f_i(x_{\pi(1)}, \ldots, x_{\pi(n)}) \mid \pi \text{ a } K\text{-permutation}\}. \tag{1}$$

For our running example, namely the ternary operator f defined by the rules on page 6 and considered in Examples 1 and 6, with the choice of representatives mentioned above, there are two auxiliary operators f_1 and f_2 with rules $\dfrac{x \xrightarrow{a} x'}{f_1(x, y, z) \xrightarrow{a} f(x', y, z)}$ $\dfrac{y \xrightarrow{a} y',\ z \xrightarrow{a} z'}{f_2(x, y, z) \xrightarrow{a} f(x, y', z')}$, where a ranges over L. Apart from the identity function over [3], the only $\{1, 2\}$-permutation is the one that swaps 1 and 2. Therefore, instantiating equation (1), we obtain that

$$f(x_1, x_2, x_3) = f_1(x_1, x_2, x_3) + f_1(x_2, x_1, x_3) + f_2(x_1, x_2, x_3) + f_2(x_2, x_1, x_3).$$

Using Definition 10, the family of partitions $\prod^{\Sigma}$ can be extended to any signature that includes the signature $\Sigma \cup \{f_i \mid i \in [\ell]\}$. Note that any disjoint extension of $\mathcal{T}'$ is in the comm-GSOS format with respect to this extension of $\prod^{\Sigma}$.

Proposition 3. *Equation (1) is sound in any disjoint extension of $\mathcal{T}'$.*

Proof. See Appendix B. □

Equation (1) can be simplified in case any of the auxiliary operators $f_1, \ldots, f_\ell$ is commutative in the set of arguments K. Indeed, let $N \subseteq [\ell]$, and assume that $\mathcal{T}'$ is in the comm-GSOS format with respect to the family of partitions that associates with each operator g the partition $\prod_g^{\Sigma}$ when $g \in \Sigma$, the partition $\{K\} \cup 1_{[n] \setminus K}$ when $g \in \{f_i \mid i \in N\}$, and the partition $1_{[n]}$ otherwise. Then we have that

Proposition 4. *The equation*

$$\begin{aligned} f(x_1, \ldots, x_n) = & \sum_{i \in [\ell] \setminus N} \sum \{f_i(x_{\pi(1)}, \ldots, x_{\pi(n)}) \mid \pi \textit{ a } K\textit{-permutation}\} + \\ & \sum_{i \in N} f_i(x_1, \ldots, x_n) \end{aligned} \tag{2}$$

is sound in any disjoint extension of $\mathcal{T}'$.

In the following section, we will see that the above simplification leads to an axiomatization of the classic parallel composition operator that is equal to an existing handcrafted one. Of course, if either N or $[\ell] \setminus N$ are empty, the corresponding $\mathbf{0}$ summand can be omitted in equation (2).

Turning non-smooth operators into smooth ones. The developments we have presented so far yield an algorithm that, given a TSS $\mathcal{T}$ with signature Σ in the comm-GSOS format with respect to a set of partitions $\prod^{\Sigma}$, can be used to generate

- a disjoint extension $\mathcal{T}'$ of $\mathcal{T}$ over some signature Σ' that includes Σ and
- a finite axiom system E

such that E is sound modulo bisimilarity over any disjoint extension of $\mathcal{T}'$ and is head normalizing for all closed Σ'-terms. Ground-completeness of E with respect to bisimilarity over $\mathcal{T}'$ (and therefore over $\mathcal{T}$) follows using standard reasoning, by possibly using the well-known Approximation Induction Principle [19] if $\mathcal{T}'$ is not semantically well founded. See [5] for details.

The algorithm has the following steps:

1. Start with the axiom system E_{BCCSP} and consider next the operators that are not in the signature for BCCSP.
2. For each non-smooth operator $f \in \Sigma$, generate a fresh smooth and discarding operator f', and add to the axiom system the equation expressing f in terms of f' as in Lemma 4.13 in [5]. (See also Proposition 5.13 in [1].)
3. For each smooth and discarding, but not distinctive, operator f in the resulting signature, generate a family of fresh good operators $f_1, \ldots f_\ell$, as indicated in this section, and add to the axiom system the instance of equation (1) or of equation (2), as appropriate, expressing f in terms of $f_1, \ldots f_\ell$.
4. For each good operator in the resulting signature, add to the axiom system the equations mentioned in the statement of Theorem 2.

5 Axiomatizing Parallel Composition

Let us concretely analyze the axiomatization derived using the procedure described above for the classic parallel composition operator $||$ from Example 2.

