

# SOS Rule Formats for Zero and Unit Elements<sup>☆</sup>

Luca Aceto<sup>a,\*</sup>, Matteo Cimini<sup>a</sup>, Anna Ingolfsdottir<sup>a</sup>, MohammadReza Mousavi<sup>b</sup>, Michel A. Reniers<sup>b,c</sup>

<sup>a</sup>*ICE-TCS, School of Computer Science, Reykjavik University, Menntavegur 1, IS 101 Reykjavik, Iceland*

<sup>b</sup>*Department of Computer Science, Eindhoven University of Technology, P.O. Box 513, NL-5600 MB Eindhoven, The Netherlands*

<sup>c</sup>*Department of Mechanical Engineering, Eindhoven University of Technology, P.O. Box 513, NL-5600 MB Eindhoven, The Netherlands*

---

## Abstract

This paper proposes rule formats for Structural Operational Semantics guaranteeing that certain constants act as left or right unit/zero elements for a set of binary operators. Examples of left and right zero, as well as unit, elements from the literature are shown to fit the rule formats offered in this study.

*Keywords:* Structural operational semantics (SOS), rule formats, bisimilarity, unit elements, zero elements

---

## 1. Introduction

Over the last three decades, Structural Operational Semantics (SOS), see, e.g., [4, 23, 25, 27], has proven to be a powerful way to specify the semantics of programming and specification languages. In this approach to semantics, languages can be given a clear behaviour in terms of states and transitions, where the collection of transitions is specified by means of a set of syntax-driven inference rules. Based on this semantics in terms of state transitions, we often want to prove the validity of general algebraic laws about the languages, which describe semantic properties of the various operators they involve modulo the notion of behavioural equivalence or preorder of interest. For example, the reader may think about the field of process algebra [7, 14, 19, 21], where one often wants to check whether certain operators are, say, commutative and associative with respect to bisimilarity [24].

---

<sup>☆</sup>The work of Aceto, Cimini and Ingolfsdottir has been partially supported by the projects ‘New Developments in Operational Semantics’ (nr. 080039021) and ‘Meta-theory of Algebraic Process Theories’ (nr. 100014021) of the Icelandic Research Fund.

\*Corresponding author

*Email addresses:* [luca@ru.is](mailto:luca@ru.is) (Luca Aceto), [matteo@ru.is](mailto:matteo@ru.is) (Matteo Cimini), [annai@ru.is](mailto:annai@ru.is) (Anna Ingolfsdottir), [m.r.mousavi@tue.nl](mailto:m.r.mousavi@tue.nl) (MohammadReza Mousavi), [m.a.reniers@tue.nl](mailto:m.a.reniers@tue.nl) (Michel A. Reniers)

This paper aims at contributing to an ongoing line of research whose goal is to ensure the validity of algebraic properties *by design*, using the so called *SOS rule formats* [5]. Results in this research area roughly state that if the specification of (parts of) the operational semantics of a language has a certain form then some semantic property is guaranteed to hold. The literature on SOS provides rule formats for basic algebraic properties of operators such as commutativity [22], associativity [17] and idempotence [1]. The main advantage of this approach is that one is able to verify the desired property by syntactic checks that can be mechanized. Moreover, it is interesting to use rule formats for establishing semantic properties since the results so obtained apply to a broad class of languages. Apart from providing one with an insight as to the semantic nature of algebraic properties and its link to the syntax of SOS rules, rule formats like those presented in the above-mentioned references may serve as a guideline for language designers who want to ensure, a priori, that the constructs under design enjoy certain basic algebraic properties.

In the present paper, we develop some rule formats guaranteeing that certain constants act as left or right unit/zero elements for a set of binary operators. A binary operator  $f$  has a left (respectively, right) unit element  $c$ , modulo some notion of behavioural equivalence, whenever the equation  $f(c, x) = x$  (respectively,  $f(x, c) = x$ ) holds. On the other hand,  $f$  has a left (respectively, right) zero element  $c$ , modulo some notion of behavioural equivalence, whenever the equation  $f(c, x) = c$  (respectively,  $f(x, c) = c$ ) holds.

A classic example of a left zero element within the realm of process algebra is provided by the constant  $\delta$ , for deadlock, from PA [13, 14], which satisfies the laws:

$$\delta \cdot x = \delta \quad \text{and} \quad \delta \parallel x = \delta \quad ,$$

where ‘ $\cdot$ ’ and ‘ $\parallel$ ’ stand for sequential composition and left merge, respectively. A standard example of a left and right unit element is instead given by the constant  $\mathbf{0}$  from Milner’s CCS [21], which satisfies the laws:

$$\mathbf{0} + x = x \quad \text{and} \quad x + \mathbf{0} = x \quad ,$$

where ‘ $+$ ’ stands for the nondeterministic choice operator.

The first two formats we provide for left or right unit/zero elements are mostly of a syntactic nature. However, even though we show how several classic examples from the literature indeed fit the formats, there are some basic, but somewhat more exotic, examples that cannot be handled by the proposed formats.

We show nevertheless that, in order to overcome some of their possible weaknesses, we can reformulate our formats within the GSOS languages of Bloom, Istrail and Meyer [16] by using a modest amount of ‘semantic reasoning’. In particular, we benefit from the logic of transition formulae developed by some of the authors in [2], which is tailored for reasoning about the satisfiability of premises of GSOS rules.

Mechanizing the rule formats in a tool-set is a long-term goal of research on SOS rule formats. We believe that the GSOS-based rule formats we present

in this paper are strong candidates for mechanization insofar as zero and unit elements are concerned.

*Roadmap of the paper.* Section 2 repeats some standard definitions from the theory of SOS. Section 3 provides the first format for left and right unit elements, and Section 4 shows how several examples of left and right unit elements from the literature fit the format and extends it to a setting with predicates. Section 5 is devoted to our first format for left and right zero elements, whose applicability is highlighted by the examples discussed in Section 6. In Section 7 we point out the main drawbacks of the syntactic formats by focussing on the one for zero elements. In Section 8 we reformulate the format for zero elements within the GSOS format using the aforementioned logic of transition formulae. In Section 9 we provide a rule format for unit elements adapting the ideas from Section 8. We conclude the paper with an overview of its main contributions and pointers for future research in Section 10.

In order to increase the readability of the main body of the paper, the proofs of some of the main technical results have been collected in the appendices.

This paper collects under one roof results that were announced, without proof, in the conference articles [3, 6]. Apart from including proofs for all the main results, this study also discusses in substantial detail the design decisions underlying our rule formats, and offers a wealth of examples of applications of the proposed formats that were not included in the conference articles for lack of space.

## 2. Preliminaries

In this section we recall some standard definitions from the theory of SOS. We refer the readers to, e.g., [4] and [23] for more information.

### 2.1. Transition system specifications and bisimilarity

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

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

We use  $s, t, u$ , possibly subscripted and/or superscripted, to range over terms. We write  $t_1 \equiv t_2$  if  $t_1$  and  $t_2$  are syntactically equal. The function  $\text{vars} : \mathbb{T}(\Sigma) \rightarrow 2^V$  gives the set of variables appearing in a term. The set  $\mathbb{C}(\Sigma) \subseteq \mathbb{T}(\Sigma)$  is the set of *closed terms*, i.e., terms that contain no variables. We use  $p, q, p', p_i, \dots$  to range over closed terms. A *substitution*  $\sigma$  is a function of type  $V \rightarrow \mathbb{T}(\Sigma)$ . We extend the domain of substitutions to terms homomorphically and write

$\sigma(t)$  for the result of applying the substitution  $\sigma$  to the term  $t$ . If the range of a substitution is included in  $\mathbb{C}(\Sigma)$ , we say that it is a *closed substitution*.

**Definition 2 (Transition system specification).** A *transition system specification* (TSS) is a triple  $(\Sigma, \mathcal{L}, D)$  where

- $\Sigma$  is a signature.
- $\mathcal{L}$  is a set of *labels* (or *actions*) ranged over by  $a, b, l$ . If  $l \in \mathcal{L}$  and  $t, t' \in \mathbb{T}(\Sigma)$ , we say that  $t \xrightarrow{l} t'$  is a *positive transition formula* and  $t \xrightarrow{l} \bar{\phantom{t}}$  is a *negative transition formula*. A transition formula (or just formula), typically denoted by  $\phi$  or  $\psi$ , is either a negative transition formula or a positive one.
- $D$  is a set of *deduction rules*, i.e., tuples of the form  $(\Phi, \phi)$  where  $\Phi$  is a set of formulae and  $\phi$  is a positive formula. We call the formulae contained in  $\Phi$  the *premises* of the rule and  $\phi$  the *conclusion*.

We write  $\text{vars}(r)$  to denote the set of variables appearing in a deduction rule  $r$ . We say that a formula or a deduction rule is *closed* if all of its terms are closed. Substitutions are also extended to formulae and sets of formulae in the natural way. For a rule  $r$  and a substitution  $\sigma$ , the rule  $\sigma(r)$  is called a substitution instance of  $r$ . A set of positive closed formulae is called a *transition relation*.

We often refer to a positive transition formula  $t \xrightarrow{l} t'$  as a *transition* with  $t$  being its *source*,  $l$  its *label*, and  $t'$  its *target*. A deduction rule  $(\Phi, \phi)$  is typically written as  $\frac{\Phi}{\phi}$ . For the sake of consistency with SOS specifications of specific operators in the literature, in examples we use  $\frac{\phi_1 \dots \phi_n}{\phi}$  in lieu of  $\frac{\{\phi_1, \dots, \phi_n\}}{\phi}$ .

An *axiom* is a deduction rule with an empty set of premises. We write  $\frac{}{\phi}$  for an axiom with  $\phi$  as its conclusion, and often abbreviate this notation to  $\phi$  when this causes no confusion. We call a deduction rule *f-defining* when the outermost function symbol appearing in the source of its conclusion is  $f$ .

*In this paper, for each constant  $c$ , we assume that each  $c$ -defining deduction rule is an axiom of the form  $\frac{}{c \xrightarrow{l} p}$  for some label  $l$  and closed term  $p$ .* This is not a real restriction since all practical cases we know of do actually satisfy this property. For GSOS languages, which are defined shortly and used in later sections of this paper, this restriction is automatically satisfied.

The meaning of a TSS is defined by the following notion of least three-valued stable model. To define this notion, we need two auxiliary definitions, namely provable transition rules and contradiction, which are given below.

**Definition 3 (Provable transition rules).** A closed deduction rule is called a *transition rule* when it is of the form  $\frac{N}{\phi}$  with  $N$  a set of *negative formulae*. A TSS  $\mathcal{T}$  *proves*  $\frac{N}{\phi}$ , denoted by  $\mathcal{T} \vdash \frac{N}{\phi}$ , when there is a well-founded upwardly branching tree with closed formulae as nodes and of which

- the root is labelled by  $\phi$ ;

- if a node is labelled by  $\psi$  and the labels of the nodes directly above it form the set  $K$  then:
  - $\psi$  is a negative formula and  $\psi \in N$ , or
  - $\psi$  is a positive formula and  $\frac{K}{\psi}$  is a substitution instance of a deduction rule in  $\mathcal{T}$ .

We often write  $\mathcal{T} \vdash \phi$  in lieu of  $\mathcal{T} \vdash \frac{\emptyset}{\phi}$ .

**Definition 4 (Contradiction and entailment).** The formula  $t \xrightarrow{l} t'$  is said to *contradict*  $t \xrightarrow{l}$ , and vice versa. For two sets  $\Phi$  and  $\Psi$  of formulae,  $\Phi$  *contradicts*  $\Psi$  when there is a  $\phi \in \Phi$  that contradicts a  $\psi \in \Psi$ . We write  $\Phi \vDash \Psi$ , read ‘ $\Phi$  is consistent with  $\Psi$ ’, when  $\Phi$  does not contradict  $\Psi$ .

A formula  $\phi$  *entails*  $\psi$  when there is a substitution  $\sigma$  such that  $\sigma(\phi) \equiv \psi$ . A set of formulae  $\Phi$  entails a formula  $\psi$  when there is some  $\phi \in \Phi$  that entails  $\psi$ .

It immediately follows from the above definition that contradiction and consistency are symmetric relations on (sets of) formulae. We now have all the necessary ingredients to define the semantics of TSSs in terms of three-valued stable models [28].

**Definition 5 (Three-valued stable model).** A pair  $(C, U)$  of disjoint sets of positive closed transition formulae is called a *three-valued stable model* for a TSS  $\mathcal{T}$  when the following conditions hold:

- for each  $\phi \in C$ , there is a set  $N$  of negative formulae such that  $\mathcal{T} \vdash \frac{N}{\phi}$  and  $C \cup U \vDash N$ , and
- for each  $\phi \in U$ , there is a set  $N$  of negative formulae such that  $\mathcal{T} \vdash \frac{N}{\phi}$  and  $C \vDash N$ .

$C$  stands for *Certainly* and  $U$  for *Unknown*; the third value is determined by the formulae not in  $C \cup U$ . The *least* three-valued stable model is a three-valued stable model that is the least one with respect to the (information-theoretic) ordering on pairs of sets of formulae defined as  $(C, U) \leq (C', U')$  iff  $C \subseteq C'$  and  $U' \subseteq U$ . We say that  $\mathcal{T}$  is *complete* when for its least three-valued stable model it holds that  $U = \emptyset$ . In a complete TSS, we say that a closed substitution  $\sigma$  satisfies a set of formulae  $\Phi$  if  $\sigma(\phi) \in C$ , for each positive formula  $\phi \in \Phi$ , and  $C \vDash \{\sigma(\phi)\}$ , for each negative formula  $\phi \in \Phi$ . If a TSS is complete, we often also write  $p \xrightarrow{l} p'$  in lieu of  $(p \xrightarrow{l} p') \in C$ .

**Definition 6 (Bisimulation and bisimilarity [21, 24]).** Let  $\mathcal{T}$  be a transition system specification with signature  $\Sigma$  and label set  $\mathcal{L}$ . A relation  $\mathcal{R} \subseteq \mathbb{C}(\Sigma) \times \mathbb{C}(\Sigma)$  is a *bisimulation relation* if and only if  $\mathcal{R}$  is symmetric and, for all  $p_0, p_1, p'_0 \in \mathbb{C}(\Sigma)$  and  $l \in \mathcal{L}$ ,

$$(p_0 \mathcal{R} p_1 \wedge \mathcal{T} \vdash p_0 \xrightarrow{l} p'_0) \Rightarrow \exists p'_1 \in \mathbb{C}(\Sigma). (\mathcal{T} \vdash p_1 \xrightarrow{l} p'_1 \wedge p'_0 \mathcal{R} p'_1).$$

Two terms  $p_0, p_1 \in \mathbb{C}(\Sigma)$  are called *bisimilar*, denoted by  $p_0 \Leftrightarrow p_1$ , when there exists a bisimulation relation  $\mathcal{R}$  such that  $p_0 \mathcal{R} p_1$ .

Bisimilarity is extended to open terms by requiring that  $s, t \in \mathbb{T}(\Sigma)$  are bisimilar when  $\sigma(s) \Leftrightarrow \sigma(t)$  for each closed substitution  $\sigma : V \rightarrow \mathbb{C}(\Sigma)$ .

In Sections 8–9 of the paper, we focus on the GSOS format of Bloom, Istrail and Meyer [16], whose definition is given below.

**Definition 7 (GSOS rule).** Suppose  $\Sigma$  is a signature. A *GSOS rule*  $r$  over  $\Sigma$  is a rule of the form:

$$\frac{\bigcup_{i=1}^{\ell} \{x_i \xrightarrow{a_{ij}} y_{ij} \mid 1 \leq j \leq m_i\} \cup \bigcup_{i=1}^{\ell} \{x_i \xrightarrow{b_{ik}} \mid 1 \leq k \leq n_i\}}{f(x_1, \dots, x_{\ell}) \xrightarrow{c} t} \quad (1)$$

where all the variables are distinct,  $m_i, n_i \geq 0$ ,  $a_{ij}, b_{ik}$ , and  $c$  are actions from a finite set,  $f$  is a function symbol from  $\Sigma$  with arity  $\ell$ , and  $t$  is a term in  $\mathbb{T}(\Sigma)$  such that  $\text{vars}(t) \subseteq \{x_1, \dots, x_{\ell}\} \cup \{y_{ij} \mid 1 \leq i \leq \ell, 1 \leq j \leq m_i\}$ .

**Definition 8.** A *GSOS language* is a triple  $G = (\Sigma_G, \mathcal{L}, R_G)$ , where  $\Sigma_G$  is a finite signature,  $\mathcal{L}$  is a finite set of action labels and  $R_G$  is a finite set of GSOS rules over  $\Sigma_G$ .

## 2.2. Predicates

Several of the examples of (left and right) zero and unit elements we will discuss in the remainder of this paper involve operators whose SOS semantics is best given using predicates as well as transition relations. For the sake of completeness, we therefore now introduce the notion of TSS extended with predicates.

**Definition 9 (Predicates).** Given a set  $\mathcal{P}$  of predicate symbols,  $Pt$  is a *positive predicate formula* and  $\neg Pt$  is a *negative predicate formula*, for each  $P \in \mathcal{P}$  and  $t \in \mathbb{T}(\Sigma)$ . We call  $t$  the *source* of both predicate formulae. In the extended setting, a (positive, negative) *formula* is either a (positive, negative) transition formula or a (positive, negative) predicate formula. The notions of deduction rule, TSS, provable transition rules and three-valued stable models are then naturally extended by adopting the more general notion of formulae. The label of a deduction rule is either the label of the transition formula or the predicate symbol of the predicate formula in its conclusion.

The definition of bisimulation is extended to a setting with predicates in the standard fashion. In particular, bisimilar terms must satisfy the same predicates.

### 3. A rule format for unit elements

We now proceed to define our rule format guaranteeing that certain constants in the language under consideration are left or right units for some binary operators. In the definition of the format proposed in the remainder of this section, we make use of a syntactic characterization of equivalence of terms up to their composition with unit elements; we call such terms *unit-context equivalent*. Intuitively, if  $s$  is unit-context equivalent to  $t$ , then  $s$  and  $t$  are bisimilar because one can be obtained from the other by applying axioms stating that some constant is a left or right unit for some binary operator. For instance, if  $c_1$  is a left unit for a binary operator  $f$  and  $c_2$  is a right unit for a binary operator  $g$ , then the terms  $f(c_1, g(t, c_2))$  and  $g(f(c_1, t), c_2)$  are both unit-context equivalent to  $t$  and also unit-context equivalent to each other.

The following definition formalizes this intuition. (While reading the technical definition, our readers may find it useful to bear in mind that  $(f, c) \in L$  means that  $c$  is a left unit for a binary operator  $f$  and  $(f, c) \in R$  means that  $c$  is a right unit for  $f$ .)

