A Rule Format for Unit Elements*

Luca Aceto[1], Anna Ingolfsdottir[1],
MohammadReza Mousavi[2], and Michel A. Reniers[2]

[1] ICE-TCS, School of Computer Science, Reykjavik University
Kringlan 1, IS-103 Reykjavik, Iceland
[2] Department of Computer Science, Eindhoven University of Technology
P.O. Box 513, NL-5600 MB Eindhoven, The Netherlands

Abstract. This paper offers a meta-theorem for languages with a Structural Operational Semantics (SOS) in the style of Plotkin. Namely, it proposes a generic rule format for SOS guaranteeing that certain constants act as left- or right-unit elements for a set of binary operators. We show the generality of our format by applying it to a wide range of operators from the literature on process calculi.

1 Introduction

In many process algebras and specification languages, one encounters constructs that are unit elements for certain composition operators. The concept of (left) unit element for a binary operator f can be concisely summarized in the following algebraic equation, where $\mathbf{0}$ is the left-unit element for f: $f(\mathbf{0}, x) = x$.

In this paper, we propose a generic rule format guaranteeing that certain constants are left- or right-unit elements for a set of binary operators, whose semantics is defined using Plotkin's style of Structural Operational Semantics (SOS) [2,12,13]. The notions of left and right unit are defined with respect to a notion of behavioural equivalence. There are various notions of behavioural equivalence presented in the literature (see, e.g., [7]), which are, by and large, weaker than bisimilarity. Thus, to be as general as possible, we prove our main result for all equivalences that contain, i.e., are weaker than, bisimilarity.

This paper is part of our ongoing line of research on capturing basic properties of composition operators in terms of syntactic rule formats, exemplified by rule formats for commutativity [11], associativity [6], determinism and idempotence [1].

This line of research serves multiple purposes. Firstly, it paves the way for a tool-set that can mechanically prove such properties without involving user interaction. Secondly, it provides us with an insight as to the semantic nature of such properties and its link to the syntax of SOS deduction rules. In other words, our rule formats may serve as a guideline for language designers who want to

* The work of Aceto and Ingolfsdottir has been partially supported by the projects "The Equational Logic of Parallel Processes" (nr. 060013021), and "New Developments in Operational Semantics" (nr. 080039021) of the Icelandic Research Fund. The first author dedicates the paper to the memory of his mother, Imelde Diomede Aceto, who passed away a year ago.

J. van Leeuwen et al. (Eds.): SOFSEM 2010, LNCS 5901, pp. 141–152, 2010.

ensure, a priori, that the constructs under design enjoy certain basic algebraic properties. There is value in determining what conditions on the SOS description of the semantics of operators guarantee that certain elements are left or right units. The fact that the constraints imposed by our general format are non-trivial indicates that the isolation of a widely applicable syntactic characterization of the semantic properties that underlie the existence of unit elements is, perhaps surprisingly, difficult.

The rest of this paper is organized as follows. In Section 2, we define some basic notions that are required for the technical developments in the rest of the paper. In Section 3, we present our rule format and prove that it guarantees the unit element property. In Section 4, we apply the rule format to various examples from the literature. In order to ease the application of our rule format to operators whose operational semantics is specified using predicates, we extend the format to that setting in Section 4.2. Section 5 concludes the paper and discusses directions for future work. Proofs can be found in [3].

2 Preliminaries

We begin by recalling the basic notions from the theory of SOS that are needed in the remainder of this study. We refer the interested readers to, e.g., [2,12] for more information and background.

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

- *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 vars : $\mathbb{T}(\Sigma) \to 2^V$ gives the set of variables appearing in a term. The set $\mathbb{C}(\Sigma) \subseteq \mathbb{T}(\Sigma)$ is the set of closed terms*, i.e., terms that contain no variables. We use $p, q, p', p_i, \ldots$ to range over closed terms. A* substitution *σ is a function of type $V \to \mathbb{T}(\Sigma)$. We extend the domain of substitutions to terms homomorphically and write $\sigma(t)$ for the result of applying the substitution σ to the term t. If the range of a substitution lies in $\mathbb{C}(\Sigma)$, we say that it is a* closed substitution*. An explicit substitution $[x \mapsto t]$ maps x to t and is the identity function on all variables but x.*