We assume henceforth that the partial synchronization function γ is commutative, so that $||$ is $\{\{1,2\}\}$-commutative. As we observed in Remark 4, the parallel composition operator is smooth but not distinctive. When we partition the set of rules for $||$ into subsets of rules that test the same arguments positively, we obtain three sets R_1, R_2 and R_3, where each R_i consists of all the instances of rule (p$_i$) from Example 2. It is easy to see that $R_1 \overset{\{1,2\}}{\smile} R_2$. Therefore, following the procedure described in Section 4.2, we can generate two auxiliary binary operators, which are the classic left merge and communication operators, denoted by $⌊⌊$ and $|$, respectively. The rules for $|$ are those in Remark 4 and those for the left merge operator are

$$\frac{x \xrightarrow{a} x'}{x ⌊⌊\, y \xrightarrow{a} x' \,||\, y} \qquad (a \in L).$$

Since we know that $|$ is $\{\{1,2\}\}$-commutative, the relationship between $||$ and the two auxiliary operators can be expressed using equation (2), whose relevant instance becomes

$$x \,||\, y = (x ⌊⌊\, y) + (y ⌊⌊\, x) + (x \mid y).$$

This is exactly equation M in Table 7.1 on page 204 of [16]. The axioms for $\mid$ produced by our methods are those given in Example 5. On the other hand, the left merge operator is axiomatized as in [5] since commutativity information is immaterial for it.

In Figure 1 we compare the axiomatization for the parallel composition operator $\parallel$ derived using the algorithm from [5] and the 'optimized axiomatization' one obtains using the algorithm mentioned above. (We omit the four equations in the axiom system E_{BCCSP} recalled in Section 4.) The axioms generated by the algorithm from [5] do resemble the original axioms of [20] to a large extent. The auxiliary operator $\mathbin{\rfloor\mkern-9mu\Vert}$ is called right merge in the literature.

Standard	Optimized
$x \parallel y = (x \mathbin{\Vert\mkern-9mu\lfloor} y) + (x \mathbin{\rfloor\mkern-9mu\Vert} y) + (x \mid y)$	$x \parallel y = (x \mathbin{\Vert\mkern-9mu\lfloor} y) + (y \mathbin{\Vert\mkern-9mu\lfloor} x) + (x \mid y)$
$(a.x) \mathbin{\Vert\mkern-9mu\lfloor} y = a.(x \parallel y)$	$(a.x) \mathbin{\Vert\mkern-9mu\lfloor} y = a.(x \parallel y)$
$x \mathbin{\rfloor\mkern-9mu\Vert} (a.y) = a.(x \parallel y)$	$(a.x) \mid (b.y) = c.(x \parallel y) \quad \text{if } \gamma(a,b) = c$
$(a.x) \mid (b.y) = c.(x \parallel y) \quad \text{if } \gamma(a,b) = c$	$(x+y) \mathbin{\Vert\mkern-9mu\lfloor} z = (x \mathbin{\Vert\mkern-9mu\lfloor} z) + (y \mathbin{\Vert\mkern-9mu\lfloor} z)$
$(x+y) \mathbin{\Vert\mkern-9mu\lfloor} z = (x \mathbin{\Vert\mkern-9mu\lfloor} z) + (y \mathbin{\Vert\mkern-9mu\lfloor} z)$	$(x+y) \mid z = (x \mid z) + (y \mid z)$
$x \mathbin{\rfloor\mkern-9mu\Vert} (y+z) = (x \mathbin{\rfloor\mkern-9mu\Vert} y) + (x \mathbin{\rfloor\mkern-9mu\Vert} z)$	
$(x+y) \mid z = (x \mid z) + (y \mid z)$	
$x \mid (y+z) = (x \mid y) + (x \mid z)$	$\mathbf{0} \mathbin{\Vert\mkern-9mu\lfloor} x = \mathbf{0}$
$\mathbf{0} \mathbin{\Vert\mkern-9mu\lfloor} x = \mathbf{0}$	
$x \mathbin{\rfloor\mkern-9mu\Vert} \mathbf{0} = \mathbf{0}$	$\mathbf{0} \mid x = \mathbf{0}$
$\mathbf{0} \mid x = \mathbf{0}$	
$x \mid \mathbf{0} = \mathbf{0}$	$(a.x) \mid (b.y) = \mathbf{0} \quad \text{if } \gamma(a,b) \text{ is undefined}$
$(a.x) \mid (b.y) = \mathbf{0} \quad \text{if } \gamma(a,b) \text{ is undefined}$	$x \mid y = y \mid x$

