On rule formats for zero and unit elements

Citation for published version (APA):

Document status and date:
Published: 01/01/2010

Document Version:
Publisher’s PDF, also known as Version of Record (includes final page, issue and volume numbers)

Please check the document version of this publication:
- A submitted manuscript is the version of the article upon submission and before peer-review. There can be important differences between the submitted version and the official published version of record. People interested in the research are advised to contact the author for the final version of the publication, or visit the DOI to the publisher’s website.
- The final author version and the galley proof are versions of the publication after peer review.
- The final published version features the final layout of the paper including the volume, issue and page numbers.

Link to publication

General rights
Copyright and moral rights for the publications made accessible in the public portal are retained by the authors and/or other copyright owners and it is a condition of accessing publications that users recognise and abide by the legal requirements associated with these rights.

- Users may download and print one copy of any publication from the public portal for the purpose of private study or research.
- You may not further distribute the material or use it for any profit-making activity or commercial gain
- You may freely distribute the URL identifying the publication in the public portal.

If the publication is distributed under the terms of Article 25fa of the Dutch Copyright Act, indicated by the “Taverne” license above, please follow below link for the End User Agreement:
www.tue.nl/taverne

Take down policy
If you believe that this document breaches copyright please contact us at:
openaccess@tue.nl
providing details and we will investigate your claim.
On Rule Formats for Zero and Unit Elements*

Luca Aceto¹, Matteo Cimini¹, Anna Ingolfsdottir¹, MohammadReza Mousavi², and Michel A. Reniers²

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

Abstract. This paper proposes rule formats for Structural Operational Semantics guaranteeing that certain constants act as left or right zero elements for a set of binary operators. Our design approach is also applied to reformulate an earlier rule format for unit elements developed by some of the authors. Examples of left and right zero, as well as unit, elements from the literature are shown to be checkable using the provided formats.

1 Introduction

In the last three decades, Structural Operational Semantics (SOS), see, e.g., [3, 17, 20, 21], has been shown 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 collection of syntax-driven inference rules. Based on this semantics in terms of state transitions, we often want to prove 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, where it is important to check whether certain operators are, say, commutative and associative.

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 [4]. 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 [18], associativity [13] 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 results so obtained apply to a broad class of languages.

* 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.
Recently, some of the authors provided in [5] a rule format guaranteeing another basic algebraic property not addressed before: the existence of left and right unit elements for operators. In the present paper, we follow the work presented in [5] and we develop some rule formats guaranteeing instead that certain constants act as left or right zero elements for a set of binary operators. Namely, a function $f$ 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 constant $c$ satisfying the above equation(s) is also said to be absorbing for the operator $f$.

A classical example of a left zero element within the realm of process algebra is provided by the constant $\delta$, for deadlock, from BPA [10], which satisfies the laws:

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

where `$\cdot$' and `$\parallel$' stand for sequential composition and left merge, respectively.

The first format we provide follows the techniques developed in [5] and is of a syntactic nature. However, even though we show how several classical examples from the literature indeed fit the format, there are some basic, but somewhat more exotic, zero elements that cannot be handled by the proposed format.

We show nevertheless that we can reformulate our zero-element format within the GSOS languages of Bloom, Istrail and Meyer [12], 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.

The final part of the paper is devoted to applying the design ideas underlying the GSOS-based format for left and right zero elements to reformulate the format for left and right unit elements from [5]. The resulting format turns out to be incomparable in power to the original one, but it is expressive enough to check all the examples discussed in [5].

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 zero elements and Section 4 shows how several examples of left and right zero elements from the literature fit the format. In Section 5 we point out the main drawbacks of the format and in Section 6 we reformulate it within the GSOS format using the aforementioned logic of transition formulae. In Section 7 we provide a rule format for unit elements adapting the ideas from Section 6. We conclude the paper with an overview of its main contributions in Section 8. In order to increase the readability of the main body of the paper, the proofs of the main technical results have been collected in Appendices A–C.
2 Preliminaries

In this section we recall some standard definitions from the theory of SOS. We refer the readers to, e.g., [3] and [17] 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 $x, x', x_1, y, y', y_1, \ldots$ 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 $f, g, \ldots$. An operator with arity zero is called a constant. We define the set $T(\Sigma)$ 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 $t_1, \ldots, t_n$ are terms, then $f(t_1, \ldots, t_n)$ is a term.

We use $s, t$, 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 $\text{vars} : T(\Sigma) \to 2^V$ gives the set of variables appearing in a term. The set $C(\Sigma) \subseteq T(\Sigma)$ is the set of closed terms, i.e., terms that contain no variables. We use $p, q, p', p_1, \ldots$ to range over closed terms. A substitution $\sigma$ is a function of type $V \to T(\Sigma)$. 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 lies in $C(\Sigma)$, we say that it is a closed substitution.

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

- $\Sigma$ is a signature.
- $L$ is a set of labels (or actions) ranged over by $a, b, l$. If $l \in L$, and $t, t' \in T(\Sigma)$ we say that $t \xrightarrow{l} t'$ is a positive transition formula and $t \xrightarrow{l}$ 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 $\frac{\Phi}{\phi}$. An axiom is a deduction rule with an empty set of premises. 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 \( c \leftarrow p \) 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 \( N \phi \) with \( N \) a set of negative formulae. A TSS \( T \) proves \( N \phi \), denoted by \( T \vdash N \phi \), 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 \( K \psi \) is a substitution instance of a deduction rule in \( T \).

**Definition 4 (Contradiction and entailment)** The formula \( t \leftarrow t' \) is said to contradict \( t \rightarrow t' \), 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 \( \Phi \models \Psi \) 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 \( \Phi \) entails a set \( \Psi \) of formulae, when there exists a substitution \( \sigma \) such that, for each \( \psi \in \Psi \), there exists a \( \phi \in \Phi \) such that \( \sigma(\phi) \equiv \psi \). In other words, \( \Phi \) entails \( \Psi \) if there is a substitution \( \sigma \) such that \( \Psi \subseteq \{ \sigma(\phi) \mid \phi \in \Phi \} \).

It immediately follows from the above definition that contradiction is a symmetric relation on (sets of) formulae. We now have all the necessary ingredients to define the semantics of TSSs in terms of three-valued stable models.

**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 \( T \) when the following conditions hold:

- for each \( \phi \in C \), there is a set \( N \) of negative formulae such that \( T \vdash N \phi \) and \( C \cup U \models N \) and
- for each \( \phi \in U \), there is a set \( N \) of negative formulae such that \( T \vdash N \phi \) and \( C \models N \).

\( C \) 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 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 \(T\) 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 \(C \models \sigma(\phi)\), for each negative formula \(\phi \in \Phi\). If a TSS is complete, we often also write \(p \rightarrow p'\) in lieu of \((p \rightarrow p') \in C\).