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

- *Σ is a signature.*
- *$\mathcal{L}$ is a set of labels (or actions) ranged over by a, b, l. If $l \in \mathcal{L}$, and $t, t' \in \mathbb{T}(\Sigma)$ we say that $t \xrightarrow{l} t'$ is a* positive transition formula *and $t \not\xrightarrow{l}$ is a* negative

transition formula. *A transition formula (or just formula), typically denoted by ϕ or ψ, is either a negative transition formula or a positive one.*
- *D is a set of* deduction rules*, i.e., pairs of the form (Φ, ϕ) where Φ is a set of formulae and ϕ is a positive formula. We call the formulae contained in Φ the* premises *of the rule and ϕ the* conclusion.

We write $vars(r)$ to denote the set of variables appearing in a deduction rule r. We say that a formula is closed *if all of its terms are closed. Substitutions are also extended to formulae and sets of formulae in the natural way. For a rule r and a substitution σ, 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 (Φ, ϕ) 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 $c \xrightarrow{l} p$ for some label l and closed term p.

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

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

Definition 4 (Contradiction and Entailment). *The formula $t \xrightarrow{l} t'$ is said to* contradict *$t \overset{l}{\nrightarrow}$, and vice versa. For a set Φ of formulae and a formula ψ, Φ* contradicts *ψ, denoted by $\Phi \nvDash \psi$, when there is a $\phi \in \Phi$ that contradicts ψ. We write $\Phi \vDash \Psi$ if Φ does not contradict any $\psi \in \Psi$. A formula ϕ* entails *ψ when there is a substitution σ such that $\sigma(\phi) \equiv \psi$. A set Φ entails a set Ψ of formulae, when there exists a substitution σ such that, for each $\psi \in \Psi$, there exists a $\phi \in \Phi$ such that $\sigma(\phi) \equiv \psi$.*

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

– *for each* $\phi \in C$, $\mathcal{T} \vdash \frac{N}{\phi}$ *for a set* N *of negative formulae such that* $C \cup U \models N$,
– *for each* $\phi \in U$, $\mathcal{T} \vdash \frac{N}{\phi}$ *for a set* N *of negative formulae such that* $C \models N$.

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* $\mathcal{T}$ *is* complete *when for its least three-valued stable model it holds that* $U = \emptyset$. *In a complete TSS, we say that a closed substitution* σ *satisfies a set of formulae* Φ *if* $\sigma(\phi) \in C$, *for each positive formula* $\phi \in \Phi$, *and* $C \models \sigma(\phi)$, *for each negative formula* $\phi \in \Phi$.

Definition 6 (Bisimulation and Bisimilarity). *Let* $\mathcal{T}$ *be a TSS with signature* Σ *and label set* $\mathcal{L}$. *A relation* $\mathcal{R} \subseteq \mathbb{C}(\Sigma) \times \mathbb{C}(\Sigma)$ *is a* bisimulation relation *if* $\mathcal{R}$ *is symmetric and, for all* $p_0, p_1, p_0' \in \mathbb{C}(\Sigma)$ *and* $l \in \mathcal{L}$,

$$(p_0 \,\mathcal{R}\, p_1 \wedge \mathcal{T} \vdash p_0 \xrightarrow{l} p_0') \Rightarrow \exists_{p_1' \in \mathbb{C}(\Sigma)} (\mathcal{T} \vdash p_1 \xrightarrow{l} p_1' \wedge p_0' \,\mathcal{R}\, p_1').$$

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

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

3 Rule Format

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

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

Definition 7 (Unit-Context Equivalent Terms). *Given sets* $L, R \subseteq \Sigma \times \Sigma$ *of pairs of binary function symbols and constants,* $\overset{L,R}{\cong}$ *is the smallest equivalence relation satisfying the following conditions, for each* $s \in \mathbb{T}(\Sigma)$*:*

1. $\forall_{(f,c)\in L}\ s \overset{L,R}{\cong} f(c,s)$, *and*
2. $\forall_{(g,c)\in R}\ s \overset{L,R}{\cong} g(s,c)$.