**Definition 10 (Unit-context equivalence).** Given sets  $L, R \subseteq \Sigma \times \Sigma$  of pairs of binary function symbols and constants,  $\cong^{L,R}$  is the smallest equivalence relation over  $\mathbb{T}(\Sigma)$  satisfying the following conditions, for each  $s \in \mathbb{T}(\Sigma)$ :

1.  $\forall (f, c) \in L. s \cong^{L,R} f(c, s)$ , and
2.  $\forall (g, c) \in R. s \cong^{L,R} g(s, c)$ .

We say that two terms  $s, t \in \mathbb{T}(\Sigma)$  are *unit-context equivalent*, if  $s \cong^{L,R} t$ .

In what follows, we abbreviate  $\cong^{L,R}$  to  $\cong$  since the sets  $L$  and  $R$  are always clear from the context.

**Lemma 1.** *For all  $s, t \in \mathbb{T}(\Sigma)$ , if  $s \cong t$  then  $\text{vars}(s) = \text{vars}(t)$  and  $\sigma(s) \cong \sigma(t)$ , for each substitution  $\sigma$ .*

Since unit-context equivalence will be used in the definition of our rule formats for unit elements, a basic sanity property that one might expect this notion to have is that it be decidable. This is the import of the following theorem.

**Theorem 1 (Decidability of unit-context equivalence).** *Let  $L, R \subseteq \Sigma \times \Sigma$  be finite sets of pairs of binary function symbols and constants. Then, for all terms  $t, u \in \mathbb{T}(\Sigma)$ , it is decidable whether  $t \cong^{L,R} u$  holds.*

**PROOF.** Let  $L$  and  $R$  be given finite sets of pairs of binary operators and constants. Suppose that we are given two terms  $t$  and  $u$ , and we want to check whether they are unit-context equivalent. From  $t$  and  $u$ , construct the (undirected) graph  $G(t, u)$  as follows.

The nodes in  $G(t, u)$  are

- $t$ ,  $u$  and all their subterms,
- all constants mentioned in  $L$  or  $R$ , and
- all terms of the form  $f(c, d)$  with  $(f, c) \in L$  and  $(f, d) \in R$ .

The edges in  $G(t, u)$  are given by items 1 and 2 in Definition 10. This graph is finite, since  $L$  and  $R$  are finite, and can be built effectively. Note that  $G(u, t)$  and  $G(t, u)$  are identical.

We claim that  $t$  is unit-context equivalent to  $u$  iff  $t$  can be reached from  $u$  in  $G(t, u)$ .

The proof of this claim is as follows. The right-to-left implication is immediate since each edge in  $G(t, u)$  corresponds to an application of item 1 or item 2 in Definition 10. For the converse, we proceed by induction on the length of a shortest proof of  $t \cong u$ . If  $t \cong u$  follows by reflexivity or by using item 1 or 2 in Definition 10 then  $t$  can be reached from  $u$  in  $G(t, u)$  in zero steps or in one step, respectively. If  $t \cong u$  follows by symmetry then the claim follows by the inductive hypothesis. Assume now that  $t \cong u$  follows by transitivity. Then there is some term  $s$  such that  $t \cong s$  (in one step) and  $s \cong u$ . By induction and the symmetry of reachability,  $s$  is reachable from  $t$  in  $G(t, s)$  and  $s$  is reachable from  $u$  in  $G(s, u)$ . To see that  $u$  is reachable from  $t$  in  $G(t, u)$ , we now observe that  $s$  can be taken to be

- a subterm of  $t$ , if  $t = f(c, s)$  for some  $(f, c) \in L$  or  $t = f(s, c)$  for some  $(f, c) \in R$ , or
- if  $t$  is a constant  $c$ , a term of one of the following forms for some constant  $d$ :
  - $f(c, d)$ , where  $(f, c) \in L$  and  $(f, d) \in R$ , or
  - $f(d, c)$ , where  $(f, c) \in R$  and  $(f, d) \in L$ .

In the former case,  $G(t, s)$  and  $G(s, u)$  are subgraphs of  $G(t, u)$ , and therefore  $t$  is reachable from  $u$  in  $G(t, u)$  as claimed.

In the latter case, let, without loss of generality,

$$t = c \cong t_1 = f(d, c) \cong t_2 \cdots t_{n-1} \cong t_n = u \quad (n \geq 2)$$

be a shortest proof of  $t \cong u$ , where  $(f, d) \in L$  and each of the intermediate equivalences is an instance of items 1 and 2 in Definition 10 or of their symmetric counterparts. Since the above is a shortest proof of  $t \cong u$ , we have that  $t_2$  can be:

1.  $d$ , if  $(f, c) \in R$ ,
2.  $g(f(d, c), e)$ , for some  $(g, e) \in R$ , or
3.  $g(e, f(d, c))$ , for some  $(g, e) \in L$ .

If  $t_2 = d$  and  $(f, c) \in R$ , then  $G(d, u)$  is a subgraph of  $G(t, u)$  and  $d$  is reachable from  $c = t$  in  $G(t, u)$ . In both the other cases, since the above is a shortest proof of  $t \cong u$ , we have that  $t_2$  must be a subterm of  $u$ . Therefore,  $G(t_2, u)$  is a subgraph of  $G(t, u)$ . Since  $t_1 = f(d, c)$  and  $c = t$  are also subterms of  $u$ , in all cases we have that  $t$  is reachable from  $u$  in  $G(t, u)$ .

It follows that both  $G(t, s)$  and  $G(s, u)$  are subgraphs of  $G(t, u)$ , and therefore  $t$  is reachable from  $u$  in  $G(t, u)$ , as claimed.  $\square$

We are now ready to define our promised rule format for unit elements.

**Definition 11 (Left- and right-aligned pairs).** Given a TSS, the sets  $L$  and  $R$  of pairs of binary function symbols and constants are the largest sets satisfying the following conditions.

1. For each  $(f, c) \in L$ , the following conditions hold:
  - (a) For each action  $a \in \mathcal{L}$ , there exists at least one deduction rule of the following form:

$$\frac{\{x_0 \xrightarrow{a_i} y_i \mid i \in I\} \cup \{x_0 \xrightarrow{a_j} \mid j \in J\} \cup \{x_1 \xrightarrow{a} z_1\}}{f(x_0, x_1) \xrightarrow{a} t'}$$

where

- i. the variables  $y_i, z_1, x_0$  and  $x_1$  are all pairwise distinct,
  - ii. for each  $j \in J$ , there is no  $c$ -defining axiom with  $a_j$  as label, and
  - iii. there exists a collection  $\{c \xrightarrow{a_i} q_i \mid i \in I\}$  of  $c$ -defining axioms such that  $\sigma(t') \cong z_1$ , where  $\sigma$  is the substitution mapping  $x_0$  to  $c$ , each  $y_i$  to  $q_i$ ,  $i \in I$ , and is the identity on all the other variables.
- (b) Each  $f$ -defining deduction rule has the following form:

$$\frac{\Phi}{f(t_0, t_1) \xrightarrow{a} t'}$$

where  $a \in \mathcal{L}$  and, for each closed substitution  $\sigma$  such that  $\sigma(t_0) \equiv c$ ,

- i. either there exists some  $t_1 \xrightarrow{a} t'' \in \Phi$  with  $\sigma(t') \cong \sigma(t'')$ , or
  - ii. there exists a premise  $\phi \in \Phi$  with  $t_0$  as its source such that
    - A. either  $\phi$  is a positive formula and the collection of conclusions of  $c$ -defining deduction rules does not entail  $\sigma(\phi)$ , or
    - B.  $\phi$  is a negative formula and the collection of conclusions of  $c$ -defining axioms contradicts  $\sigma(\phi)$ .
2. The definition of right-aligned pairs of operators and constant symbols – that is, those such that  $(f, c) \in R$  – is symmetric and is not repeated here.

For a function symbol  $f$  and a constant  $c$ , we call  $(f, c)$  *left aligned* (respectively, *right aligned*) if  $(f, c) \in L$  (respectively,  $(f, c) \in R$ ).

Condition 1a in the above definition ensures that, whenever  $(f, c)$  is in  $L$ , each transition of the form  $p \xrightarrow{a} p'$ , for some closed terms  $p$  and  $p'$  and action  $a$ , can be used to infer a transition  $f(c, p) \xrightarrow{a} q'$  for some  $q'$  that is bisimilar to  $p'$ . This means that if  $(f, c)$  is in  $L$  then, in the context of the constant  $c$ , the operator  $f$  does not ‘prune away’ any of the behaviour of its second argument.

Condition 1(b)i, on the other hand, ensures that, whenever  $(f, c)$  is in  $L$ , each transition  $f(c, p) \xrightarrow{a} q'$  is due to a transition  $p \xrightarrow{a} p'$  for some  $p'$  that is bisimilar to  $q'$ . Thus, if  $(f, c)$  is in  $L$  then, in the context of the constant  $c$ , a term of the form  $f(c, p)$  can only mimic the behaviour of  $p$ . As will become clear from the examples to follow, condition 1(b)ii ensures that the  $f$ -defining rule cannot be used to derive a transition for  $f(c, p)$  and hence it is exempted from further conditions; the presence of this condition enhances the generality of our format and allows us to handle common examples of unit constants from the literature (see, e.g., Example 5 to follow).

**Remark 1.** Note that the requirement that  $\sigma(t') \cong z_1$  in condition 1a of the above definition implies that  $\text{vars}(\sigma(t')) = \{z_1\}$ . Therefore  $x_1, z_1$  and the  $y_i, i \in I$ , are the only variables that may possibly occur in  $t'$ .

We are now ready to state the correctness of the proposed rule format for left and right unit elements.

**Theorem 2.** *Let  $\mathcal{T}$  be a complete TSS in which each rule is  $f$ -defining for some function symbol  $f$ . Assume that  $L$  and  $R$  are the sets of left- and right-aligned function symbols according to Definition 11. For each  $(f, c) \in L$ , it holds that  $f(c, x) \Leftrightarrow x$ . Symmetrically, for each  $(f, c) \in R$ , it holds that  $f(x, c) \Leftrightarrow x$ .*

We omit the proof of the above result since it is a simplified version of that of Theorem 3 to follow.

Note that Theorem 2 trivially extends to any notion of behavioural equivalence weaker than bisimilarity.

### 3.1. Discussion of Definition 11

We shall now discuss some of the conditions in Definition 11 in a slightly more technical fashion. In what follows, we focus on the conditions that left-aligned pairs must meet.

First of all, note that relaxing the requirement that  $x_0 \neq x_1$  in condition 1a would jeopardize Theorem 2. To see this, consider the TSS with constants  $\mathbf{0}$  and  $a$ , and binary operator  $f$  with rules

$$\frac{}{a \xrightarrow{a} \mathbf{0}} \quad \frac{x \xrightarrow{a} y}{f(x, x) \xrightarrow{a} y}.$$

It is easy to check that  $L = \{(f, \mathbf{0})\}$  satisfies all the requirements in condition 1 apart from  $x_0 \neq x_1$ . However, the term  $f(\mathbf{0}, a)$  affords no transition unlike  $a$ . Therefore  $\mathbf{0}$  is not a left unit for  $f$ .

In general, the requirement that variables be all different in condition 1a is needed in all the congruence rule formats for bisimilarity – see, for instance, the survey papers [4, 23] for an overview of such formats.

The role played by requirements 1(a)ii and 1(a)iii in ensuring that, modulo bisimilarity,  $f(c, p)$  affords ‘the same transitions as  $p$ ’, for each  $p$ , is highlighted by the following two examples.

**Example 1.** Consider the TSS with constants  $\mathbf{0}$  and  $a$ , and a binary operator  $f$  with rules:

$$\frac{}{a \xrightarrow{a} \mathbf{0}} \quad \frac{x_0 \xrightarrow{a} \quad x_1 \xrightarrow{a} y_1}{f(x_0, x_1) \xrightarrow{a} y_1} .$$

It is easy to check that  $L = \{(f, a)\}$  satisfies all the requirements in condition 1 apart from 1(a)ii. However, the term  $f(a, a)$  affords no transition unlike  $a$ . Therefore  $a$  is not a left unit for  $f$ .

**Example 2.** Consider the TSS with constants  $\mathbf{0}$  and  $a$ , and a binary operator  $f$  with rules:

$$\frac{}{a \xrightarrow{a} \mathbf{0}} \quad \frac{x_0 \xrightarrow{a} y_0 \quad x_1 \xrightarrow{a} y_1}{f(x_0, x_1) \xrightarrow{a} y_1} .$$

It is easy to check that  $L = \{(f, \mathbf{0})\}$  satisfies all the requirements in condition 1 apart from 1(a)iii. However, the term  $f(\mathbf{0}, a)$  affords no transition unlike  $a$ . Therefore  $\mathbf{0}$  is not a left unit for  $f$ .

Condition 1b is there to ensure that, for each  $p$ , the term  $f(c, p)$  only affords transitions that can be mimicked by  $p$ . Although condition 1(b)i can be removed without jeopardizing Theorem 2, its presence certainly increases the applicability of the format. For instance, the second rule for nondeterministic choice in Example 5 to follow does not meet condition 1(b)ii.

As we remarked previously, the role of requirement 1(b)ii is also to enhance the generality of our format.

Removing condition 1b altogether would jeopardize Theorem 2. To see this, consider the TSS with constant  $\mathbf{0}$  and binary operator  $f$  with rules

$$\frac{x_1 \xrightarrow{a} y_1}{f(x_0, x_1) \xrightarrow{a} y_1} \quad \frac{}{f(x_0, x_1) \xrightarrow{a} \mathbf{0}} .$$

It is easy to check that  $L = \{(f, \mathbf{0})\}$  satisfies all the requirements in condition 1a. However, the term  $f(\mathbf{0}, \mathbf{0})$  affords an  $a$ -labelled transition unlike  $\mathbf{0}$ . Therefore  $\mathbf{0}$  is not a left unit for  $f$ .

Note that, since the sets  $L$  and  $R$  are defined as the *largest* sets of pairs satisfying the conditions from Definition 11, in order to show that  $(f, c)$  is a left-aligned pair, say, it suffices only to exhibit two sets  $L$  and  $R$  satisfying these conditions, such that  $(f, c)$  is contained in  $L$ .

The following two examples illustrate that it is in general advantageous to consider sets of left- and/or right-aligned operators instead of just a single one.

**Example 3.** Assume that  $a$  is the only action and consider the binary operators  $f_i$ ,  $i \geq 0$ , with rules

$$\frac{x_1 \xrightarrow{a} y_1}{f_i(x_0, x_1) \xrightarrow{a} f_{i+1}(x_0, y_1)} .$$

Let  $\mathbf{0}$  be a constant with no rules. Then each of the pairs  $(f_i, \mathbf{0})$  is left aligned because the sets  $L = \{(f_i, \mathbf{0}) \mid i \geq 0\}$  and  $R = \emptyset$  meet the conditions from Definition 11. In particular, note that  $f_{i+1}(x_0, y_1)[x_0 \mapsto \mathbf{0}] \equiv f_{i+1}(\mathbf{0}, y_1) \cong y_1$ , for each  $i \geq 0$ . Observe that, for each  $i \geq 0$ , the equations  $f_i(\mathbf{0}, x) = x$  hold modulo bisimilarity. This fact can be checked directly by showing that the symmetric closure of the relation  $\mathcal{R} = \{(f_i(\mathbf{0}, p), p) \mid p \text{ a closed term}\}$  is a bisimulation, and is also a consequence of Theorem 2, which states the correctness of the rule format we described in Definition 11.

**Example 4.** Consider the following TSS, which is defined for a signature with  $\mathbf{0}$  and  $a$  as constants and  $f$  and  $g$  as binary function symbols.

$$\frac{}{a \xrightarrow{a} \mathbf{0}} \quad \frac{y \xrightarrow{a} y'}{f(x, y) \xrightarrow{a} g(y', x)} \quad \frac{x \xrightarrow{a} x'}{g(x, y) \xrightarrow{a} f(y, x')}$$

The TSS fits our rule format with  $L = \{(f, \mathbf{0})\}$  and  $R = \{(g, \mathbf{0})\}$ . Note that it is essential for the above example to consider both  $L$  and  $R$  simultaneously.

#### 4. Applications and extensions of the format for unit elements

Apart from its correctness, the acid test for the usefulness of a rule format is that it be expressive enough to cover examples from the literature that afford the property they were designed to ensure. Our order of business in this section will be to offer examples of applications of the format for unit elements we introduced in Definition 11 and to show how the format can be extended to deal with operators whose semantic definition involves the use of predicates.

##### 4.1. Applications of the basic rule format

We start by presenting examples of applications of the format for unit elements we introduced in Definition 11.

**Example 5 (Nondeterministic choice).** Consider the nondeterministic choice operator from Milner's CCS [21] specified by the rules below, where  $a \in \mathcal{L}$ .

$$\frac{x \xrightarrow{a} x'}{x + y \xrightarrow{a} x'} \quad \frac{y \xrightarrow{a} y'}{x + y \xrightarrow{a} y'}$$

The sets  $R = L = \{(+, \mathbf{0})\}$  meet the conditions in Definition 11. Indeed, condition 1a and its symmetric version are trivially satisfied by the right-hand and the left-hand rule schemas, respectively. (Note that the substitution  $\sigma$

associated with the empty collection of axioms in condition 1(a)iii is the identity function over the set of variables.) To see that condition 1b is also met, let  $\sigma$  be a closed substitution such that  $\sigma(x) = \mathbf{0}$ . Observe that

- each instance of the right-hand rule schema meets condition 1(b)i, and
- each instance of the left-hand rule schema meets condition 1(b)iiA because the set of rules for  $\mathbf{0}$  is empty and therefore does not entail  $\sigma(x) = \mathbf{0} \xrightarrow{a} \sigma(x')$ .

The reasoning for condition 2 is symmetric. Therefore, Theorem 2 yields the soundness of the well known equations [18]:  $\mathbf{0} + x = x = x + \mathbf{0}$ .

**Example 6 (Synchronous parallel composition).** Assume, for the sake of simplicity, that  $a$  is the only action. Consider a constant  $\text{RUN}_a$  and the synchronous parallel composition from CSP [19]<sup>1</sup> specified by the rules

$$\frac{}{\text{RUN}_a \xrightarrow{a} \text{RUN}_a} \quad \frac{x \xrightarrow{a} x' \quad y \xrightarrow{a} y'}{x \parallel_{\mathcal{L}} y \xrightarrow{a} x' \parallel_{\mathcal{L}} y'}$$

Take  $L = R = \{(\parallel_{\mathcal{L}}, \text{RUN}_a)\}$ . These sets  $L$  and  $R$  meet the conditions in Definition 11. To see that condition 1a and its symmetric version are satisfied by the above rule for  $\parallel_{\mathcal{L}}$ , observe that the substitution  $\sigma$  associated with the singleton set containing the only axiom for  $\text{RUN}_a$  in condition 1(a)iii maps both the variables  $x$  and  $x'$  to  $\text{RUN}_a$  and is the identity function over the other variables. For such a  $\sigma$ , we have that  $\sigma(x' \parallel_{\mathcal{L}} y') = \text{RUN}_a \parallel_{\mathcal{L}} y' \cong y'$ .