Definition 6 (Bisimulation and bisimilarity [16, 19]) Let \(T\) be a transition system specification with signature \(\Sigma\) and label set \(L\). A relation \(R \subseteq \mathcal{C}(\Sigma) \times \mathcal{C}(\Sigma)\) is a bisimulation relation if and only if \(R\) is symmetric and, for all \(p_0, p_1, p'_0 \in \mathcal{C}(\Sigma)\) and \(l \in L\),

\[
(p_0 R p_1 \land T \vdash p_0 \xrightarrow{l} p'_0) \Rightarrow \exists p'_1 \in \mathcal{C}(\Sigma). (T \vdash p_1 \xrightarrow{l} p'_1 \land p'_0 R p'_1).
\]

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

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

In Sections 6–7 of the paper, we focus on the GSOS format of Bloom, Istrail and Meyer [12], 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:

\[
\bigcup_{i=1}^{l} \left\{ x_i \xrightarrow{a_{ij}} y_{ij} \mid 1 \leq j \leq m_i \right\} \cup \bigcup_{i=1}^{l} \left\{ x_i \xrightarrow{b_{ik}} \mid 1 \leq k \leq n_i \right\} \xrightarrow{f(x_1, \ldots, x_l)} t
\]

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 \(l\), and \(t\) is a term in \(T(\Sigma)\) such that \(\text{vars}(t) \subseteq \{x_1, \ldots, x_l\} \cup \{y_{ij} \mid 1 \leq i \leq l, 1 \leq j \leq m_i\}\).

Definition 8 A GSOS language is a triple \(G = (\Sigma_G, L, R_G)\), where \(\Sigma_G\) is a finite signature, \(L\) 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 proceed to introduce the notion of TSS extended with predicates.
**Definition 9 (Predicates)** Given a set $\mathcal{P}$ of predicate symbols, $P \, t$ is a positive predicate formula and $\neg P \, t$ is a negative predicate formula, for each $P \in \mathcal{P}$ and $t \in T(\Sigma)$. 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 Rule format

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 technique developed by some of the authors in [5] for left or right unit elements.

As in [5], we make use of an equivalence relation between terms called zero-context equivalence, which is the counterpart of the unit-context equivalence from [5]. 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 10 (Zero-context equivalent terms)** Given sets $L, R \subseteq \Sigma \times \Sigma$ of pairs of binary function symbols and constants, $\cong_0$ is the smallest equivalence relation satisfying the following constraints, for each $s \in T(\Sigma)$:

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

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

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 $\cong_0^L \times \cong_0^R$.

**Theorem 11 (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 $t, u \in T(\Sigma)$, it is decidable whether $t \cong_0^L \, u$ holds.
Proof. Let $L$ and $R$ be given. Suppose the 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 $R$,
- all terms of the form $f(c, d)$ with $(f, c) \in L$ and $(f, d) \in R$, and

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 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 10. For the converse, we proceed by induction on the length of a shortest proof of $t \equiv_0 u$. If $t \equiv_0 u$ 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 $t \equiv_0 u$ is proven using symmetry then the claim follows by the inductive hypothesis. Assume now that $t \equiv_0 u$ is a shortest proof of $t \equiv_0 u$. Then there is some term $s$ such that $t \equiv_0 s$ (in one step) and $s \equiv_0 u$. 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 $s \equiv_0 u$ could only proceed in the next step by going back to $t = c$, contradicting our assumption that it was a shortest proof of $t \equiv_0 u$.

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.\[\qed\]

We now proceed to define the rule format for left and right zero elements, which is the first 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 1. Assume that \( a \) is the only action. Let 0 be a constant with deduction rule

\[
0 \mapsto 0
\]

Furthermore consider the binary operators \( \llbracket n \rrbracket_\_ \) for \( n \geq 0 \), with deduction rules

\[
\begin{align*}
  x & \mapsto x' \\
  x & \mp x' \\
  x[0]y & \mapsto x' \\
  x[n+1]y & \mapsto x'[n]y \\
  x[n]y & \mp x'[n]y \\
  x[n]y & \mp y
\end{align*}
\]

Assuming that the transition relation \( \mp \) denotes unit time steps, \( p[\llbracket n \rrbracket_\_] q \) 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 \( \llbracket n \rrbracket_\_ \) one needs to show also that it is a left zero element for all operators \( \llbracket i \rrbracket_\_ \) with \( 0 \leq i < n \). It is not hard to see that the relation

\[
R_n = \{(0[i]p,0) \mid 0 \leq i \leq n, \ p \in C(\Sigma)\} \cup \{(0,0)\}
\]

is a bisimulation. Therefore, 0 is a left zero element for the operator \( \llbracket n \rrbracket_\_ \).

\[\square\]

In the previous example, the zero element property for \( \llbracket n \rrbracket_\_ \) depends on that property for all \( \llbracket i \rrbracket_\_ \) with \( 0 \leq i < n \). The next example illustrates that this dependency can even be worse.

Example 2. Assume that \( a \) is the only action and consider the binary operators \( f_i, i \geq 0 \), with rules

\[
\begin{align*}
  x & \mapsto y \\
  f_i(x, x) & \mp f_i+1(y, x)
\end{align*}
\]

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

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

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, such a set cannot be inductively defined since, 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 \textit{well-founded}.

\[\square\]

Example 3. Consider the following TSS with constant \( \text{RUN}_a \) and binary function symbols \( f \) and \( g \) with rules

\[
\begin{align*}
  x & \mapsto y \\
  f(x, x) & \mp g(x, y) \\
  g(x, x) & \mp f(y, x)
\end{align*}
\]

8
It is not hard to see that \( f(RUN_a, p) \leftrightarrow RUN_a \leftrightarrow g(p, RUN_a) \), for each closed term \( p \). Therefore \( RUN_a \) 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 [5], in the following definition we talk about left- and right-aligned pairs.