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

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

Lemma 8. *For all* $s,t \in \mathbb{T}(\Sigma)$, *if* $s \cong t$ *then* $vars(s) = vars(t)$ *and* $\sigma(s) \cong \sigma(t)$, *for each substitution* σ.

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

Definition 9 (Left- and Right-Aligned Pairs). *Given a TSS, the sets* L *and* R *of pairs of binary function symbols and constants are the largest sets satisfying the following conditions.*

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

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

   where

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

   (b) *Each* f*-defining deduction rule has the following form:*

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

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

   i. *either there exists some* $t_1 \xrightarrow{a} t'' \in \Phi$ *with* $\sigma(t') \cong \sigma(t'')$, *or*
   ii. *there exists a premise* $\phi \in \Phi$ *with* t_0 *as its source such that*
      A. *either* ϕ *is a positive formula and the collection of conclusions of c-defining deduction rules does not entail* $\sigma(\phi)$, *or*
      B. ϕ *is a negative formula and the collection of conclusions of c-defining axioms contradicts* $\sigma(\phi)$.

2. *The definition of right-aligned pairs of operators and constant symbols – that is, those such that* $(f, c) \in R$ *– is symmetric and is not repeated here.*

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

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

Condition 1(b)i, on the other hand, ensures that, whenever (f, c) is in L, each transition $f(c, p) \xrightarrow{a} q'$ is due to a transition $p \xrightarrow{a} p'$ for some p' that is bisimilar to q'. Thus, if (f, c) is in L then, in the context of the constant c, a term of the form $f(c, p)$ can only mimic the behaviour of p. As will become clear from the examples to follow, condition 1(b)ii ensures that the f-defining rule cannot be used to derive a transition for $f(c, p)$ and hence it is exempted from further conditions; the presence of this condition enhances the generality of our format and allows us to handle common examples of unit constants from the literature (see, e.g., Example 3). A slightly more technical discussion of the conditions is given in [3].

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

Note that, since the sets L and R are defined as the *largest* sets of pairs satisfying the conditions from Definition 9, in order to show that (f, c) is a left-aligned pair, say, it suffices only to exhibit two sets L and R satisfying these conditions, such that (f, c) is contained in L.

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

Example 1. Assume that a is the only action and consider the binary operators f_i, $i \geq 0$, with rules

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

Let $\mathbf{0}$ be a constant with no rules. Then each of the pairs $(f_i, \mathbf{0})$ is left aligned because the sets $L = \{(f_i, \mathbf{0}) \mid i \geq 0\}$ and $R = \emptyset$ meet the conditions from Definition 9. In particular, note that $f_{i+1}(x_0, y_1)[x_0 \mapsto \mathbf{0}] \equiv f_{i+1}(\mathbf{0}, y_1) \cong y_1$, for each $i \geq 0$. Note that, for each $i \geq 0$, the equations $f_i(\mathbf{0}, x) = x$ hold modulo bisimilarity. This fact can be checked directly by showing that the symmetric closure of the relation $\mathcal{R} = \{(f_i(\mathbf{0}, p), p) \mid p \text{ a closed term}\}$ is a bisimulation, and is also a consequence of Theorem 10 to follow, which states the correctness of the rule format we described in Definition 9.

Example 2. Consider the following TSS, which is defined for a signature with $\mathbf{0}$ and a as constants and f and g as binary function symbols.

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

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

Theorem 10. *Let $\mathcal{T}$ be a complete TSS in which each rule is f-defining for some function symbol f. Assume that L and R are the sets of left- and right-aligned function symbols according to Definition 9. For each $(f, c) \in L$, it holds that $f(c, x) \underline{\leftrightarrow} x$. Symmetrically, for each $(f, c) \in R$, it holds that $f(x, c) \underline{\leftrightarrow} x$.*

Note that Theorem 10 trivially extends to any notion of behavioural equivalence weaker than bisimilarity.

4 Applications and Extensions

Apart from its correctness, the acid test for the usefulness of a rule format is that it be expressive enough to cover examples from the literature that afford the property they were designed to ensure. Our order of business in this section will be to offer examples of applications of the format for unit elements we introduced in Definition 9 and to show how the format can be extended to deal with operators whose semantic definition involves the use of predicates.

