

A Rule Format for Associativity

Sjoerd Cranen, MohammadReza Mousavi and Michel A. Reniers

Department of Computer Science, Eindhoven University of Technology, P.O. Box 513,
NL-5600 MB Eindhoven, The Netherlands

Abstract. We propose a rule format that guarantees associativity of binary operators with respect to all notions of behavioral equivalence that are defined in terms of (im)possibility of transitions, e.g., the notions below strong bisimilarity in van Glabbeek’s spectrum. The initial format is a subset of the De Simone format. We show that all trivial generalizations of our format are bound for failure. We further extend the format in a few directions and illustrate its application to several formalisms in the literature. A subset of the format is studied to obtain associativity with respect to graph isomorphism.

1 Introduction

Structural Operational Semantics (SOS) [15] provides a convenient and intuitive way of specifying behavior of systems in terms of states and (labeled) transitions. There are essential properties that must be repeatedly proven for each instance of SOS specification, and the proofs of such properties often follow standard yet tedious lines of reasoning. To facilitate such proofs, many rule formats have been developed for SOS (for an overview, see [1, 14]), each with their own syntactic features such as predicates, data stores, and negative formulae. Around each of these formats different meta-theorems have been formulated and proven. These meta-theorems aim at providing a quick and syntactical way of proving the aforementioned properties for the semantics specified by the deduction rules. Examples of such meta-theorems include those concerning congruence of behavioral equivalences [7] and commutativity of operators [12].

Associativity (with respect to a given notion of equivalence) is an interesting property of binary operators, which does not lend itself easily to such syntactic checks. Proofs of associativity are usually much more laborious than proofs of both congruence and commutativity. For example, proofs of congruence, by and large, make use of induction on the proof structure for transitions and are confined to look at the proof-tree *up to depth one*. Thus, they can be performed by looking at deduction rules individually. Proofs of associativity, however, are usually concerned with proof-trees of depth two and hence, if there are n deduction rules for a certain binary operator each with m premises, the number of case-distinctions in its associativity proof are n^m in the worst case (for each deduction rule and each premises, there are n possible deduction rules responsible for the transition mentioned in the premise). This is why in [12, Section 5], we report that our initial attempt to devise a property for associativity did not

lead to a concrete rule format. We are not aware of any other rule format for associativity to date. In [10], an abstract 2-categorical framework is presented, which guarantees algebraic properties such as commutativity and associativity. Deriving concrete rule formats from this abstract framework is mentioned as future research in [10].

In this paper we revisit this problem and propose a syntactic rule format that guarantees associativity of a binary operator with respect to any notion of behavioral equivalence that is specified in terms of transitions, e.g., all notions below strong bisimilarity in van Glabeek’s spectrum [9, 8]. We take the De Simone format [7] as our starting point and add a number of other ingredients, such as predicates [3], to it. Our choice of De Simone format is motivated by the inherent complexity of associativity proofs and is thus aimed to reduce the size and the number of the proof trees as much as possible. The extensions are motivated by practical examples as illustrated in the remainder of this paper. Despite the simple setting of our format, as we show in this paper, our format is widely applicable to most practical examples we encountered so far. Moreover, we show that dropping any of these restrictions jeopardizes the meta-result, even for associativity with respect to the weaker notions of behavioral equivalence, e.g., trace equivalence.

The rest of this paper is structured as follows. We start in Section 2 by presenting some preliminary notions concerning associativity and SOS, used throughout the rest of the paper. In Section 3, we present our basic rule format for associativity and state and prove our associativity meta-theorem with respect to all notions of equivalence that are weaker than (i.e., contain) strong bisimilarity. Section 4 extends our rule format in various directions. In Section 5, we impose some constraints on our format in order to obtain associativity with respect to isomorphism (and all weaker notions). Finally, Section 6 concludes the paper and points out possible directions for future research.

2 Preliminaries

For sake of completeness, we quote standard definitions from concurrency theory and the meta-theory of SOS, which are used in the rest of this paper. A reader familiar with these standard definitions may skip this section altogether.

Definition 1 (Signature and terms). *We assume an infinite set of variables V (with typical members $x, y, x', y', x_i, y_i, \dots$). A signature Σ is a set of function symbols (operators) with fixed arities. Functions with zero arity are called constants. A term $t \in \mathbb{T}(\Sigma)$ is defined inductively as follows:*

- *A variable $x \in V$ is a term.*
- *If t_1, \dots, t_n are terms then for all $f \in \Sigma$ with arity n , $f(t_1, \dots, t_n)$ is a term. Terms are typically denoted by t, t', t_i, t'_i, \dots . Syntactic equality on terms is denoted by \equiv . A closed term $p \in \mathbb{C}(\Sigma)$, is a term which does not contain any variables. A substitution σ is a function from V to $\mathbb{T}(\Sigma)$. The domain of substitution σ is lifted naturally to terms. The range of a closing substitution σ is $\mathbb{C}(\Sigma)$.*

Definition 2 (Transition System Specification (TSS), formula). A transition system specification (TSS) is a tuple (Σ, Rel, D) where

- Σ is a signature
- Rel is a set of relation symbols, where for all $\longrightarrow \in Rel$ and $t, t' \in \mathbb{T}(\Sigma)$. We define that $(t, t') \in \longrightarrow$ is a formula.
- D is a set of deduction rules. A deduction rule is defined as a tuple (H, c) , where H is a set of formulae and c is a formula. The formula c is called the conclusion and the formulae from H are called the premises of the deduction rule.

A formula (also called transition) $(t, t') \in \longrightarrow$ is usually denoted by the more intuitive notation $t \longrightarrow t'$. We refer to t as the *source* and to t' as the *target* of the transition. Mostly, in case there is more than one relation symbol, we use $Rel = \{ \xrightarrow{l} \mid l \in L \}$ for some set of labels L . A deduction rule (H, c) is usually denoted by $\frac{H}{c}$. Such a deduction rule is an (f, l) -defining rule of an n -ary operator $f \in \Sigma$ and label $l \in L$ if and only if c is of the form $f(x_1, \dots, x_n) \xrightarrow{l} t$. A deduction rule is an f -defining rule if it is an (f, l) -defining rule for some $l \in L$.