To see that condition 1b is also met, let  $\sigma$  be a closed substitution mapping  $x$  to  $\text{RUN}_a$ , and assume that  $\text{RUN}_a \xrightarrow{a} \text{RUN}_a$  entails  $\text{RUN}_a \xrightarrow{a} \sigma(x')$ . It follows that  $\sigma(x') = \text{RUN}_a$ . Therefore,

$$\sigma(x' \parallel_{\mathcal{L}} y') = \text{RUN}_a \parallel_{\mathcal{L}} \sigma(y') \cong \sigma(y')$$

and condition 1(b)i is met. Theorem 2 thus yields the soundness of the well known equations  $\text{RUN}_a \parallel_{\mathcal{L}} x = x = x \parallel_{\mathcal{L}} \text{RUN}_a$ . These are just equation L3B from [19, page 69] and its symmetric counterpart.

**Example 7 (Left merge and interleaving parallel composition).** The operational semantics of the classic left merge and interleaving parallel composition operators [7, 13, 14, 21] is given by the rules below.

$$\frac{x \xrightarrow{a} x'}{x \parallel y \xrightarrow{a} x' \parallel y} \quad \frac{x \xrightarrow{a} x'}{x \parallel y \xrightarrow{a} x' \parallel y} \quad \frac{y \xrightarrow{a} y'}{x \parallel y \xrightarrow{a} x \parallel y'}$$

<sup>1</sup>In [19], Hoare uses the symbol  $\parallel$  to denote the synchronous parallel composition operator. Here we will use that symbol for asynchronous parallel composition.

Take  $L = \{(\parallel, \mathbf{0})\}$  and  $R = \{(\parallel, \mathbf{0}), (\ll, \mathbf{0})\}$ . It is easy to see that these sets  $L$  and  $R$  meet the condition in Definition 11. Therefore, Theorem 2 yields the well known equalities  $\mathbf{0} \parallel x = x$ ,  $x \parallel \mathbf{0} = x$ , and  $x \ll \mathbf{0} = x$ .

Note that the pair  $(\ll, \mathbf{0})$  cannot be added to  $L$  while preserving condition 1a in Definition 11. Indeed,  $\mathbf{0}$  is not a left unit for the left merge operator  $\ll$ .

**Example 8 (Disrupt).** Consider the following disrupt operator  $\blacktriangleright$  [9] with rules

$$\frac{x \xrightarrow{a} x'}{x \blacktriangleright y \xrightarrow{a} x' \blacktriangleright y} \quad \frac{y \xrightarrow{a} y'}{x \blacktriangleright y \xrightarrow{a} y'}$$

Note that the equation  $\mathbf{0} \blacktriangleright x = x$  holds modulo bisimilarity. We now argue that its soundness is a consequence of Theorem 2. Indeed, take  $L = \{(\blacktriangleright, \mathbf{0})\}$  and  $R = \emptyset$ . It is easy to see that these sets  $L$  and  $R$  meet the conditions in Definition 11. In particular, to see that condition 1b is met by the first rule, observe that the set of rules for  $\mathbf{0}$  is empty and therefore does not entail  $\mathbf{0} \xrightarrow{a} p$  for any closed term  $p$ . A symmetric reasoning shows that the valid equation  $x \blacktriangleright \mathbf{0} = x$  is also a consequence of Theorem 2.

**Example 9 (Timed nondeterministic choice).** Consider the following version of the nondeterministic choice operator in a timed setting from [8]. It is defined by means of the deduction rules from Example 5 and additionally the deduction rules

$$\frac{x \xrightarrow{1} x' \quad y \xrightarrow{1} y'}{x + y \xrightarrow{1} x' + y'} \quad \frac{x \xrightarrow{1} x' \quad y \xrightarrow{1} \dashv}{x + y \xrightarrow{1} x'} \quad \frac{x \xrightarrow{1} \dashv \quad y \xrightarrow{1} y'}{x + y \xrightarrow{1} y'}$$

where the action label 1 denotes the passage of one unit of time. The equations  $\mathbf{0} + x = x$  and  $x + \mathbf{0} = x$  hold modulo bisimilarity. This is a consequence of Theorem 2 by taking  $L = R = \{(+, \mathbf{0})\}$ . For label 1, condition 1a is met by the third deduction rule. The first two deduction rules satisfy condition 1(b)iiA, and the third deduction rule satisfies condition 1(b)i trivially.

#### 4.2. Extension of the rule format with predicates

In the literature concerning the theory of rule formats for SOS (especially, the work devoted to congruence formats for various notions of bisimilarity), most of the time predicates are neglected at first and are only added to the considerations at a later stage. The reason is that one can encode predicates quite easily by means of transition relations. One can find a number of such encodings in the literature – see, for instance, [17, 30]. In each of these encodings, a predicate  $P$  is represented as a transition relation  $\xrightarrow{P}$  (assuming that  $P$  is a fresh label) with some fixed target. However, choosing the ‘right’ target term to cope with the examples in the literature (and the new ones that may appear in the future) within our format is extremely intricate, if not impossible. That is why we introduce an extension of our proposed rule format for unit elements that

handles predicates as first-class objects, rather than coding them as transitions with dummy targets.

Next, we define the extension of our rule format to cater for predicates. As we did in the earlier developments, in this section we assume that, for each constant  $c$ , each  $c$ -defining deduction rule for predicates is an axiom of the form  $P c$ .

**Definition 12 (Extended left- and right-aligned pairs).** Given a TSS, the sets  $L$  and  $R$  of pairs of binary function symbols and constants are the largest sets satisfying the following conditions.

1. For each  $(f, c) \in L$ , the following conditions hold:
  - (a) For each action  $a \in \mathcal{L}$ , there exists a deduction rule of the following form:

$$\frac{\{x_0 \xrightarrow{a_i} y_i \mid i \in I\} \cup \{P_k x_0 \mid k \in K\} \cup \{x_0 \xrightarrow{a_j} \text{ or } \neg P_j x_0 \mid j \in J\} \cup \{x_1 \xrightarrow{a} z_1\}}{f(x_0, x_1) \xrightarrow{a} t'}$$

where

- i. the variables  $y_i, z_1, x_0$  and  $x_1$  are all pairwise distinct,
  - ii. for each  $j \in J$ , there is no  $c$ -defining deduction rule with  $a_j$  or  $P_j$  as label (depending on whether the formula with index  $j$  is a transition or a predicate formula),
  - iii. there exists a collection  $\{P_k c \mid k \in K\}$  of  $c$ -defining axioms, and
  - iv. there exists a collection  $\{c \xrightarrow{a_i} q_i \mid i \in I\}$  of  $c$ -defining axioms such that  $\sigma(t') \cong z_1$ , where  $\sigma$  is the substitution mapping  $x_0$  to  $c$ , each  $y_i$  to  $q_i$ ,  $i \in I$ , and is the identity on all the other variables.
- (b) For each predicate  $P \in \mathcal{P}$ , there exists a deduction rule, of the following form:

$$\frac{\{P_i x_0 \mid i \in I\} \cup \{\neg P_j x_0 \mid j \in J\} \cup \{P x_1\}}{P f(x_0, x_1)}$$

where

- i. for each  $j \in J$ , there is no  $c$ -defining axiom with  $P_j$  as label, and
  - ii. there exists a collection  $\{P_i c \mid i \in I\}$  of  $c$ -defining axioms.
- (c) Each  $f$ -defining deduction rule has one of the following forms:

$$\frac{\Phi}{f(t_0, t_1) \xrightarrow{a} t'} \quad \text{or} \quad \frac{\Phi}{P f(t_0, t_1)}$$

where  $a \in \mathcal{L}$ ,  $P \in \mathcal{P}$  and for each closed substitution  $\sigma$  with  $\sigma(t_0) \equiv c$ ,

- i. either there exists some  $t_1 \xrightarrow{a} t'' \in \Phi$  with  $\sigma(t') \cong \sigma(t'')$  (if the conclusion is a transition formula), or  $P t_1 \in \Phi$  (if the conclusion is a predicate formula), or
- ii. there exists a premise  $\phi \in \Phi$  with  $t_0$  as its source such that

- A. either  $\phi$  is a positive formula and the collection of conclusions of  $c$ -defining deduction rules does not entail  $\sigma(\phi)$ , or
  - B.  $\phi$  is a negative formula and the collection of conclusions of  $c$ -defining axioms contradicts  $\sigma(\phi)$ .
2. The definition of right-aligned pairs of operators and constant symbols – that is, those such that  $(f, c) \in R$  – is symmetric and is not repeated here.

We are now ready to state the counterpart of Theorem 2 in a setting with predicates.

**Theorem 3.** *Let  $\mathcal{T}$  be a complete TSS in which each rule is  $f$ -defining for some function symbol  $f$ . Assume that  $L$  and  $R$  are the sets of extended left- and right-aligned function symbols according to Definition 12. For each  $(f, c) \in L$ , it holds that  $f(c, x) \Leftrightarrow x$ . Symmetrically, for each  $(f, c) \in R$ , it holds that  $f(x, c) \Leftrightarrow x$ .*

Appendix A contains the proof of Theorem 3.

We now provide some examples of the application of the extended rule format.

**Example 10 (Sequential Composition).** A standard operator whose operational semantics can be given using predicates is that of sequential composition. Consider the following deduction rules, where  $p \downarrow$  means that ‘ $p$  can terminate successfully.’ (As usual in the literature, we write the termination predicate  $\downarrow$  in postfix notation.)

$$\frac{}{1 \downarrow} \quad \frac{x \downarrow \quad y \downarrow}{x \cdot y \downarrow} \quad \frac{x \xrightarrow{a} x'}{x \cdot y \xrightarrow{a} x' \cdot y} \quad \frac{x \downarrow \quad y \xrightarrow{a} y'}{x \cdot y \xrightarrow{a} y'}$$

Take  $L = R = \{(\cdot, 1)\}$ . The TSS conforms to our extended rule format. The second deduction rule matches conditions 1b and 1c of Definition 12 (and the symmetric ones omitted for the right-aligned operators). The third deduction rule satisfies condition 1(c)iiA of Definition 12 (and the omitted 2(a) and 2(c) conditions). The rightmost deduction rule satisfies conditions 1a and 1(c)i of Definition 12, as well as the omitted condition 2(c)iiA because  $1$  has no transitions.

**Example 11 (Fair parallel composition operators).** Consider the operators  $\parallel^m$  and  $\parallel_n$  with  $m, n \geq 0$ , which are inspired by those introduced in [26]. The idea is that  $\parallel^m$  is a left-parallel composition for  $m + 1$  steps, unless it terminates successfully before completing a sequence of  $m + 1$  transitions, and then turns into  $\parallel_n$  for some  $n \geq 0$ . The operator  $\parallel_n$  behaves symmetrically.

The SOS rules for these operators are as follows, with  $m, n \geq 0$ .

$$\frac{x \downarrow \quad y \downarrow}{x \parallel^m y \downarrow} \quad \frac{x \xrightarrow{a} x'}{x \parallel^0 y \xrightarrow{a} x' \parallel_n y} \quad \frac{x \downarrow \quad y \xrightarrow{a} y'}{x \parallel^m y \xrightarrow{a} y'} \quad \frac{x \xrightarrow{a} x'}{x \parallel^{m+1} y \xrightarrow{a} x' \parallel^m y}$$

$$\frac{x \downarrow \quad y \downarrow}{x \parallel_n y \downarrow} \quad \frac{y \xrightarrow{a} y'}{x \parallel_0 y \xrightarrow{a} x \parallel^m y'} \quad \frac{x \xrightarrow{a} x' \quad y \downarrow}{x \parallel_n y \xrightarrow{a} x'} \quad \frac{y \xrightarrow{a} y'}{x \parallel_{n+1} y \xrightarrow{a} x \parallel_n y'}$$

Take  $L = R = \{(\parallel^m, 1), (\parallel_n, 1) \mid m, n \geq 0\}$ , where 1 is the constant specified in the above example. The TSS conforms to our extended rule format. By way of example, consider a pair of the form  $(\parallel^m, 1) \in L$  with  $m \geq 0$ . The first rule for  $\parallel^m$  satisfies condition 1b. The third rule for  $\parallel^m$  meets requirements 1a and 1(c)i in Definition 12 for each label  $a$ . On the other hand, the second and the fourth rule meet requirement 1(c)iiA because 1 affords no transitions. The situation is entirely symmetric if we consider  $(\parallel^m, 1)$  as a member of the set  $R$ .

A similar reasoning shows that the conditions in Definition 12 are met by  $(\parallel_n, 1)$  for each  $n \geq 0$ .

**Example 12 (Fair Parallel Composition Operators, Reprise).** Consider the following variation on the above example, which uses only operators of the form  $\parallel^m$  with  $m \geq 0$ . Again, the idea is that  $\parallel^m$  is a left-parallel composition for  $m + 1$  steps. However, when the ‘time slice’ for its left argument is over, a process of the form  $p \parallel^m q$  now turns into  $q \parallel^n p$  for some  $n$ . Intuitively, this family of operators implements a round-robin policy with a flexible allocation of the time slice for each process.

The SOS rules for these operators are as follows, with  $m, n \geq 0$ .

$$\frac{x \downarrow \quad y \downarrow}{x \parallel^m y \downarrow} \quad \frac{x \xrightarrow{a} x'}{x \parallel^0 y \xrightarrow{a} y \parallel^n x'} \quad \frac{x \downarrow \quad y \xrightarrow{a} y'}{x \parallel^m y \xrightarrow{a} y'} \quad \frac{x \xrightarrow{a} x'}{x \parallel^{m+1} y \xrightarrow{a} x' \parallel^m y}$$

Take  $L = R = \{(\parallel^m, 1) \mid m \geq 0\}$ . Reasoning as in the previous example, it is not hard to see that the TSS conforms to our extended rule format. Note that, in order to handle this example, it is fundamental to consider the sets  $L$  and  $R$  simultaneously.

## 5. A rule format for zero elements

In this section we provide a rule format guaranteeing that certain constants act as left or right zero elements for a set of binary operators. To this end we employ a variation on the techniques we developed in Sections 3–4 for left or right unit elements.

As before, we make use of an equivalence relation between terms called *zero-context equivalence*, which is the counterpart of the unit-context equivalence from Definition 10. Intuitively if  $c$  is a left zero element for an operator  $f$  and  $c$  is also a right zero element for  $g$ , then the terms  $f(c, t_1)$  and  $g(t_2, c)$  are both zero-context equivalent to  $c$  and zero-context equivalent to each other.

In the following formal definition of zero-context equivalence, it is useful to consider  $(f, c) \in L$  as stating that ‘ $c$  acts as a left zero element for the operator  $f$ ’ and analogously  $(f, c) \in R$  indicates that the constant  $c$  is a right zero element for  $f$ .

**Definition 13 (Zero-context equivalent terms).** Given sets  $L, R \subseteq \Sigma \times \Sigma$  of pairs of binary function symbols and constants,  $\cong_0^{L,R}$  is the smallest equivalence relation over  $\mathbb{T}(\Sigma)$  satisfying the following constraints, for each  $s \in \mathbb{T}(\Sigma)$ :

1.  $\forall (f, c) \in L. c \cong_0^{L,R} f(c, s)$ , and
2.  $\forall (g, d) \in R. d \cong_0^{L,R} g(s, d)$ .

We say that two terms  $s, t \in \mathbb{T}(\Sigma)$  are *zero-context equivalent*, if  $s \cong_0^{L,R} t$ .

Since the sets  $L$  and  $R$  are always clear from the context, in the remainder of the paper we write  $\cong_0$  in place of  $\cong_0^{L,R}$ .

**Theorem 4 (Decidability of zero-context equivalence).** *Let  $L, R \subseteq \Sigma \times \Sigma$  be finite sets of pairs of binary function symbols and constants. Then, for all terms  $t, u \in \mathbb{T}(\Sigma)$ , it is decidable whether  $t \cong_0^{L,R} u$  holds.*

PROOF. Let  $L$  and  $R$  be given. Suppose that we are given two terms  $t$  and  $u$  and we want to check whether they are zero-context equivalent. From  $t$  and  $u$ , construct the (undirected) graph  $G(t, u)$  as follows.

The nodes in  $G(t, u)$  are

- $t$  and  $u$ ,
- the constants mentioned in  $L$  and  $R$ , and
- all terms of the form  $f(c, d)$  with  $(f, c) \in L$  and  $(f, d) \in R$ .

The edges in  $G(t, u)$  are given by items 1 and 2 in Definition 13. This graph is finite, since  $L$  and  $R$  are finite, and can be built effectively. Note that  $G(u, t)$  and  $G(t, u)$  are identical.

We claim that  $t$  is zero-context equivalent to  $u$  iff  $t$  can be reached from  $u$  in  $G(t, u)$ .

The proof of this claim is as follows. The right-to-left implication is immediate since each edge in  $G(t, u)$  corresponds to an application of item 1 or item 2 in Definition 13. For the converse, we proceed by induction on the length of a shortest proof of  $t \cong_0^{L,R} u$ . If  $t \cong_0^{L,R} u$  follows by reflexivity or by using item 1 or 2 in Definition 13 then  $t$  can be reached from  $u$  in  $G(t, u)$  in zero steps or in one step, respectively. If  $t \cong_0^{L,R} u$  is proven using symmetry then the claim follows by the inductive hypothesis. Assume now that  $t \cong_0^{L,R} u$  follows by transitivity.

Then there is some term  $s$  such that  $t \cong_0^{L,R} s$  (in one step) and  $s \cong_0^{L,R} u$ . By induction and the symmetry of reachability,  $s$  is reachable from  $t$  in  $G(t, s)$  and  $s$  is reachable from  $u$  in  $G(s, u)$ . To see that  $u$  is reachable from  $t$  in  $G(t, u)$ , we now observe that  $s$  can be taken to be

- a constant mentioned in  $L$  or  $R$ , if  $t = f(c, t')$  for some  $(f, c) \in L$  or  $t = f(t', c)$  for some  $(f, c) \in R$ , or
- if  $t$  is a constant  $c$ , a term of one of the following forms for some constant  $d$ :
  - $f(c, d)$ , where  $(f, c) \in L$  and  $(f, d) \in R$ , or
  - $f(d, c)$ , where  $(f, c) \in R$  and  $(f, d) \in L$ .

Indeed, assume, by way of example, that  $t = c$  and  $s = f(c, t')$ , where  $(f, c) \in L$  and  $t'$  is not a constant  $d$  such that  $(f, d) \in R$ . Then the proof of  $s \stackrel{L,R}{\cong}_0 u$  could only proceed in the next step by going back to  $t = c$ , contradicting our assumption that it was a shortest proof of  $t \stackrel{L,R}{\cong}_0 u$ .

It follows that both  $G(t, s)$  and  $G(s, u)$  are subgraphs of  $G(t, u)$ , and therefore  $t$  is reachable from  $u$  in  $G(t, u)$ , as claimed.  $\square$

We now proceed to define the rule format for left and right zero elements, which is the second main contribution of the paper. Before doing so, however, it may be useful to discuss some examples, which highlight two of the key design criteria in the definition to follow.