Fig. 1. Axiomatizing $\parallel$

We implemented the axiomatizations in the rewriting logic specification and programming language Maude [25]. The optimized axiomatization consists of fewer axioms and uses only two auxiliary operators instead of three, yet it brings terms into normal forms, which only contain the operators from the signature of BCCSP, only negligibly faster than the standard axiomatization. For example, the term $a^3 \parallel b^3 \parallel c^3$ is reduced to its normal form by performing 5736 rewrites using the optimized axiomatization and 5748 when using the standard one, at the same average speed of $\sim$350.000 rewrites/second. For the purpose of the experiment, we considered the communication function defined by $\gamma(l_1, l_2) = \min(l_1, l_2)$, where $l_1, l_2 \in \{a, b, c\}$ and $a < b < c$.

6 Conclusions and Future Work

In this paper, we have taken a first step towards marrying two lines of development within the field of the meta-theory of SOS, viz. the study of algorithms for the automatic generation of ground-complete axiomatizations for bisimilarity from SOS specifications (see, for instance, [3,5,6,18,22,30,46]) and the development of rule formats guaranteeing the validity of algebraic laws, such as those presented in [4,8,9,15,26,37]. More specifically, we have presented a rule format for commutativity that refines the one offered in [37] in that it allows one to consider various sets of commutative arguments, and we have used the information provided by that rule format to refine the algorithm for the automatic generation of ground-complete axiomatizations for bisimilarity from [5]. The resulting procedure produces axiom systems that use fewer auxiliary operators than the one from [5]. Moreover, in some important cases, the mechanically produced axiomatizations of some operators are identical to the hand-crafted ones that had been presented in the literature.

Admittedly, for the operators we tested, neither making use of the commutativity format to reduce the number of additional operators, nor the alternative approaches for axiomatizing negative premises we investigated helped significantly in improving the performance of the axiomatizations when reducing terms to 'normal forms'. However, to the best of our knowledge, the ideas we have presented in this paper have never been explored before, and they enrich the toolbox one can use when reasoning about bisimilarity by means of axiomatizations. Moreover, the combination of two closely related strands of research on the meta-theory of SOS we have begun in this paper is of theoretical interest and may lead to further improvements on algorithms for the automatic generation of axiomatic characterizations of bisimilarity.

As future work, we will implement the algorithm presented in this paper in the PREG Axiomatizer tool [7]. We also intend to explore the use of other rule formats for algebraic properties in improving mechanized axiomatizations for bisimilarity. The ultimate goals of this research are to make automatically generated axiomatizations comparable to the known ones from the literature and to make the first steps towards the automatic generation of axiomatizations that are complete for open terms. The latter goal is a very ambitious one since obtaining complete axiomatizations of bisimilarity is a very hard research problem even for sufficiently rich fragments of specific process calculi; see, for instance, [10,11,14].

References

1. L. Aceto. Deriving complete inference systems for a class of GSOS languages generating regular behaviours. Report IR 93–2009, Institute for Electronic Systems, Department of Mathematics and Computer Science, Aalborg University, Aalborg, 1993. Also available as Computer Science Report 1/94, University of Sussex, January 1994.
2. L. Aceto. GSOS and finite labelled transition systems. *Theoretical Computer Science*, 131:181–195, 1994.
3. Luca Aceto. Deriving complete inference systems for a class of GSOS languages generating regular behaviours. In Bengt Jonsson and Joachim Parrow, editors, *Proceedings of the fifth International Conference on Concurrency Theory (CONCUR'94)*, volume 836 of *Lecture Notes in Computer Science*, pages 449–464. Springer-Verlag, Berlin, Germany, 1994.
4. Luca Aceto, Arnar Birgisson, Anna Ingólfsdóttir, Mohammad Reza Mousavi, and Michel A. Reniers. Rule formats for determinism and idempotence. *Sci. Comput. Program.*, 77(7–8):889–907, 2012.
5. Luca Aceto, Bard Bloom, and Frits W. Vaandrager. Turning SOS rules into equations. *Information and Computation (I&C)*, 111:1–52, 1994.

