Rule Formats for Distributivity*

Luca Aceto[1], Matteo Cimini[1], Anna Ingolfsdottir[1],
MohammadReza Mousavi[2], and Michel A. Reniers[3]

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

Abstract. This paper proposes rule formats for Structural Operational Semantics guaranteeing that certain binary operators are left distributive with respect to a set of binary operators. Examples of left-distributivity laws from the literature are shown to be instances of the provided formats.

1 Introduction

Over the last three decades, Structural Operational Semantics (SOS), see, e.g., [7, 20, 22], has proven to be a powerful way to specify the semantics of programming and specification languages. In this approach to semantics, languages can be given a clear behaviour in terms of states and transitions, where the collection of transitions is specified by means of a set of syntax-driven inference rules. This behavioural description of the semantics of a language essentially tells one how the expressions in the language under definition behave when run on an idealized abstract machine.

Designers of languages often have expected algebraic properties of language constructs in mind when defining a language. For example, one expects that a sequential composition operator be associative and, in the field of process algebra [11, 16, 17], operators such as nondeterministic and parallel composition are often meant to be commutative and associative with respect to bisimilarity. Once the semantics of a language has been given in terms of state transitions, a natural question to ask is whether the intended algebraic properties do hold modulo the notion of behavioural equivalence or preorder of interest. The typical approach to answer this question is to perform an *a posteriori verification*: based

* The work of Aceto, Cimini and Ingolfsdottir has been partially supported by the projects 'New Developments in Operational Semantics' (nr. 080039021) and 'Metatheory of Algebraic Process Theories' (nr. 100014021) of the Icelandic Research Fund. The work on the paper was partly carried out while Luca Aceto held an Abel Extraordinary Chair at Universidad Complutense de Madrid, Spain, supported by the NILS Mobility Project.

on the semantics in terms of state transitions, one proves the validity of the desired algebraic laws, which describe semantic properties of the various operators in the language. An alternative approach is to ensure the validity of algebraic properties *by design*, using the so called *SOS rule formats* [2]. In this approach, one gives *syntactic templates* for the inference rules used in defining the operational semantics for certain operators that guarantee the validity of the desired laws by design. Not surprisingly, the definition of rule formats is based on finding a reasonably good trade-off between generality and ease of application. On the one hand, one strives to define a rule format that can capture as many examples from the literature as possible, including ones that may arise in the future. On the other, the rule format should be as easy to apply as possible and, preferably, the syntactic constraints of the format should be algorithmically checkable.

The literature on SOS provides rule formats for basic algebraic properties of operators such as commutativity [19], associativity [15], idempotence [3] and the existence of unit and zero elements [5, 8]. The main advantage of this approach is that one is able to verify the desired property by syntactic checks that can be mechanized. Moreover, it is interesting to use rule formats for establishing semantic properties since the results so obtained apply to a broad class of languages. Apart from providing one with an insight as to the semantic nature of algebraic properties and its link to the syntax of SOS rules, rule formats like those presented in the above-mentioned references may serve as a guideline for language designers who want to ensure, a priori, that the constructs under design enjoy certain basic algebraic properties.

In the present paper, we develop two rule formats guaranteeing that certain binary operators are left distributive with respect to others modulo bisimilarity. A binary operator $\circledast$ is *left distributive* with respect to a binary operator $\oplus$, modulo some notion of behavioural equivalence, whenever the equation $(x \oplus y) \circledast z = (x \circledast z) \oplus (y \circledast z)$ holds.

A classic example of left-distributivity law within the realm of process algebra is $(x + y) \between z = (x \between z) + (y \between z)$, where '+' and '$\between$' stand for nondeterministic choice and left merge, respectively, from [11, 17]. (The reader may find many other examples in the main body of this paper.) Distributivity laws like the aforementioned one play a crucial role in (ground-)complete axiomatizations of behavioural equivalences over fragments of process algebras (see, e.g., the above-mentioned references and [4]), and their lack of validity with respect to choice-like operators is often the key to the nonexistence of finite (in)equational axiomatizations of behavioural semantics—see, for instance, [6, 18].