Definition 3 (De Simone format [7]). A deduction rule is in the De Simone format if it is of the form

$$\frac{\{x_i \xrightarrow{l_i} y_i \mid i \in I\}}{f(x_1, \dots, x_n) \xrightarrow{l} t}, \quad P$$

where $f \in \Sigma$ has arity n , $I \subseteq \{1, \dots, n\}$ is a finite set of indices, P is a predicate on the labels in the deduction rule and moreover,

- for $1 \leq i < j \leq n$, x_i and x_j are different variables and for $1 \leq i \leq n$ and $j \in I$, x_i and y_j are distinct variables,
- t is a term in which the variables from $\{x_i \mid i \notin I\} \cup \{y_i \mid i \in I\}$ occur at most once.

A TSS is in the De Simone format if and only if all of its deduction rules are.

Definition 4 (Provability). A derived deduction rule $\frac{P}{c}$, also written as the tuple (P, c) , is provable from a TSS \mathcal{T} , denoted $\mathcal{T} \vdash (P, c)$, when there exists a proof structure, i.e., well-founded upwardly branching tree with formulae as nodes and of which

- the root node is labeled by c ,
- if a node is labeled by ψ and the labels of the nodes above it form the set K then either
 - $\psi \in P \wedge K = \emptyset$ or
 - $\frac{K}{\psi}$ is an instance of a deduction rule of \mathcal{T} .

A formula φ is provable from \mathcal{T} , denoted $\mathcal{T} \vdash \varphi$ if and only if $\mathcal{T} \vdash (\emptyset, \varphi)$.

Definition 5 (Bisimulation). Let \mathcal{T} be a TSS with signature Σ . A relation $\mathcal{R} \subseteq \mathbb{C}(\Sigma) \times \mathbb{C}(\Sigma)$ is a bisimulation relation if and only if \mathcal{R} is symmetric and for all $p_0, p_1, p'_0 \in \mathbb{C}(\Sigma)$ and $l \in L$

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

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

It is easy to check that bisimilarity is indeed an equivalence. Bisimilarity can be extended to open terms by requiring that $t_0 \Leftrightarrow t_1$ when $\sigma(t_0) \Leftrightarrow \sigma(t_1)$ for all closing substitutions $\sigma : V \rightarrow \mathbb{C}(\Sigma)$. In the remainder of this paper, we restrict our attention to the notions of equivalence on *closed terms* that contain strong bisimilarity. However, all our results carry over (without any change) to the notions on *open terms* that contain strong bisimilarity on open terms in the above sense. Another notion of equivalence that we use in the remainder of this paper is isomorphism, as defined below.

Definition 6. (Isomorphism) Two closed terms p and q are isomorphic, denoted by $p \sim_i q$, when there exists a bijective function $h : \text{reach}(p) \rightarrow \text{reach}(q)$ such that $h(p) = q$ and if $h(p_0) = q_0$ and $h(p_1) = q_1$, then $p_0 \xrightarrow{l} p_1$ if and only if $q_0 \xrightarrow{l} q_1$, where $\text{reach}(p)$ is the smallest set satisfying $p \in \text{reach}(p)$ and if $p' \in \text{reach}(p)$ and $p' \xrightarrow{l} q'$, then $q' \in \text{reach}(p)$ (i.e., the set of closed terms reachable from p).

Definition 7 (Associativity). A binary operator $f \in \Sigma$ is associative w.r.t. an equivalence \sim on closed terms if and only if for each $p_0, p_1, p_2 \in \mathbb{C}(\Sigma)$, it holds that $f(p_0, f(p_1, p_2)) \sim f(f(p_0, p_1), p_2)$.

3 The ASSOC-De Simone Format

In this section, we first specify a limited number of rule types from the De Simone format. Then, we define a number of constraints on the occurrences of rules of such types that guarantee associativity of operators defined by such rules. The format is illustrated by means of a number of examples from the literature. The constraints are obtained by analyzing all proof structures that can be constructed using this restricted set of rule types. The proof of the meta-result, which is a detailed analysis of the proof structures described above, is given next.

3.1 Format

Definition 8 (The ASSOC-De Simone Rule Format). Consider the following types of rules which are all in the De Simone format. Let $\gamma : L \times L \rightarrow L$ be an associative partial function (w.r.t. syntactic equality).

1_l. Left-conforming rules

$$\frac{x \xrightarrow{l} x'}{f(x, y) \xrightarrow{l} f(x', y)}$$

3_l. Left-choice rules

$$\frac{x \xrightarrow{l} x'}{f(x, y) \xrightarrow{l} x'}$$

2_l. Right-conforming rules

$$\frac{y \xrightarrow{l} y'}{f(x, y) \xrightarrow{l} f(x, y')}$$

4_l. Right-choice rules

$$\frac{y \xrightarrow{l} y'}{f(x, y) \xrightarrow{l} y'}$$

5_l. Left-choice axioms 6_l. Right-choice axioms 7_(l₀, l₁). Communicating rules

$$\frac{}{f(x, y) \xrightarrow{l} x} \quad \frac{}{f(x, y) \xrightarrow{l} y} \quad \frac{x \xrightarrow{l_0} x' \quad y \xrightarrow{l_1} y'}{f(x, y) \xrightarrow{\gamma(l_0, l_1)} f(x', y')}$$

A TSS is in the ASSOC-De Simone format with respect to $f \in \Sigma$ when for each $l \in L$, each f -defining rule is of a type given above and the set of all f -defining rules satisfies the following constraints. (Each proposition P in the following constraints should be read as “there exists a deduction rule of type P in the set of f -defining rules”. To avoid repeated uses of parentheses, we assume that \vee and \wedge take precedence over \Rightarrow and \Leftrightarrow .)

1. $5_l \Rightarrow 2_l \wedge 3_l$,
2. $6_l \Rightarrow 1_l \wedge 4_l$,
3. $7_{(l, l')} \Rightarrow (1_l \Leftrightarrow 2_{l'}) \wedge (3_l \Leftrightarrow 4_{l'})$
 $\quad \wedge (2_l \Leftrightarrow 2_{\gamma(l, l')}) \wedge (4_l \Leftrightarrow 4_{\gamma(l, l')}) \wedge (1_{l'} \Leftrightarrow 1_{\gamma(l, l')}) \wedge (3_{l'} \Leftrightarrow 3_{\gamma(l, l')})$,
4. $1_l \wedge 3_l \Leftrightarrow \exists l' \gamma(l, l') = l \wedge 7_{(l, l')} \wedge 5_{l'} \wedge 6_{l'}$,
5. $2_l \wedge 4_l \Leftrightarrow \exists l' \gamma(l', l) = l \wedge 7_{(l', l)} \wedge 5_{l'} \wedge 6_{l'}$,
6. $(1_l \vee 4_l) \wedge (2_l \vee 3_l) \Rightarrow (5_l \Leftrightarrow 6_l)$.