6. Luca Aceto, Georgiana Caltais, Eugen-Ioan Goriac, and Anna Ingólfsdóttir. Axiomatizing GSOS with predicates. In Michel A. Reniers and Pawel Sobocinski, editors, *Proceedings Eight Workshop on Structural Operational Semantics 2011*, volume 62 of *EPTCS*, pages 1–15, 2011.
7. Luca Aceto, Georgiana Caltais, Eugen-Ioan Goriac, and Anna Ingólfsdóttir. PREG Axiomatizer - a ground bisimilarity checker for GSOS with predicates. In Andrea Corradini, Bartek Klin, and Corina Cîrstea, editors, *Algebra and Coalgebra in Computer Science - 4th International Conference, CALCO 2011, Winchester, UK, August 30-September 2, 2011. Proceedings*, volume 6859 of *Lecture Notes in Computer Science*, pages 378–385. Springer, 2011.
8. Luca Aceto, Matteo Cimini, Anna Ingólfsdóttir, Mohammad Reza Mousavi, and Michel A. Reniers. Rule formats for distributivity. In Adrian Horia Dediu, Shunsuke Inenaga, and Carlos Martín-Vide, editors, *Language and Automata Theory and Applications - 5th International Conference, LATA 2011, Tarragona, Spain, May 26–31, 2011. Proceedings*, volume 6638 of *Lecture Notes in Computer Science*, pages 80–91. Springer, 2011.
9. Luca Aceto, Matteo Cimini, Anna Ingólfsdóttir, Mohammad Reza Mousavi, and Michel A. Reniers. SOS rule formats for zero and unit elements. *Theor. Comput. Sci.*, 412(28):3045–3071, 2011.
10. Luca Aceto, Wan Fokkink, Anna Ingólfsdóttir, and Bas Luttik. Finite equational bases in process algebra: Results and open questions. In Aart Middeldorp, Vincent van Oostrom, Femke van Raamsdonk, and Roel C. de Vrijer, editors, *Processes, Terms and Cycles: Steps on the Road to Infinity, Essays Dedicated to Jan Willem Klop, on the Occasion of His 60th Birthday*, volume 3838 of *Lecture Notes in Computer Science*, pages 338–367. Springer, 2005.
11. Luca Aceto, Wan Fokkink, Anna Ingólfsdóttir, and Bas Luttik. A finite equational base for CCS with left merge and communication merge. *ACM Trans. Comput. Log.*, 10(1), 2009.
12. Luca Aceto, Willem Jan (Wan) Fokkink, and Chris Verhoef. Structural operational semantics. In Jan A. Bergstra, Alban Ponse, and Scott A. Smolka, editors, *Handbook of Process Algebra, Chapter 3*, pages 197–292. Elsevier Science, Dordrecht, The Netherlands, 2001.
13. Luca Aceto and Anna Ingolfsdottir. CPO models for compact GSOS languages. *Information and Computation*, 129(2):107–141, 1996.
14. Luca Aceto, Anna Ingólfsdóttir, Bas Luttik, and Paul van Tilburg. Finite equational bases for fragments of CCS with restriction and relabelling. In Giorgio Ausiello, Juhani Karhumäki, Giancarlo Mauri, and C.-H. Luke Ong, editors, *Fifth IFIP International Conference On Theoretical Computer Science - TCS 2008, IFIP 20th World Computer Congress, TC 1, Foundations of Computer Science, September 7–10, 2008, Milano, Italy*, volume 273 of *IFIP*, pages 317–332. Springer, 2008.
15. Luca Aceto, Anna Ingolfsdottir, MohammadReza Mousavi, and Michel A. Reniers. Algebraic properties for free! *Bulletin of the European Association for Theoretical Computer Science (BEATCS)*, 99:81–104, 2009.
16. J.C.M. Baeten, T. Basten, and M.A. Reniers. *Process Algebra: Equational Theories of Communicating Processes*, volume 50 of *Cambridge Tracts in Theoretical Computer Science*. Cambridge University Press, 2009.
17. J.C.M. (Jos) Baeten and Jan A. Bergstra. Discrete time process algebra. *Formal Aspects of Computing*, 8(2):188–208, 1996.
18. J.C.M. (Jos) Baeten and Erik P. de Vink. Axiomatizing GSOS with termination. *Journal of Logic and Algebraic Programming (JLAP)*, 60-61:323–351, 2004.
19. Jan A. Bergstra and Jan Willem Klop. Fixedpoint semantics in process algebra. Technical Report IW 206/82, Center for Mathematics, Amsterdam, The Netherlands, 1982.
20. Jan A. Bergstra and Jan Willem Klop. Process algebra for synchronous communication. *Information and Control*, 60(1-3):109–137, 1984.