**Definition 12 (Left- and right-aligned pairs)** Given a TSS with set of predicate symbols \( P \) and set of labels \( L \), 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 \( c \overset{a}{\to} t \) (or \( P c \)) does exist then there is a rule:

   \[
   \frac{\{x_0 \overset{a}{\to} t_i \mid i \in I\} \cup \{P_k x_0 \mid k \in K\} \cup \{x_0 \overset{a}{\to} \text{ or } \neg P_j x_0 \mid j \in J\}}{f(x_0, x_1) \overset{a}{\to} t'} \quad \text{(or } P f(x_0, x_1))
   \]

   where
   \( i. \ x_1 \notin \{x_0\} \cup \bigcup_{i \in I} \text{vars}(t_i) \),
   \( ii. \ \text{for each } j \in J, \text{ there is no } c\text{-defining axiom with } a_j \text{ or } P_j \text{ as label (depending on whether the formula with index } j \text{ is a transition or a predicate formula)}, \)
   \( iii. \ \text{there exists a collection } \{P_k c \mid k \in K\} \text{ of } c\text{-defining axioms, and} \)
   \( iv. \ \text{there exists some substitution } \sigma \text{ such that } \sigma(x_0) = c, \{c \overset{a}{\to} \sigma(t_i) \mid i \in I\} \text{ is included in the collection of } c\text{-defining axioms, and if the conclusion is a transition formula, } \sigma(t') \equiv_0 t. \)

   (b) Each \( f\)-defining deduction rule has one of the following forms:

   \[
   \frac{\Phi}{f(t_0, t_1) \overset{a}{\to} t'} \quad \text{or} \quad \frac{\Phi}{P f(t_0, t_1)}
   \]

   where \( a \in L, P \in P \) and, for each closed substitution \( \sigma \) such that \( \sigma(t_0) \equiv c \), one of the following cases holds:
   \( i. \ \text{there exists an axiom } c \overset{a}{\to} t \text{ with } \sigma(t') \equiv_0 t \) (if the conclusion is a transition formula), or an axiom \( P c \) (if the conclusion is a predicate formula), or
   \( ii. \ \text{there exists a premise } \phi \in \Phi \text{ with } t_0 \text{ as its source such that} \)
   \( A. \ \text{either } \phi \text{ is a positive formula and the collection of } c\text{-defining axioms does not entail } \sigma(\phi), \text{ or} \)
   \( B. \ \phi \text{ is a negative formula and the collection of } c\text{-defining axioms contradicts } \sigma(\phi). \)

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 inherited directly from [5], 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 [5], the sets $L$ and $R$ are defined as the largest sets of pairs satisfying the constraints from Definition 12. 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 12.

**Theorem 13** Let $T$ 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 12. For each $(f, c) \in L$, it holds that $f(c, x) \equiv c$. Symmetrically, for each $(f, c) \in R$, it holds that $f(x, c) \equiv c$.

Proof. Appendix A contains the proof of Theorem 13. ⊠

**Example 4.** Consider Example 2. We now show that RUN$_a$ is a left zero element for each $f_i$ using Theorem 13. 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 $\sigma(f_{i+1}(y_0, x_1)) \equiv f_{i+1}(\text{RUN}_a, \sigma(x_1)) \equiv_0 \text{RUN}_a$, 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 $\sigma(f_{i+1}(y_0, x_1)) \not\equiv_0 \text{RUN}_a$, and therefore case 1(b)i does not apply. This means that $\sigma(y_0)$ is not $\text{RUN}_a$ and therefore the condition in case 1(b)iiA is met. ⊠

**Example 5.** Consider now Example 3. We show that RUN$_a$ is a left zero element for $f$ and a right zero element for $g$ using Theorem 13. 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 $L$. 

10
– 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 12 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 12 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 13. 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 12 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

$$
x_1 \not\in \bigcup_{i \in I} \text{vars}(t_i)
$$

in condition 1(a)ii would also invalidate Theorem 13. 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 12 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 6. Consider the TSS with constants 0 and \( a\&b \), and a binary operator \( f \) with rules:

\[
\begin{array}{ccc}
\text{0} & \text{0} \\
\text{a\&b} & \text{0} & \text{0} \\
\text{f(x, y)} & \text{a\&b} & \text{0}
\end{array}
\]

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 \( f(a\&b, 0) \) affords no transition unlike \( a\&b \). Therefore \( a\&b \) is not a left zero element for \( f \).

Example 7. Consider the TSS over set of labels \( \{ a, b \} \) with constant \( \text{RUN}_a \) and a binary operator \( f \) with rule:

\[
\begin{array}{ccc}
x_0 \xrightarrow{a} y_0 & x_0 \xrightarrow{b} y_1 \\
f(x_0, x_1) \xrightarrow{a} y_0 & f(x_0, x_1) \xrightarrow{b} y_0
\end{array}
\]

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 11 to follow, constraint 1(b)ii 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 12, 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 \cite{7}, 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 \cite{6}, which is a left zero element for sequential composition.

4 Examples

In this section we show that several examples of zero elements from the literature indeed fit the format described in Section 3.

Example 8 (Synchronous parallel composition). Consider the synchronous parallel composition from CSP \cite{15} over a set of actions \( \mathcal{L} \) with rules:

\[
\begin{array}{ccc}
\xrightarrow{a} x' & \xrightarrow{a} y' \\
x \parallel_{\mathcal{L}} y \parallel \mathcal{L} x' & y \parallel_{\mathcal{L}} y'
\end{array}
\]

We know that the inaction constant \( 0 \), with no rules, is a left and right zero element for \( \parallel_{\mathcal{L}} \). Let \( L = R = \{ (\parallel_{\mathcal{L}}, 0) \} \). We claim that \( L \) and \( R \) meet the
constraints in Definition 12. First of all, \( 0 \) 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 \( \sigma(x) \equiv 0 \), 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
\[
0 \parallel_L y \leadsto 0 \quad \text{and} \quad x \parallel_L 0 \leadsto 0
\]
thus follow from Theorem 13.

Example 9 (Left merge operator). Consider the left merge operator from [10].
\[
x \xrightarrow{a} x' \\
x \parallel y \xrightarrow{a} x' \parallel y
\]
Here \( \parallel \) stands for the merge operator from [10], whose SOS specification is immaterial for this example; see Example 13 to follow. Let \( L = \{ (\parallel, 0) \} \) and \( R = \emptyset \). We claim that \( L \) meets the constraints in Definition 12. It is easy to check that the claim is true by the same reasoning used in Example 8. This time it is sufficient to check conditions 1a and 1b because \( 0 \) is just a left zero element for \( \parallel \). By Theorem 13 the validity of the law \( (0 \parallel y) \leadsto 0 \) follows. Note that the pair \( \{ (\parallel, 0) \} \) cannot be added to \( R \) because the symmetric version of condition 1b would be violated. Indeed \( 0 \) is not a right zero element for \( \parallel \).