The types of rules presented above give us a nice starting point while covering many practical applications. These rule types cover all rules that are in the De Simone format with four additional restrictions: Firstly, the target of the conclusion can contain at most one (binary) operator, secondly, the aforementioned operator is the same as the one appearing in the source, thirdly, the labels of the premises and the conclusion coincide (apart from the communicating rule), and finally testing is disallowed.

The main sources of complication in our format are pairs $1_l \wedge 3_l$ and $2_l \wedge 4_l$. If no such pairs are present in the TSS under consideration and moreover, label l generated by 5_l or 6_l cannot synchronize with other labels, i.e., $\nexists l' (5_l \vee 6_l) \wedge (7_{(l, l')} \vee 7_{(l', l)})$, then the last three constraints need not be checked. (The last constraint can be dropped in the light of the above-mentioned facts and constraints 1 and 2.) In all practical cases that we have encountered thus far, these conditions hold. Moreover, for each operator with a rule of type $7_{(l_0, l_1)}$, if the presence of an f -defining rule X_l , for $X \in \{1, 2, 3, 4\}$, implies the presence of $X_{l'}$ for all l' , then for such an operator, constraint 3 can be simplified to $7_{(l, l')} \Rightarrow ((1_{l''} \Leftrightarrow 2_{l''}) \wedge (3_{l''} \Leftrightarrow 4_{l''}))$. The former assumption on X_l holds for

most associative operators in practice but fails for few, such as CSP's parallel composition [16, Chapter 7]. Obviously for operators without defining rules of type $7_{(l_0, l_1)}$, constraint 3 is trivially satisfied.

Theorem 1. *For a TSS in the ASSOC-De Simone format with respect to $f \in \Sigma$, it holds that f is associative for each notion of equivalence \sim containing strong bisimilarity.*

3.2 Examples

In this section we illustrate the ASSOC-De Simone format by means of operators from the literature.

Example 1 (Alternative composition). In most process languages for the description of sequential and parallel systems a form of alternative composition or choice is present. Here we present nondeterministic alternative composition as present in CCS [11] and ACP [4].

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

In this example we find deduction rules of types 3 and 4. Therefore, the requirements are met and it can be concluded that alternative composition is associative.

Example 2 (Parallel composition). Another frequently occurring associative operator is parallel composition. It appears in amongst others ACP, CCS, and CCS. Here we discuss parallel composition with communication in the style of ACP [4], for which the others are special cases. It is assumed that an associative (and commutative) partial function γ on labels is given that defines the result of communication and determines the absence or presence of the right-most rule.

$$\frac{x \xrightarrow{l} x'}{x \parallel y \xrightarrow{l} x' \parallel y} \quad \frac{y \xrightarrow{l} y'}{x \parallel y \xrightarrow{l} x \parallel y'} \quad \frac{x \xrightarrow{l} x' \quad y \xrightarrow{l'} y'}{x \parallel y \xrightarrow{\gamma(l, l')} x' \parallel y'}$$

Thus, in terms of the types of deduction rules, we have deduction rules of type 1, 2, and 7. Therefore, the requirements are met and it can be concluded that parallel composition is associative.

Example 3 (Disrupt). The disrupt is originally introduced in the language LOTOS [6], where it is used to model for example exception handling. Also, it is used, for example in [2], for the description of mode switches.

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

Here we see that only deduction rules of types 1 and 4 are present. As a consequence also disrupt is associative.

Example 4 (External choice). The external choice operator \square from CSP [16] has the following deduction rules, where $l \neq \tau$:

$$\frac{x \xrightarrow{\tau} x'}{x \square y \xrightarrow{\tau} x' \square y} \quad \frac{y \xrightarrow{\tau} y'}{x \square y \xrightarrow{\tau} x \square y'} \quad \frac{x \xrightarrow{l} x'}{x \square y \xrightarrow{l} x'} \quad \frac{y \xrightarrow{l} y'}{x \square y \xrightarrow{l} y'}$$

These are of the types 1, 2, 3 and 4, respectively. The constraints of the ASSOC-De Simone format are satisfied since the rules of type 1 and 3 (and 2 and 4) are only there for different labels. Therefore, external choice is associative.

3.3 Proof of Theorem 1

We start with the following auxiliary definition, which will be used in the remainder of our proof.

Definition 9. (*Syntactic Equality Modulo Associativity*) *Equality modulo associativity of an operator f , denoted by \simeq_f , is the smallest reflexive and symmetric relation satisfying $f(p_0, f(p_1, p_2)) \simeq_f f(f(p_0, p_1), p_2)$, for each $p_0, p_1, p_2 \in \mathbb{C}(\Sigma)$.*

The following lemma gives us a stronger thesis from which the theorem follows.

Lemma 1. *An operator $f \in \Sigma$ is associative w.r.t. \sim if \simeq_f is a bisimulation relation.*

Proof. If we prove that \simeq_f is a bisimulation relation, then the theorem follows, because we then have that $f(p_0, f(p_1, p_2)) \leftrightarrow f(f(p_0, p_1), p_2)$ and from $\leftrightarrow \subseteq \sim$, it follows that $f(p_0, f(p_1, p_2)) \sim f(f(p_0, p_1), p_2)$.

To obtain that \simeq_f is a bisimulation relation, we construct all possible proof structures Pr with $f(p_0, f(p_1, p_2)) \xrightarrow{l} p'$ as conclusion using only rules of the types given in Definition 8. We then show that for each such proof, there is a proof Pr' with $f(f(p_0, p_1), p_2) \xrightarrow{l} p''$ as a conclusion for some p'' such that $p' \simeq_f p''$ and that Pr' uses the premises of Pr . We do the same thing the other way around, so we may conclude that \simeq_f is a bisimulation. By lemma 1 we then conclude that f is associative.