21. Bard Bloom. *Ready Simulation, Bisimulation, and the Semantics of CCS-like Languages*. PhD thesis, Department of Electrical Engineering and Computer Science, Massachusetts Institute of Technology, 1989.
22. Bard Bloom. Structural operational semantics for weak bisimulations. *Theoretical Computer Science (TCS)*, 146:25–68, 1995.
23. Bard Bloom, Sorin Istrail, and Albert R. Meyer. Bisimulation can't be traced. *Journal of the ACM (JACM)*, 42(1):232–268, January 1995.
24. D. Bosscher. Term rewriting properties of SOS axiomatisations. In Masami Hagiya and John C. Mitchell, editors, *Theoretical Aspects of Computer Software, International Conference TACS '94, Sendai, Japan, April 19–22, 1994, Proceedings*, volume 789 of *Lecture Notes in Computer Science*, pages 425–439. Springer, 1994.
25. Manuel Clavel, Francisco Durán, Steven Eker, Patrick Lincoln, Narciso Martí-Oliet, José Meseguer, and Carolyn L. Talcott, editors. *All About Maude - A High-Performance Logical Framework, How to Specify, Program and Verify Systems in Rewriting Logic*, volume 4350 of *Lecture Notes in Computer Science*. Springer, 2007.
26. Sjoerd Cranen, Mohammad Reza Mousavi, and Michel A. Reniers. A rule format for associativity. In Franck van Breugel and Marsha Chechik, editors, *CONCUR 2008 - Concurrency Theory, 19th International Conference, CONCUR 2008, Toronto, Canada, August 19–22, 2008. Proceedings*, volume 5201 of *Lecture Notes in Computer Science*, pages 447–461. Springer, 2008.
27. R. De Nicola and M. Hennessy. Testing equivalences for processes. *Theoretical Computer Science*, 34:83–133, 1984.
28. Willem Jan (Wan) Fokkink and Chris Verhoef. A conservative look at operational semantics with variable binding. *Information and Computation (I&C)*, 146(1):24–54, 1998.
29. Robert Jan (Rob) van Glabbeek. The linear time - branching time spectrum I. In Jan A. Bergstra, Alban Ponse, and Scott A. Smolka, editors, *Handbook of Process Algebra, Chapter 1*, pages 3–100. Elsevier Science, Dordrecht, The Netherlands, 2001.
30. Robert Jan (Rob) van Glabbeek. On cool congruence formats for weak bisimulations (extended abstract). In *Proceedings of the 2nd International Colloquium on Theoretical Aspects of Computing (ICTAC'05)*, Lecture Notes in Computer Science. Springer-Verlag, Berlin, Germany, 2005. To appear.
31. Robert Jan (Rob) van Glabbeek. The meaning of negative premises in transition system specifications II. *Journal of Logic and Algebraic Programming (JLAP)*, 60-61:229–258, 2004.
32. Matthew Hennessy and Gordon D. Plotkin. Full abstraction for a simple parallel programming language. In Jirí Becvár, editor, *Proceedings of the 8th Symposium on Mathematical Foundations of Computer Science (MFCS'79)*, volume 74 of *Lecture Notes in Computer Science*, pages 108–120. Springer-Verlag, Berlin, Germany, 1979.
33. Matthew C. B. Hennessy and A.J.R.G. (Robin) Milner. Algebraic laws for non-determinism and concurrency. *Journal of the ACM*, 32(1):137–161, 1985.
34. C.A.R. (Tony) Hoare. *Communicating Sequential Processes*. Prentice Hall, 1985.
35. A.J.R.G. (Robin) Milner. *A Calculus of Communicating Systems*, volume 92 of *Lecture Notes in Computer Science*. Springer-Verlag, 1980.
36. A.J.R.G. (Robin) Milner. *Communication and Concurrency*. Prentice Hall, 1989.
37. MohammadReza Mousavi, Michel Reniers, and Jan Friso Groote. A syntactic commutativity format for SOS. *Information Processing Letters (IPL)*, 93:217–223, March 2005.
38. MohammadReza Mousavi and Michel A. Reniers. Orthogonal extensions in structural operational. In *Proceedings of the 32nd International Colloquium on Automata, Languages and Programming (ICALP'05)*, volume 3580 of *Lecture Notes in Computer Science*, pages 1214–1225. Springer-Verlag, Berlin, Germany, 2005.