In the rule formats, for the sake of simplicity, the $\oplus$ operator 'behaves like' some form of nondeterministic choice operator. Both rule formats are based on syntactic conditions that are decidable over finite language specifications.

We provide a wealth of examples showing that the validity of several left-distributivity laws from the literature on process algebras can be proved using the proposed rule formats.

Roadmap of the paper The paper is organized as follows. Section 2 reviews some standard definitions from the theory of SOS that will be used in the remainder

of this study. Section 3 presents our first rule format guaranteeing that a binary operator $\circledast$ is left-distributive with respect to a binary operator $\oplus$ modulo bisimilarity. The rule format is defined in Section 3.2 and some examples of its application are given in Section 3.3. We extend our rule format in Section 4 by allowing for a wider set of terms appearing in the target of deduction rules. Examples that can be handled using the second rule format are offered in the same section. We refer the reader to [1] for proofs and further results.

2 Preliminaries

In this section we recall some standard definitions from the theory of SOS. We refer the readers to, e.g., [7] and [20] for more information.

2.1 Transition System Specifications and Bisimilarity

Definition 1 (Signatures, terms and substitutions) *We let V denote an infinite set of variables and use $x, x', x_i, y, y', y_i, \ldots$ 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 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, u, possibly subscripted and/or superscripted, to range over terms. We write $t_1 \equiv t_2$ if t_1 and t_2 are syntactically equal. The function $\mathrm{vars} : \mathbb{T}(\Sigma) \to 2^V$ *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 is included in $\mathbb{C}(\Sigma)$, we say that it is a* closed substitution*. For a sequence $x_1, \ldots, x_n$ of distinct variables and a sequence $t_1, \ldots, t_n$ of terms, we write $[x_1 \mapsto t_1, \ldots, x_n \mapsto t_n]$ for a substitution that maps each x_i to t_i, $1 \leq i \leq n$.*

Definition 2 (Transition system specification) *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*. Such formulae are called* t-testing*. 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., tuples 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 $\mathrm{vars}(\Phi)$ *to denote the set of variables appearing in a set of formulae Φ. 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. 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 $\frac{\Phi}{\phi}$. For the sake of consistency with SOS specifications of specific operators in the literature, in examples we use $\frac{\phi_1 \ldots \phi_n}{\phi}$ in lieu of $\frac{\{\phi_1, \ldots, \phi_n\}}{\phi}$.

An *axiom* is a deduction rule with an empty set of premises. We write $\frac{}{\phi}$ for an axiom with ϕ as its conclusion, and often abbreviate this notation to ϕ when this causes no confusion.

Definition 3 *Given a rule d of the form $\frac{\Phi}{f(t_1, \ldots, t_n) \xrightarrow{a} t}$, we say that d is f-*defining*, and write $\mathrm{op}(d) = f$, d is a-*emitting*, and $\mathrm{toc}(d) = t$, the* target *of the conclusion of d. We also denote by $D(f, a)$ the set of a-emitting and f-defining rules in a set of deduction rules D.*

Example 1 (Choice operators). The choice operator from [17] is defined by the following rules, where a ranges over the set of actions:

$$(chl_a)\ \frac{x \xrightarrow{a} x'}{x + y \xrightarrow{a} x'} \qquad (chr_a)\ \frac{y \xrightarrow{a} y'}{x + y \xrightarrow{a} y'}\ .$$

For each action a, the rules (chl_a) and (chr_a) are a-emitting and $+$-defining. For rule (chl_a), we have that $\mathrm{toc}(chl_a) = x'$.

The meaning of a TSS is defined by the notion of least three-valued stable model [23]. We write $\mathcal{T} \vdash p \xrightarrow{a} p'$ if the transition $p \xrightarrow{a} p'$ is in the least three-valued stable model of $\mathcal{T}$. Since the precise definition of this notion does not play a role in the remainder of this paper, we omit it for the sake of brevity and refer our readers to [1] for details.

Definition 4 (Bisimulation and bisimilarity) *Let $\mathcal{T}$ be a transition system specification 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 and only if $\mathcal{R}$ is symmetric and, for all $p_0, p_1, p_0' \in \mathbb{C}(\Sigma)$ and $l \in \mathcal{L}$,*

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

Two terms $p_0, p_1 \in \mathbb{C}(\Sigma)$ are called bisimilar*, denoted by $p_0 \leftrightarroweq p_1$, 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 $\sigma(s) \leftrightarroweq \sigma(t)$ for each closed substitution $\sigma : V \to \mathbb{C}(\Sigma)$.