**Example 13.** Assume that  $a$  is the only action. Let  $0$  be a constant with deduction rule

$$\frac{}{0 \mapsto 0}$$

Furthermore consider the binary operators  $_{[n]}-$ , for  $n \geq 0$ , with deduction rules

$$\frac{x \mapsto x'}{x[0]y \mapsto x'} \quad \frac{x \mapsto x'}{x[n+1]y \mapsto x'[n]y} \quad \frac{x \xrightarrow{a} x'}{x[n]y \xrightarrow{a} x'[n]y} \quad \frac{x \not\mapsto \quad x \xrightarrow{a}}{x[n]y \mapsto y}$$

Assuming that the transition relation  $\mapsto$  denotes unit time steps,  $p[n]q$  denotes that  $q$  will start only when  $p$  has finished in at most  $n$  time units. In order to prove that  $0$  is a left zero element for the operator  $_{[n]}-$  one needs to show also that it is a left zero element for all operators  $_{[i]}-$  with  $0 \leq i < n$ . It is not hard to see that the relation

$$\mathcal{R}_n = \{(0[i]p, 0) \mid 0 \leq i \leq n, p \in \mathbb{C}(\Sigma)\} \cup \{(0, 0)\}$$

is a bisimulation. Therefore,  $0$  is a left zero element for the operator  $_{[n]}-$ .

In the previous example, the zero element property for  $_{[n]}-$  depends on that property for all  $_{[i]}-$  with  $0 \leq i < n$ . The next example illustrates that this dependency can even be worse.

**Example 14.** Assume that  $a$  is the only action and consider the binary operators  $f_i$ ,  $i \geq 0$ , with rules

$$\frac{x_0 \xrightarrow{a} y_0}{f_i(x_0, x_1) \xrightarrow{a} f_{i+1}(y_0, x_1)} .$$

Let  $\text{RUN}_a$  be the constant mentioned in Example 6 with rule  $\text{RUN}_a \xrightarrow{a} \text{RUN}_a$ . Then  $f_i(\text{RUN}_a, p) \Leftrightarrow \text{RUN}_a$ , for each closed term  $p$  and  $i \geq 0$ . Indeed, it is not hard to see that the relation

$$\mathcal{R} = \{(f_i(\text{RUN}_a, p), \text{RUN}_a) \mid i \geq 0, p \in \mathbb{C}(\Sigma)\}$$

is a bisimulation. Therefore,  $\text{RUN}_a$  is a left zero element for each of the operators  $f_i$ ,  $i \geq 0$ . Note that, in order to show that  $\text{RUN}_a$  is a left zero element for, say,  $f_0$ , we need to consider a set of operators, namely  $\{f_i \mid i \geq 0\}$ . Moreover, in order to show that  $\text{RUN}_a$  is a left zero element for  $f_i$ ,  $i \geq 0$ , we need to prove that  $\text{RUN}_a$  is a left zero element for  $f_{i+1}$ . Therefore the set of proof obligations is not *well-founded*.

**Example 15.** Consider the following TSS with constant  $\text{RUN}_a$  and binary function symbols  $f$  and  $g$  with rules

$$\frac{x_0 \xrightarrow{a} y_0}{f(x_0, x_1) \xrightarrow{a} g(x_1, y_0)} \quad \frac{x_1 \xrightarrow{a} y_1}{g(x_0, x_1) \xrightarrow{a} f(y_1, x_0)}$$

It is not hard to see that  $f(\text{RUN}_a, p) \Leftrightarrow \text{RUN}_a \Leftrightarrow g(p, \text{RUN}_a)$ , for each closed term  $p$ . Therefore  $\text{RUN}_a$  is a left zero element for  $f$  and a right zero element for  $g$ . In the light of the mutual dependency between  $f$  and  $g$ , this example indicates that a widely applicable rule format for left zero elements will need to be based at the same time on a rule format for right zero elements, and vice versa.

In order to remain in line with the terminology in Definition 11, in the following definition we talk about left- and right-aligned pairs (for zero elements).

**Definition 14 (Left- and right-aligned pairs for zero elements).** Given a TSS with set of predicate symbols  $\mathcal{P}$  and set of labels  $\mathcal{L}$ , the sets  $L$  and  $R$  of pairs of binary function symbols and constants are the largest sets satisfying the following constraints.

1. For each  $(f, c) \in L$ , the following conditions hold.
  - (a) Whenever an axiom  $c \xrightarrow{a} t$  (or  $P c$ ) does exist then there is a rule:

$$\frac{\{x_0 \xrightarrow{a_i} t_i \mid i \in I\} \cup \{P_k x_0 \mid k \in K\} \cup \{x_0 \xrightarrow{a_j} \text{ or } \neg P_j x_0 \mid j \in J\}}{f(x_0, x_1) \xrightarrow{a} t' \quad (\text{or } P f(x_0, x_1))}$$

where

- i.  $x_1 \notin \{x_0\} \cup \bigcup_{i \in I} \text{vars}(t_i)$ ,

- ii. for each  $j \in J$ , there is no  $c$ -defining axiom with  $a_j$  or  $P_j$  as label (depending on whether the formula with index  $j$  is a transition or a predicate formula),
  - iii. there exists a collection  $\{P_k c \mid k \in K\}$  of  $c$ -defining axioms, and
  - iv. there exists some substitution  $\sigma$  such that  $\sigma(x_0) = c$ ,  $\{c \xrightarrow{a_i} \sigma(t_i) \mid i \in I\}$  is included in the collection of  $c$ -defining axioms, and if the conclusion is a transition formula,  $\sigma(t') \cong_0 t$ .
- (b) Each  $f$ -defining deduction rule has one of the following forms:

$$\frac{\Phi}{f(t_0, t_1) \xrightarrow{a} t'} \quad \text{or} \quad \frac{\Phi}{P f(t_0, t_1)}$$

where  $a \in \mathcal{L}$ ,  $P \in \mathcal{P}$  and, for each closed substitution  $\sigma$  such that  $\sigma(t_0) \equiv c$ , one of the following cases holds:

- i. there exists an axiom  $c \xrightarrow{a} t$  with  $\sigma(t') \cong_0 t$  (if the conclusion is a transition formula), or an axiom  $P c$  (if the conclusion is a predicate formula), or
  - ii. there exists a premise  $\phi \in \Phi$  with  $t_0$  as its source such that
    - A. either  $\phi$  is a positive formula and the collection of  $c$ -defining axioms does not entail  $\sigma(\phi)$ , or
    - B.  $\phi$  is a negative formula and the collection of  $c$ -defining axioms contradicts  $\sigma(\phi)$ .
2. The definition of right-aligned pairs of operators and constant symbols—that is, those such that  $(f, c) \in R$ —is symmetric and is not repeated here.

For a function symbol  $f$  and a constant  $c$ , we call  $(f, c)$  *left aligned* (respectively, *right aligned*) if  $(f, c) \in L$  (respectively,  $(f, c) \in R$ ).

The structure of the above definition is akin to that of Definition 11, but there are, however, significant differences in the details. Intuitively, the aim of condition 1a is to ensure that whenever the constant  $c$  performs, say, an  $a$ -transition then also  $f(c, p)$  does so for each closed term  $p$ , and the two transitions lead to terms that are zero-context equivalent. Conversely, condition 1b guarantees that each transition that  $f(c, p)$  can perform actually simulates one of the steps of the constant  $c$ . The clauses play the corresponding role also for predicates.

Note that, as in Definition 11, the sets  $L$  and  $R$  are defined as the largest sets of pairs satisfying the constraints from Definition 14. As remarked earlier, this means that, in order to check whether a constant  $c$  is, for example, a left zero element for an operator  $f$ , it is sufficient that the pair  $(f, c)$  be contained in  $L$  for a pair of sets  $L$  and  $R$  that satisfy the conditions above.

The following theorem states the correctness of the rule format in Definition 14.

**Theorem 5.** *Let  $\mathcal{T}$  be a complete TSS in which each rule is  $f$ -defining for some function symbol  $f$ . Assume that  $L$  and  $R$  are the sets of left- and right-aligned function symbols according to Definition 14. For each  $(f, c) \in L$ , it holds that  $f(c, x) \Leftrightarrow c$ . Symmetrically, for each  $(f, c) \in R$ , it holds that  $f(x, c) \Leftrightarrow c$ .*

Appendix B contains the proof of Theorem 5.

**Example 16.** Consider Example 14. We now show that  $\text{RUN}_a$  is a left zero element for each  $f_i$  using Theorem 5. To this end, let  $L = \{(f_i, \text{RUN}_a) \mid i \geq 0\}$  and take  $R = \emptyset$ . Let us focus on a generic function symbol  $f_i$ . We prove that conditions 1a and 1b are met.

- 1a: For the only axiom  $\text{RUN}_a \xrightarrow{a} \text{RUN}_a$  we can use the only  $f_i$ -defining rule. Here we can associate the axiom  $\text{RUN}_a \xrightarrow{a} \text{RUN}_a$  to the premise  $x_0 \xrightarrow{a} y_0$  and consider a substitution  $\sigma$  such that  $\sigma(x_0) \equiv \sigma(y_0) \equiv \text{RUN}_a$ . Since  $\sigma(y_0) \equiv \text{RUN}_a$  and  $(f_{i+1}, \text{RUN}_a) \in L$ , it follows that

$$\sigma(f_{i+1}(y_0, x_1)) \equiv f_{i+1}(\text{RUN}_a, \sigma(x_1)) \cong_0 \text{RUN}_a ,$$

and we are done.

- 1b: We can associate the only  $f_i$ -defining rule to the axiom  $\text{RUN}_a \xrightarrow{a} \text{RUN}_a$ . Assume that  $\sigma(x_0) \equiv \text{RUN}_a$  but  $\sigma(f_{i+1}(y_0, x_1)) \not\cong_0 \text{RUN}_a$ , and therefore case 1(b)i does not apply. This means that  $\sigma(y_0) \not\equiv \text{RUN}_a$  and therefore the condition in case 1(b)iiA is met.

**Example 17.** Consider now Example 15. We show that  $\text{RUN}_a$  is a left zero element for  $f$  and a right zero element for  $g$  using Theorem 5. Let  $L = \{(f, \text{RUN}_a)\}$  and  $R = \{(g, \text{RUN}_a)\}$ . We limit ourselves to checking that conditions 1a and 1b are met by the pair  $(f, \text{RUN}_a)$  contained in  $L$ .

- 1a: For the only axiom  $\text{RUN}_a \xrightarrow{a} \text{RUN}_a$ , we can use the only rule for  $f$ . Indeed, the obvious substitution  $\sigma$  constructed as required in item 1(a)iv of Definition 14 satisfies that  $\sigma(g(x_1, y_0)) \cong_0 \text{RUN}_a$  because  $(g, \text{RUN}_a) \in R$ .
- 1b: The only  $f$ -defining rule is the one on the left. For that we can consider the axiom  $\text{RUN}_a \xrightarrow{a} \text{RUN}_a$ . If  $\sigma(y_0) \equiv \text{RUN}_a$  then case 1(b)i applies since  $(g, \text{RUN}_a) \in R$ . Otherwise, the condition in case 1(b)iiA is met.

A similar reasoning can be applied to the pair  $(g, \text{RUN}_a)$  in  $R$ .

We conclude this section by discussing some of the constraints in Definition 14 in order to argue that they cannot be easily relaxed. In what follows, we focus on the conditions that left-aligned pairs must meet. First of all, note that relaxing the requirement that  $x_0 \not\equiv x_1$  in condition 1(a)i would jeopardize Theorem 5. To see this, consider the TSS with constant  $\text{RUN}_a$  and binary operator  $f$  with rule

$$\frac{x_0 \xrightarrow{a} y_0}{f(x_0, x_0) \xrightarrow{a} y_0} .$$

It is not hard to check that  $L = \{(f, \text{RUN}_a)\}$  and  $R = \emptyset$  satisfy all the constraints in Definition 14 apart from  $x_0 \not\equiv x_1$ . For example, let us examine condition 1b. Let  $\sigma$  be a closed substitution such that  $\sigma(x_0) \equiv \text{RUN}_a$  and assume that

the axiom for  $\text{RUN}_a$  entails  $\sigma(x_0) \equiv \text{RUN}_a \xrightarrow{a} \sigma(y_0)$ —or else condition 1(b)iiA would be met. It follows that  $\sigma(y_0) \equiv \text{RUN}_a$  and therefore condition 1(b)i is satisfied.

However,  $\text{RUN}_a$  is *not* a left zero element for  $f$ . For example, the term  $f(\text{RUN}_a, f(\text{RUN}_a))$  affords no transition and therefore cannot be bisimilar to  $\text{RUN}_a$ .

The following example shows that relaxing the requirement that

$$x_1 \notin \bigcup_{i \in I} \text{vars}(t_i)$$

in condition 1(a)i would also invalidate Theorem 5. To see this, consider the TSS with constant  $\text{RUN}_a$  and binary operator  $f$  with rule

$$\frac{x_0 \xrightarrow{a} x_1}{f(x_0, x_1) \xrightarrow{a} x_1} .$$

Again, it is not hard to check that  $L = \{(f, \text{RUN}_a)\}$  and  $R = \emptyset$  satisfy all the constraints in Definition 14 apart from the requirement that  $x_1$  should not occur in the target of a positive premise. However,  $f(\text{RUN}_a, f(\text{RUN}_a, \text{RUN}_a))$  affords no transition and therefore cannot be bisimilar to  $\text{RUN}_a$ . This means that  $\text{RUN}_a$  is *not* a left zero element for  $f$ .

The role played by requirements 1(a)ii and 1(a)iv in ensuring that, modulo bisimilarity,  $f(c, p)$  affords ‘the same transitions as  $c$ ’, for each  $p$ , is highlighted by the following two examples.

**Example 18.** Consider the TSS with constants  $\mathbf{0}$  and  $a\&b$ , and a binary operator  $f$  with rules:

$$\frac{}{a\&b \xrightarrow{a} \mathbf{0}} \quad \frac{}{a\&b \xrightarrow{b} \mathbf{0}} \quad \frac{x_0 \xrightarrow{b} \quad x_0 \xrightarrow{a} y_0}{f(x_0, x_1) \xrightarrow{a} y_0} \quad \frac{x_0 \xrightarrow{a} \quad x_0 \xrightarrow{b} y_0}{f(x_0, x_1) \xrightarrow{b} y_0} .$$

It is not hard to check that  $L = \{(f, a\&b)\}$  and  $R = \emptyset$  satisfy all the constraints in Definition 1 apart from 1(a)ii. However, the term  $f(a\&b, \mathbf{0})$  affords no transition unlike  $a\&b$ . Therefore  $a\&b$  is not a left zero element for  $f$ .

**Example 19.** Consider the TSS over set of labels  $\{a, b\}$  with constant  $\text{RUN}_a$  and a binary operator  $f$  with rule:

$$\frac{x_0 \xrightarrow{a} y_0 \quad x_0 \xrightarrow{b} y_1}{f(x_0, x_1) \xrightarrow{a} x_1} .$$

It is easy to check that  $L = \{(f, \text{RUN}_a)\}$  satisfies all the constraints in Definition 1 apart from 1(a)iv. However,  $f(\text{RUN}_a, \text{RUN}_a)$  affords no transition unlike  $\text{RUN}_a$ . Therefore  $\text{RUN}_a$  is *not* a left zero element for  $f$ .

As witnessed, e.g., by Example 23 to follow, constraint 1(b)i enhances the generality of our format. Indeed, if we removed constraint 1(b)i and a left-aligned pair  $(f, c)$  satisfied condition 1(b)ii, then no rule for  $f$  would be applicable to a closed term of the form  $f(c, p)$ . Therefore, no term of the form  $f(c, p)$  would afford a transition. Since  $(f, c)$  satisfies condition 1 in Definition 14, the collection of  $c$ -defining axioms must be empty. As a consequence, the resulting format would be unable to handle left zero elements such as  $\text{RUN}_a$  that afford some transition. Examples of constants with deduction axioms in the literature are immediate deadlock [10], which acts as a left zero element for sequential composition, parallel composition, left merge and communication merge and as a right zero element for parallel composition and communication merge, and delayable deadlock from [7], which is a left zero element for sequential composition.

## 6. Examples of application of the format for zero elements

In this section we show that several examples of zero elements from the literature indeed fit the format described in Section 5. For the sake of readability, where appropriate we repeat the definition of some operators that were introduced in Section 4.1.

**Example 20 (Synchronous parallel composition).** Recall the synchronous parallel composition from CSP [20] over a set of actions  $\mathcal{L}$  with rules:

$$\frac{x \xrightarrow{a} x' \quad y \xrightarrow{a} y'}{x \parallel_{\mathcal{L}} y \xrightarrow{a} x' \parallel_{\mathcal{L}} y'} \quad (a \in \mathcal{L}) .$$

We know that the inaction constant  $\mathbf{0}$ , with no rules, is a left and right zero element for  $\parallel_{\mathcal{L}}$ . Let  $L = R = \{(\parallel_{\mathcal{L}}, \mathbf{0})\}$ . We claim that  $L$  and  $R$  meet the constraints in Definition 14. First of all,  $\mathbf{0}$  has no axioms so the clauses 1a and its symmetric counterpart 2a are vacuously satisfied. To show that also the clause 1b is met, we consider the rule above and note that, for every possible substitution  $\sigma$  such that  $\sigma(x) \equiv \mathbf{0}$ , the empty set of deduction rules does not entail the premise  $\sigma(x) \xrightarrow{a} \sigma(x')$ . This meets constraint 1(b)iiA. The symmetric counterpart of clause 1b is handled in similar fashion. The well-known laws

$$\mathbf{0} \parallel_{\mathcal{L}} y \Leftrightarrow \mathbf{0} \quad \text{and} \quad x \parallel_{\mathcal{L}} \mathbf{0} \Leftrightarrow \mathbf{0}$$

thus follow from Theorem 5.

**Example 21 (Left merge operator).** Consider the left merge operator from [13].

$$\frac{x \xrightarrow{a} x'}{x \ll y \xrightarrow{a} x' \parallel y}$$

Here  $\parallel$  stands for the merge operator from [13], whose SOS specification is immaterial for this example; see the earlier Example 7 and Example 25 to follow. Let  $L = \{(\parallel, \mathbf{0})\}$  and  $R = \emptyset$ . We claim that  $L$  meets the constraints in Definition 14. It is easy to check that the claim is true by the same reasoning used in Example 6. This time it is sufficient to check conditions 1a and 1b because  $\mathbf{0}$  is just a *left* zero element for  $\parallel$ . By Theorem 5 the validity of the law  $\mathbf{0} \parallel y \Leftrightarrow \mathbf{0}$  follows. Note that the pair  $\{(\parallel, \mathbf{0})\}$  cannot be added to  $R$  because the symmetric version of condition 1b would be violated. Indeed  $\mathbf{0}$  is *not* a right zero element for  $\parallel$ .