39. Mohammadreza Mousavi, Michel A. Reniers, and Jan Friso Groote. SOS formats and meta-theory: 20 years after. *Theoretical Computer Science (TCS)*, 373:238–272, 2007.
40. David M.R. Park. Concurrency and automata on infinite sequences. In *Proceedings of the 5th GI Conference*, volume 104 of *Lecture Notes in Computer Science*, pages 167–183. Springer-Verlag, Berlin, Germany, 1981.
41. Gordon D. Plotkin. A structural approach to operational semantics. Technical Report DAIMI FN-19, Computer Science Department, Aarhus University, Aarhus, Denmark, September 1981.
42. Gordon D. Plotkin. The origins of structural operational semantics. *Journal of Logic and Algebraic Programming (JLAP)*, 60:3–15, 2004.
43. Gordon D. Plotkin. A structural approach to operational semantics. *Journal of Logic and Algebraic Progamming (JLAP)*, 60:17–139, 2004. This article first appeared as [41].
44. A. W. Roscoe, C. A. R. Hoare, and Richard Bird. *The Theory and Practice of Concurrency*. Prentice Hall PTR, Upper Saddle River, NJ, USA, 1997.
45. R. de Simone. Higher-level synchronising devices in MEIJE–SCCS. *Theoretical Computer Science*, 37:245–267, 1985.
46. Irek Ulidowski. Finite axiom systems for testing preorder and De Simone process languages. *Theoretical Computer Science*, 239(1):97–139, 2000.
47. Irek Ulidowski. Priority rewrite systems for OSOS process languages. In Roberto M. Amadio and Denis Lugiez, editors, *Proceedings of CONCUR 2003 - Concurrency Theory, 14th International Conference, Marseille, France, September 3–5, 2003*, volume 2761 of *Lecture Notes in Computer Science*, pages 87–101. Springer, 2003.

A Proving Theorem 1

Let R be the relation $\sim_{cc}$ restricted to closed terms. By definition, this relation contains all the desired pairs of terms of the form

$$(f(p_1, \dots, p_j, \dots, p_k, \dots, p_n), f(p_1, \dots, p_k, \dots, p_j, \dots, p_n)),$$

where $j, k \in K$ for some $K \in \prod_f$ and $j < k$. It therefore suffices to prove that R is a bisimulation relation. To this end, note, first of all, that R is symmetric since so is $\sim_{cc}$.

Consider now an arbitrary pair $(p, q) \in R$. Suppose that $p \xrightarrow{l} p'$ for some $l \in L$ and p'. We have to prove the existence of a q' such that $q \xrightarrow{l} q'$ and $(p', q') \in R$. This we show by induction on the definition of R, and proceed with a case analysis on the reason why $(p, r) \in R$ holds. (Recall that R is the restriction of $\sim_{cc}$ to closed terms.)

1. If $p \equiv q$ then the claim holds since $\sim_{cc}$ is reflexive.
2. Assume that (p, q) is in R because $p = f(p_1, \dots, p_n)$ and $q = f(q_1, \dots, q_n)$, for some n-ary $f \in \Sigma$ and closed terms p_i and q_i such that $(p_i, q_i) \in R$ $(1 \leq i \leq n)$. As $p \xrightarrow{l} p'$, there are a rule

$$\text{(d)}\frac{\{x_i \xrightarrow{l_{ij}} y_{ij} \mid 1 \leq i \leq n, 1 \leq j \leq m_i\} \cup \{x_i \overset{l_{ik}}{\not\rightarrow} \mid 1 \leq i \leq n, 1 \leq k \leq n_i\}}{f(\overrightarrow{x}) \xrightarrow{l} t}$$

and a substitution σ such that
– $\sigma(x_i) = p_i \xrightarrow{l_{ij}} \sigma(y_{ij})$ for all $1 \leq i \leq n$ and $1 \leq j \leq m_i$,
– $\sigma(x_i) = p_i \overset{l_{ik}}{\not\rightarrow}$ for all $1 \leq i \leq n$ and $1 \leq k \leq n_i$, and
– $\sigma(t) = p'$.
Let

$$X \doteq \{x_i \mid 1 \leq i \leq n\} \text{ and}$$
$$Y \doteq \{y_{ij} \mid 1 \leq i \leq n, 1 \leq j \leq m_i\}.$$

Our aim is to use the above rule to show that $q = f(q_1, \dots, q_n) \xrightarrow{l} q'$ for some q' such that $(p', q') \in R$. To this end, we will now define a new substitution σ' in such a way that
– σ' 'satisfies the premises of rule d',
– $\sigma'(f(\overrightarrow{x})) = q$ and
– $(\sigma(x), \sigma'(x)) \in R$, for each $x \in V$.
We start with the following partial definition:

$$\sigma'(x) \doteq \begin{cases} q_i & \text{if } x = x_i \ (1 \leq i \leq n) \\ \sigma(x) & \text{if } x \in V \setminus (X \cup Y). \end{cases}$$