3 The Left-distributivity Rule Formats

In this section, we present a rule format guaranteeing that a binary operator $\circledast$ is left-distributive with respect to a binary operator $\oplus$ modulo bisimilarity. The rule format suffices to handle many examples from the literature.

Definition 5 (Left-distributivity law) *We say that a binary operator $\circledast$ is* left-distributive *with respect to a binary operator $\oplus$ (modulo bisimilarity) if the following equality holds:*

$$(x \oplus y) \circledast z \leftrightarrows (x \circledast z) \oplus (y \circledast z). \tag{1}$$

For all closed terms p, q, r, proving the algebraic law (1) involves two proof obligations:

- **Firability:** ensuring that $(p \oplus q) \circledast r \xrightarrow{a}$ if, and only if, $(p \circledast r) \oplus (q \circledast r) \xrightarrow{a}$, for each action a;
- **Matching conclusions:** ensuring that, for each closed term p_1, if $(p \oplus q) \circledast r \xrightarrow{a} p_1$, then there exists some closed term p_2 such that $(p \circledast r) \oplus (q \circledast r) \xrightarrow{a} p_2$ and $p_1 \leftrightarrows p_2$, and vice versa.

3.1 The Firability Condition

We begin by introducing the conditions on sets of rules for two binary operators $\circledast$ and $\oplus$ that we shall use to guarantee the firability condition for them. First of all, we present syntactic constraints on the rules for those operators that we shall use throughout the remainder of the paper.

Definition 6 *We say that a deduction rule is of* the form (R1) *when it has the structure*

$$\frac{\Phi_y}{x \circledast y \xrightarrow{a} t} \quad \textit{or} \quad \frac{\{x \xrightarrow{a} x'\} \cup \Phi_y}{x \circledast y \xrightarrow{a} t},$$

where

- *the variables x, x', y are pairwise distinct, and*
- *Φ_y is a (possibly empty) set of (positive or negative) y-testing formulae such that $x, x' \notin \mathrm{vars}(\Phi_y)$.*

A deduction rule is of the form (R2) *when it has the structure*

$$\frac{\{x \xrightarrow{a} x'\}}{x \oplus y \xrightarrow{a} t} \quad \textit{or} \quad \frac{\{y \xrightarrow{a} y'\}}{x \oplus y \xrightarrow{a} t} \quad \textit{or} \quad \frac{\{x \xrightarrow{a} x', y \xrightarrow{a} y'\}}{x \oplus y \xrightarrow{a} t},$$

where the variables x, x', y, y' are pairwise distinct. A rule of the form $(R1)$ or $(R2)$ is non-left-inheriting *if $x \notin \mathrm{vars}(t)$, that is, if x does not appear in the target of the conclusion of the rule. An operation f specified by rules of the form $(R1)$ or $(R2)$ is* non-left-inheriting *if so are all of the f-defining rules.*

Definition 7 (Firability constraint) *Given a TSS T, let $\circledast$ and $\oplus$ be binary operators in the signature of T. For each action a, we write* $\mathrm{Fire}(\circledast, \oplus, a)$ *whenever the following conditions are met:*

- *if $D(\circledast, a) \neq \emptyset$ then $D(\oplus, a) \neq \emptyset$,*
- *each $d \in D(\circledast, a)$ is of the form $(R1)$, and*
- *each $d \in D(\oplus, a)$ is of the form $(R2)$.*

Example 2. Recall the choice operator $+$, presented in Example 1. As our readers can easily check, $\mathrm{Fire}(+, +, a)$ holds for each action a.

The firability constraint in Definition 7 is sufficient to guarantee the aforementioned firability condition.

Theorem 1 (Firability Theorem). *Given a TSS T, let $\circledast$ and $\oplus$ be binary operators from the signature of T. Suppose that* $\mathrm{Fire}(\circledast, \oplus, a)$ *holds for some action a. Then,*

$$(p \oplus q) \circledast r \xrightarrow{a} \text{ if, and only if, } (p \circledast r) \oplus (q \circledast r) \xrightarrow{a},$$

for all closed terms p, q, r.

