Rule formats for determinism and idempotence

Citation for the original published paper (version of record):
Aceto, L., Birgisson, A., Ingolfsdottir, A. et al (2012). Rule formats for determinism and idempotence. Science of Computer Programming, 77(7-8): 889-907.
http://dx.doi.org/10.1016/j.scico.2010.04.002

Science of Computer Programming 77 (2012) 889–907

Contents lists available at SciVerse ScienceDirect

Science of Computer Programming

journal homepage: www.elsevier.com/locate/scico

Rule formats for determinism and idempotence☆

Luca Aceto [a,*], Arnar Birgisson [b], Anna Ingolfsdottir [a], MohammadReza Mousavi [c], Michel A. Reniers [c]

[a] ICE-TCS, School of Computer Science, Reykjavik University, Menntavegur 1, IS-101 Reykjavik, Iceland

[b] Department of Computer Science and Engineering, Chalmers University of Technology, Sweden

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

ARTICLE INFO

Article history:
Received 30 September 2009
Received in revised form 12 March 2010
Accepted 15 April 2010
Available online 27 April 2010

Keywords:
Structural operational semantics
Rule formats
Determinism
Idempotence

ABSTRACT

Determinism is a semantic property of (a fragment of) a language that specifies that a program cannot evolve operationally in several different ways. Idempotence is a property of binary composition operators requiring that the composition of two identical specifications or programs will result in a piece of specification or program that is equivalent to the original components. In this paper, we propose (related) meta-theorems for guaranteeing the determinism and idempotence of binary operators. These meta-theorems are formulated in terms of syntactic templates for operational semantics, called rule formats. In order to obtain a powerful rule format for idempotence, we make use of the determinism of certain transition relations in the definition of the format for idempotence. We show the applicability of our formats by applying them to various operational semantics from the literature.

1. Introduction

Structural operational semantics (SOS) [25] is a popular method for assigning a rigorous meaning to specification and programming languages. In this approach to semantics, the behaviour of (terms in) programming and specification languages is clearly given in terms of states and transitions, where the collection of transitions is specified by means of a collection of syntax-driven inference rules. Such a rule-based specification of the operational semantics of languages has proven itself to be very flexible, and naturally lends itself to proofs of properties of languages using structural or rule induction.

The meta-theory of SOS provides powerful tools for proving semantic properties for programming and specification languages without investing too much time on the actual proofs; it offers syntactic templates for SOS rules, called *rule formats*, which guarantee semantic properties once the SOS rules conform to the templates (see, e.g., the references [3,23] for surveys on the meta-theory of SOS). There are various rule formats in the literature for many different semantic properties, ranging from basic properties such as commutativity [21] and associativity [10] of operators, the existence of unit elements [4] and congruence of behavioral equivalences (see, e.g., [15,30]) to more technical and involved ones such as (semi-)stochasticity [18] and non-interference [26]. In this paper, we propose rule formats for two (related) properties, namely determinism and idempotence.

☆ The work of Aceto, Birgisson and Ingolfsdottir has been partially supported by the projects "The Equational Logic of Parallel Processes" (nr. 060013021), "New Developments in Operational Semantics" (nr. 080039021) and "Meta-theory of Algebraic Process Theories" (nr. 100014021) of the Icelandic Research Fund. Birgisson has been further supported by research-student grant nr. 080890008 of the Icelandic Research Fund.

* Corresponding author.
E-mail addresses: luca@ru.is, luca.aceto@gmail.com (L. Aceto).

0167-6423/$ – see front matter © 2010 Elsevier B.V. All rights reserved.
doi:10.1016/j.scico.2010.04.002

Determinism is a semantic property of (a fragment of) a language that specifies that a program cannot evolve operationally in several different ways. It holds for sub-languages of many process calculi and programming languages, and it is also a crucial property for many formalisms for the description of timed systems, where time transitions are required to be deterministic because the passage of time should not resolve any choice.