To be able to show the results in a compact way, we introduce the following acronym for proof structures. Let us denote the instantiation of rule r ($r \in \{1, \dots, 7\}$) with n premises with $r(r_1, r_2, \dots, r_n)$, where r_i is the rule that is instantiated on premise i . Furthermore, if no rule is instantiated then we denote this with the symbol ‘-’, to indicate that no rule is used.

Now the proof for a transition $t \xrightarrow{l} c$ can be denoted with an expression $r \cdot (r_1, r_2, \dots, r_n)$ when the conclusion of r matches $t \xrightarrow{l} c$ and premise i of rule r matches the conclusion of rule r_i . When the rule used to derive a proof for premises r_i is not relevant, we may write $r \cdot (r_1, \dots, r_{i-1}, -, \dots, r_n)$. For readability and when no confusion may arise, we write $r \cdot r_1$ for $r(-, r_1)$, $r(r_1, -)$ or $r \cdot (r_1)$, and write r for $r(-, -)$, $r \cdot (-)$, or $r \cdot ()$.

Example 5. A proof for the derived deduction rule

$$\frac{x \xrightarrow{l_0} x' \quad z \xrightarrow{l_1} z'}{f(x, f(y, z)) \xrightarrow{\gamma(l_0, l_1)} f(x', f(y, z'))}$$

is the following

$$\frac{x \xrightarrow{l_0} x' \quad \frac{z \xrightarrow{l_1} z'}{f(y, z) \xrightarrow{l_1} f(y, z')}}{f(x, f(y, z)) \xrightarrow{\gamma(l_0, l_1)} f(x', f(y, z'))} .$$

It consists of an instantiation of rule type $7_{(l_0, l_1)}$ and one of type 2_{l_1} , and may therefore be written as $7_{(l_0, l_1)} \cdot (-, 2_{l_1})$ or simply $7_{(l_0, l_1)} \cdot 2_{l_1}$.

| T_r | T_l | c_r | c_l | further req. |
|--|--|--------------------|--------------------|---|
| 1_l | $1_l \cdot 1_l$ | $f(x', f(y, z))$ | $f(f(x', y), z)$ | |
| $2_l \cdot 1_l$ | $1_l \cdot 2_l$ | $f(x, f(y', z))$ | $f(f(x, y'), z)$ | |
| $2_l \cdot 2_l$ | 2_l | $f(x, f(y, z'))$ | $f(f(x, y), z')$ | |
| $2_l \cdot 3_l$ | $3_l \cdot 2_l$ | $f(x, y')$ | $f(x, y')$ | |
| $2_l \cdot 4_l$ | $7_{(l', l)} \cdot 5_{l'}$ | $f(x, z')$ | $f(x, z')$ | $\gamma(l', l) = l$ |
| $2_l \cdot 5_l$ | 5_l | $f(x, y)$ | $f(x, y)$ | |
| $2_l \cdot 6_l$ | $1_l \cdot 5_l$ | $f(x, z)$ | $f(x, z)$ | |
| $2_{\gamma(l, l')} \cdot 7_{(l, l')}$ | $7_{(l, l')} \cdot 2_l$ | $f(x, f(y', z'))$ | $f(f(x, y'), z')$ | |
| 3_l | $3_l \cdot 3_l$ | x' | x' | |
| $4_l \cdot 1_l$ | $1_l \cdot 4_l$ | $f(y', z)$ | $f(y', z)$ | |
| $4_l \cdot 2_l$ | $7_{(l', l)} \cdot 6_{l'}$ | $f(y, z')$ | $f(y, z')$ | $\gamma(l', l) = l$ |
| $4_l \cdot 3_l$ | $3_l \cdot 4_l$ | y' | y' | |
| $4_l \cdot 4_l$ | 4_l | z' | z' | |
| $4_l \cdot 5_l$ | $3_l \cdot 6_l$ | y | y | |
| $4_l \cdot 6_l$ | 6_l | z | z | |
| $4_{\gamma(l, l')} \cdot 7_{(l, l')}$ | $7_{(l, l')} \cdot 4_l$ | $f(y', z')$ | $f(y', z')$ | |
| 5_l | $3_l \cdot 5_l$ | x | x | |
| 6_l | $1_l \cdot 6_l$ | $f(y, z)$ | $f(y, z)$ | |
| $7_{(l, l')} \cdot 1_{l'}$ | $1_{\gamma(l, l')} \cdot 7_{(l, l')}$ | $f(x', f(y', z))$ | $f(f(x', y'), z)$ | |
| $7_{(l, l')} \cdot 2_{l'}$ | $7_{(l, l')} \cdot 1_l$ | $f(x', f(y, z'))$ | $f(f(x', y), z')$ | |
| $7_{(l, l')} \cdot 3_{l'}$ | $3_{\gamma(l, l')} \cdot 7_{(l, l')}$ | $f(x', y')$ | $f(x', y')$ | |
| $7_{(l, l')} \cdot 4_{l'}$ | $7_{(l, l')} \cdot 3_l$ | $f(x', z')$ | $f(x', z')$ | |
| $7_{(l, l')} \cdot 5_{l'}$ | $3_l \cdot 1_l$ | $f(x', y)$ | $f(x', y)$ | $\gamma(l, l') = l$ |
| $7_{(l, l')} \cdot 6_{l'}$ | $1_l \cdot 3_l$ | $f(x', z)$ | $f(x', z)$ | $\gamma(l, l') = l$ |
| $7_{(l, \gamma(l_0, l_1))} \cdot 7_{(l_0, l_1)}$ | $7_{(\gamma(l, l_0), l_1)} \cdot 7_{(l, l_0)}$ | $f(x', f(y', z'))$ | $f(f(x', y'), z')$ | $\gamma(l, \gamma(l_0, l_1)) = \gamma(\gamma(l, l_0), l_1)$ |

Table 1. All proofs with $f(x, f(y, z))$ or $f(f(x, y), z)$ in the source of the conclusion.

Table 1 shows all the mentioned proofs in an abbreviated way. Column T_r lists the proof structure of proofs with the source of the conclusion $f(x, f(y, z))$,

T_l lists the proof structures of those with source of the conclusion $f(f(x, y), z)$. The corresponding targets of the conclusions are listed in columns c_r and c_l .

What is not listed in the table, but what is relevant to the proof of correctness, are the labels of the conclusion transition. These labels are always equal except for the last row, where the labels are $\gamma(l, \gamma(l_0, l_1))$ and $\gamma(\gamma(l, l_0), l_1)$, respectively. By associativity of the function γ these labels are also equal.