The import of Theorem 1 is that, when proving the validity of (1), we can guarantee the firability condition for action a just by showing that $\mathrm{Fire}(\circledast, \oplus, a)$ holds. Theorem 1 underlies the soundness of the rule formats we present in what follows.

The reader will have already noticed that the rule form $(R1)$ does not place any restriction on tests for the variable y. This is possible because the second argument of the terms $(p \oplus q) \circledast r$, $p \circledast r$ and $q \circledast r$ is always the same, i.e. the term r. This means that, for each $\circledast$-defining rule, the same tests performed on the second argument on one side of (1) are performed on the other. Roughly speaking, one side of (1) may fire as much as the other does, insofar the second argument is concerned.

3.2 The Matching-conclusion Condition

Theorem 1 tells us that any rule format, whose constraints imply condition $\mathrm{Fire}(\circledast, \oplus, a)$ for each action a, guarantees the validity of (1) provided that the matching-conclusion condition is met. Intuitively, in order to guarantee syntactically that the matching-conclusion condition is satisfied, the targets of the conclusions of $\circledast$-defining and $\oplus$-defining rules should 'match' when those operators are used in the specific contexts of the left- and the right-hand sides of (1). In what follows, we shall examine two different ways of ensuring the above-mentioned 'match' of the targets of the conclusions of $\circledast$-defining and $\oplus$-defining rules. The first relies on assuming that the targets of the conclusions of $\oplus$-defining rules are target variables of premises of rules of the form $(R2)$. The resulting rule format, which we present in this section, is based on easily checkable syntactic constraints and covers a large number of left-distributivity laws from the literature.

The First Rule Format The rule format that we present deals with examples of left distributivity with respect to operators whose semantics is given by rules of the form $(R2)$ that, like those for the choice operator we mentioned in Example 1, have target variables of premises as targets of their conclusions. The following definition presents the syntactic constraints of the rule format.

Definition 8 (First rule format) *Let T be a TSS, and let $\circledast$ and $\oplus$ be binary operators in the signature of T. We say that the rules for $\circledast$ and $\oplus$ are in the* first rule format for left distributivity *if the following conditions are met:*

1. $\mathrm{Fire}(\circledast, \oplus, a)$ *holds for each action a,*
2. *$\circledast$ is non-left-inheriting,*
3. *each $\oplus$-defining rule has a target variable of one of its premises as target of its conclusion and*
4. *for each action a, either there is no a-emitting and $\oplus$-defining rule that tests both x and y, or if some a-emitting and $\circledast$-defining rule tests its left argument x then so do all a-emitting and $\circledast$-defining rules.*

Theorem 2 (Left distributivity over choice-like operators). *Let T be a TSS, and let $\circledast$ and $\oplus$ be binary operators in the signature of T. Assume that the rules for $\circledast$ and $\oplus$ are in the* first rule format for left distributivity. *Then*

$$(x \oplus y) \circledast z \leftrightarrows (x \circledast z) \oplus (y \circledast z) \ .$$

Remark 1. Condition 4 in Definition 8 is necessary for the soundness of the rule format for left distributivity proved in the above theorem. To see this, consider the operations $\oplus$ and $\circledast$ with rules

$$\frac{\{x \xrightarrow{a} x', y \xrightarrow{a} y'\}}{x \oplus y \xrightarrow{a} x'} \qquad \frac{\{x \xrightarrow{a} x', y \xrightarrow{a} y'\}}{x \circledast y \xrightarrow{a} x' \circledast y} \qquad \frac{\{y \xrightarrow{a} y'\}}{x \circledast y \xrightarrow{a} y'} \ .$$

The above rules satisfy all the conditions in Definition 8 apart from condition 4. Now, let a be a constant with rule $a \xrightarrow{a} \mathbf{0}$, where $\mathbf{0}$ is a constant with no rules. As our readers can easily check,

$$(a \circledast a) \oplus (\mathbf{0} \circledast a) \not\leftrightarrows (a \oplus \mathbf{0}) \circledast a.$$

Indeed, the term $(a \circledast a) \oplus (\mathbf{0} \circledast a)$ can perform a sequence of two a-labelled transitions, whereas $(a \oplus \mathbf{0}) \circledast a$ cannot because $a \oplus \mathbf{0}$ affords no transitions.