Hitherto, we already have that $\sigma'(f(\overrightarrow{x})) = q$ and that $(\sigma(x), \sigma'(x)) \in R$ for each $x \in V \setminus Y$. Moreover, σ' satisfies the negative premises of d. Indeed, as $(p_i, q_i) \in$

R by assumption, the induction hypothesis yields that p_i and q_i have the same set of initial actions. Therefore, since p_i satisfies the negative premises of d, so does q_i.
It remains to complete the definition of σ' for the variables in Y in such a way that σ' satisfies the positive premises of rule d and $(\sigma(x), \sigma'(x)) \in R$, for each $x \in Y$. To this end, consider a premise of d of the form $x_i \xrightarrow{l_{ij}} y_{ij}$, for some $1 \leq i \leq n$ and $1 \leq j \leq m_i$. Since $\sigma(x_i) = p_i$, $(p_i, q_i) \in R$ and $\sigma(x_i) \xrightarrow{l_{ij}} \sigma(y_{ij})$, it follows from the induction hypothesis that there exists some p'_{ij} such that $q_i \xrightarrow{l} p'_{ij}$ and $(\sigma(y_{ij}, p'_{ij}) \in R$. We let $\sigma'(y_{ij}) \doteq p'_{ij}$. Defining σ' for each $y_{ij} \in Y$ inductively, in this manner, concludes the proof since rule d instantiated with σ' gives us that $q \xrightarrow{l} \sigma'(t)$ and, by Lemma 2, $(\sigma(t), \sigma'(t)) \in R$. (Recall that R is just $\sim_{cc}$ restricted to closed terms.)

3. We are left to examine the case where

$$p = f(p_1, \ldots, p_j, \ldots, p_k, \ldots, p_n) \text{ and}$$
$$q = f(p_1, \ldots, p_k, \ldots, p_j, \ldots, p_n),$$

for some $f \in \Sigma$ and $j, k \in K \in \prod_f$, with $j < k$. Since $p \xrightarrow{l} p'$, there are a rule

$$(\text{d})\ \frac{\{x_i \xrightarrow{l_{i\ell}} y_{i\ell} \mid 1 \leq i \leq n, 1 \leq \ell \leq m_i\} \cup \{x_i \overset{l_{i\ell'}}{\nrightarrow} \mid 1 \leq i \leq n, 1 \leq \ell' \leq n_i\}}{f(\overrightarrow{x}) \xrightarrow{l} t}$$

and a substitution σ such that
- $\sigma(x_i) = p_i \xrightarrow{l_{i\ell}} \sigma(y_{i\ell})$ for all $1 \leq i \leq n$ and $1 \leq \ell \leq m_i$,
- $\sigma(x_i) = p_i \overset{l_{i\ell'}}{\nrightarrow}$ for all $1 \leq i \leq n$ and $1 \leq \ell' \leq n_i$, and
- $\sigma(t) = p'$.

As before, our aim is to show that $q = f(p_1, \ldots, p_k, \ldots, p_j, \ldots, p_n) \xrightarrow{l} q'$ for some q' such that $(p', q') \in R$. In what follows, for the sake of brevity, we refer to the set of premises of rule d as H. According to Definition 12, the transition system specification contains another rule of the following form:

$$(\text{d'})\ \frac{\{x'_i \xrightarrow{l'_{i\ell}} y'_{i\ell} \mid 1 \leq i \leq n, 1 \leq \ell \leq m'_i\} \cup \{x'_i \overset{l'_{i\ell'}}{\nrightarrow} \mid 1 \leq i \leq n, 1 \leq \ell' \leq n'_i\}}{f(x'_1, \ldots, x'_n) \xrightarrow{l} t'}.$$

Moreover, there is a bijective mapping $\hbar$ over the set of variables such that
- $\hbar(x'_i) = x_i$ for $1 \leq i \leq n$ such that $i \neq j$ and $i \neq k$,
- $\hbar(x'_j) = x_k$ and $\hbar(x'_k) = x_j$,
- $\hbar(t') \sim_{cc} t$ and
- $\hbar(H') = H$, where H' stands for the collection of premises of d'.

Let $\sigma' = \sigma \circ \hbar$. It is not hard to see that σ' satisfies the premises of d'. Therefore,

$$\sigma'(f(x'_1, \ldots, x'_n)) = f(p_1, \ldots, p_k, \ldots, p_j, \ldots, p_n) = q \xrightarrow{l} \sigma'(t').$$

Since $\hbar(t') \sim_{cc} t$, by Lemma 2, we have that

$$\sigma'(t') = \sigma(\hbar(t')) \sim_{cc} \sigma(t).$$

Hence $(\sigma(t), \sigma'(t')) \in R$ and we are done.