We have proved f to be associative if our format guarantees that whenever the rules needed for a proof in the T_r column are present, then the rules needed for the corresponding proof in T_l are present too and the other way around. This is trivially true for those rows in the table where the sets of rules used to construct the proof are the same. In Table 2 we have eliminated these rows. This yields the requirements, listed in the column ‘To Prove’, on the presence of certain rules. In the column ‘Follows From’, we indicate which constraints in Definition 8 discharge the proof obligations in the ‘To Prove’ column.

| T_r | T_l | To Prove | Follows From |
|--|--|---|---|
| $2_l \cdot 4_l$ | $7_{(\nu, l)} \cdot 5_l$ | $(2_l \wedge 4_l) \Leftrightarrow (\exists \nu 7_{(\nu, l)} \wedge 5_{\nu'})$ | Constraint 5 |
| $2_l \cdot 5_l$ | 5_l | $5_l \Rightarrow 2_l$ | Constraint 1 |
| $2_l \cdot 6_l$ | $1_l \cdot 5_l$ | $(2_l \wedge 6_l) \Leftrightarrow (1_l \wedge 5_l)$ | \Rightarrow Constraints 6, 2 \Leftarrow Constraints 6, 1 |
| $2_{\gamma(l, \nu')} \cdot 7_{(l, \nu')}$ | $7_{(l, \nu')} \cdot 2_l$ | $7_{(l, \nu')} \Rightarrow (2_l \Leftrightarrow 2_{\gamma(l, \nu')})$ | Constraint 3 |
| $4_l \cdot 2_l$ | $7_{(\nu', l)} \cdot 6_{\nu'}$ | $(2_l \wedge 4_l) \Leftrightarrow (7_{(\nu', l)} \wedge 6_{\nu'})$ | Constraint 5 |
| $4_l \cdot 5_l$ | $3_l \cdot 6_l$ | $(4_l \wedge 5_l) \Leftrightarrow (3_l \wedge 6_l)$ | \Leftarrow Constraints 6, 1 \Rightarrow Constraints 6, 2 |
| $4_l \cdot 6_l$ | 6_l | $6_l \Rightarrow 4_l$ | Constraint 2 |
| $4_{\gamma(l, \nu')} \cdot 7_{(l, \nu')}$ | $7_{(l, \nu')} \cdot 4_l$ | $7_{(l, \nu')} \Rightarrow (4_l \Leftrightarrow 4_{\gamma(l, \nu')})$ | Constraint 3 |
| 5_l | $3_l \cdot 5_l$ | $5_l \Rightarrow 3_l$ | Constraint 1 |
| 6_l | $1_l \cdot 6_l$ | $6_l \Rightarrow 1_l$ | Constraint 2 |
| $7_{(l, \nu')} \cdot 1_{\nu'}$ | $1_{\gamma(l, \nu')} \cdot 7_{(l, \nu')}$ | $7_{(l, \nu')} \Rightarrow (1_{\nu'} \Leftrightarrow 1_{\gamma(l, \nu')})$ | Constraint 3 |
| $7_{(l, \nu')} \cdot 2_{\nu'}$ | $7_{(l, \nu')} \cdot 1_l$ | $7_{(l, \nu')} \Rightarrow (1_l \Leftrightarrow 2_{\nu'})$ | Constraint 3 |
| $7_{(l, \nu')} \cdot 3_{\nu'}$ | $3_{\gamma(l, \nu')} \cdot 7_{(l, \nu')}$ | $7_{(l, \nu')} \Rightarrow (3_{\nu'} \Leftrightarrow 3_{\gamma(l, \nu')})$ | Constraint 3 |
| $7_{(l, \nu')} \cdot 4_{\nu'}$ | $7_{(l, \nu')} \cdot 3_l$ | $7_{(l, \nu')} \Rightarrow (3_l \Leftrightarrow 4_l)$ | Constraint 3 |
| $7_{(l, \nu')} \cdot 5_{\nu'}$ | $3_l \cdot 1_l$ | $7_{(l, \nu')} \cdot 5_{\nu'} \Leftrightarrow 3_l \cdot 1_l$ | Constraint 4 |
| $7_{(l, \nu')} \cdot 6_{\nu'}$ | $1_l \cdot 3_l$ | $7_{(l, \nu')} \cdot 6_{\nu'} \Leftrightarrow 1_l \cdot 3_l$ | Constraint 4 |
| $7_{(l, \gamma(l_0, l_1))} \cdot 7_{(l_0, l_1)}$ | $7_{(\gamma(l, l_0), l_1)} \cdot 7_{(l, l_0)}$ | $7_{(l, \gamma(l_0, l_1))} \wedge 7_{(l_0, l_1)} \Leftrightarrow 7_{(\gamma(l, l_0), l_1)} \wedge 7_{(l, l_0)}$ | Associativity of γ |

Table 2. Table 1 without the trivial cases.

3.4 Counter-Examples

The seven basic types of rules that are allowed by the ASSOC-De Simone format are more restrictive than arbitrary rules in the De Simone format in two respects. First, the De Simone format allows for complex terms as the target of

the conclusion. However, we only allow for either a variable or applications of the operator being defined (i.e., appearing in the source of the conclusion) on variables. Second, for rules of the first six types, the premise has the same label as the conclusion. In this section, we show that dropping the first restriction jeopardizes our associativity meta-result even with respect to trace equivalence, which is one of the weakest notions of behavioral equivalence. The following two counter-examples witness that we cannot trivially relax this condition. The first counter-example uses an unary function and the second one uses an associative binary operator (different from the one being defined).

Example 6 (Complex target, I). Consider the terms $f(a, f(a, a))$ and $f(f(a, a), a)$ w.r.t. the following TSS

$$\overline{f(x, y) \xrightarrow{a} g(x)} \quad \overline{g(x) \xrightarrow{b} x}$$

The term $f(a, f(a, a))$ can make an a -transition followed by a b -transition and then it deadlocks. However, the $f(f(a, a), a)$ term can make two consecutive ab -traces: $f(f(a, a), a) \xrightarrow{a} g(f(a, a)) \xrightarrow{b} f(a, a) \xrightarrow{a} g(a) \xrightarrow{b} b$.