3.3 Examples of Application of the Rule Format

Theorem 2 provides us with a simple, yet rather powerful, syntactic condition in order to infer left-distributivity laws for operators like $+$. Many of the common left-distributivity laws are automatically derived from Theorem 2, as witnessed by the examples we now proceed to discuss.

Example 3 (Left merge and interleaving parallel composition). The operational semantics of the classic left-merge and interleaving parallel composition operators [11, 13, 17] is given by the rules below:

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

Theorem 2 yields the validity of the following law.

$$(x + y) \between z \leftrightarrows (x \between z) + (y \between z)$$

Example 4 (Synchronous parallel composition). Consider the synchronous parallel composition from CSP [16][4] specified by the rules below, where a ranges over the set of actions:

$$\frac{x \xrightarrow{a} x' \quad y \xrightarrow{a} y'}{x \parallel_s y \xrightarrow{a} x' \parallel_s y'} \ .$$

Theorem 2 yields the validity of the following law.

$$(x + y) \parallel_s z \leftrightarrows (x \parallel_s z) + (y \parallel_s z)$$

Example 5 (Join and '/' operators). Consider the join operator $\bowtie$ from [12] and the 'hourglass' operator / from [4] specified by the rules below, where a, b range over the set of actions:

$$\frac{x \xrightarrow{a} x' \quad y \xrightarrow{a} y'}{x \bowtie y \xrightarrow{a} x' \mp y'} \qquad \frac{x \xrightarrow{a} x' \quad y \xrightarrow{b} y'}{x / y \xrightarrow{a} x' / y'} \ ,$$

where $\mp$ denotes the delayed choice operator from [12]. (The operational specification of the delayed choice operator is immaterial for the analysis of this example.) Theorem 2 yields the validity of the following laws.

$$(x + y) \bowtie z \leftrightarrows (x \bowtie z) + (y \bowtie z) \qquad (x + y) \,/\, z \leftrightarrows (x \,/\, z) + (y \,/\, z)$$

Example 6 (Disrupt). Consider the following disrupt operator $\blacktriangleright$ [9, 14] with rules

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

Theorem 2 yields the validity of the following law.

$$(x + y) \blacktriangleright z \leftrightarrows (x \blacktriangleright z) + (y \blacktriangleright z)$$

[4] In [16], Hoare uses the symbol $\parallel$ to denote the synchronous parallel composition operator. Here we use that symbol for parallel composition.

Example 7 (Unless operator). The unless operator $\triangleleft$ from [10] and the operator $\mathit{\Delta}$ from [4, page 23] are specified by the rules

$$\frac{x \xrightarrow{a} x' \quad y \overset{b}{\not\to} \text{ for } a < b}{x \triangleleft y \xrightarrow{a} x'} \qquad \frac{x \xrightarrow{a} x' \quad y \overset{b}{\not\to} \text{ for } a < b}{x \mathit{\Delta}\, y \xrightarrow{a} \theta(x')} \ ,$$

where $<$ is an irreflexive partial order over the set of actions and θ denotes the priority operator from [10]. (The operational specification of the priority operator is immaterial for the analysis of this example.) Theorem 2 yields the validity of the following laws.

$$(x + y) \triangleleft z \leftrightarrows (x \triangleleft z) + (y \triangleleft z) \qquad (x + y) \mathit{\Delta}\, z \leftrightarrows (x \mathit{\Delta}\, z) + (y \mathit{\Delta}\, z)$$

4 Analyzing Targets of Conclusions of Deduction Rules

In this section, we extend the first rule format by generalizing the matching-conclusion conditions. We do so by examining different possible targets of the conclusions of the $\circledast$- and $\oplus$-defining rules. By analyzing different possible syntactic shapes for terms, we check which pairs of shapes can be related (possibly under some further requirements) while preserving the left-distributivity law.