Example 10 (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.)
\[
\begin{align*}
\text{(seq1)} \quad & x \xrightarrow{a} x' \\
& x \cdot y \xrightarrow{a} x' \cdot y \\
\text{(seq2)} \quad & x \downarrow \xrightarrow{a} y' \\
& x \cdot y \xrightarrow{a} y' \\
\text{(seq3)} \quad & x \downarrow \quad y \downarrow \\
& (x \cdot y) \downarrow
\end{align*}
\]
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 12. 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 13, the validity of the well-known law \( \delta \cdot y \equiv \delta \) 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 12. Indeed \( \delta \) is not a right zero element for \( \cdot \).
Example 11 (Sequential Composition (2)). Focusing again on the sequential composition operator from the previous example, consider once more the constant RUN\textsubscript{a} from Example 2 with axiom

\[
\text{RUN}_{a} \xrightarrow{a} \text{RUN}_{a}.
\]

This constant simply displays \textit{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 RUN\textsubscript{a} \cdot y \Leftrightarrow \text{RUN}_{a} with \( a \in \mathcal{L} \) using Theorem 13.

Let \( L = \{(\cdot, \text{RUN}_{a})\} \) and \( R = \emptyset \). We claim that \( L \) meets the constraints in Definition 12. 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 \textit{a}. The substitution \( \sigma \) constructed in order to meet the requirements in condition 1(a)iv is such that \( \sigma(\textit{x}) \equiv \text{RUN}_{a} \) and \( \sigma(\textit{x}') \equiv \text{RUN}_{a} \). Moreover, \text{RUN}_{a} is zero-context equivalent to \text{RUN}_{a} \cdot \textit{y} and we are done.

- 1b: Since \text{RUN}_{a} \downarrow \textit{0} does not hold, with the rules (seq2) and (seq3) we fall in the subcase 1(b)iA. 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 \textit{c} \cdot \textit{y} \Leftrightarrow \textit{c}, where \textit{c} is any constant whose behaviour is defined by a collection of axioms of the form \( \{c \xrightarrow{a} \textit{c} \mid i \in I\} \), where \( I \) is any index set.

Example 12 (Predictable failure constant of BPA\textsubscript{0δ}). In this example we focus on the language BPA\textsubscript{0δ} of Baeten and Bergstra—see [8]. 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 \textit{x} \cdot 0 \Leftrightarrow 0 and 0 \cdot \textit{x} \Leftrightarrow 0 both hold. The following SOS rules for the language BPA\textsubscript{0δ} make use of the predicate \( \not= 0 \) 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 \textit{a} action.

\[
\begin{align*}
\frac{\textit{a} \not= 0}{\textit{x} \not= 0} & \quad \frac{\delta \not= 0}{\textit{x} \not= 0} \quad \frac{\textit{a} \xrightarrow{a} \checkmark}{\textit{a} \xrightarrow{a} \checkmark} \quad \frac{\textit{x} \not= 0}{\textit{x} + \textit{y} \not= 0} \quad \frac{\textit{y} \not= 0}{\textit{x} + \text{y} \not= 0} \\
\frac{\textit{x} \xrightarrow{a} \textit{x}'}{\textit{x} + \textit{y} \xrightarrow{a} \textit{x}'} & \quad \frac{\textit{x} \xrightarrow{a} \checkmark}{\textit{x} + \textit{y} \xrightarrow{a} \checkmark} \quad \frac{\textit{y} \xrightarrow{a} \textit{y}'}{\textit{x} + \text{y} \xrightarrow{a} \textit{y}'} & \quad \frac{\textit{y} \xrightarrow{a} \checkmark}{\textit{x} + \text{y} \xrightarrow{a} \checkmark} \\
\frac{\textit{x} \not= 0 \quad \textit{y} \not= 0}{\textit{x} \cdot \textit{y} \not= 0} & \quad \frac{\textit{x} \xrightarrow{a} \textit{x}'} \quad \frac{\textit{y} \not= 0}{\textit{x} \cdot \textit{y} \xrightarrow{a} \textit{x}' \cdot \textit{y}} \quad \frac{\textit{x} \xrightarrow{a} \checkmark}{\textit{x} \cdot \textit{y} \xrightarrow{a} \checkmark} \quad \frac{\textit{y} \xrightarrow{a} \checkmark}{\textit{x} \cdot \textit{y} \xrightarrow{a} \checkmark}
\end{align*}
\]

Let \( L = R = \{(\cdot, 0)\} \). We claim that \( L \) and \( R \) meet the constraints in Definition 12. 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 · one by one. Since $0 \neq 0$ does not hold we fall into case 1(b)ii with the leftmost rule. Since $0 \not\xrightarrow{a}$, and $0 \not\xrightarrow{a}$ ✓ 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 13.

Example 13 (Merge operator). Let $\mathcal{L}$ be the set of actions. Consider the classic merge operator $\parallel$ with the following rules, where $a \in \mathcal{L}$.

\[
\begin{align*}
\frac{x \xrightarrow{a} x'}{x \parallel y \xrightarrow{a} x' \parallel y} & \quad \frac{y \xrightarrow{a} y'}{x \parallel y \xrightarrow{a} x \parallel y'}
\end{align*}
\]

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 13. Indeed, let $L = R = \{ (\parallel, \text{RUN}_\mathcal{L}) \}$. It is easy to see that condition 1a in Definition 12 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 14 (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 [16], 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:

\[
\begin{align*}
\frac{x \xrightarrow{a} x'}{x + y \xrightarrow{a} x'} & \quad \frac{y \xrightarrow{a} y'}{x + y \xrightarrow{a} y'}
\end{align*}
\]

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 13, we are able to prove the validity of the law $x + c \xrightarrow{a} c$. We leave the details to the reader. The operator studied in this example bears resemblance with the preferential choice operator $\rightarrow$ from [11].

5 Discussion of the format

The format for left and right zero elements we presented in Definition 12 is based on rather intuitive design decisions and, as witnessed by the examples discussed
in Section 4, 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 12. This discussion will motivate the development of an alternative format for left and right zero elements that we present in Section 6 to follow.

5.1 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 \(f\) proving an \(a\)-labelled transition does fire (leading to a term that is bisimilar to \(t\)). The way we guarantee this property stems from [5], 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 \(c\)-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 fireability/unfireability of the rules in question, when the first argument of the operator \(f\) is the considered left zero element. Premises about 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 12. However, this choice does not allow us to handle left zero elements such as the one in the following example.

**Example 15.** Consider a TSS with constants \(0\) and \(\text{RUN}_a\) (from Example 2), and a function symbol \(f\) defined as follows

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

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 12.
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 16.** Assume that $a$ is the only action. Consider the TSS with constant $\text{RUN}_a$ from Example 2 and binary operator $f$ with rule

$$
\begin{array}{c}
x \xrightarrow{a} x' \\
y \xrightarrow{a} y'
\end{array} \quad \Rightarrow \\
\begin{array}{c}
f(x, y) \xrightarrow{a} f(x', y')
\end{array}
$$

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. \(\Box\)

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 6.2, 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.

### 5.2 Checking the format, algorithmically

We are aware that checking the requirements in Definition 12 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 [5].

Even though the requirements of the proposed format are frequently easy to check in practice, as the examples in Section 4 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 4.