Example 7 (Complex target, II). Consider the following TSS with the signature containing two constants 0 and a and binary operators f and g (which respectively represent left-merge and parallel-composition operators).

$$\begin{array}{cc} \text{(a)} \frac{}{a \xrightarrow{a} 0} & \text{(f)} \frac{x \xrightarrow{a} x'}{f(x, y) \xrightarrow{a} g(x', y)} \\ \text{(g0)} \frac{x \xrightarrow{a} x'}{g(x, y) \xrightarrow{a} g(x', y)} & \text{(g1)} \frac{y \xrightarrow{a} y'}{g(x, y) \xrightarrow{a} g(x, y')} \end{array}$$

Consider the terms $f(a, f(0, a))$ and $f(f(a, 0), a)$. The former term can only make an a -transition into $g(0, f(0, a))$ (by the proof structure $\mathbf{f} \cdot \mathbf{a}$); the target of this transition, in turn, deadlocks. The latter term can only make an a -transition into $g(g(0, 0), a)$ (by the proof structure $\mathbf{g0} \cdot \mathbf{g0} \cdot \mathbf{a}$); however the target of this transition can make one further a -transition into $g(g(0, 0), 0)$.

The restriction of the treatment to deduction rules without relabeling (excluding the communicating rule) is not essential, but allows for a simpler presentation of the format. It remains future work to formulate and prove a rule format for associativity in the presence of relabeling.

In Section 4, we extend our format by introducing testing in the premises and predicates.

4 Possible Extensions

In this section, we investigate extensions of our format in various directions.

4.1 Testing Rules

De Simone format does not allow for premises of which the targets do not appear in the target of the conclusion. This phenomenon is called testing in the SOS literature and, albeit disallowed by De Simone format, is of practical relevance, e.g., in modeling predicates. Thus, in this section, we introduce the concept of testing to our ASSOC-De Simone format to cover these practical cases. The only relevant type of testing which allows us to model predicates is given by the following type of rules. As we demonstrate in Section 4.3, other sorts of testing (predicate) rules can be coded using a combination of rules already present in our ASSOC-De Simone format.

$$\begin{array}{c}
 8_{(l,l')}. \text{ Left-choice + test} \\
 \frac{x \xrightarrow{l} x' \quad y \xrightarrow{l'} y'}{f(x,y) \xrightarrow{l} x'}
 \end{array}
 \qquad
 \begin{array}{c}
 9_{(l',l)}. \text{ Right-choice + test} \\
 \frac{x \xrightarrow{l'} x' \quad y \xrightarrow{l} y'}{f(x,y) \xrightarrow{l} y'}
 \end{array}$$

The constraints we give next for the above two types of testing rules can be generalized to arbitrary testing rules (with relabeling and changing operators).

Definition 10 (The ASSOC-De Simone Rule Format with Testing). *A TSS is in the ASSOC-De Simone format with testing w.r.t. f , when each defining rule is of one of the previously given types, the rules of types $\{1_l, \dots, 6_l, 7_{(l,l')}\}$ satisfy the constraints of the ASSOC-De Simone format w.r.t. f and moreover, the following constraints hold for the rules of types $8_{(l,l')}$ and $9_{(l,l')}$:*

1. $8_{(l,l')} \wedge X_p \Rightarrow 3_l$ and $9_{(l',l)} \wedge X_p \Rightarrow 4_l$, for each $X_p \in \{1_{l'}, \dots, 6_{l'}, 7_{(l_0,l_1)} \mid \gamma(l_0, l_1) = l' \wedge l_0 \neq l' \vee l_1 \neq l'\}$,
2. $(8_{(l,l')} \wedge 1_l) \Rightarrow \exists l'' \gamma(l'', l) = l \wedge 7_{(l'',l)} \wedge 5_{l''}$ and $(9_{(l',l)} \wedge 2_l) \Rightarrow \exists l'' \gamma(l'', l) = l \wedge 7_{(l'',l)} \wedge 5_{l''}$,
3. $8_{(l,l')} \wedge 6_l \Rightarrow 5_l$ and $9_{(l',l)} \wedge 5_l \Rightarrow 6_l$,
4. $7_{(l_0,l_1)} \Rightarrow (8_{(l_1,l')} \Leftrightarrow 8_{(\gamma(l_0,l_1),l')}) \wedge (8_{(l_0,l')} \Leftrightarrow 9_{(l',l_1)}) \wedge (9_{(l',\gamma(l_0,l_1))} \Leftrightarrow 9_{(l',l_0)})$,
5. $8_{(l,l')} \wedge 8_{(l,l'')} \Rightarrow 8_{(l',l'')} \vee (7_{(l',l'')} \wedge 8_{(l,\gamma(l',l''))})$ and $9_{(l',l)} \wedge 9_{(l',l'')} \Rightarrow 9_{(l',l'')} \vee (7_{(l',l'')} \wedge 9_{(\gamma(l',l''),l)})$.

Theorem 2. *For a TSS in the ASSOC-De Simone format with testing w.r.t. $f \in \Sigma$, it holds that f is associative for each notion of equivalence \sim containing strong bisimilarity.*

Due to space limitations, proof of the theorem is omitted. Later in Section 4.3, we show how rules of type 8 and 9 can be used to obtain associativity of operators in the definition of which predicates are involved, e.g., sequential composition operator.

4.2 Changing Operators

In the ASSOC-De Simone format, the only operator that may appear in the target of the conclusion is the same as the operator appearing in the source.

This assumption may not hold in practice and is not essential to our meta-result, either. Thus, in this section, we relax this assumption and allow for rules of the following shape.

$$\begin{array}{ccc}
1_l. \text{ Left-conf. rules} & 2_l. \text{ Right-conf. rules} & 7_{(l_0, l_1)}. \text{ Comm. rules} \\
\frac{x \xrightarrow{l} x'}{f(x, y) \xrightarrow{l} g(x', y)} & \frac{y \xrightarrow{l} y'}{f(x, y) \xrightarrow{l} g(x, y')} & \frac{x \xrightarrow{l_0} x' \quad y \xrightarrow{l_1} y'}{f(x, y) \xrightarrow{\gamma^{(l_0, l_1)}} g(x', y')}
\end{array}$$

Note that if g is taken to be the same as f , then we recover the original types of rules allowed in the ASSOC-De Simone format.