**Example 22 (Sequential composition (1)).** We now examine an example that involves the use of predicates. Consider the standard sequential composition operator  $\cdot$ , which makes use of the predicate symbol  $\downarrow$ . (The formula  $x \downarrow$  means that  $x$  terminates successfully.)

$$\text{(seq1)} \quad \frac{x \xrightarrow{a} x'}{x \cdot y \xrightarrow{a} x' \cdot y} \quad \text{(seq2)} \quad \frac{x \downarrow \quad y \xrightarrow{a} y'}{x \cdot y \xrightarrow{a} y'} \quad \text{(seq3)} \quad \frac{x \downarrow \quad y \downarrow}{(x \cdot y) \downarrow}$$

Consider the deadlock constant  $\delta$ , defined by no axioms. In particular,  $\delta \downarrow$  does *not* hold.

Let  $L = \{(\cdot, \delta)\}$  and  $R = \emptyset$ . We claim that  $L$  meets the constraints in Definition 14. Here again condition 1a is vacuously true. In order to show that constraint 1b is also satisfied, consider a substitution  $\sigma$  that maps  $x$  to  $\delta$ . It suffices only to observe that each of the above rules has a positive premise  $\phi$  such that  $\sigma(\phi)$  is not entailed by the empty set of rules. Therefore, once again, we fall under case 1(b)iiA. By Theorem 5, the validity of the well-known law  $\delta \cdot y \Leftrightarrow \delta$  follows.

Note that the pair  $\{(\cdot, \delta)\}$  cannot be added to  $R$  because rule (seq1) would invalidate the symmetric counterpart of condition 1b in Definition 14. Indeed  $\delta$  is *not* a right zero element for  $\cdot$ .

**Example 23 (Sequential composition (2)).** Focusing again on the sequential composition operator from the previous example, consider once more the constant  $\text{RUN}_a$  from Example 14 with axiom

$$\frac{}{\text{RUN}_a \xrightarrow{a} \text{RUN}_a} \cdot$$

This constant simply displays  $a$  infinitely many times. This behaviour is enough to preempt the execution of the right-hand argument of  $\cdot$  and our order of business in this example is indeed to check the validity of the laws  $\text{RUN}_a \cdot y \Leftrightarrow \text{RUN}_a$  with  $a \in \mathcal{L}$  using Theorem 5.

Let  $L = \{(\cdot, \text{RUN}_a)\}$  and  $R = \emptyset$ . We claim that  $L$  meets the constraints in Definition 14. To prove this claim, we consider each constraint in turn.

- 1a: We need to match the above axiom for  $\text{RUN}_a$  with a rule that defines  $\cdot$ . The rule we pick is the instance of (seq1) for action  $a$ . The substitution

$\sigma$  constructed in order to meet the requirements in condition 1(a)iv is such that  $\sigma(x) \equiv \text{RUN}_a$  and  $\sigma(x') \equiv \text{RUN}_a$ . Moreover,  $\text{RUN}_a$  is zero-context equivalent to  $\text{RUN}_a \cdot y$  and we are done.

- 1b: Since  $\text{RUN}_a \downarrow$  does not hold, with the rules (seq2) and (seq3) we fall in the subcase 1(b)iiA. The rule (seq1) falls instead in the subcase 1(b)i for the same reason of the case 1a examined above.

Note that, following the above reasoning, we can show the validity of laws of the form  $c \cdot y \Leftrightarrow c$ , where  $c$  is any constant whose behaviour is defined by a collection of axioms of the form  $\{c \xrightarrow{a_i} c \mid i \in I\}$ , where  $I$  is any index set.

**Example 24 (Predictable failure constant of  $BPA_{0\delta}$ ).** In this example we focus on the language  $BPA_{0\delta}$  of Baeten and Bergstra—see [11]. The *predictable failure*  $0$  is a non-standard constant that ‘absorbs the computation’ no matter where it appears within the context of the sequential composition operator  $\cdot$ . Namely, the laws  $x \cdot 0 \Leftrightarrow 0$  and  $0 \cdot x \Leftrightarrow 0$  both hold. The following SOS rules for the language  $BPA_{0\delta}$  make use of the predicate  $\neq 0$  that determines whether or not a process can be proved equal to  $0$ , and of predicates  $\xrightarrow{a} \checkmark$  that tell us when a process can terminate by performing an  $a$  action.

$$\begin{array}{c}
\frac{}{a \neq 0} \quad \frac{}{\delta \neq 0} \quad \frac{}{a \xrightarrow{a} \checkmark} \quad \frac{x \neq 0}{x + y \neq 0} \quad \frac{y \neq 0}{x + y \neq 0} \\
\\
\frac{x \xrightarrow{a} x'}{x + y \xrightarrow{a} x'} \quad \frac{x \xrightarrow{a} \checkmark}{x + y \xrightarrow{a} \checkmark} \quad \frac{y \xrightarrow{a} y'}{x + y \xrightarrow{a} y'} \quad \frac{y \xrightarrow{a} \checkmark}{x + y \xrightarrow{a} \checkmark} \\
\\
\frac{x \neq 0 \quad y \neq 0}{x \cdot y \neq 0} \quad \frac{x \xrightarrow{a} x' \quad y \neq 0}{x \cdot y \xrightarrow{a} x' \cdot y} \quad \frac{x \xrightarrow{a} \checkmark \quad y \neq 0}{x \cdot y \xrightarrow{a} y}
\end{array}$$

Let  $L = R = \{(\cdot, 0)\}$ . We claim that  $L$  and  $R$  meet the constraints in Definition 14. Firstly,  $0$  has no axioms so the clause 1a and its symmetric counterpart are vacuously satisfied. To show that clause 1b is satisfied, we must consider the three rules for  $\cdot$  one by one. Since  $0 \neq 0$  does not hold we fall into case 1(b)ii with the leftmost rule. Since  $0 \xrightarrow{a}$  and  $0 \xrightarrow{a} \checkmark$  for any  $a$ , the remaining rules also fall into the case 1(b)iiA. The symmetric counterpart of condition 1b is satisfied for each of the rules because  $0 \neq 0$  does not hold. The laws

$$x \cdot 0 \Leftrightarrow 0 \quad \text{and} \quad 0 \cdot x \Leftrightarrow 0$$

thus follow by Theorem 5.

**Example 25 (Merge operator).** Let  $\mathcal{L}$  be the set of actions. Consider the classic merge operator  $\parallel$  with the following rules, where  $a \in \mathcal{L}$ .

$$\frac{x \xrightarrow{a} x'}{x \parallel y \xrightarrow{a} x' \parallel y} \quad \frac{y \xrightarrow{a} y'}{x \parallel y \xrightarrow{a} x \parallel y'}$$

Let  $\text{RUN}_{\mathcal{L}}$  be a constant defined by axioms  $\text{RUN}_{\mathcal{L}} \xrightarrow{a} \text{RUN}_{\mathcal{L}}$  for each action  $a \in \mathcal{L}$ . We claim that the constant  $\text{RUN}_{\mathcal{L}}$  is both a left and right zero element for  $\parallel$ . This can be checked using Theorem 5. Indeed, let  $L = R = \{\{\parallel, \text{RUN}_{\mathcal{L}}\}\}$ . It is easy to see that condition 1a in Definition 14 is met for  $\text{RUN}_{\mathcal{L}} \xrightarrow{a} \text{RUN}_{\mathcal{L}}$  by taking the instance of the left-hand rule for  $\parallel$  with action  $a$ . Moreover, such a rule also meets condition 1(b)i.

Consider now the right-hand rule for  $\parallel$  with action  $a$ . That rule also meets condition 1(b)i. Indeed, for each closed substitution  $\sigma$  such that  $\sigma(x) \equiv \text{RUN}_{\mathcal{L}}$ , we have that

$$\sigma(x \parallel y') \equiv \text{RUN}_{\mathcal{L}} \parallel \sigma(y') \cong_0 \text{RUN}_{\mathcal{L}}$$

and  $\text{RUN}_{\mathcal{L}} \xrightarrow{a} \text{RUN}_{\mathcal{L}}$  is one of the axioms for the constant  $\text{RUN}_{\mathcal{L}}$ .

**Example 26 (A right-choice operator).** In this example we apply our format to a non-standard operator. For the sake of simplicity we assume that  $a$  is the only action. Consider a variant of the choice operator of Milner's CCS [21], where the right-hand argument has a higher priority than the left-hand argument, i.e., the scheduler executes the left-hand argument only when the other one has no transitions. The rules for such an operator are as follows:

$$\frac{x \xrightarrow{a} x' \quad y \not\xrightarrow{a}}{x \leftarrow y \xrightarrow{a} x'} \quad \frac{y \xrightarrow{a} y'}{x \leftarrow y \xrightarrow{a} y'}$$

Let  $c$  be any constant whose behaviour is defined by a non-empty, finite collection of axioms  $\{c \xrightarrow{a} p_i \mid i \in I\}$ , where  $I$  is some index set. Reasoning as in the previous examples, using Theorem 5, we are able to prove the validity of the law  $x \leftarrow c \Leftrightarrow c$ . We leave the details to the reader. The operator studied in this example bears resemblance with the preferential choice operator  $\leftarrow$  from [15].

## 7. Discussion of the format for zero elements

The format for left and right zero elements we presented in Definition 14 is based on rather intuitive design decisions and, as witnessed by the examples discussed in Section 6, it is applicable to several operators from the literature. However, the format does have some, mostly theoretical, limitations and can be modified in several ways in order to improve some of its features. After all, the design of rule formats for SOS is always based on a trade-off between generality and applicability, and is, to some extent, an 'experimental science'.

Below we discuss two features of the rule format described in Definition 14. This discussion will motivate the development of an alternative format for left and right zero elements that we present in Section 8 to follow.

### 7.1. Premises of rules

One of the main potential limitations of the format for left zero elements is that the form of the rules in condition 1a does *not* allow one to test the variable  $x_1$  in the premises; that is, we are able to test only the variable that

will be instantiated with the left zero element  $c$ . The reason for this design choice is as follows. When an axiom  $c \xrightarrow{a} t$  is present, we must ensure that, regardless of the second argument of  $f$ , at least one rule for  $f$  having an  $a$ -labelled transition as its conclusion does fire (leading to a term that is bisimilar to  $t$ ). The way we guarantee this property stems from Definition 11, i.e., we judiciously manage the presence/absence of  $c$ -defining axioms in order to satisfy the premises. Moreover, we require a strong syntactic connection between the closed term that is the target of the axiom  $c \xrightarrow{a} t$  and the instantiated target of the conclusion of the rule for  $f$ . The same reasoning underlies our design choices for rules ‘matching’  $c$ -defining axioms of the form  $P c$ , where  $P$  is a predicate symbol.

In condition 1(b)ii, we must ensure that the rule under consideration either *cannot* fire when the first argument of  $f$  is instantiated with  $c$  or otherwise it would lead to a term that is bisimilar to a derivative of the left zero element.

In both of the aforementioned situations, it is important to reason about the satisfiability of premises of rules. The conditions we give in 1a and 1(b)ii can be indeed regarded as a basic, syntactic approximation of our attempt to ensure firability/unfirability of the rules in question, when the first argument of the operator  $f$  is the considered left zero element. Premises involving the argument  $x_1$  are a challenge, because, since  $x_1$  can be replaced by an arbitrary closed term, there is no easy, purely syntactic way to ensure their satisfiability in the context of a left zero element  $c$ . For this reason, testing  $x_1$  is forbidden by the format for left zero elements in Definition 14. However, this choice does not allow us to handle left zero elements such as the one in the following example.

**Example 27.** Consider a TSS with constants  $\mathbf{0}$  and  $\text{RUN}_a$  (from Example 14), and a function symbol  $f$  defined as follows

$$(y) \frac{x \xrightarrow{a} x' \quad y \xrightarrow{a} y'}{f(x, y) \xrightarrow{a} x'} \quad (\text{not-}y) \frac{x \xrightarrow{a} x' \quad y \not\xrightarrow{a}}{f(x, y) \xrightarrow{a} x'} .$$

The constant  $\text{RUN}_a$  is a left zero element for  $f$ , but the pair  $(f, \text{RUN}_a)$  is not left aligned because the test on the variable  $y$  is forbidden by condition 1a in Definition 14.

This example is admittedly highly artificial. (Indeed, we are not aware of any operator from the literature that is specified using rules akin to the ones given above.) The following one is perhaps less so.

**Example 28.** Assume that  $a$  is the only action. Consider the TSS with constant  $\text{RUN}_a$  from Example 14 and binary operator  $f$  with rule

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

We claim that the constant  $\text{RUN}_a$  is a left and a right zero element for  $f$ . Indeed, each closed term in the TSS above is bisimilar to  $\text{RUN}_a$ . On the other hand,

the pair  $(f, \text{RUN}_a)$  is neither left- nor right-aligned because of the premises involving  $y$  and  $x$  in the rule for  $f$ , respectively.

Admittedly, neither of the examples given above is to be found in the literature. However, we feel that the study of versions of our rule format that allow one to test both arguments of a binary operator is a natural question to address. In Section 8.2, we propose a format, based on the GSOS format of Bloom, Istrail and Meyer, that is able to handle the examples we mentioned above and that has independent interest.

### 7.2. Checking the format, algorithmically

We are aware that checking the requirements in Definition 14 may involve hard work. Namely, they require the user to provide proofs of zero-context equivalence between terms and of entailment/contradiction between collections of transition formulae. This is not an unexpected drawback because it is inherited from the design of the format for left and right unit elements from Definition 11.

Even though the requirements of the proposed format are frequently easy to check in practice, as the examples in Section 6 clearly indicate, their verification may be very lengthy in general and steps toward alternative mechanizable solutions are desirable.

In the next section, our order of business is to provide an alternative rule format for zero elements, which is a candidate for automated checking and retains enough expressiveness to cover relevant examples from the literature, such as those we presented in Section 6.

## 8. A rule format for zero elements based on GSOS

In what follows, we adapt the format from Section 5 in the context of GSOS languages. By employing the *logic of initial transitions* developed in [2], we are able to reason easily about firability of rules, and the resulting rule format is a step towards addressing both the issues discussed in Sections 7.1 and 7.2.

### 8.1. The logic of initial transitions

In this section, for the sake of completeness, we discuss the logic we employ in the definition of our revised rule format for left and right zero elements based on GSOS. The *logic of initial transitions* has been recently introduced by some of the authors in [2] in order to reason about the satisfiability of the premises of GSOS rules. The set of initial transitions formulae over a finite set of actions  $\mathcal{L}$  is defined by the following grammar, where  $a \in \mathcal{L}$ :

$$F ::= \text{True} \mid x \xrightarrow{a} \mid \neg F \mid F \wedge F .$$

As usual, we write  $\text{False}$  for  $\neg \text{True}$ , and  $F \vee F'$  for  $\neg(\neg F \wedge \neg F')$ .

The semantics of this logic is given by a satisfaction relation  $\models$  that is defined, relative to a GSOS language  $G = (\Sigma_G, \mathcal{L}, R_G)$ , by structural recursion

on  $F$  in the following way, where  $\sigma$  is a closed substitution and  $\rightarrow_G$  is the collection of transitions that can be proven using the rules in  $R_G$ :

$$\begin{aligned}
\rightarrow_G, \sigma \models \text{True} & \quad \text{always} \\
\rightarrow_G, \sigma \models x \xrightarrow{a} & \Leftrightarrow \sigma(x) \xrightarrow{a}_G p, \text{ for some } p \\
\rightarrow_G, \sigma \models \neg F & \Leftrightarrow \text{not } \rightarrow_G, \sigma \models F \\
\rightarrow_G, \sigma \models F \wedge F' & \Leftrightarrow \rightarrow_G, \sigma \models F \text{ and } \rightarrow_G, \sigma \models F' .
\end{aligned}$$

The reader familiar with Hennessy-Milner logic [18] will have noticed that the propositions of the form  $x \xrightarrow{a}$  correspond to Hennessy-Milner formulae of the form  $\langle a \rangle \text{True}$ . In what follows, we consider formulae up to commutativity and associativity of  $\wedge$ .

We use the logic to turn the set of premises  $\Phi$  of a GSOS rule into a formula that describes the collection of closed substitutions that satisfy  $\Phi$ . The conversion procedure  $\text{hyps}$  is borrowed from [2]. Formally,

$$\begin{aligned}
\text{hyps}(\emptyset) & = \text{True} \\
\text{hyps}(\{x \xrightarrow{a} y\} \cup \Phi) & = (x \xrightarrow{a}) \wedge \text{hyps}(\Phi \setminus \{x \xrightarrow{a} y\}) \\
\text{hyps}(\{x \not\xrightarrow{a}\} \cup \Phi) & = \neg(x \xrightarrow{a}) \wedge \text{hyps}(\Phi \setminus \{x \not\xrightarrow{a}\}) .
\end{aligned}$$

Intuitively, if  $\Phi$  is the set of premises of a rule then  $\text{hyps}(\Phi)$  is the conjunction of the corresponding initial transition formulae. For example,

$$\text{hyps}(\{x \xrightarrow{a} y, z \not\xrightarrow{b}\}) = (x \xrightarrow{a}) \wedge \neg(z \xrightarrow{b}) .$$

If  $J$  is a finite set of GSOS rules, we overload  $\text{hyps}$  and write:

$$\text{hyps}(J) = \bigvee_{r \in J} \text{hyps}(\Phi_r) ,$$

where  $\Phi_r$  is the set of premises of rule  $r$ .

**Lemma 2.** *Assume that  $G$  is a GSOS language. Let  $\Phi = \Phi_1 \cup \Phi_2$ , where  $\Phi_1$  and  $\Phi_2$  are disjoint, be the set of premises of a rule in  $G$  of the form (1) on page 6. Let  $\sigma$  be a closed substitution such that  $\rightarrow_G, \sigma \models \text{hyps}(\Phi)$  and  $\sigma$  satisfies  $\Phi_1$ . Then there is a closed substitution  $\sigma'$  such that*

- $\sigma'(x_i) = \sigma(x_i)$  for each  $i \in \{1, \dots, l\}$ ,
- $\sigma'(y) = \sigma(y)$  for each target variable  $y$  of a positive premise in  $\Phi_1$  and
- $\sigma'$  satisfies  $\Phi$ .

**PROOF.** We construct a substitution  $\sigma'$  meeting the requirements stated in the lemma by induction on the cardinality of  $\Phi_2$ . If  $\Phi_2$  is empty, then take  $\sigma'$  to be  $\sigma$ . Otherwise, pick an arbitrary transition formula in  $\Phi_2$ . If the transition formula is of the form  $x_i \xrightarrow{b}$ , for some  $i \in \{1, \dots, l\}$  and label  $b$ , then  $\neg(x_i \xrightarrow{b})$  is a conjunct

of  $\text{hyps}(\Phi)$ . As  $\rightarrow_G, \sigma \models \text{hyps}(\Phi)$ , we have that  $\sigma$  satisfies  $x_i \xrightarrow{b}$ . Therefore  $\sigma$  satisfies  $\Phi_1 \cup \{x_i \xrightarrow{b}\}$  and the existence of a substitution  $\sigma'$  meeting the requirements stated in the lemma follows by induction applied to  $\Phi_2 \setminus \{x_i \xrightarrow{b}\}$ .