### 6 A rule format for zero elements based on GSOS

In what follows, we adapt the format from Section 3 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 5.1 and 5.2.
6.1 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 $L$ is defined by the following grammar, where $a \in L$:

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

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

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

$$
\begin{align*}
\rightarrow_G, \sigma &\models \text{True} \quad \text{always} \\
\rightarrow_G, \sigma &\models x \xrightarrow{a} \iff \sigma(x) \xrightarrow{a} p, \text{ for some } p \\
\rightarrow_G, \sigma &\models \neg F \iff \text{not } \rightarrow_G, \sigma \models F \\
\rightarrow_G, \sigma &\models F \land F' \iff \rightarrow_G, \sigma \models F \text{ and } \rightarrow_G, \sigma \models F' .
\end{align*}
$$

The reader familiar with Hennessy-Milner logic [14] 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 $\land$.

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,

$\text{hyps}(\emptyset) = \text{True}$

$\text{hyps}(\{ x \xrightarrow{a} y \} \cup \Phi) = (x \xrightarrow{a}) \land \text{hyps}(\Phi \setminus \{ x \xrightarrow{a} y \})$

$\text{hyps}(\{ x \not\xrightarrow{a} \} \cup \Phi) = \neg(x \xrightarrow{a}) \land \text{hyps}(\Phi \setminus \{ x \xrightarrow{a} \})$ .

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,

$$
\text{hyps}(\{ x \xrightarrow{a} y, z \xrightarrow{b} \}) = (x \xrightarrow{a}) \land \neg(z \xrightarrow{b}) .
$$

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 14 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 5. Let $\sigma$ be a closed substitution such that $\rightarrow_G, \sigma \models \hyps(\Phi)$ and $\sigma$ satisfies $\Phi_1$. Then there is a closed substitution $\sigma'$ such that

- $\sigma'(x_i) = \sigma(x_i)$ for each $i \in \{1, \ldots, l\}$,
- $\sigma'(y) = \sigma(y)$ 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 $x_i \overset{b}{\rightarrow}$, for some $i \in \{1, \ldots, l\}$ and label $b$, then $\neg(x_i \overset{a}{\rightarrow})$ is a conjunct of $\hyps(\Phi)$. As $\rightarrow_G, \sigma \models \hyps(\Phi)$, we have that $\sigma$ satisfies $x_i \overset{b}{\rightarrow}$. Therefore $\sigma$ satisfies $\Phi_1 \cup \{x_i \overset{b}{\rightarrow}\}$ and the existence of a substitution $\sigma'$ meeting the requirements stated in the lemma follows by induction applied to $\Phi_2 \setminus \{x_i \overset{b}{\rightarrow}\}$.

Consider now the case that $x_i \overset{a}{\rightarrow} y \in \Phi_2$ for some variable $y$ and label $a$. As $\rightarrow_G, \sigma \models \hyps(\Phi)$ and $x_i \overset{a}{\rightarrow}$ is a conjunct of $\hyps(\Phi)$, we have that $\sigma(x_i) \overset{a}{\rightarrow} p$ 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 $\Phi_1 \cup \{x_i \overset{a}{\rightarrow}\}$. 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 $\Phi_2 \setminus \{x_i \overset{a}{\rightarrow} y\}$.

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 15 (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 $\models_G \text{True} \Rightarrow \hyps(\Phi)$ holds is equivalent to checking whether the rule $r$ is always firable. Conversely, checking whether $\models_G \hyps(\Phi) \Rightarrow \text{False}$ 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 element constants 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 16 Let $G = (\Sigma_G, L, R_G)$ 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:

\[ F[x \mapsto c] = \begin{cases} \text{True} & \text{if } F = \text{True} \\ F \land \text{True} & \text{if } F = \text{True} \\ F \lor \text{True} & \text{if } F = \text{True} \\ F \land \text{False} & \text{if } F = \text{False} \\ F \lor \text{False} & \text{if } F = \text{False} \\ F[x \mapsto c] & \text{if } F = x \text{ or } F \text{ contains } x \text{ directly} \\ \text{False} & \text{otherwise} \end{cases} \]
True\([x \mapsto c] = \text{True}\\)

\((x \mapsto \alpha)[x \mapsto c] = \begin{cases} \text{True} & \text{if there is a } c\text{-defining axiom } c \mapsto p \text{ for some } p \\ \text{False} & \text{otherwise} \end{cases}\\)

\((y \mapsto \alpha)[x \mapsto c] = y \mapsto \alpha & \text{if } x \neq y\\)

\((-F)[x \mapsto c] = \neg(F[x \mapsto c])\\)

\((F_1 \land F_2)[x \mapsto c] = (F_1[x \mapsto c]) \land (F_2[x \mapsto c]) .\\)

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

**Lemma 17** Let \(G = (\Sigma_G, L, R_G)\) 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 \(\sigma\),

\[-G, \sigma \models F[x \mapsto c] \iff -G, \sigma[x \mapsto c] \models F ,\]

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 \(G\).

### 6.2 An alternative rule format for zero elements

We now have all the ingredients to reformulate the format we presented in Section 3 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 12, 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 6.1 and by the disjunctive nature of \(\text{hyps}(J)\), with \(J\) 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 18 (Left- and right-aligned pairs, 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 \(c \mapsto t\), there exists a set \(J\) of rules of the form

\[
\begin{array}{c}
\Phi \\
\hline
f(x_0, x_1) \mapsto t' \\
\end{array}
\]

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 \( \sigma(t') \equiv_0 t \), where \( \sigma \) is the substitution mapping \( x_0 \) to \( c \), \( y \) to \( t \) and is the identity on all the other variables, or

B. \( \sigma(t') \equiv_0 t \), 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 \( \sigma(t') \equiv_0 t \), where \( \sigma \) is the substitution mapping \( x_0 \) to \( c \), \( y \) to \( t \) and is the identity on all the other variables, or

B. \( \sigma(t') \equiv_0 t \), 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 1. 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 15, 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 18, these two properties are guaranteed, respectively, by conditions 1(a)i and 1(a)ii.

Theorem 19 Let \( G \) be a GSOS language. Assume that \( L \) and \( R \) are the sets of left- and right-aligned function symbols according to Definition 18. For each \((f, c) \in L\), it holds that \( f(c, x) \equiv c \). Symmetrically, for each \((f, c) \in R\), it holds that \( f(x, c) \equiv c \).

Appendix B contains the proof of Theorem 19.

The following result is a consequence of Theorems 11 and 15.
For GSOS languages, the sets $L$ and $R$ can be effectively constructed.

Remark 2 (Handling predicates using the format of Definition 18). The formats in Definition 12 and the one in Definition 18 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 12 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 18, one can check Example 15, which cannot be dealt with by the format from Definition 12.