Idempotence is a property of binary composition operators requiring that the composition of two identical specifications or programs will result in a piece of specification or program that is equivalent to the original components. The idempotence of a binary operator f is concisely expressed by the following algebraic equation.

$$f(x, x) = x.$$

Determinism and idempotence may seem unrelated at first sight. However, perhaps surprisingly, it turns out that, in order to obtain a powerful rule format for idempotence, we need to have the determinism of certain transition relations in place. Therefore, having a syntactic condition for determinism, apart from its intrinsic value, results in a powerful, yet syntactic framework for idempotence.

To our knowledge, our rule format for idempotence has no precursor in the literature. As for determinism, in [13], a rule format for bounded nondeterminism is presented but the case for determinism is not studied. Also, in [27] a rule format is proposed to guarantee several time-related properties, including time determinism, in the settings of ordered SOS. In the case of time determinism, the format considered in [27] corresponds to a subset of our rule format when translated to the setting of ordinary SOS, by means of the recipe given in [20].

We made a survey of existing deterministic process calculi and of idempotent binary operators in the literature and we have applied our formats to them. Our formats could cover all practical cases that we have discovered so far, which is an indication of their expressiveness and relevance. However, in Section 4.4 of the paper, we present a generalized format for idempotence that may have future applications. Even though we are not aware of applications of this more general format in the current literature, we find it worthwhile to include it in this paper since one of the goals of research on the meta-theory of SOS is to present rule formats that might be applicable not only to extant languages, but also to those that might be developed in the future.

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 [21], associativity [10], and left and right unit elements [4].

This line of research can serve multiple purposes. Firstly, it can pave the way for a toolset 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 can serve as a guideline for language designers who want to ensure, a priori, that the constructs under design enjoy certain basic properties.

The rest of this paper is organized as follows. In Section 2, we recall some basic definitions from the meta-theory of SOS. In Section 3, we present our rule format for determinism and prove that it does guarantee determinism for certain transition relations. Section 4 introduces a rule format for idempotence and proves it correct. In Sections 3 and 4, we also provide several examples to motivate the constraints of our rule formats and to demonstrate their practical applications. Finally, Section 5 concludes the paper and presents some directions for future research.

This article is an expanded version of the conference paper [1]. Apart from including the proofs of the technical results that were announced without proof in the conference publication, the following scientific contributions are new in this version of the paper:

- Theorem 9, to the effect that determining whether a closed term in a finite transition system specification is deterministic for a given label is undecidable, and its proof in the Appendix;
- Theorem 20, to the effect that determining whether a finite transition system specification is in the syntactic determinism format with respect to a set of labels L is decidable;
- Section 3.4, which offers rule formats for other forms of determinism not considered in [1];
- Example 49, which introduces a generalization of the format for idempotence presented in [1].

The presentation of the paper has also undergone some changes in reaction to the comments of the expert reviewers.

2. Preliminaries

In this section we present, for the sake of completeness, some standard definitions from the meta-theory of SOS that will be used in the remainder of the paper.

Definition 1 (*Signature and Terms*)**.** We let V represent a countably 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 denote 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 $t, t', t_i, \ldots$ 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 $p, p', p_i, \ldots$ 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. If the range of a substitution lies in $\mathbb{C}(\Sigma)$, we say that it is a *closed substitution*.

Definition 2 (*Transition System Specifications*)**.** A *transition system specification* (TSS) is a triple (Σ, L, D), where

- Σ is a signature.
- L is a set of labels. If $l \in L$, and $t, t' \in \mathbb{T}(\Sigma)$ we say that $t \xrightarrow{l} t'$ is a *positive formula* and $t \xnrightarrow{l}$ is a *negative formula*. A formula, typically denoted by $\phi, \psi, \phi', \phi_i, \ldots$ is either a negative formula or a positive one. We often refer to a formula $t \xrightarrow{l} t'$ as a *transition* with t being its *source*, l its *label*, and t' its *target*.
- 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*.

A TSS is *finite* if Σ, L and D are all finite, and the set of premises in each deduction rule in D is finite.

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, sets of formulae and rules in the natural way. If r is a rule and σ is a (closed) substitution, then $\sigma(r)$ is called a *(closed) substitution instance* of r.