Consider now the case that  $x_i \xrightarrow{a} y \in \Phi_2$  for some variable  $y$  and label  $a$ . As  $\rightarrow_G, \sigma \models \text{hyps}(\Phi)$  and  $x_i \xrightarrow{a}$  is a conjunct of  $\text{hyps}(\Phi)$ , we have that  $\sigma(x_i) \xrightarrow{a} p$  for some closed term  $p$ . Let  $\sigma''$  be the closed substitution that maps the variable  $y$  to  $p$  and agrees with  $\sigma$  on all the other variables. Since all the variables in a GSOS rule are distinct, and  $\Phi_1$  and  $\Phi_2$  are disjoint,  $\sigma''$  satisfies  $\Phi_1 \cup \{x_i \xrightarrow{a} y\}$ . Moreover, by construction,  $\sigma$  and  $\sigma''$  agree on the variables occurring in the source of the conclusion of the rule and on each target variable  $y'$  of a premise in  $\Phi_1$ . The existence of a substitution  $\sigma'$  meeting the requirements stated in the lemma follows now by induction applied to  $\Phi_2 \setminus \{x_i \xrightarrow{a} y\}$ .  $\square$

We write  $\models_G F \Rightarrow F'$  iff every substitution that satisfies  $F$  also satisfies  $F'$ . This semantic entailment preorder is decidable, as shown in [2].

**Theorem 6 (Decidability of entailment [2]).** *Let  $G$  be a GSOS language. Then, for all formulae  $F$  and  $F'$ , it is decidable whether  $\models_G F \Rightarrow F'$  holds.*

As a matter of fact, when  $\Phi$  is the set of the premises of a rule  $r$ , checking whether  $\models_G \text{True} \Rightarrow \text{hyps}(\Phi)$  holds is equivalent to checking whether the rule  $r$  is always fireable. Conversely, checking whether  $\models_G \text{hyps}(\Phi) \Rightarrow \text{False}$  holds is equivalent to checking whether the rule  $r$  never fires. These considerations will be useful in the remainder of the paper. Our definition of the alternative rule format for left and right zero elements makes use of the logic and especially of these two kinds of entailment. The semantic entailment is, moreover, used in a simplified fashion where one does not need to check all the closed substitutions, but only those that map one variable to the left or right zero element constant under consideration. We now proceed to formalize this notion.

**Definition 15.** Let  $G = (\Sigma_G, \mathcal{L}, R_G)$  be a GSOS language. For each formula  $F$ , constant  $c \in \Sigma_G$  and variable  $x$ , we define the formula  $F[x \mapsto c]$  by structural recursion on  $F$  as follows:

$$\begin{aligned}
\text{True}[x \mapsto c] &= \text{True} \\
(x \xrightarrow{a})[x \mapsto c] &= \begin{cases} \text{True} & \text{if there is a } c\text{-defining axiom } c \xrightarrow{a} p \text{ for some } p \\ \text{False} & \text{otherwise} \end{cases} \\
(y \xrightarrow{a})[x \mapsto c] &= y \xrightarrow{a} \quad \text{if } x \neq y \\
(\neg F)[x \mapsto c] &= \neg(F[x \mapsto c]) \\
(F_1 \wedge F_2)[x \mapsto c] &= (F_1[x \mapsto c]) \wedge (F_2[x \mapsto c]) .
\end{aligned}$$

The connection between  $F$  and  $F[x \mapsto c]$  is provided by the following lemma.

**Lemma 3.** *Let  $G = (\Sigma_G, \mathcal{L}, R_G)$  be a GSOS language. Let  $F$  be a formula,  $c$  be a constant in  $\Sigma_G$  and  $x$  be a variable. Then, for each closed substitution  $\sigma$ ,*

$$\rightarrow_G, \sigma \models F[x \mapsto c] \text{ iff } \rightarrow_G, \sigma[x \mapsto c] \models F ,$$

where  $\sigma[x \mapsto c]$  denotes the substitution that maps  $x$  to  $c$  and acts like  $\sigma$  on all the other variables.

As a consequence of the above lemma, checking whether  $F$  holds for all substitutions that map variable  $x$  to a constant  $c$  amounts to showing that the formula  $F[x \mapsto c]$  is satisfied by all substitutions—that is, showing that  $F[x \mapsto c]$  is a tautology over  $G$ .

### 8.2. An alternative rule format for zero elements

We now have all the ingredients to reformulate the format we presented in Section 5 within the GSOS format. This time the conditions of our format will not try to ensure firability/unfirability of rules by purely syntactic means as in Definition 14, but they instead exploit the logic of initial transition formulae to incorporate a modicum of semantic reasoning within the rule format.

In what follows the reader should bear in mind that, by the considerations in Section 8.1 and by the disjunctive nature of  $\text{hyps}(J)$ , with  $J$  a set of rules, the semantic entailment  $\models_G \text{True} \Rightarrow \text{hyps}(J)$  actually holds whenever, no matter what closed substitution  $\sigma$  we pick, at least one of the rules in the set  $J$  does fire, when it is instantiated with  $\sigma$ .

**Definition 16 (Left- and right-aligned pairs for zero elements, anew).** Let  $G$  be a GSOS language. The sets  $L$  and  $R$  of pairs of binary function symbols and constants are the largest sets satisfying the following constraints.

1. For each  $(f, c) \in L$ , the following conditions hold.
  - (a) For each axiom  $c \xrightarrow{a} t$ , there exists a set  $J$  of rules of the form

$$\frac{\Phi}{f(x_0, x_1) \xrightarrow{a} t'}$$

such that

- i.  $\models_G \text{True} \Rightarrow \text{hyps}(J)[x_0 \mapsto c]$ , and
  - ii. for each rule in  $J$ , one of the following cases holds:
    - A. there is some variable  $y \in \text{vars}(t')$  such that  $x_0 \xrightarrow{a} y \in \Phi$  and  $\sigma(t') \cong_0 t$ , where  $\sigma$  is the substitution mapping  $x_0$  to  $c$ ,  $y$  to  $t$  and is the identity on all the other variables, or
    - B.  $\sigma(t') \cong_0 t$ , where  $\sigma$  is the substitution mapping  $x_0$  to  $c$  and is the identity on all the other variables.
- (b) For each  $f$ -defining deduction rule

$$\frac{\Phi}{f(x_0, x_1) \xrightarrow{a} t'}$$

one of the following cases holds:

- i. there exists an axiom  $c \xrightarrow{a} t$  such that
    - A. there is some variable  $y \in \text{vars}(t')$  such that  $x_0 \xrightarrow{a} y \in \Phi$  and  $\sigma(t') \cong_0 t$ , where  $\sigma$  is the substitution mapping  $x_0$  to  $c$ ,  $y$  to  $t$  and is the identity on all the other variables, or
    - B.  $\sigma(t') \cong_0 t$ , where  $\sigma$  is the substitution mapping  $x_0$  to  $c$  and is the identity on all the other variables.
  - ii.  $\models_G \text{hyps}(\Phi)[x_0 \mapsto c] \Rightarrow \text{False}$ .
2. The definition of right-aligned pairs of operators and constant symbols—that is, those such that  $(f, c) \in R$ —is symmetric and is not repeated here.

For a function symbol  $f$  and a constant  $c$ , we call  $(f, c)$  *left aligned* (respectively, *right aligned*) if  $(f, c) \in L$  (respectively,  $(f, c) \in R$ ).

**Remark 2.** Let  $G$  be a GSOS language over a signature including at least one constant. Since  $\text{hyps}(J)$  is a disjunctive formula, condition 1(a)i in the above definition implies that the set  $J$  is non-empty. On the other hand, condition 1(b)ii says that the premises of the rule under consideration cannot be satisfied by any closed substitution that maps the variable  $x_0$  to the constant  $c$ .

In condition 1a and its symmetric counterpart, one must identify a *set*  $J$  of rules. To understand why, the reader should consider Example 27, where the rules  $(y)$  and  $(\text{not-}y)$  only together allow the operator  $f$  to simulate the behaviour of the constant  $\text{RUN}_a$ : no matter what closed term is substituted for the argument variable  $y$ , we are sure that one of the two rules fires and that the transition leads to  $\text{RUN}_a$ . In Definition 16, these two properties are guaranteed, respectively, by conditions 1(a)i and 1(a)ii.

**Theorem 7.** *Let  $G$  be a GSOS language. Assume that  $L$  and  $R$  are the sets of left- and right-aligned function symbols according to Definition 16. For each  $(f, c) \in L$ , it holds that  $f(c, x) \Leftrightarrow c$ . Symmetrically, for each  $(f, c) \in R$ , it holds that  $f(x, c) \Leftrightarrow c$ .*

Appendix C contains the proof of Theorem 7.

The following result is a consequence of Theorems 4 and 6.

**Theorem 8.** *For GSOS languages, the sets  $L$  and  $R$  can be effectively constructed.*

**Remark 3 (Handling predicates using the format of Definition 16).** The format in Definition 14 and the one in Definition 16 are incomparable. Indeed the former allows for complex terms in the source of the conclusions of rules and in premises of rules. In addition, the format from Definition 14 does not require all variables in the premises of rules to be distinct and permits the use of predicates. GSOS languages forbid all of these features. On the other hand, it is easy to see that, using the format from Definition 16, one can check Example 27, which cannot be dealt with by the format from Definition 14.

It is important to note, however, that the GSOS-based format we presented in Definition 16 can indeed be used to reason about the examples from Section 6

that use predicates. In fact, one can encode a finite collection of predicates specified using rules of the form

$$\frac{\{x_0 \xrightarrow{a_i} y_i \mid i \in I\} \cup \{P_k x_0 \mid k \in K\} \cup \{x_0 \xrightarrow{a_j} \text{ or } \neg P_j x_0 \mid j \in J\}}{P f(x_0, x_1)},$$

where

- the index sets  $I, K$  and  $J$  are finite and
- the variables  $x_0, x_1$  and  $y_i, i \in I$ , are pairwise different,

rather easily by means of transition relations specified by GSOS rules. One can find a number of such encodings in the literature—see, for instance, [17, 30]. The key idea in each of these encodings is that a predicate  $P$  is represented as a transition relation  $\xrightarrow{P}$  (assuming that  $P$  is a fresh action label) with some fixed fresh constant  $c_P$  as target and a fresh variable for the target of each of the premises.

For example, using this encoding strategy, the above rule becomes the standard GSOS rule

$$\frac{\{x_0 \xrightarrow{a_i} y_i \mid i \in I\} \cup \{x_0 \xrightarrow{P_k} z_k \mid k \in K\} \cup \{x_0 \xrightarrow{a_j} \text{ or } x_0 \xrightarrow{P_j} \mid j \in J\}}{f(x_0, x_1) \xrightarrow{P} c_P},$$

where the variables  $z_k$  are fresh and pairwise distinct.

With this encoding of predicates, which preserves finiteness of a language specification, the format proposed in Definition 16 is immediately applicable to all the examples we discussed in Section 6, as well as to those mentioned in, e.g., [12].

**Example 29.** Consider again the TSS discussed in Example 28. We will now argue that the format proposed in Definition 16 is capable of proving the validity of the laws

$$f(x, \text{RUN}_a) \Leftrightarrow \text{RUN}_a \text{ and } f(\text{RUN}_a, y) \Leftrightarrow \text{RUN}_a,$$

unlike the purely syntactic one we introduced in Section 5. To see this, take  $L = R = \{(f, \text{RUN}_a)\}$ . We limit ourselves to checking that conditions 1a and 1b in Definition 16 are met.

- 1a : The only axiom for  $\text{RUN}_a$  is  $\text{RUN}_a \xrightarrow{a} \text{RUN}_a$ . Take  $J$  as the set containing the single rule for  $f$ . Then

$$(x \xrightarrow{a} \wedge y \xrightarrow{a})[x \mapsto \text{RUN}_a] = \text{True} \wedge y \xrightarrow{a}.$$

As we observed in Example 28, each closed term in the TSS under consideration affords an  $a$ -labelled transition. Therefore, the formula  $\text{True} \wedge y \xrightarrow{a}$  is a tautology and condition 1(a)i is met. Note, moreover, that  $x \xrightarrow{a} x'$  is a premise of the only rule for  $f$ ,  $x \in \text{vars}(f(x', y'))$  and  $f(\text{RUN}_a, y') \cong_0 \text{RUN}_a$ . Therefore condition 1(a)ii is also met.

1b : Reasoning as above, we can easily check that the only rule for  $f$  meets condition 1(b)iA.

**Example 30.** Consider now the synchronous parallel composition of Example 6. We claim that the format proposed in Definition 16 is capable of proving the validity of the laws  $\mathbf{0} \parallel_{\mathcal{L}} y \Leftrightarrow \mathbf{0}$  and  $x \parallel_{\mathcal{L}} \mathbf{0} \Leftrightarrow \mathbf{0}$ .

Let  $L = R = \{(\parallel_{\mathcal{L}}, \mathbf{0})\}$ . Since the constant  $\mathbf{0}$  has no axioms, condition 1a is vacuously satisfied. In order to see that also condition 1b is satisfied, it is sufficient to notice that the only rule for  $\parallel_{\mathcal{L}}$  can never fire because  $\mathbf{0}$  has no transitions. Indeed, the entailment  $\models_G (x \xrightarrow{a} \wedge y \xrightarrow{a})[x \mapsto \mathbf{0}] \Rightarrow \text{False}$  holds and condition 1(b)ii is met.

Following the same line of the previous two examples, we are able to show that the proposed format applies to all of the examples in Section 6.

Consider now Example 27. This example is interesting because, in order to meet condition 1a for the only axiom  $\text{RUN}_a \xrightarrow{a} \text{RUN}_a$ , we must choose  $J$  to be the set containing both of the rules  $(y)$  and  $(\text{not-}y)$ . Choosing  $J$  to be a singleton set containing one of the rules is not enough, because neither

$$\models_G \text{True} \Rightarrow (x \xrightarrow{a} \wedge y \xrightarrow{a})[x \mapsto \text{RUN}_a]$$

nor

$$\models_G \text{True} \Rightarrow (x \xrightarrow{a} \wedge y \xrightarrow{\text{not-}a})[x \mapsto \text{RUN}_a]$$

hold. On the other hand, when  $J = \{(y), (\text{not-}y)\}$ , the entailment

$$\models_G \text{True} \Rightarrow ((x \xrightarrow{a} \wedge y \xrightarrow{a}) \vee (x \xrightarrow{a} \wedge y \xrightarrow{\text{not-}a}))[x \mapsto \text{RUN}_a]$$

holds and, moreover,  $\text{RUN}_a \cong_0 \text{RUN}_a$ , meeting condition 1(a)iiA. Therefore the proposed format can check Example 27, which cannot be dealt with by the format from Definition 14.

Example 29 offers a scenario within the realm of GSOS languages that can be handled using the format proposed in Definition 16, but not by means of the purely syntactic one we introduced in Definition 14. As shown by the following example, the format for left and right zero elements in Definition 16 is incomparable with the one in Definition 14 even when restricted to GSOS languages.

**Example 31.** Assume that the set of actions is  $\{a, b\}$ , and consider the GSOS language whose signature consists of a constant  $c$  and a binary operation  $f$ , and whose rules are listed below.

$$\frac{}{c \xrightarrow{a} c} \quad \frac{}{c \xrightarrow{b} c} \quad \frac{x_0 \xrightarrow{a} y_1}{f(x_0, x_1) \xrightarrow{b} f(y_1, x_1)} \quad \frac{x_0 \xrightarrow{b} y_1}{f(x_0, x_1) \xrightarrow{a} f(y_1, x_1)}$$

It is not hard to see that  $L = \{(f, c)\}$  and  $R = \emptyset$  meet the constraints in Definition 14. On the other hand, condition 1(a)ii in Definition 16 fails.

## 9. From zero to unit

In this section we reformulate the unit element format of Definition 11 following the lines of Definition 16.

### Definition 17 (Left- and right-aligned pairs for unit elements, reprise).

Given a GSOS language  $G$ , the sets  $L$  and  $R$  of pairs of binary function symbols and constants are the largest sets satisfying the following constraints.

1. For each  $(f, c) \in L$ , the following conditions hold:
  - (a) For each action  $a \in \mathcal{L}$ , there exists at least one deduction rule of the form

$$\frac{\Phi \cup \{x_1 \xrightarrow{a} y_1\}}{f(x_0, x_1) \xrightarrow{a} t'}$$

where

- i.  $\models_G x_1 \xrightarrow{a} \Rightarrow \text{hyps}(\Phi)[x_0 \mapsto c]$ , and
  - ii. one of the following cases holds:
    - A. there are a premise  $x_0 \xrightarrow{b} y \in \Phi$ , for some  $b \in \mathcal{L}$  and  $y \in \text{vars}(t')$ , and an axiom  $c \xrightarrow{b} t$  such that  $\sigma(t') \cong y_1$ , where  $\sigma$  is the substitution mapping  $x_0$  to  $c$ ,  $y$  to  $t$  and is the identity on all the other variables, or
    - B.  $\sigma(t') \cong y_1$ , where  $\sigma$  is the substitution mapping  $x_0$  to  $c$  and is the identity on all the other variables.
- (b) For each  $f$ -defining deduction rule

$$\frac{\Phi}{f(x_0, x_1) \xrightarrow{a} t'}$$

one of the following cases holds:

- i.  $x_1 \xrightarrow{a} y_1 \in \Phi$  for some variable  $y_1$  and
    - A. either there is a premise  $x_0 \xrightarrow{b} y \in \Phi$ , for some  $b \in \mathcal{L}$  and variable  $y \in \text{vars}(t')$ , such that  $c$  has a single axiom with label  $b$ —say,  $c \xrightarrow{b} t$ —and  $\sigma(t') \cong y_1$ , where  $\sigma$  is the substitution mapping  $x_0$  to  $c$ ,  $y$  to  $t$  and is the identity on all the other variables,
    - B. or  $\sigma(t') \cong y_1$ , where  $\sigma$  is the substitution mapping  $x_0$  to  $c$  and is the identity on all the other variables.
  - ii.  $\models_G \text{hyps}(\Phi)[x_0 \mapsto c] \Rightarrow \text{False}$ .
2. The definition of right-aligned pairs of operators and constant symbols—that is, those such that  $(f, c) \in R$ —is symmetric and is not repeated here.

For a function symbol  $f$  and a constant  $c$ , we call  $(f, c)$  *left aligned* (respectively, *right aligned*) if  $(f, c) \in L$  (respectively,  $(f, c) \in R$ ).

The following theorem states the correctness of the rule format defined above.