It is important to note, however, that the GSOS-based format we presented in Definition 18 can indeed be used to reason about the examples from Section 4 that use predicates. In fact, one can encode a finite collection of predicates specified using rules of the form

\[
\{ x_0 \xrightarrow{a_i} y_i \mid i \in I \} \cup \{ P_k x_0 \mid k \in K \} \cup \{ x_0 \xrightarrow{a_j} \text{ or } \neg P_j x_0 \mid j \in J \},
\]

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, [13, 22]. The key idea in each of these encodings is that a predicate $P$ is represented as a transition relation $P \to$ (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

\[
\{ x_0 \xrightarrow{a_i} y_i \mid i \in I \} \cup \{ x_0 \xrightarrow{P_k} z_k \mid k \in K \} \cup \{ x_0 \xrightarrow{a_j} \text{ or } x_0 \not \xrightarrow{P_j} \mid j \in J \},
\]

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 18 is immediately applicable to all the examples we discussed in Section 4, as well as to those mentioned in, e.g., [9].

Example 17. Consider again the TSS discussed in Example 16. We will now argue that the format proposed in Definition 18 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 3. To see this, take \( L = R = \{(f, \text{RUN}_a)\} \). We limit ourselves to checking that conditions 1a and 1b in Definition 18 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

\[
(x \xrightarrow{a} \land y \xrightarrow{a}) \left[ x \mapsto \text{RUN}_a \right] = \text{True} \land y \xrightarrow{a} .
\]

As we observed in Example 16, each closed term in the TSS under consideration affords an \( a \)-labelled transition. Therefore, the formula \( \text{True} \land y \xrightarrow{a} \) 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 \( f(\text{RUN}_a, y') \equiv_0 \text{RUN}_a \). 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.

\[ \text{Example 18.} \text{ Consider now the synchronous parallel composition of Example 8. We claim that the format proposed in Definition 18 is capable of proving the validity of the laws } (0 \parallel_L 0) \iff 0 \text{ and } (x \parallel_L 0) \iff 0 . \]

Let \( L = R = \{(0, 0)\} \). Since the constant 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 \( \parallel_L \) can never fire because 0 has no transitions. Indeed, the entailment \( \models_G (x \xrightarrow{a} \land y \xrightarrow{a}) \left[ x \mapsto 0 \right] \Rightarrow \text{False} \) 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 4.

Consider now Example 15. 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 \((\neg y)\). Choosing \( J \) to be a singleton set containing one of the rules is not enough, because neither

\[
\models_G \text{True} \Rightarrow (x \xrightarrow{a} \land y \xrightarrow{a}) \left[ x \mapsto \text{RUN}_a \right]
\]

nor

\[
\models_G \text{True} \Rightarrow (x \xrightarrow{a} \land y \xrightarrow{a}) \left[ x \mapsto \text{RUN}_a \right]
\]

hold. On the other hand, when \( J = \{(y), (\neg y)\} \), the entailment

\[
\models_G \text{True} \Rightarrow ((x \xrightarrow{a} \land y \xrightarrow{a}) \lor (x \xrightarrow{a} \land y \xrightarrow{a})) \left[ x \mapsto \text{RUN}_a \right]
\]

holds and, moreover, \( \text{RUN}_a \equiv_0 \text{RUN}_a \), meeting condition 1(a)iiA. Therefore the proposed format can check Example 15, which cannot be dealt with by the format from Definition 12.
7 From zero to unit

In this section we reformulate the unit element format of [5] following the lines of Definition 12.
For the sake of clarity and completeness we repeat here the definition of unit-context equivalence from [5].

Definition 21 (Unit-context equivalence [5]) Given sets $L, R \subseteq \Sigma \times \Sigma$ of pairs of binary function symbols and constants, $\cong^{L,R}$ is the smallest equivalence relation satisfying the following constraints, for each $s \in \mathcal{T}(\Sigma)$:

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

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

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

Theorem 22 (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 $t, u \in \mathcal{T}(\Sigma)$, it is decidable whether $t^{L,R} \cong u$ 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 21. 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 21. 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 21 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 21 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.

**Definition 23 (Left- and right-aligned pairs for unit elements)** 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 $a \in L$, there exists at least one deduction rule of the form
      $$\Phi \cup \{x_1 \xrightarrow{a} y_1\} \quad \frac{f(x_0, x_1) \xrightarrow{a} t'}{f(x_0, x_1) \xrightarrow{a} t'}$$
      where
      i. $\models_G x_0 \xrightarrow{a} \Rightarrow \text{hyps}(\Phi)[x_0 \mapsto c]$, 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

\[
\Phi \\
f(x_0, x_1) \xrightarrow{a} t'
\]

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. \( \models_{\mathcal{L}} \mathbf{hyps}(\Phi)[x_0 \mapsto c] \Rightarrow \text{False} \).

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 24**

Let \( G \) be a GSOS language. Assume that \( L \) and \( R \) are the sets of left- and right-aligned function symbols according to Definition 23. For each \( (f,c) \in L \), it holds that \( f(c,x) \models x \). Symmetrically, for each \( (f,c) \in R \), it holds that \( f(x,c) \models x \).

Appendix C contains the proof of Theorem 24.

**Remark 3.** The constraint that \( c \xrightarrow{b} t \) be the only \( c \)-defining axiom with label \( b \) in condition 1(b)iA of Definition 23 is necessary for the validity of Theorem 24. To see this, consider, for instance, the TSS over set of labels \{a\} with constants \( 0 \), \( \text{RUN}_a \) (see Example 2) and \( c \), and the binary operator \( \| \mathcal{L} \) defined in Example 8.

The rules for the constant \( c \) are

\[
\begin{align*}
&c \xrightarrow{a} c & &c \xrightarrow{a} 0
\end{align*}
\]

Observe that the sets \( L = \{\| \mathcal{L}, c\} \) and \( R = \emptyset \) would satisfy the conditions in Definition 23 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

\[
c \| \mathcal{L} \text{RUN}_a \xrightarrow{a} 0 \| \mathcal{L} \text{RUN}_a \xrightarrow{a} ,
\]

26
while RUN\(_a\) can only perform action \(a\) forever. Therefore \(c\) is not a left unit element for \(\|_L\).

The following result is a consequence of Theorems 15 and 22.

**Theorem 25** 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 [5]. 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 in [5]. A concrete, albeit admittedly inexpressive, example of a TSS exploiting this feature is discussed below.