Definition 11 (The ASSOC-De Simone Rule Format with Changing Operators). Let $\Sigma' \subseteq \Sigma$. A TSS is in the ASSOC-De Simone format with changing operators w.r.t. Σ' when for each $f \in \Sigma'$ the following constraints are satisfied:

- the f -defining rules are in the ASSOC-De Simone format (with relaxed targets of conclusions as described above),
- for each $l \in L$, the (f, l) -defining rules of type 7 all have the same operator $g \in \Sigma'$ in the target of the conclusion, and there are no (f, l) -defining rules of type 1 or 2 with $f \neq g$, where g is the operator appearing in the target of the conclusion.

Theorem 3. For a TSS in the ASSOC-De Simone format with changing operators w.r.t. $\Sigma' \in \Sigma$, it holds that each $f \in \Sigma'$ is associative for each notion of equivalence \sim containing strong bisimilarity.

Proof. The proof is obtained by adapting Table 1 for those occurrences of rules of types 1, 2 and 7 with a changing operator, say g . The targets of the conclusions, i.e., c_l and c_r then stay the same terms with all occurrences of f replaced by g .

Example 8 (Communication merge). Consider the communication merge operator $|$ from ACP [4] with the following deduction rule

$$\frac{x \xrightarrow{l} x' \quad y \xrightarrow{l'} y'}{x | y \xrightarrow{\gamma^{(l, l')}} x' || y'}$$

and additionally those of the parallel composition operator from Example 2. Recall that the communication function γ is assumed to be associative. Note that this rule is of type $7_{(l, l')}$. The constraints of the ASSOC-De Simone format with changing operators are satisfied. Thus, as a consequence, communication merge is associative.

4.3 Predicates

Assume that a predicate \mathcal{P} is given by deduction rules of the following form.

$$\frac{\mathcal{P}_x}{\mathcal{P}_{f(x, y)}} \quad \frac{\mathcal{P}_y}{\mathcal{P}_{f(x, y)}} \quad \frac{\mathcal{P}_x \quad \mathcal{P}_y}{\mathcal{P}_{f(x, y)}}$$

In order to accommodate such predicates in our framework and our format, we use a translation inspired by [17]. The following deduction rules of types $1_{\mathcal{P}}$, $2_{\mathcal{P}}$, and $7_{(\mathcal{P},\mathcal{P})}$, respectively, code predicate \mathcal{P} in the ASSOC-De Simone format.

$$\frac{x \xrightarrow{\mathcal{P}} x'}{f(x,y) \xrightarrow{\mathcal{P}} f(x',y)} \quad \frac{y \xrightarrow{\mathcal{P}} y'}{f(x,y) \xrightarrow{\mathcal{P}} f(x,y')} \quad \frac{x \xrightarrow{\mathcal{P}} x' \quad y \xrightarrow{\mathcal{P}} y'}{f(x,y) \xrightarrow{\mathcal{P}} f(x',y')}$$

Other possible types of rules for defining predicates can be coded similarly inside the ASSOC-De Simone rule format. The major difficulty here is in combining predicates with transitions. This can be done in our framework, using either communicating rules (type $7_{(l,l')}$) or testing rules (types $8_{(l,l')}$ or $9_{(l,l')}$). The following example illustrates this.

Example 9 (Sequential Composition). Consider the following deduction rules defining the sequential composition operator.

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

The second deduction rule uses the termination predicate as a premise. Translation of the later two deduction rules to a setting without predicates gives

$$\frac{x \xrightarrow{\downarrow} x' \quad y \xrightarrow{l} y'}{x \cdot y \xrightarrow{l} y'} \quad \frac{x \xrightarrow{\downarrow} x' \quad y \xrightarrow{\downarrow} y'}{x \cdot y \xrightarrow{\gamma(\downarrow,\downarrow)} x' \cdot y'}$$

with $\gamma(\downarrow,\downarrow) = \downarrow$ and undefined otherwise. These rules are of type 1_l , $9_{(\downarrow,l)}$, and $7_{(\downarrow,\downarrow)}$. In Definition 10, the only constraints of which the left-hand-side of the implications hold are 4 and 5. Thus, we only need to check that $(8_{(\downarrow,l')} \Leftrightarrow 8_{(\downarrow,l')} \wedge (8_{(\downarrow,l')} \Leftrightarrow 9_{(l',\downarrow)}) \wedge (9_{(l',\downarrow)} \Leftrightarrow 9_{(l',\downarrow)})$ and $9_{(\downarrow,\downarrow)} \vee (7_{(\downarrow,\downarrow)} \wedge 9_{(\downarrow,l)})$. The former holds trivially since none of the propositions appearing in the three bimplications hold. The latter holds, as well, because we have that $7_{(\downarrow,\downarrow)} \wedge 9_{(\downarrow,l)}$. Thus, we can conclude that all constraints of the ASSOC-De Simone format with testing are satisfied and hence sequential composition is associative.

5 Associativity for Isomorphism

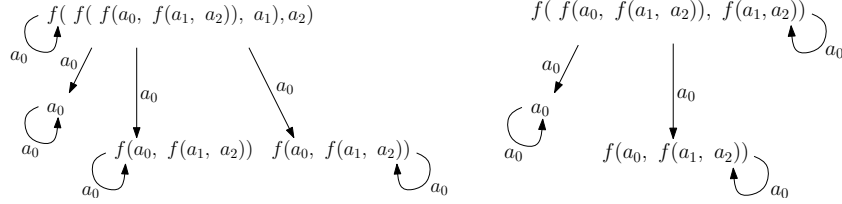
Although associativity w.r.t. strong bisimilarity provides us with a strong meta-result that is capable of dealing with all applications in practice, an even stronger result can be obtained, if we prove associativity w.r.t. isomorphism as given in Definition 6.

In this section, we first show that our meta-result does not trivially carry over to the case where isomorphism is considered as the notion of equivalence. Then, we seek extra conditions under which associativity w.r.t. isomorphism indeed holds. The following example shows why the ASSOC-De Simone format cannot be used as is for proving associativity w.r.t. isomorphism.