4.1 Applications of the Basic Rule Format

We start by presenting examples of applications of the format for unit elements we introduced in Definition 9.

Example 3 (Nondeterministic Choice). Consider the nondeterministic choice operator from Milner's CCS [10] specified by the rules below, where $a \in \mathcal{L}$.

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

The sets $R = L = \{(+, \mathbf{0})\}$ meet the conditions in Definition 9. Indeed, condition 1a and its symmetric version are trivially satisfied by the right-hand and the left-hand rule schemas, respectively. (Note that the substitution σ associated with the empty collection of axioms in condition 1(a)iii is the identity function over the set of variables.) To see that condition 1b is also met, let σ be a closed substitution such that $\sigma(x) = \mathbf{0}$. Observe that

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

The reasoning for condition 2 is symmetric. Therefore, Theorem 10 yields the soundness of the well known equations [8]: $\mathbf{0} + x = x = x + \mathbf{0}$.

Example 4 (Synchronous Parallel Composition). Assume, for the sake of simplicity, that a is the only action. Consider a constant RUN_a and the synchronous parallel composition from CSP [9][1] specified by the rules

$$\frac{}{\mathrm{RUN}_a \xrightarrow{a} \mathrm{RUN}_a} \qquad \frac{x \xrightarrow{a} x' \quad y \xrightarrow{a} y'}{x \parallel_a y \xrightarrow{a} x' \parallel_a y'} \ .$$

Take $L = R = \{(\parallel_a, \mathrm{RUN}_a)\}$. These sets L and R meet the conditions in Definition 9. To see that condition 1a and its symmetric version are satisfied by the above rule for $\parallel_a$, observe that the substitution σ associated with the singleton set containing the only axiom for RUN_a in condition 1(a)iii maps both the variables x and x' to RUN_a and is the identity function over the other variables. For such a σ, $\sigma(x' \parallel_a y') = \mathrm{RUN}_a \parallel_a y' \cong y'$.

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

$$\sigma(x' \parallel_a y') = \mathrm{RUN}_a \parallel_a \sigma(y') \cong \sigma(y')$$

and condition 1(b)i is met. Theorem 10 thus yields the soundness of the well known equations $\mathrm{RUN}_a \parallel_a x = x = x \parallel_a \mathrm{RUN}_a$. These are just equation L3B from [9, page 69] and its symmetric counterpart.

Example 5 (Left Merge and Interleaving Parallel Composition). The following rules describe the operational semantics of the classic left merge and interleaving parallel composition operators [5,10].

$$\frac{x \xrightarrow{a} x'}{x \mathbin{\underline{\parallel}} y \xrightarrow{a} x' \parallel y} \qquad \frac{x \xrightarrow{a} x'}{x \parallel y \xrightarrow{a} x' \parallel y} \qquad \frac{y \xrightarrow{a} y'}{x \parallel y \xrightarrow{a} x \parallel y'}$$

Take $L = \{(\parallel, \mathbf{0})\}$ and $R = \{(\parallel, \mathbf{0}), (\mathbin{\underline{\parallel}}, \mathbf{0})\}$. It is easy to see that these sets L and R meet the condition in Definition 9. Therefore, Theorem 10 yields the well known equalities $\mathbf{0} \parallel x = x$, $x \parallel \mathbf{0} = x$, and $x \mathbin{\underline{\parallel}} \mathbf{0} = x$.

Note that the pair $(\mathbin{\underline{\parallel}}, \mathbf{0})$ cannot be added to L while preserving condition 1a in Definition 9. Indeed, $\mathbf{0}$ is not a left unit for the left merge operator $\mathbin{\underline{\parallel}}$.

Example 6 (Disrupt). Consider the following disrupt operator $\blacktriangleright$ [4] with rules

$$\frac{x \xrightarrow{a} x'}{x \blacktriangleright y \xrightarrow{a} x' \blacktriangleright y} \qquad \frac{y \xrightarrow{a} y'}{x \blacktriangleright y \xrightarrow{a} y'} \ .$$

[1] In [9], Hoare uses the symbol $\parallel$ to denote the synchronous parallel composition operator. Here we will use that symbol for parallel composition.