**Example 19.** Consider a TSS, over the set of labels \(\{a, b\}\), with constants RUN\(_a\) and RUN\(_b\), and a binary function symbol \(f\) defined by the rules below.

\[
\begin{align*}
y \xrightarrow{a} &\ y' \\
y \xrightarrow{b} &\ y'
\end{align*}
\]

\[
\begin{align*}
f(x, y) \xrightarrow{a} &\ y' \\
f(x, y) \xrightarrow{b} &\ y'
\end{align*}
\]

The constants RUN\(_a\) and 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 RUN\(_a\) and 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 23. On the other hand, the format from [5] fails on this basic scenario since \(y\) is tested twice in the rules for \(f\).

All the examples from the literature mentioned in [5] can be handled by the rule format presented in Definition 23. (Indeed, predicates can be dealt with within the proposed format along the lines discussed in Remark 2.) By way of illustration, we limit ourselves to discussing just a couple of examples addressed in [5].

**Example 20 (Nondeterministic Choice).** Consider the classic nondeterministic choice operator from Milner’s CCS [16] specified by the rules below, where \(a \in \mathcal{L}\).

\[
\begin{align*}
x \xrightarrow{a} &\ x' \\
x + a &\xrightarrow{a} x'
\end{align*}
\]

\[
\begin{align*}
y \xrightarrow{a} &\ y' \\
x + y &\xrightarrow{a} y'
\end{align*}
\]

The sets \(R = L = \{(+, 0)\}\) meet the constraints in Definition 23. Let us discuss the constraints relative to the set \(L\). (The constraints for the set \(R\) can be checked using a similar reasoning.)
Example 21 (Synchronous Parallel Composition). Assume that $a$ is the only action in $\mathcal{L}$. Consider the constant $\text{RUN}_a$ and the synchronous parallel composition operator $\parallel_{\mathcal{L}}$ from Example 8. For ease of reference, we recall that $\parallel_{\mathcal{L}}$ is specified by the rule

\[
\frac{x \xrightarrow{a} x' \quad y \xrightarrow{a} y'}{x \parallel_{\mathcal{L}} y \xrightarrow{a} x' \parallel_{\mathcal{L}} y'}.
\]

Take $L = R = \{(\parallel_{\mathcal{L}}, \text{RUN}_a)\}$. These sets $L$ and $R$ meet the constraints in Definition 23. Let us discuss first the set $L$.

1a: Consider the rule above. Since $(x \xrightarrow{a})[x \mapsto \text{RUN}_a] = \text{True}$, the entailment

\[ \models_G y \xrightarrow{a} \Rightarrow (\text{hyps}(\Phi))[x \mapsto 0] \]

is trivially satisfied. Therefore condition 1(a)i is met. Note, moreover, that $y' \cong y'$. Therefore condition 1(a)ii is met too.

1b: Consider the first rule. Since $0 \not\xrightarrow{a}$, the entailment $\models_G (x \xrightarrow{a})[x \mapsto 0] \Rightarrow \text{False}$ holds and condition 1(b)ii is met. Moreover, reasoning as above, we can easily check that the second rule above meets condition 1(b)i.

8 Conclusions

In this paper we have provided two rule formats ensuring that certain constants in a language act as left or right zero elements for a set of binary operators. The format for left and right zero elements presented in Section 3 follows the techniques developed by some of the authors in [5], where a format for left and right unit elements was offered, but the actual details are rather different.

To overcome some drawbacks of that format, in Section 6.2 we reformulated it within the GSOS format, making use of the logic of initial transitions proposed
The new format is able to check relevant cases from the literature and some instances of zero elements left out by the format in Section 3.

Following the design of the revised format for zero elements, we also provided an alternative rule format for left and right unit elements. This format does not include advanced features such as complex terms in the source of the conclusions of rules, like the one in [5] instead does, but is still able to check relevant cases and basic unit elements not addressed by the format from [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.

References


A Proof of Theorem 13

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

**Lemma 26** For all $s, t \in T(\Sigma)$, if $s \cong_0 t$ then $\sigma(s) \cong_0 \sigma(t)$, for each substitution $\sigma$.

From Lemma 26, 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 13 given below we make use of this observation.

**Proof.** (of Theorem 13)

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 $p \xrightarrow{\alpha} p'$ then $q \xrightarrow{\alpha} q'$ for some $q'$ such that $p' \cong_0 q'$,
We prove these statements by an induction on the definition of \( \equiv_0 \). The cases that \( p \equiv_0 q \) is due to reflexivity, symmetry and transitivity of \( \equiv_0 \) are trivial or follow easily using the inductive hypothesis. So, two relevant cases remain to be proved.

1. Suppose that \( p \equiv_0 q \) is due to \( p \equiv c \) and \( q \equiv f(c, p) \) for some \((f, c) \in L\).
   (a) Assume that \( c \xrightarrow{\alpha} p' \), for some \( p' \in \mathbb{C}(\Sigma) \). This is because there exists an axiom \( c \xrightarrow{\alpha} p' \). We shall show that there exists a \( p'' \in \mathbb{C}(\Sigma) \) such that \( q \equiv f(c, p) \xrightarrow{\alpha} p'' \) and \( p' \equiv_0 p'' \).

   By Definition 12, we have a deduction rule of the following form

   \[
   \begin{array}{c}
   \{ x_0 \xrightarrow{\alpha} t_i \mid i \in I \} \cup \{ P_k x_0 \mid k \in K \} \cup \{ x_0 \xrightarrow{\alpha} \text{ or } \neg P_j x_0 \mid j \in J \} \\
   f(x_0, x_1) \xrightarrow{\alpha} t' \\
   \end{array}
   \]

   where
   i. \( x_1 \not\in \{ x_0 \} \cup \bigcup_{i \in I} \text{vars}(t_i) \),
   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 \( \sigma(x_0) = c \), \( \{ c \xrightarrow{\alpha} \sigma(t_i) \mid i \in I \} \) is included in the collection of \( c \)-defining axioms, and \( \sigma(t') \equiv_0 p' \).

   Since \( x_1 \not\in \{ x_0 \} \cup \bigcup_{i \in I} \text{vars}(t_i) \), 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, \( f(c, p) \xrightarrow{\alpha} \sigma'(t') \) is a provable transition. As \( \sigma'(t') \equiv_0 p' \) by clause (iv) above and \( \sigma'(\sigma(t')) \equiv \sigma'(t') \) by construction, Lemma 26 yields that \( \sigma'(t') \equiv_0 p' \), and we are done.

   (b) Assume that \( q \equiv f(c, p) \xrightarrow{\alpha} q' \in C \), for some \( q' \in \mathbb{C}(\Sigma) \). We shall show that there is some \( p' \in \mathbb{C}(\Sigma) \) such that \( c \xrightarrow{\alpha} p' \) and \( p' \equiv_0 q' \).