Table 1 summarizes our results. Even though the offered list is not exhaustive, which, at first sight, seems a challenging task to achieve, we believe Table 1 offers enough cases to cover almost all practical cases, as demonstrated by the examples presented in the remainder of this section and in the full version of this paper.

Table 1. Analysis of the targets of conclusions

	$\mathrm{toc}(d_1)$	$\mathrm{toc}(d_2)$	result	further requirements
1	$x' \circledast y$	x	$p \circledast r$	
2	$x' \circledast y$	y	$q \circledast r$	
3	x	$x' \oplus y'$	$p \oplus q$	$D(\circledast, a) = \{d_1\}$
4	x'	$x' \oplus y'$	$p' \oplus q'$	$D(\circledast, a) = \{d_1\}$
5	$x \circledast t$	$x' \oplus y'$	$(p \oplus q) \circledast \sigma(t)$	$D(\circledast, a) = \{d_1\}$, $x, x' \notin \mathrm{vars}(t)$
6	$x' \circledast t$	$x' \oplus y'$	$(p' \oplus q') \circledast \sigma(t)$	$D(\circledast, a) = \{d_1\}$, $x, x' \notin \mathrm{vars}(t)$
7	t	$x' \oplus y'$	$\sigma(t)$	$\oplus$ idempotent, $D(\circledast, a) = \{d_1\}$, $x, x' \notin \mathrm{vars}(t)$
8	t	x'	$\sigma'(t)$	Condition 4 of Definition 8, $x \notin \mathrm{vars}(t)$
9	t	y'	$\sigma'(t)$	Condition 4 of Definition 8, $x \notin \mathrm{vars}(t)$

with $\sigma = [y \mapsto r,\ y_i \mapsto r_i\ (i \in I)]$ and $\sigma' = [y \mapsto r,\ x' \mapsto p',\ y_i \mapsto r_i\ (i \in I)]$

In Table 1, x and y are considered as the variables for the first and second argument, respectively, for both $\circledast$- and $\oplus$-defining rules. When the variable x' is mentioned, implicitly the considered rule has a premise $x \xrightarrow{a} x'$ (for a-emitting

rules). Similarly, when the variable y' is mentioned, implicitly the rule considered has a premise $y \xrightarrow{a} y'$. The term t stands for a generic open term from the signature, and p, q and r are hypothetical closed terms applied to the distributivity equation in this way: $(p \oplus q) \circledast r \leftrightarrows (p \circledast r) \oplus (q \circledast r)$. The symbols p', q', and r_i are considered as targets of possible transitions from p, q and r.

Table 1 is to be read as follows. First of all, $d_1 \in D(\circledast, a)$ and $d_2 \in D(\oplus, a)$, for some action a. In each row, the first column (column $\mathrm{toc}(d_1)$) specifies the form of the target of the conclusion of the $\circledast$-defining rule d_1 (e.g., x in case of row 3), and the second column (column $\mathrm{toc}(d_2)$) specifies the form of the target of the conclusion of the $\oplus$-defining rule d_2 (e.g., $x' \oplus y'$ in case of row 3). If the conditions in the column *further requirements* are satisfied (e.g., in row 3, d_1 is the only $\circledast$-defining and a-emitting rule), then the result of the transition of terms $(p \oplus q) \circledast r$ and $(p \circledast r) \oplus (q \circledast r)$ is specified by the term given in column *result* (e.g., $p \oplus q$ in row 3). In rows 5–6, the stated result is up to one application of the left-distributivity equation (1). The requirement $\oplus$ *idempotent* means that the operator $\oplus$ can be proved idempotent, e.g., by means of the rule format offered in [3].

The reader may want to notice that the first rule format of Section 3.2 is partly based on the analysis which leads to rows 8 and 9.

Theorem 3 (Soundness of Table 1). *Let T be a TSS. Let $\circledast$ and $\oplus$ be binary operations in the signature of T satisfying*

1. $\mathrm{Fire}(\circledast, \oplus, a)$, *and*
2. *if $D(\circledast, a) \neq \emptyset$ then for each $d_1 \in D(\circledast, a)$ and for each $d_2 \in D(\oplus, a)$, the rules d_1 and d_2 match a row in Table 1.*

It holds that:

$$(x \oplus y) \circledast z \leftrightarrows (x \circledast z) \oplus (y \circledast z).$$

In what follows, we apply the rule format provided in this section in order to check some examples of left-distributivity laws whose validity cannot be inferred using Theorem 2.