**Theorem 9.** *Let  $G$  be a GSOS language. Assume that  $L$  and  $R$  are the sets of left- and right-aligned function symbols according to Definition 17. For each  $(f, c) \in L$ , it holds that  $f(c, x) \Leftrightarrow x$ . Symmetrically, for each  $(f, c) \in R$ , it holds that  $f(x, c) \Leftrightarrow x$ .*

Appendix D contains the proof of Theorem 9.

**Remark 4.** The constraint that  $c \xrightarrow{b} t$  be the only  $c$ -defining axiom with label  $b$  in condition 1(b)iA of Definition 17 is necessary for the validity of Theorem 9. To see this, consider, for instance, the TSS over set of labels  $\{a\}$  with constants  $\mathbf{0}$ ,  $\text{RUN}_a$  (see Example 6) and  $c$ , and the binary operator  $\|\mathcal{L}$  defined in Example 6. The rules for the constant  $c$  are

$$\frac{}{c \xrightarrow{a} c} \quad \frac{}{c \xrightarrow{a} \mathbf{0}} .$$

Observe that the sets  $L = \{\|\mathcal{L}, c\}$  and  $R = \emptyset$  would satisfy the conditions in Definition 17 if the uniqueness requirement were dropped from condition 1(b)iA. On the other hand,  $c \|\mathcal{L} \text{RUN}_a$  is not bisimilar to  $\text{RUN}_a$  because

$$c \|\mathcal{L} \text{RUN}_a \xrightarrow{a} \mathbf{0} \|\mathcal{L} \text{RUN}_a \not\xrightarrow{a} ,$$

while  $\text{RUN}_a$  can only perform action  $a$  forever. Therefore  $c$  is *not* a left unit element for  $\|\mathcal{L}$ .

The following result is a consequence of Theorems 1 and 6.

**Theorem 10.** *For GSOS languages, the sets  $L$  and  $R$  can be effectively constructed.*

The format for left and right unit elements proposed above is incomparable to the one offered in Section 3. Indeed, the latter allows for complex terms as source of the conclusions and in the premises, which the GSOS format forbids. On the other hand, in condition 1a above, the set of premises  $\Phi$  may contain several tests on the argument variable  $x_1$ , which is forbidden by the purely syntactic format presented in Section 3. A concrete, albeit admittedly inexpressive, example of a TSS exploiting this feature is discussed below.

**Example 32.** Consider a TSS, over the set of labels  $\{a, b\}$ , with constants  $\text{RUN}_a$  and  $\text{RUN}_b$ , and a binary function symbol  $f$  defined by the rules below.

$$\frac{y \xrightarrow{a} y' \quad y \not\xrightarrow{b}}{f(x, y) \xrightarrow{a} y'} \quad \frac{y \xrightarrow{b} y' \quad y \not\xrightarrow{a}}{f(x, y) \xrightarrow{b} y'}$$

The constants  $\text{RUN}_a$  and  $\text{RUN}_b$  are both left unit elements for  $f$ . Indeed, *every* closed term is a left unit element for  $f$ . This holds true because each closed term is bisimilar to one of the constants  $\text{RUN}_a$  and  $\text{RUN}_b$ . Therefore, every process is either able to perform initially an  $a$ -transition or is able to perform initially a  $b$ -transition, but never both.

It is not hard to check that the sets  $L = \{(f, \text{RUN}_a), (f, \text{RUN}_b)\}$  and  $R = \emptyset$  satisfy the conditions in Definition 17. On the other hand, the format from Section 3 fails on this basic scenario since  $y$  is tested twice in the rules for  $f$ .

As shown by the following example, the format for left and right unit elements in Definition 17 is incomparable with the one in Definition 11 even when restricted to GSOS languages.

**Example 33.** Observe, first of all, that the TSS in Example 32 is a GSOS language. Therefore that example shows that there are scenarios that fit the GSOS format and that can be handled using Definition 17, but not by means of Definition 11.

We shall now provide an example within the realm of GSOS languages that can be dealt with using Definition 11, but not by means of Definition 17. To this end, assume that the set of actions is  $\{a, b\}$ , and consider the GSOS language whose signature consists of a constant  $c$  and a binary operation  $f$ , and whose rules are listed below, where  $\alpha$  ranges over  $\{a, b\}$ .

$$\frac{}{c \xrightarrow{\alpha} c} \quad \frac{x_0 \xrightarrow{a} y_1 \quad x_0 \xrightarrow{b} y_2 \quad x_1 \xrightarrow{\alpha} z_1}{f(x_0, x_1) \xrightarrow{\alpha} f(y_1, f(y_2, z_1))}$$

It is not hard to see that  $L = \{(f, c)\}$  and  $R = \emptyset$  meet the constraints in Definition 11. On the other hand, condition 1(a)ii in Definition 17 fails.

All the examples from the literature mentioned in Section 4 can be handled by the rule format presented in Definition 17. (Indeed, predicates can be dealt with within the proposed format along the lines discussed in Remark 3.)

## 10. Conclusions

In this paper we have provided rule formats ensuring that certain constants in a language act as left or right unit/zero elements for a set of binary operators. The formats for left and right unit/zero elements presented in Sections 3–5 are mostly of a syntactic nature. To overcome some of the limitations of those formats, in Sections 8–9 we reformulated them within the GSOS format. The resulting formats make use of a modicum of semantic information and employ the logic of initial transitions proposed in [2]. The new formats are able to check relevant examples from the literature and some instances of unit/zero elements left out by the formats in Sections 3–5.

We believe that the formats we propose in this paper for GSOS languages are good candidates for mechanization in a tool-set for checking algebraic laws based on rule formats. The development of such a tool based on the work presented here and on the results surveyed in [5] is an interesting topic for future research.

A natural question that we plan to address in future work is whether there are interesting classes of SOS specifications for which our rule formats provide a complete proof method for determining whether a constant is a left or right

unit/zero element for some binary operator. Not surprisingly, our rule formats are not complete in general. For example, consider the TSS with the following rules

$$\frac{}{c \xrightarrow{a} !c} \quad \frac{x \xrightarrow{a} x'}{f(x, y) \xrightarrow{a} (c \parallel \mathbf{0}) \parallel !c} \quad \frac{x \xrightarrow{a} x'}{!x \xrightarrow{a} x' \parallel !x} ,$$

where ‘ $\parallel$ ’ is the interleaving parallel composition operator from Example 7 and ‘!’ is the unary replication operator familiar from the  $\pi$ -calculus (see, for instance, the book [29]). It is not hard to see that the constant  $c$  is a left zero element for the binary operator  $f$  because the closed terms  $!c$  and  $(c \parallel \mathbf{0}) \parallel !c$  are bisimilar. On the other hand, this fact cannot be shown using our rule formats because those two terms are not zero-context equivalent.

The development of rule formats always involves a trade-off between generality and ease of application. We hope that the wealth of examples from the literature we have provided in this paper have convinced our readers that the formats we have proposed are indeed widely applicable. However, as the above-mentioned TSS indicates, there is room for improving our formats. Developing more powerful, but still easily applicable, formats is another topic for future research.

For many contemporary process algebras the SOS framework used in this paper is still too restricted. Indeed, the SOS semantics of those languages involves more advanced features such as configurations that consist of more than only a process term, i.e., SOS with data, or the presence of structural congruences as an addendum to the SOS specification. Future work will show whether our formats can be generalized to deal with such additions.

Finally, we remark that the rule formats we have offered in this paper can be easily extended to deal with operators with arity greater than or equal to three. In this study we have focussed on rule formats for binary operators since, as the applications in the paper indicate, this suffices for most of the examples we are aware of in the literature on process calculi. We trust that this choice of ours has also allowed us to present our work in the simplest possible setting of interest.

## References

- [1] L. Aceto, A. Birgisson, A. Ingólfssdóttir, M. R. Mousavi, and M. A. Reniers. Rule formats for determinism and idempotence. In F. Arbab and M. Sirjani, editors, *Fundamentals of Software Engineering, Third IPM International Conference, FSEN 2009, Kish Island, Iran, April 15-17, 2009, Revised Selected Papers*, volume 5961 of *Lecture Notes in Computer Science*, pages 146–161. Springer-Verlag, 2010. A full version of this paper will appear in *Science of Computer Programming* and is available at <http://dx.doi.org/10.1016/j.scico.2010.04.002>.
- [2] L. Aceto, M. Cimini, and A. Ingólfssdóttir. A bisimulation-based method for proving the validity of equations in GSOS languages. In *Proceedings of*

*Structural Operational Semantics 2009, August 31, 2009, Bologna (Italy)*, volume 18 of *Electronic Proceedings in Theoretical Computer Science*, pages 1–16, 2010.

- [3] L. Aceto, M. Cimini, A. Ingolfsdottir, M. Mousavi, and M. A. Reniers. On rule formats for zero and unit elements. In *Proceedings of the 26th Conference on the Mathematical Foundations of Programming Semantics (MFPS XXVI)*, Electronic Notes in Theoretical Computer Science, Ottawa, Canada, 2010. Elsevier B.V., The Netherlands.
- [4] L. Aceto, W. Fokkink, and C. Verhoef. Structural operational semantics. In *Handbook of Process Algebra*, pages 197–292. Elsevier, 1999.
- [5] L. Aceto, A. Ingolfsdottir, M. Mousavi, and M. A. Reniers. Algebraic properties for free! *Bulletin of the EATCS*, 99:81–104, October 2009.
- [6] L. Aceto, A. Ingolfsdottir, M. Mousavi, and M. A. Reniers. Rule formats for unit elements. In J. van Leeuwen, A. Muscholl, D. Peleg, J. Pokorný, and B. Rumpe, editors, *SOFSEM 2010, 36th Conference on Current Trends in Theory and Practice of Computer Science, Špindleruv Mlýn, Czech Republic, January 23-29, 2010. Proceedings*, volume 5901 of *Lecture Notes in Computer Science*, pages 141–152. Springer-Verlag, 2010.
- [7] J. Baeten, T. Basten, and M. Reniers. *Process Algebra: Equational Theories of Communicating Processes*, volume 50 of *Cambridge Tracts in Theoretical Computer Science*. Cambridge University Press, 2009.
- [8] J. Baeten and J. Bergstra. Discrete time process algebra. *Formal Aspects of Computing*, 8(2):188–208, 1996.
- [9] J. Baeten and J. Bergstra. Mode transfer in process algebra. Technical Report Report CSR 00–01, Eindhoven University of Technology, 2000.
- [10] J. Baeten and C. Middelburg. *Process Algebra with Timing*. Monographs in Theoretical Computer Science, An EATCS Series. Springer-Verlag, Berlin, 2002.
- [11] J. C. M. Baeten and J. A. Bergstra. Process algebra with a zero object. In *CONCUR '90: Proceedings of the Theories of Concurrency: Unification and Extension*, pages 83–98, Amsterdam, The Netherlands, 1990. Springer-Verlag.
- [12] J. C. M. Baeten and E. P. de Vink. Axiomatizing GSOS with termination. *Journal of Logic and Algebraic Programming*, 60–61:323–351, 2004.
- [13] J. Bergstra and J. W. Klop. Fixed point semantics in process algebras. Report IW 206, Mathematisch Centrum, Amsterdam, 1982.
- [14] J. Bergstra and J. W. Klop. Process algebra for synchronous communication. *Information and Control*, 60(1/3):109–137, 1984.

- [15] J. A. Bergstra and C. A. Middelburg. Preferential choice and coordination conditions. *J. Log. Algebr. Program.*, 70(2):172–200, 2007.
- [16] B. Bloom, S. Istrail, and A. R. Meyer. Bisimulation can’t be traced. *J. ACM*, 42(1):232–268, 1995.
- [17] S. Cranen, M. Mousavi, and M. A. Reniers. A rule format for associativity. In F. van Breugel and M. Chechik, editors, *Proceedings of the 19th International Conference on Concurrency Theory (CONCUR’08)*, volume 5201 of *Lecture Notes in Computer Science*, pages 447–461, Toronto, Canada, 2008. Springer-Verlag, Berlin, Germany.
- [18] M. Hennessy and R. Milner. Algebraic laws for nondeterminism and concurrency. *J. ACM*, 32(1):137–161, 1985.
- [19] C. Hoare. *Communicating Sequential Processes*. Prentice-Hall International, Englewood Cliffs, 1985.
- [20] C. A. R. Hoare. Communicating sequential processes. *Commun. ACM*, 21(8):666–677, 1978.
- [21] R. Milner. *Communication and concurrency*. Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 1989.
- [22] M. Mousavi, M. Reniers, and J. F. Groote. A syntactic commutativity format for SOS. *Information Processing Letters*, 93:217–223, Mar. 2005.
- [23] M. R. Mousavi, M. A. Reniers, and J. F. Groote. SOS formats and meta-theory: 20 years after. *Theor. Comput. Sci.*, 373(3):238–272, 2007.
- [24] D. Park. Concurrency and automata on infinite sequences. In *Proceedings of the 5th GI-Conference on Theoretical Computer Science*, pages 167–183, London, UK, 1981. Springer-Verlag.
- [25] G. D. Plotkin. A Structural Approach to Operational Semantics. Technical Report DAIMI FN-19, University of Aarhus, 1981.
- [26] G. D. Plotkin. A powerdomain for countable non-determinism (extended abstract). In M. Nielsen and E. M. Schmidt, editors, *Automata, Languages and Programming, 9th Colloquium, Aarhus, Denmark, July 12-16, 1982, Proceedings*, volume 140 of *Lecture Notes in Computer Science*, pages 418–428. Springer, 1982.
- [27] G. D. Plotkin. A structural approach to operational semantics. *J. Log. Algebr. Program.*, 60-61:17–139, 2004.
- [28] T. Przymusiński. The well-founded semantics coincides with the three-valued stable semantics. *Fundamenta Informaticae*, 13(4):445–463, 1990.
- [29] D. Sangiorgi and D. Walker. *The Pi-calculus: A Theory of Mobile Processes*. Cambridge University Press, Cambridge, UK, 2001.

- [30] C. Verhoef. A congruence theorem for structured operational semantics with predicates and negative premises. *Nordic Journal on Computing*, 2(2):274–302, 1995.

### Appendix A. Proof of Theorem 3

We prove that  $\cong$  is a bisimulation relation. The claim then follows since  $f(c, p) \cong p$  and  $g(p, c') \cong p$  for each closed term  $p$ ,  $(f, c) \in L$  and  $(g, c') \in R$ . We prove this statement by an induction on the definition of  $\cong$ . The cases that  $p \cong q$  is due to reflexivity, symmetry and transitivity of  $\cong$  are trivial. So, two relevant cases remain to be proven.

1. Suppose that  $p \cong q$  is due to  $q \equiv f(c, p)$  for some  $(f, c) \in L$ .
  - (a) Assume that  $p \xrightarrow{a} p'$ , for some  $p' \in \mathbb{C}(\Sigma)$ . We shall show that there exists a  $p'' \in \mathbb{C}(\Sigma)$  such that  $f(c, p) \xrightarrow{a} p''$  and  $p' \cong p''$ .  
From condition 1a in Definition 12, we have that there exists a deduction rule of the following form

$$\frac{\{x_0 \xrightarrow{a_i} y_i \mid i \in I\} \cup \{P_k x_0 \mid k \in K\} \cup \{x_0 \xrightarrow{a_j} \text{ or } \neg P_j x_0 \mid j \in J\} \cup \{x_1 \xrightarrow{a} z_1\}}{f(x_0, x_1) \xrightarrow{a} t'}$$

where

- i. the variables  $y_i, z_1, x_0$  and  $x_1$  are all pairwise distinct,
- ii. for each  $j \in J$ , there is no  $c$ -defining deduction rule with  $a_j$  or  $P_j$  as label (depending on whether the formula with index  $j$  is a transition or a predicate formula),
- iii. there exists a collection  $\{P_k c \mid k \in K\}$  of  $c$ -defining axioms, and
- iv. there exists a collection  $\{c \xrightarrow{a_i} q_i \mid i \in I\}$  of  $c$ -defining axioms such that  $\sigma(t') \cong z_1$ , where  $\sigma$  is the substitution mapping  $x_0$  to  $c$ , each  $y_i$  to  $q_i$ ,  $i \in I$ , and is the identity on all the other variables.

Define a substitution  $\sigma'$  such that  $\sigma'(x_0) \equiv c$ ,  $\sigma'(x_1) \equiv p$ ,  $\sigma'(z_1) \equiv p'$  and  $\sigma'(y_i) \equiv \sigma(y_i)$  for each  $i \in I$ . Note that  $\sigma'$  satisfies all the premises in the above-mentioned rule. Therefore a proof tree for  $q \equiv f(c, p) \xrightarrow{a} \sigma'(t')$  is completed. Since  $\sigma(t') \cong z_1$ , the last claim in Lemma 1 yields that  $\sigma'(t') \cong \sigma'(z_1) \equiv p'$ . Hence,  $p' \cong \sigma'(t')$  and we are done.

- (b) Assume that  $q \xrightarrow{a} q'$ , for some  $q' \in \mathbb{C}(\Sigma)$ .

By the proviso of the theorem, the transition  $q \equiv f(c, p) \xrightarrow{a} q'$  must be proved using an  $f$ -defining rule. Therefore, it follows from condition 1c in Definition 12 that the transition of  $q \equiv f(c, p) \xrightarrow{a} q'$  is due to a deduction rule of the following form

$$\frac{\Phi}{f(t_0, t_1) \xrightarrow{a} t'}$$

and a closed substitution  $\sigma$  such that  $\sigma(t_0) = c$ ,  $\sigma(t_1) = p$ ,  $\sigma(t') = q'$  and  $\sigma$  satisfies  $\Phi$ .

Since  $\sigma$  satisfies  $\Phi$  and  $\sigma(t_0) \equiv c$ , item 1(c)ii of Definition 12 cannot apply. (Otherwise, either item 1(c)iiA holds, which means that there exists some  $\phi \in \Phi$  such that  $\sigma(\phi)$  is not provable, or item 1(c)iiB holds, which means that the collection of provable transitions contradicts  $\sigma(\phi)$ . Any of these two implies that  $\sigma(f(t_0, t_1) \xrightarrow{a} t')$  is not provable using the above-given rule and the substitution  $\sigma$ .)

Thus according to item 1(c)i of Definition 12, there exists some  $\phi \equiv t_1 \xrightarrow{a} t'' \in \Phi$  such that  $q' \equiv \sigma(t') \cong \sigma(t'')$ . By applying  $\sigma$  to  $\phi$ , we obtain  $p \equiv \sigma(t_1) \xrightarrow{a} \sigma(t'')$  and we are done.

- (c) Assume that  $Pp$  for some predicate symbol  $P \in \mathcal{P}$ .

We proceed to show that  $Pq$  also holds. Since  $(f, c) \in L$ , condition 1b in Definition 12 yields the presence of a rule of the form

$$\frac{\{P_i x_0 \mid i \in I\} \cup \{\neg P_j x_0 \mid j \in J\} \cup \{P x_1\}}{P f(x_0, x_1)}$$

where

- i. for each  $j \in J$ , there is no  $c$ -defining deduction rule with  $P_j$  as label,
- ii. there exists a collection  $\{P_i c \mid i \in I\}$  of  $c$ -defining axioms.