Note that the equation $\mathbf{0} \blacktriangleright x = x$ holds modulo bisimilarity. We now argue that its soundness is a consequence of Theorem 10. Indeed, take $L = \{(\blacktriangleright, \mathbf{0})\}$ and $R = \emptyset$. It is easy to see that these sets L and R meet the conditions in Definition 9. In particular, to see that condition 1b is met by the first rule, observe that the set of rules for $\mathbf{0}$ is empty and therefore does not entail $\mathbf{0} \xrightarrow{a} p$ for any closed term p. A symmetric reasoning shows that the valid equation $x \blacktriangleright \mathbf{0} = x$ is also a consequence of Theorem 10.

Example 7 (Timed Nondeterministic Choice). Consider nondeterministic choice in a timed setting. It is defined by means of the deduction rules from Example 3 and additionally the deduction rules

$$\frac{x \xrightarrow{1} x' \quad y \xrightarrow{1} y'}{x + y \xrightarrow{1} x' + y'} \qquad \frac{x \xrightarrow{1} x' \quad y \not\xrightarrow{1}}{x + y \xrightarrow{1} x'} \qquad \frac{x \not\xrightarrow{1} \quad y \xrightarrow{1} y'}{x + y \xrightarrow{1} y'} \ .$$

The equations $\mathbf{0} + x = x$ and $x + \mathbf{0} = x$ hold modulo bisimilarity. This is a consequence of Theorem 10 by taking $L = R = \{(+, \mathbf{0})\}$. For label 1, condition 1a is met by the third deduction rule. The first deduction rule satisfies condition 1(b)iiA, the second deduction rule satisfies condition 1(b)iiB, and the third deduction rule satisfies condition 1(b)i trivially.

4.2 Predicates

In the literature concerning the theory of rule formats for SOS (especially, the work devoted to congruence formats for various notions of bisimilarity), most of the time predicates are neglected at first and are only added to the considerations at a later stage. The reason is that one can encode predicates quite easily by means of transition relations. One can find a number of such encodings in the literature – see, for instance, [6,15]. In each of these encodings, a predicate P is represented as a transition relation $\xrightarrow{P}$ (assuming that P is a fresh label) with some fixed target. However, choosing the "right" target term to cope with the examples in the literature (and the new ones appearing in the future) within our format is extremely intricate, if not impossible. That is why we introduce an extension of our rule format that handles predicates as first-class objects, rather than coding them as transitions with dummy targets. To this end, we extend the basic notions presented in Section 2 to a setting with predicates.

Definition 11 (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 $t \in \mathbb{T}(\Sigma)$. We call t the* source *of both predicate formulae. In the extended setting, a (positive, negative)* formula *is either a (positive, negative) transition formula or (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 of the predicate formula in its conclusion.*

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

Definition 12 (Extended Left- and Right-Aligned Pairs). *Given a TSS, the sets L and R of pairs of binary function symbols and constants are the largest sets satisfying the following conditions.*

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

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

   where

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

   (b) *For each predicate $P \in \mathcal{P}$, there exists a deduction rule, of the following form:*

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

   where

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

   (c) *Each f-defining deduction rule has one of the following forms:*

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

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

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

A. either ϕ is a positive formula and the collection of conclusions of c-defining deduction rules does not entail $\sigma(\phi)$, or
B. ϕ is a negative formula and the collection of conclusions of c-defining axioms contradicts $\sigma(\phi)$.

2. The definition of right-aligned pairs of operators and constant symbols – that is, those such that $(f, c) \in R$ – is symmetric and is not repeated here.

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

We are now ready to state the counterpart of Theorem 10 in a setting with predicates.

Theorem 13. *Let $\mathcal{T}$ be a complete TSS in which each rule is f-defining for some function symbol f. Assume that L and R are the sets of extended left- and right-aligned function symbols according to Definition 12. For each $(f, c) \in L$, it holds that $f(c, x) \underline{\leftrightarrow} x$. Symmetrically, for each $(f, c) \in R$, it holds that $f(x, c) \underline{\leftrightarrow} x$.*

We now provide an example of the application of the rule format. In [3], we give two additional examples involving the use of predicates.