Example 10 (Associativity w.r.t. Isomorphism). Consider the following TSS.

$$\begin{array}{c}
(\mathbf{a}_i) \frac{}{a_i \xrightarrow{a_i} a_i} \quad (\alpha, \beta) \frac{x \xrightarrow{\alpha} x' \quad y \xrightarrow{\beta} y'}{f(x, y) \xrightarrow{\gamma(\alpha, \beta)} f(x', y')} \\
(\mathbf{la}_0) \frac{x \xrightarrow{a_0} x'}{f(x, y) \xrightarrow{a_0} x'} \quad (\mathbf{ra}_0) \frac{y \xrightarrow{a_0} y'}{f(x, y) \xrightarrow{a_0} y'}
\end{array}
\quad
\begin{array}{|c|c|c|}
\hline
\alpha & \beta & \gamma(\alpha, \beta) \\
\hline
a_0 & a_1 & a' \\
\hline
a_1 & a_2 & a' \\
\hline
a_0 & a' & a_0 \\
\hline
a' & a_2 & a_0 \\
\hline
\end{array}$$

where rule (\mathbf{a}_i) is present only for $i = 0, 1, 2$ and rule (α, β) is only defined for all pairs of α and β for which the table on the right provides an entry. Next, the LTS's of the terms $f(f(f(a_0, f(a_1, a_2)), a_1), a_2)$ and $f(f(a_0, f(a_1, a_2)), f(a_1, a_2))$ are depicted. Note that these two LTS's are not isomorphic since one of them comprises three states and the other one comprises four states.



Theorem 4 gives sufficient conditions for associativity w.r.t. isomorphism.

Theorem 4. *For a TSS in the ASSOC-De Simone format w.r.t. $f \in \Sigma$ such that, disregarding the labels, the set of all f -defining rules satisfies $(1 \vee 2 \vee 7) \Leftrightarrow \neg(3 \vee 4 \vee 5 \vee 6 \vee 8 \vee 9)$ (all f -defining rules are either of types 1, 2 and 7, or are all of the other types), then f' is associative w.r.t. isomorphism.*

Proof. The proviso of Theorem 4 requires that the f -defining rules are either of the types 1,2,7 or of the types 3,4,5,6,8,9. We call the deduction rules of the first type f -preserving and those of the second type f -eliminating.

If all f -defining rules are f -preserving, then all states in the LTS of $f(p_0, f(p_1, p_2))$ are of the form $f(q_0, f(q_1, q_2))$. Then, define $h(f(q_0, f(q_1, q_2))) \doteq f(f(q_0, q_1), q_2)$. Then, it is straightforward to check (by consulting the corresponding rows of Table 1) that h is the bijective function satisfying the constraints of Definition 6.

If all f -defining rules are f -eliminating, then define $h(f(p_0, f(p_1, p_2))) \doteq f(f(p_0, p_1), p_2)$ (for the initial state) and $f(p') = p'$ for all nodes reachable from the initial state. Note that the initial state cannot have a self-loop, because the size of the term strictly decreases in each transition. Thus, the above definition gives rise to a (function and a) bijection. It is also straightforward to check that in the rows of Table 1 when the applied rules are all f -eliminating, then the targets of the transitions are syntactically equal and thus our bijection satisfies the constraints of Definition 6.

6 Conclusions

In the context of binary operators specified by means of deduction rules in the well-known De Simone format, we developed a rule format guaranteeing associa-

tivity w.r.t. any notion of behavioral equivalence containing strong bisimilarity. The format is adapted for the setting with predicates, testing rules and ‘changing operators’. Applicability of the format is shown by means of examples from literature.

We plan to extend the ASSOC-De Simone format to deal with relabeling and negative premises [5]. Another direction for future research is a more general format in the setting of weak equivalences such as branching bisimilarity and weak bisimilarity. An extension of the ASSOC-De Simone format to deal with a notion of state/date is also anticipated (see [13]).

Acknowledgment. Insightful comments of CONCUR’08 reviewers led to a number of improvements and are gratefully acknowledged.

References

1. L. Aceto, W. Fokkink, and C. Verhoef. Structural Operational Semantics. In *Handbook of Process Algebra, Chapter 3*, pages 197–292. Elsevier, 2001.
2. J.C.M. Baeten and J.A. Bergstra. Mode transfer in process algebra. Technical Report CSR-00-01, Dept. of Computer Science, TU/Eindhoven, 2000.
3. J.C.M. Baeten and C. Verhoef. A congruence theorem for structured operational semantics with predicates. In *Proc. of CONCUR’93*, volume 715 of *LNCS*, pages 477–492. Springer, 1993.
4. J.C.M. Baeten and W.P. Weijland. *Process Algebra*. Cambridge, 1990.
5. R. Bol and J.F. Groote. The meaning of negative premises in transition system specifications. *JACM*, 43(5):863–914, September 1996.
6. E. Brinksma. A tutorial on LOTOS. In *Proc. of Protocol Specification, Testing and Verification V*, pages 171–194. North-Holland, 1985.
7. R. de Simone. Higher-level synchronizing devices in MEIJE-SCCS. *TCS*, 37:245–267, 1985.
8. R.J. van Glabbeek. The linear time - branching time spectrum I. In *Handbook of Process Algebra, Chapter 1*, pages 3–99. Elsevier, 2001.
9. R.J. van Glabbeek. The linear time - branching time spectrum II. In *Proc. of CONCUR’93*, volume 715 of *LNCS*, pages 66–81. Springer, 1993.
10. I. Hasuo, B. Jacobs, and A. Sokolova. The microcosm principle and concurrency in coalgebra. In *Proc. of FOSSACS’08*, LNCS. Springer, 2008. To appear.
11. A.J.R.G. Milner. *A Calculus of Communicating Systems*, volume 92 of *LNCS*. Springer, 1980.
12. M.R. Mousavi, M.A. Reniers, and J.F. Groote. A syntactic commutativity format for SOS. *IPL*, 93:217–223, 2005.
13. M.R. Mousavi, M.A. Reniers, and J.F. Groote. Notions of bisimulation and congruence formats for SOS with data. *I&C*, 200(1):107–147, 2005.
14. M.R. Mousavi, M.A. Reniers, and J.F. Groote. SOS formats and meta-theory: 20 years after. *TCS*, (373):238–272, 2007.
15. G.D. Plotkin. A structural approach to operational semantics. *JLAP*, 60:17–139, 2004.
16. A.W. Roscoe. *The Theory and Practice of Concurrency*. Prentice Hall, 1997.
17. C. Verhoef. A congruence theorem for structured operational semantics with predicates and negative premises. *Nordic Journal of Computing*, 2(2):274–302, 1995.