Since  $Pp$  by assumption, the substitution instance of the above rule associated with any closed substitution  $\sigma$  mapping  $x_0$  to  $c$  and  $x_1$  to  $p$  proves  $P f(c, p)$ . Since  $q \equiv f(c, p)$  we have that  $Pq$ , which was to be shown.

- (d) Assume that  $Pq$  for some predicate symbol  $P \in \mathcal{P}$ .

By the proviso of the theorem, the statement  $Pp$  must be proved using an  $f$ -defining rule. Therefore, it follows from condition 1c in Definition 12 that  $Pp$  is due to a deduction rule of the following form

$$\frac{\Phi}{P f(t_0, t_1)}$$

and a closed substitution  $\sigma$  such that  $\sigma(t_0) \equiv c$ ,  $\sigma(t_1) \equiv p$ , and  $\sigma$  satisfies  $\Phi$ . Since  $\sigma$  satisfies  $\Phi$  and  $\sigma(t_0) \equiv c$ , item 1(c)ii of Definition 12 cannot apply. Therefore, condition 1(b)i yields that  $P t_1 \in \Phi$ . Thus  $Pp$  holds, as  $\sigma(t_1) \equiv p$  and  $\sigma$  satisfies  $\Phi$ .

2. Suppose that  $p \cong q$  is due to  $q \equiv g(p, c)$  for some  $(g, c) \in R$ .

This case is similar to the previous case and we omit the details.  $\square$

## Appendix B. Proof of Theorem 5

The proof will rely on the following lemma, which can be shown by a straightforward induction on the definition of  $\cong_0$ .

**Lemma 4.** For all  $s, t \in \mathbb{T}(\Sigma)$ , if  $s \cong_0 t$  then  $\sigma(s) \cong_0 \sigma(t)$ , for each substitution  $\sigma$ .

From Lemma 4, it trivially follows that, when  $t$  is a closed term,  $s \cong_0 t$  implies  $\sigma(s) \cong_0 t$  for each substitution  $\sigma$ . In the proof of Theorem 5 given below we make use of this observation.

PROOF. (of Theorem 5)

We prove that  $\cong_0$  is a bisimulation relation. The claim then follows since  $f(c, p) \cong_0 p$  and  $g(p, c') \cong_0 p$  for each closed term  $p$ ,  $(f, c) \in L$  and  $(g, c') \in R$ . In order to show that  $\cong_0$  is a bisimulation it suffices to prove that whenever  $p \cong_0 q$  then

- if  $p \xrightarrow{a} p'$  then  $q \xrightarrow{a} q'$  for some  $q'$  such that  $p' \cong_0 q'$ ,
- if  $P p$  then  $P q$ , for each predicate  $P$ ,
- if  $q \xrightarrow{a} q'$  then  $p \xrightarrow{a} p'$  for some  $p'$  such that  $p' \cong_0 q'$ , and
- if  $P q$  then  $P p$ , for each predicate  $P$ .

We prove these statements by an induction on the definition of  $\cong_0$ . The cases that  $p \cong_0 q$  is due to reflexivity, symmetry and transitivity of  $\cong_0$  are trivial or follow easily using the inductive hypothesis. So, two relevant cases remain to be proved.

1. Suppose that  $p \cong_0 q$  is due to  $p \equiv c$  and  $q \equiv f(c, p)$  for some  $(f, c) \in L$ .
  - (a) Assume that  $c \xrightarrow{a} p'$ , for some  $p' \in \mathbb{C}(\Sigma)$ . This is because there exists an axiom  $c \xrightarrow{a} p'$ . We shall show that there exists a  $p'' \in \mathbb{C}(\Sigma)$  such that  $q \equiv f(c, p) \xrightarrow{a} p''$  and  $p' \cong_0 p''$ .

By Definition 14, we have a deduction rule of the following form

$$\frac{\{x_0 \xrightarrow{a_i} t_i \mid i \in I\} \cup \{P_k x_0 \mid k \in K\} \cup \{x_0 \xrightarrow{a_j} \text{ or } \neg P_j x_0 \mid j \in J\}}{f(x_0, x_1) \xrightarrow{a} t'}$$

where

- i.  $x_1 \notin \{x_0\} \cup \bigcup_{i \in I} \text{vars}(t_i)$ ,
- ii. for each  $j \in J$ , there is no  $c$ -defining deduction rule with  $a_j$  or  $P_j$  as label (depending on whether the formula with index  $j$  is a transition or a predicate formula),
- iii. there exists a collection  $\{P_k c \mid k \in K\}$  of  $c$ -defining axioms, and
- iv. there is a substitution  $\sigma$  such that  $\sigma(x_0) = c$ ,  $\{c \xrightarrow{a_i} \sigma(t_i) \mid i \in I\}$  is included in the collection of  $c$ -defining axioms, and  $\sigma(t') \cong_0 p'$ .

Since  $x_1 \notin \{x_0\} \cup \bigcup_{i \in I} \text{vars}(t_i)$ , we can extend  $\sigma$  to a closed substitution  $\sigma'$  mapping  $x_1$  to  $p$  and all the variables not contained in  $\{x_0, x_1\} \cup \bigcup_{i \in I} \text{vars}(t_i)$  to  $c$ . It is easy to see that the substitution  $\sigma'$  constructed in that fashion satisfies all the premises of the above rule. Thus,  $f(c, p) \xrightarrow{a} \sigma'(t')$  is a provable transition. As  $\sigma(t') \cong_0 p'$  by clause (iv) above and  $\sigma'(\sigma(t')) \equiv \sigma'(t')$  by construction, Lemma 4 yields that  $\sigma'(t') \cong_0 p'$ , and we are done.

- (b) Assume that  $q \equiv f(c, p) \xrightarrow{a} q' \in C$ , for some  $q' \in \mathbb{C}(\Sigma)$ . We shall show that there is some  $p' \in \mathbb{C}(\Sigma)$  such that  $c \xrightarrow{a} p'$  and  $p' \cong_0 q'$ .  
 By the proviso of the theorem, the transition  $q \equiv f(c, p) \xrightarrow{a} q'$  must be proved using an  $f$ -defining rule. Therefore, it follows from constraint 1(b) in Definition 14 that the transition  $q \equiv f(c, p) \xrightarrow{a} q'$  is due to a deduction rule of the following form

$$\frac{\Phi}{f(x_0, x_1) \xrightarrow{a} t'}$$

and a closed substitution  $\sigma$  such that  $\sigma(x_0) \equiv c$ ,  $\sigma(x_1) \equiv p$ ,  $\sigma(t') \equiv q'$  and  $\sigma$  satisfies  $\Phi$ .

Since  $\sigma$  satisfies  $\Phi$  and  $\sigma(x_0) \equiv c$ , then the condition 1(b)ii does not apply and we fall in the case of constraint 1(b)i. Thus, by the proviso of the clause, we can identify an axiom  $c \xrightarrow{a} p'$  for some  $p'$  such that  $p' \cong_0 \sigma(t')$ , and we are done.

- (c) The two cases involving predicates, namely when  $P c$  holds and  $P f(c, p)$  holds, follow the same lines.
2. Suppose that  $p \cong_0 q$  is due to  $p \equiv c$  and  $q \equiv g(p, c)$  for some  $(g, c) \in R$ .  
 This case is similar to the previous one and we omit the details.  $\square$

### Appendix C. Proof of Theorem 7

We prove that the relation  $\cong_0$  is a bisimulation. The claim then follows since  $f(c, p) \cong_0 c$  and  $g(p, c') \cong_0 c'$  for each closed term  $p$ ,  $(f, c) \in L$  and  $(g, c') \in R$ . To this end, we show that, when  $p \cong_0 q$ , the transfer conditions of Definition 6 are met by an induction on the definition of  $\cong_0$ . The cases that  $p \cong_0 q$  is due to reflexivity, symmetry and transitivity of  $\cong_0$  are trivial or follow easily by induction. So, two relevant cases remain to be considered.

1. Suppose that  $p \cong_0 q$  is due to  $p \equiv c$  and  $q \equiv f(c, q')$  for some  $(f, c) \in L$  and closed term  $q'$ .
- (a) Assume that  $c \xrightarrow{a} p'$ , for some  $p' \in \mathbb{C}(\Sigma)$ . This is because there exists an axiom  $c \xrightarrow{a} p'$ . We shall show that there exists a  $p'' \in \mathbb{C}(\Sigma)$  such that  $q \equiv f(c, q') \xrightarrow{a} p''$  and  $p' \cong_0 p''$ .  
 From constraint 1(a)ii in Definition 16, we have a non-empty set  $J$  of deduction rules of the following form

$$\frac{\Phi}{f(x_0, x_1) \xrightarrow{a} t'}$$

such that

- i.  $\models_G \text{True} \Rightarrow \text{hyps}(J)[x_0 \mapsto c]$ , and
- ii. for each rule in  $J$ , one of the following cases holds:

- A. there is some variable  $y \in \text{vars}(t')$  such that  $x_0 \xrightarrow{a} y \in \Phi$  and  $\sigma(t') \cong_0 p'$ , where  $\sigma$  is the substitution mapping  $x_0$  to  $c$ ,  $y$  to  $p'$  and is the identity on all the other variables, or
- B.  $\sigma(t') \cong_0 p'$ , where  $\sigma$  is the substitution mapping  $x_0$  to  $c$  and is the identity on all the other variables.

Let  $\sigma'$  be an arbitrary closed substitution mapping  $x_0$  to  $c$  and  $x_1$  to  $q'$ . Since  $\models_G \text{True} \Rightarrow \text{hyps}(J)[x_0 \mapsto c]$ , we have that  $\sigma'$  satisfies the formula  $\text{hyps}(\Phi)[x_0 \mapsto c]$ , where  $\Phi$  is the set of premises of some rule  $r$  in the set  $J$ . If, for this rule  $r$ , we are in case 1(a)iiA above, let  $\sigma''$  be the substitution that maps  $y$  to  $p'$  and acts like  $\sigma'$  on all the other variables. (In this case, the substitution  $\sigma''$  satisfies the premise  $x_0 \xrightarrow{b} y \in \Phi$ .) Otherwise, let  $\sigma'' = \sigma'$ . By Lemma 2, we can construct a substitution  $\sigma'''$  that

- ‘extends’  $\sigma''$  defined above,
- maps  $x_1$  to  $q'$  and
- satisfies  $\Phi$ .

Instantiating  $r$  with  $\sigma'''$  yields the transition  $q \equiv f(c, q') \xrightarrow{a} \sigma'''(t')$ . Since  $\sigma(t') \cong_0 p'$ , the term  $p'$  is a closed and  $\sigma'''$  ‘extends’  $\sigma$ , Lemma 4 yields that  $p' \cong_0 \sigma'''(t')$ , and we are done.

- (b) Assume that  $q \equiv f(c, q') \xrightarrow{a} p''$ , for some  $p'' \in \mathbb{C}(\Sigma)$ . We shall show that  $c \xrightarrow{a} p'$ , for some  $p' \in \mathbb{C}(\Sigma)$  such that  $p' \cong_0 p''$ .

It follows from constraint 1b in Definition 16 that the transition  $q \equiv f(c, q') \xrightarrow{a} p''$  is due to a deduction rule of the following form

$$\frac{\Phi}{f(x_0, x_1) \xrightarrow{a} t'}$$

and a closed substitution  $\sigma'$  such that  $\sigma'(x_0) \equiv c$ ,  $\sigma'(x_1) \equiv q'$ ,  $\sigma'(t') \equiv p''$  and  $\sigma'$  satisfies  $\Phi$ .

Since  $\sigma'$  satisfies  $\Phi$  and  $\sigma'(x_0) \equiv c$ , condition 1(b)ii in Definition 16 cannot apply and we fall in the case of constraint 1(a)ii. Thus we can find an axiom  $c \xrightarrow{a} p'$  and show that  $\sigma'(t') \cong_0 p'$  reasoning as in the case above.

- 2. Suppose that  $p \cong_0 q$  is due to  $q \equiv g(p, c)$  for some  $(g, c) \in R$ .

This proof is similar to the one for the previous case and we omit the details.  $\square$

## Appendix D. Proof of Theorem 9

We prove that  $\cong$  is a bisimulation relation. The claim then follows since  $f(c, p) \cong p$  and  $g(p, c') \cong p$  for each closed term  $p$ ,  $(f, c) \in L$  and  $(g, c') \in R$ . We prove that whenever  $p \cong q$  the transfer conditions of Definition 6 are met by an induction on the definition of  $\cong$ . The cases that  $p \cong q$  is due to reflexivity, symmetry and transitivity of  $\cong$  are trivial or follow easily by induction. So, two relevant cases remain to be proved.

1. Suppose that  $p \cong q$  is due to  $q \equiv f(c, p)$  for some  $(f, c) \in L$ .
  - (a) Assume that  $p \xrightarrow{a} p'$ , for some  $p' \in \mathbb{C}(\Sigma)$ . We shall show that there exists a  $p'' \in \mathbb{C}(\Sigma)$  such that  $q \equiv f(c, p) \xrightarrow{a} p''$  and  $p' \cong p''$ .  
From constraint 1a in Definition 17, we have that there exists a deduction rule of the following form

$$\frac{\Phi \cup \{x_1 \xrightarrow{a} y_1\}}{f(x_0, x_1) \xrightarrow{a} t'}$$

where

- i.  $\models_G x_1 \xrightarrow{a} \Rightarrow \text{hyps}(\Phi)[x_0 \mapsto c]$ , and
- ii. one of the following cases holds:
  - A. there are a premise  $x_0 \xrightarrow{b} y \in \Phi$ , for some  $b \in \mathcal{L}$  and  $y \in \text{vars}(t')$ , and an axiom  $c \xrightarrow{b} t$  such that  $\sigma(t') \cong y_1$ , where  $\sigma$  is the substitution mapping  $x_0$  to  $c$ ,  $y$  to  $t$  and is the identity on all the other variables, or
  - B.  $\sigma(t') \cong y_1$ , where  $\sigma$  is the substitution mapping  $x_0$  to  $c$  and is the identity on all the other variables.

We now show that there exists a closed substitution  $\sigma'$  such that  $\sigma'$  satisfies  $\Phi$ ,  $f(c, p) \xrightarrow{a} \sigma'(t')$  is a provable transition and  $\sigma'(t') \cong p'$ . Consider an arbitrary closed substitution  $\sigma''$  mapping  $x_0$  to  $c$ ,  $x_1$  to  $p$  and  $y_1$  to  $p'$  and not precisely specified elsewhere at the moment. Such a substitution  $\sigma''$  satisfies the premise  $x_1 \xrightarrow{a} y$ . If we are in case 1(a)iiA, let  $\sigma''(y) \equiv t$ , so that  $\sigma''$  also satisfies the premise  $x_0 \xrightarrow{b} y \in \Phi$ .

As  $\sigma''$  satisfies the premise  $x_1 \xrightarrow{a} y_1$ ,  $\sigma''(x_0) \equiv c$  and  $\models_G x_1 \xrightarrow{a} \Rightarrow \text{hyps}(\Phi)[x_0 \mapsto c]$ , we have that  $\rightarrow_G, \sigma'' \models \text{hyps}(\Phi)$ . Therefore Lemma 2 yields a closed substitution  $\sigma'$  such that

- $\sigma'(x_i) = \sigma''(x_i)$  for each  $i \in \{0, 1\}$ ,
- $\sigma'(y_1) = \sigma''(y_1) = p'$ ,
- $\sigma'(y) = \sigma''(y) = t$  if we are in case 1(a)iiA and
- $\sigma'$  satisfies  $\Phi$ .

Instantiating the rule

$$\frac{\Phi \cup \{x_1 \xrightarrow{a} y_1\}}{f(x_0, x_1) \xrightarrow{a} t'}$$

with such a closed substitution  $\sigma'$  yields the transition

$$\sigma'(f(x_0, x_1)) \equiv f(c, p) \xrightarrow{a} \sigma'(t') .$$

Recall that  $\sigma(t') \cong y_1$ , where  $\sigma$  is either the substitution defined in case 1(a)iiA or 1(a)iiB. In both cases, by Lemma 1, we have that

$$\sigma'(t') = \sigma'(\sigma(t')) \cong \sigma'(y_1) = p'$$

and we are done.

(b) Assume that  $q \equiv f(c, p) \xrightarrow{a} q'$ , for some  $q' \in \mathbb{C}(\Sigma)$ .

The transition  $q \equiv f(c, p) \xrightarrow{a} q'$  must be proved using an  $f$ -defining rule of the form

$$\frac{\Phi}{f(x_0, x_1) \xrightarrow{a} t'}$$

and a closed substitution  $\sigma'$  such that  $\sigma'(x_0) \equiv c$ ,  $\sigma'(x_1) \equiv p$ ,  $\sigma'(t') \equiv q'$  and  $\sigma'$  satisfies  $\Phi$ . Since  $\sigma'$  satisfies  $\Phi$  and  $\sigma'(x_0) \equiv c$ , condition 1(b)ii in Definition 17 cannot apply and we fall in the case of constraint 1(b)i. Thus  $x_1 \xrightarrow{a} y_1 \in \Phi$  for some variable  $y_1$ . As  $\sigma'$  satisfies  $\Phi$ , it follows that  $\sigma'(x_1) \equiv p \xrightarrow{a} \sigma'(y_1)$ . We claim that  $\sigma'(y_1) \cong q'$ . To see that this claim does hold true, recall that, since constraint 1(b)i in Definition 17 is met,

- i. either there is a premise  $x_0 \xrightarrow{b} y \in \Phi$ , for some  $b \in \mathcal{L}$  and variable  $y \in \text{vars}(t')$ , such that  $c$  has a single axiom with label  $b$ —say,  $c \xrightarrow{b} t$ —and  $\sigma(t') \cong y_1$ , where  $\sigma$  is the substitution mapping  $x_0$  to  $c$ ,  $y$  to  $t$  and is the identity on all the other variables,
- ii. or  $\sigma(t') \cong y_1$ , where  $\sigma$  is the substitution mapping  $x_0$  to  $c$  and is the identity on all the other variables.

In the former case,  $\sigma'$  satisfies the premise  $x_0 \xrightarrow{b} y \in \Phi$ . Therefore,  $\sigma'(x_0) \equiv c \xrightarrow{b} t \equiv \sigma'(y)$ , as  $c \xrightarrow{b} t$  is the only  $c$ -defining axiom with label  $b$ . By Lemma 1, since  $\sigma(t') \cong y_1$  holds, we have that

$$q' = \sigma'(t') = \sigma'(\sigma(t')) \cong \sigma'(y_1) = p'$$

and we are done.

The latter case is handled similarly.

2. Suppose that  $p \cong q$  is due to  $q \equiv g(p, c)$  for some  $(g, c) \in R$ .

This case is similar to the previous case and we omit the details. □