Example 8 (Sequential Composition). A standard operator whose operational semantics can be given using predicates is that of sequential composition. Consider the following deduction rules, where $p \downarrow$ means that "p can terminate successfully". (As usual in the literature, we write the termination predicate $\downarrow$ in postfix notation.)

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

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

5 Conclusions

In this paper, we proposed a rule format for Structural Operational Semantics, guaranteeing constants to be left- or right-unit elements of certain operators. The rule format encompasses advanced features such as negative premises and complex terms appearing nearly anywhere in the deduction rules. We further extended the proposed format to accommodate predicates, which are among the common ingredients in the SOS of many contemporary process description languages. The rule format is applied to a number of examples from the literature, motivating its applicability.

A straightforward extension of our rule format allows one to deal with unit elements that are complex closed terms (instead of constants). We are not aware of many practical examples in which such unit elements are present. Another algebraic property, which can be captured using the same technique, is the existence of a (left or right) zero element, i.e., a constant $\mathbf{0}$ such that $f(\mathbf{0}, x) = f(x, \mathbf{0}) = \mathbf{0}$. Mechanizing the existing rule formats for algebraic properties in a tool-set is another direction for future work.

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

References

1. Aceto, L., Birgisson, A., Ingolfsdottir, A., Mousavi, M.R., Reniers, M.A.: Rule Formats for Determinism and Idempotence. In: FSEN 2009. LNCS, vol. 5961. Springer, Heidelberg (to appear, 2010)
2. Aceto, L., Fokkink, W.J., Verhoef, C.: Structural Operational Semantics. In: Handbook of Process Algebra, ch. 3, pp. 197–292. Elsevier, Amsterdam (2001)
3. Aceto, L., Ingolfsdottir, A., Mousavi, M.R., Reniers, M.A.: A Rule Format for Unit Elements. Tech. Rep. CSR-0913, Eindhoven University of Technology (2009)
4. Baeten, J.C.M., Bergstra, J.: Mode Transfer in Process Algebra. Tech. Rep. CSR-0001, Eindhoven University of Technology (2000)
5. Bergstra, J.A., Klop, J.W.: Fixedpoint Semantics in Process Algebra. Tech. Rep. IW 206/82, Center for Mathematics, Amsterdam (1982)
6. Cranen, S., Mousavi, M.R., Reniers, M.A.: A Rule Format for Associativity. In: van Breugel, F., Chechik, M. (eds.) CONCUR 2008. LNCS, vol. 5201, pp. 447–461. Springer, Heidelberg (2008)
7. van Glabbeek, R.J.: The Linear Time - Branching Time Apectrum I. In: Handbook of Process Algebra, ch. 1, pp. 3–100. Elsevier, Amsterdam (2001)
8. Hennessy, M., Milner, R.: Algebraic Laws for Non-Determinism and Concurrency. J. ACM 32(1), 137–161 (1985)
9. Hoare, C.A.R.: Communicating Sequential Processes. Prentice-Hall, Englewood Cliffs (1985)
10. Milner, R.: Communication and Concurrency. Prentice-Hall, Englewood Cliffs (1989)
11. Mousavi, M.R., Reniers, M.A., Groote, J.F.: A Syntactic Commutativity Format for SOS. IPL 93, 217–223 (2005)
12. Mousavi, M.R., Reniers, M.A., Groote, J.F.: SOS Formats and Meta-Theory: 20 Years after. TCS 373(3), 238–272 (2007)
13. Plotkin, G.D.: A Structural Approach to Operational Semantics. JLAP 60-61, 17–140 (2004)
14. Plotkin, G.D.: A Powerdomain for Countable Non-Determinism (extended abstract). In: Nielsen, M., Schmidt, E.M. (eds.) ICALP 1982. LNCS, vol. 140, pp. 418–428. Springer, Heidelberg (1982)
15. Verhoef, C.: A Congruence Theorem for Structured Operational Semantics with Predicates and Negative Premises. Nordic Journal of Computing 2(2), 274–302 (1995)