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

   \[
   \begin{array}{c}
   \Phi \\
   f(x_0, x_1) \xrightarrow{\alpha} t' \\
   \end{array}
   \]

   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 \( p' \equiv_0 \sigma(t') \), and we are done.

(c) The two cases involving predicates, namely when \( P \) holds and \( Pf(c, p) \) holds, follow the same lines.

2. Suppose that \( p \equiv_0 q \) 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. \( \square \)

### B Proof of Theorem 19

We prove that the relation \( \equiv_0 \) is a bisimulation. The claim then follows since \( f(c, p) \equiv_0 c \) and \( g(p, c') \equiv_0 c' \) for each closed term \( p \), \((f, c) \in L \) and \((g, c') \in R \).
To this end, we show that, when \( p \equiv_0 q \), the transfer conditions of Definition 6 are met by an induction on the definition of \( \equiv_0 \). The cases that \( p \equiv_0 q \) is due to reflexivity, symmetry and transitivity of \( \equiv_0 \) are trivial or follow easily by induction. So, two relevant cases remain to be considered.

1. Suppose that \( p \equiv_0 q \) 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 \( c \xrightarrow{a} p' \in C \), for some \( p' \in C(\Sigma) \). This is because there exists an axiom \( c \xrightarrow{a} p' \). We shall show that there exists a \( p'' \in C(\Sigma) \) such that \( q \equiv f(c, q') \xrightarrow{a} p'' \) and \( p' \equiv_0 p'' \).

   From constraint 1(a)ii in Definition 18, we have a non-empty set \( J \) of deduction rules of the following form

   \[
   \frac{}{f(x_0, x_1) \xrightarrow{a} t'}
   \]

   such that

   i. \( |_G \text{True} \Rightarrow \text{hyps}(J)[x_0 \mapsto c] \), 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 \( \sigma(t') \equiv_0 p' \), where \( \sigma \) is the substitution mapping \( x_0 \) to \( c \), \( y \) to \( p' \) and is the identity on all the other variables, or
   B. \( \sigma(t') \equiv_0 p' \), 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 \( |_G \text{True} \Rightarrow \text{hyps}(J)[x_0 \mapsto c] \), 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 14, 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 26 yields that \( p' \cong_0 \sigma''(t') \), and we are done.

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

It follows from constraint 1b in Definition 18 that the transition \( q \equiv f(c, q') \xrightarrow{a} p'' \) is due to a deduction rule of the following form

\[
\Phi \quad \frac{f(x_0, x_1) \xrightarrow{a} t'}{f(x_0, x_1) \xrightarrow{a} t'}
\]

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)iii in Definition 18 cannot apply and we fall in the case of constraint 1(a)iii. 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.

C Proof of Theorem 24

The proof relies on the following lemma proved in [5].

Lemma 27 For all \( s, t \in T(\Sigma) \), if \( s \equiv t \) then \( \text{vars}(s) = \text{vars}(t) \) and \( \sigma(s) \equiv \sigma(t) \), for each substitution \( \sigma \).

The proof of Theorem 24 is given below.

Proof. (of Theorem 24) We prove that \( \equiv \) is a bisimulation relation. The claim then follows since \( f(c, p) \equiv p \) and \( g(p, c') \equiv p \) for each closed term \( p, (f, c) \in L \) and \( (g, c') \in R \). We prove that whenever \( p \equiv q \) the transfer conditions of Definition 6 are met by an induction on the definition of \( \equiv \). The cases that \( p \equiv q \) is due to reflexivity, symmetry and transitivity of \( \equiv \) are trivial or follow easily by induction. So, two relevant cases remain to be proved.

1. Suppose that \( p \equiv q \) is due to \( q \equiv f(c, p) \) for some \( (f, c) \in L \).

   (a) Assume that \( p \xrightarrow{a} p' \in C \), for some \( p' \in C(\Sigma) \). We shall show that there exists a \( p'' \in C(\Sigma) \) such that \( q \equiv f(c, p) \xrightarrow{a} p'' \) and \( p' \equiv p'' \).

   From constraint 1a in Definition 23, we have that there exists a deduction rule of the following form

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

   where
(b) Assume that $q \sigma$ with such a closed substitution $\sigma$ and a closed substitution $\sigma_{\text{case 1(a)iiA or 1(a)iiB.}}$. In both cases, by Lemma 27, we have that $\sigma$ in Definition 23 cannot apply and we fall in the case of constraint 1(b)i. $\sigma$ and $\sigma_{\text{case 1(a)iiA or 1(a)iiB.}}$.

We now show that there exists a closed substitution $\sigma'$ such that $\sigma'$ satisfies $\Phi$, $f(c,p) \sigma' \to t'$ is a provable transition and $\sigma'(t') \equiv p'$. 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 $x_1 \sigma' \to q$. If we are in case 1(a)iiA, let $\sigma''(y) \equiv t$, so that $\sigma''$ also satisfies the premise $x_0 \sim y \in \Phi$. As $\sigma''$ satisfies the premise $x_1 \sigma' \to y$, $\sigma''(x_0) \equiv c$ and $|G, x_1 \sigma' \to \text{hyps} | = \Phi$. Therefore Lemma 14 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

\[
\Phi \cup \{x_1 \sigma' \to y_1\}.
\]

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

\[
\sigma'(f(x_0, x_1)) \equiv f(c, p) \sigma' \to t'.
\]

Recall that $\sigma'(t') \equiv y_1$, where $\sigma$ is either the substitution defined in case 1(a)iiA or 1(a)iiB. In both cases, by Lemma 27, we have that

\[
\sigma'(t') = \sigma'(\sigma(t')) \equiv \sigma'(y_1) = p'
\]

and we are done.

(b) Assume that $q \equiv f(c, p) \sigma' \to q' \in C$, for some $q' \in C(\Sigma)$.

The transition $q \equiv f(c, p) \sigma' \to q' \in C$ must be proved using an $f$-defining rule of the form

\[
\Phi
\]

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 23 cannot apply and we fall in the case of constraint 1(b)i. Thus $x_1 \sigma' \to y_1 \in \Phi$ for some variable $y_1$. As $\sigma'$ satisfies $\Phi$, it follows that $\sigma'(y_1) \equiv p \sigma'(y_1)$. We claim that $\sigma'(y_1) \equiv q'$. To see that this claim does hold true, recall that, since constraint 1(b)i in Definition 23 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 $\sigma(t') \equiv y_1$, where $\sigma$ is the substitution mapping $x_0$ to $c$, $y$ to $t$ and is the identity on all the other variables,

ii. or $\sigma(t') \equiv y_1$, 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 27, since $\sigma(t') \equiv y_1$ holds, we have that

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

and we are done.

The latter case is handled similarly.

2. Suppose that $p \equiv q$ 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. ☑