4. Assume that $(p, q) \in R$ because, by shorter inferences, $(p, r) \in R$ and $(r, q) \in R$. By Lemma 1, r is closed. Suppose that $p \xrightarrow{l} p'$ for some closed term p'. By the inductive hypothesis, there is some closed term r' such that $r \xrightarrow{l} r'$ and $(p', r') \in R$. Using again the induction hypothesis, we have that $q \xrightarrow{l} q'$ and $(r', q') \in R$, for some q'. Since R is transitive, $(p', q') \in R$ and we are done.

This completes the proof of the theorem.

B Proof of Proposition 3

Consider an arbitrary disjoint extension of $\mathcal{T}'$. Assume that $f(p_1, \ldots, p_n) \xrightarrow{l} p$, for some closed terms $p_1, \ldots, p_n, p$. This is because there are a rule $d = \dfrac{H}{f(x_1, \ldots, x_n) \xrightarrow{l} t}$ and a closed substitution σ such that

- σ satisfies H,
- $\sigma(x_i) = p_i$, for each $i \in [n]$, and
- $\sigma(t) = p$.

Since the TSS under consideration is a disjoint extension of $\mathcal{T}'$, and thus of $\mathcal{T}$, there is some set of rules ρ_j, $j \in [\ell]$, and some rule $d' \in \rho_j$ such that $d \overset{K}{\smile} d'$. Letting, without loss of generality, $d' = \dfrac{H'}{f(x_1, \ldots, x_n) \xrightarrow{l} t'}$, this means that there are some K-permutation π and some bijection $\hbar$ over variables such that

- $\hbar(x_i) = x_{\pi(i)}$, for each $i \in [n]$,
- $\hbar(t') \sim_{cc} t$, and
- $\hbar(H') = H$.

Consider now the closed substitution $\sigma \circ \hbar$. This substitution satisfies H' because σ satisfies H and $\hbar(H') = H$. Moreover,

$$(\sigma \circ \hbar)(f(x_1, \ldots, x_n)) = \sigma(f(x_{\pi(1)}, \ldots, x_{\pi(n)})$$

and $(\sigma \circ \hbar)(t') \sim_{cc} \sigma(t) = p$, by Lemma 2. Furthermore, from the proof of Theorem 1 in Appendix A, we have that $\sim_{cc}$ is a bisimulation, and therefore $(\sigma \circ \hbar)(t') \leftrightarrow p$. Since there is a rule for f_j of the form $\dfrac{H'}{f_j(x_1, \ldots, x_n) \xrightarrow{l} t'}$, we have that

$$\sigma(f_j(x_{\pi(1)}, \ldots, x_{\pi(n)})) \xrightarrow{l} (\sigma \circ \hbar)(t') \leftrightarrow p.$$

Therefore it holds that

$$\sigma(\sum_{i=1}^{\ell}\sum\{f_i(x_{\pi(1)},\ldots,x_{\pi(n)}) \mid \pi \text{ is a } K\text{-permutation}\}) \xrightarrow{l} (\sigma \circ \hbar)(t') \underline{\leftrightarrow}\, p,$$

and we have found a matching transition, up to bisimilarity, from the instantiation of the right-hand side of equation (1) with σ.

We now check that each transition

$$t = \sum_{i=1}^{\ell}\sum\{f_i(p_{\pi(1)},\ldots,p_{\pi(n)}) \mid \pi \text{ a } K\text{-permutation}\} \xrightarrow{l} p$$

can be matched by $f(p_1,\ldots,p_n)$ up to bisimilarity. To this end, assume that, for some p, $t \xrightarrow{l} p$ This means that there are some $j \in [\ell]$ and some K-permutation π such that $f_j(p_{\pi(1)},\ldots,p_{\pi(n)}) \xrightarrow{l} p$. Since each rule for f_j is a rule for f, we have that $f(p_{\pi(1)},\ldots,p_{\pi(n)}) \xrightarrow{l} p$ also holds. As π is a K-permutation, $K \in \prod_f$ and $\mathcal{T}$ is in the comm-GSOS format with respect to $\prod^{\Sigma}$, by repeated use of Theorem 1, we have that $f(p_{\pi(1)},\ldots,p_{\pi(n)}) \underline{\leftrightarrow}\, f(p_1,\ldots,p_n)$. Therefore there is some p' such that $f(p_1,\ldots,p_n) \xrightarrow{l} p'$ and $p' \underline{\leftrightarrow}\, p$, which was to be shown.