Example 8 (Unit-delay operator and the choice operator from ATP). Consider any TSS T containing the unit-delay operator $\lfloor\ \rfloor$ and the choice operator $+^*$ from ATP [21][5] and for which the transition relation $\xrightarrow{\chi}$ is deterministic. (The distinguished symbol χ denotes the passage of one unit of time.) The semantics of those operators is defined by the following rules, where $a \neq \chi$:

$$(ud_a)\ \frac{x \xrightarrow{a} x'}{\lfloor x \rfloor(y) \xrightarrow{a} x'} \qquad (ud_\chi)\ \frac{}{\lfloor x \rfloor(y) \xrightarrow{\chi} y} \qquad (extTime)\ \frac{x \xrightarrow{\chi} x' \quad y \xrightarrow{\chi} y'}{x +^* y \xrightarrow{\chi} x' +^* y'}$$

[5] In [21], the symbol of this operator is $\oplus$, whose use we prefer to avoid in this paper for the sake of clarity.

$$(extChl_a)\ \frac{x \xrightarrow{a} x'}{x +^* y \xrightarrow{a} x'} \qquad (extChr_a)\ \frac{y \xrightarrow{a} y'}{x +^* y \xrightarrow{a} y'}\ .$$

Table 1 can be used to match the targets of the conclusions as follows: the combination of ud_a and $extChl_a$ follows from row 8, the combination of ud_a and $extChr_a$ follows from row 9, and finally the combination of ud_χ and $extTime$ follows from row 7.

Example 9 (Timed left merge and the choice operator from ATP). Consider the TSS for ATP with the timed extension of the left-merge operator from Example 3 specified by the following rules, where $a \neq \chi$:

$$(merge_a)\frac{x \xrightarrow{a} x'}{x \between y \xrightarrow{a} x' \parallel y} \qquad (merge_\chi)\frac{x \xrightarrow{\chi} x' \quad y \xrightarrow{\chi} y'}{x \between y \xrightarrow{\chi} x' \between y'}\ .$$

Table 1 can be used to match the targets of the conclusions as follows: the combination $merge_a$, $extChl_a$ follows from row 8, the combination $merge_a$, $extChr_a$ follows from row 9 and the combination $merge_\chi$, $extTime$ follows from row 6.

In the extended version of this paper [1], we apply our rule formats to several more examples and also show how they can be applied to obtain distributivity for unary operators. The full version of the paper also offers a much more general format for left distributivity based on a notion of distributivity compliance between rules of which Table 1 is an approximation.

References

1. Aceto, L., Cimini, M., Ingolfsdottir, A., Mousavi, M.R., Reniers, M.A.: Rule formats for distributivity. Technical Report CSR-10-16, TU/Eindhoven (2010)
2. Aceto, L., Ingolfsdottir, A., Mousavi, M.R., Reniers, M.A.: Algebraic properties for free! Bulletin of the European Association for Theoretical Computer Science **99** (October 2009) 81–104 Columns: Concurrency.
3. Aceto, L., Birgisson, A., Ingolfsdottir, A., Mousavi, M.R., Reniers, M.A.: Rule formats for determinism and idempotence. In Arbab, F., Sirjani, M., eds.: Fundamentals of Software Engineering, Third IPM International Conference, FSEN 2009, Kish Island, Iran, April 15-17, 2009, Revised Selected Papers. Volume 5961 of Lecture Notes in Computer Science, Springer-Verlag (2010) 146–161
4. Aceto, L., Bloom, B., Vaandrager, F.W.: Turning SOS rules into equations. Inf. Comput. **111**(1) (1994) 1–52
5. Aceto, L., Cimini, M., Ingolfsdottir, A., Mousavi, M.R., Reniers, M.A.: On rule formats for zero and unit elements. In: Proceedings of the 26th Conference on the Mathematical Foundations of Programming Semantics (MFPS XXVI). Volume 265 of Electronic Notes in Theoretical Computer Science, Ottawa, Canada, Elsevier B.V., The Netherlands (2010) 145–160

6. Aceto, L., Fokkink, W., Ingolfsdottir, A., Luttik, B.: Finite equational bases in process algebra: Results and open questions. In: Processes, Terms and Cycles. Volume 3838 of Lecture Notes in Computer Science, Springer-Verlag (2005) 338–367
7. Aceto, L., Fokkink, W., Verhoef, C.: Structural operational semantics. In: Handbook of Process Algebra. Elsevier (2001) 197–292
8. Aceto, L., Ingolfsdottir, A., Mousavi, M.R., Reniers, M.A.: Rule formats for unit elements. In van Leeuwen, J., Muscholl, A., Peleg, D., Pokorný, J., Rumpe, B., eds.: SOFSEM 2010, 36th Conference on Current Trends in Theory and Practice of Computer Science, Špindleruv Mlýn, Czech Republic, January 23-29, 2010. Proceedings. Volume 5901 of Lecture Notes in Computer Science, Springer-Verlag (2010) 141–152
9. Baeten, J., Bergstra, J.: Mode transfer in process algebra. Technical Report Report CSR 00–01, Eindhoven University of Technology (2000)
10. Baeten, J., Bergstra, J., Klop, J.W.: Syntax and defining equations for an interrupt mechanism in process algebra. Fundamenta Informaticae **IX**(2) (1986) 127–168
11. Baeten, J., Basten, T., Reniers, M.A.: Process Algebra: Equational Theories of Communicating Processes. Volume 50 of Cambridge Tracts in Theoretical Computer Science. Cambridge University Press (2009)
12. Baeten, J., Mauw, S.: Delayed choice: An operator for joining Message Sequence Charts. In Hogrefe, D., Leue, S., eds.: Formal Description Techniques VII, Proceedings of the 7th IFIP WG6.1 International Conference on Formal Description Techniques, Berne, Switzerland, 1994. Volume 6 of IFIP Conference Proceedings, Chapman & Hall (1995) 340–354
13. Bergstra, J., Klop, J.W.: Fixed point semantics in process algebras. Report IW 206, Mathematisch Centrum, Amsterdam (1982)
14. Brinksma, E.: A tutorial on LOTOS. In: Proceedings of the IFIP WG6.1 Fifth International Conference on Protocol Specification, Testing and Verification V, Amsterdam, The Netherlands, The Netherlands, North-Holland Publishing Co. (1985) 171–194
15. Cranen, S., Mousavi, M.R., Reniers, M.A.: A rule format for associativity. In van Breugel, F., Chechik, M., eds.: Proceedings of the 19th International Conference on Concurrency Theory (CONCUR'08). Volume 5201 of Lecture Notes in Computer Science, Toronto, Canada, Springer-Verlag, Berlin, Germany (2008) 447–461
16. Hoare, C.: Communicating Sequential Processes. Prentice-Hall International, Englewood Cliffs (1985)
17. Milner, R.: Communication and Concurrency. Prentice-Hall, Inc., Upper Saddle River, NJ, USA (1989)
18. Moller, F.: The importance of the left merge operator in process algebras. In Paterson, M., ed.: Proceedings 17^{th} ICALP, Warwick. Volume 443 of Lecture Notes in Computer Science, Springer-Verlag (July 1990) 752–764
19. Mousavi, M.R., Reniers, M.A., Groote, J.F.: A syntactic commutativity format for SOS. Information Processing Letters **93** (March 2005) 217–223
20. Mousavi, M.R., Reniers, M.A., Groote, J.F.: SOS formats and meta-theory: 20 years after. Theor. Comput. Sci. **373**(3) (2007) 238–272
21. Nicollin, X., Sifakis, J.: The algebra of timed processes, ATP: Theory and application. Information and Computation **114**(1) (1994) 131–178
22. Plotkin, G.D.: A structural approach to operational semantics. J. Log. Algebr. Program. **60-61** (2004) 17–139
23. Przymusinski, T.: The well-founded semantics coincides with the three-valued stable semantics. Fundamenta Informaticae **13**(4) (1990) 445–463