A set of positive closed formulae is called a *transition relation*. For T a transition relation and $l \in L$, we write $\xrightarrow{l}$ for the collection of l-labelled transitions in T; that is,

$$\xrightarrow{l} = \{p \xrightarrow{a} p' \mid a = l \text{ and } p \xrightarrow{a} p' \in T\}.$$

A deduction rule (Φ, ϕ) is typically written as $\frac{\Phi}{\phi}$. An *axiom* is a deduction rule without premises. In what follows, an axiom will be usually written as $\frac{}{\phi}$ or just ϕ. For a deduction rule r, we write $conc(r)$ to denote its conclusion and $prem(r)$ to denote its premises. We call a deduction rule *f-defining* when the outermost function symbol appearing in the source of its conclusion is f.

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 consistency, 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* the closed transition rule $\frac{N}{\phi}$, 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 nodes directly 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 closed substitution instance of a deduction rule in $\mathcal{T}$.

Example 4. As a running example, we consider in this section the TSS with constant a, labels l_1 and l_2, and the deduction rules given below.

$$\frac{a \xnrightarrow{l_2}}{a \xrightarrow{l_1} a} \qquad \frac{a \xnrightarrow{l_1}}{a \xrightarrow{l_2} a}.$$

Both the above rules are a-defining and are provable transition rules. Indeed, they are the only transition rules that are provable in this TSS.

Definition 5 (*Contradiction and Consistency*)**.** Formula $t \xrightarrow{l} t'$ is said to *contradict* $t \xnrightarrow{l}$, and vice versa. For two sets Φ and Ψ of formulae, Φ *contradicts* Ψ when there is a $\phi \in \Phi$ that contradicts a $\psi \in \Psi$. Φ is *consistent* with Ψ, denoted by $\Phi \nvDash \Psi$, when Φ does not contradict Ψ.

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

Definition 6 (*The 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

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

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 with respect to the ordering on pairs of sets of formulae defined as $(C, U) \leq (C', U')$ iff $C \subseteq C'$ and $U' \subseteq U$. When for the least three-valued stable model it holds that $U = \emptyset$, we say that $\mathcal{T}$ is *complete*.

Example 7. The TSS in Example 4 has

- $(\{a \xrightarrow{l_1} a\}, \emptyset)$,
- $(\{a \xrightarrow{l_2} a\}, \emptyset)$ and
- $(\emptyset, \{a \xrightarrow{l_1} a, a \xrightarrow{l_2} a\})$

as its three-valued stable models. Its least three-valued stable model is $(\emptyset, \{a \xrightarrow{l_1} a, a \xrightarrow{l_2} a\})$. Therefore that TSS is not complete.

Complete TSSs unequivocally define a transition relation, i.e., the C component of their least three-valued stable model. Completeness is central to almost all meta-results in the SOS meta-theory and, as it turns out, it also plays an essential role in our meta-results concerning determinism and idempotence. All practical instances of TSSs are complete and there are sufficient syntactic conditions guaranteeing completeness; see, for example, [14].

3. Determinism

The agenda of this section is to define a rule format for determinism. We start with a general format for determinism in Section 3.1; that format captures the "essence" of determinism as a semantic property. Although the general format presented in Section 3.1 is natural and elegant, it lacks the practicality of a syntactic format. This is why, in Section 3.2, we provide a syntactic variation on our general rule format that is sufficient to guarantee the determinism of certain transition relations. In Section 3.3, we apply our rule formats to several examples from the literature. In Section 3.4, we present some alternative definitions for determinism and show how our formats can easily be adapted to these definitions.

3.1. The general determinism format

For the sake of precision, we begin by defining the notion of determinism that is typically considered in the literature on process calculi and related languages.

Definition 8 (*Determinism*)**.** A transition relation T is called *deterministic for label l* when, if $p \xrightarrow{l} p' \in T$ and $p \xrightarrow{l} p'' \in T$, then $p' \equiv p''$.

Given a complete transition system specification $\mathcal{T} = (\Sigma, L, D)$, a term $p \in \mathbb{C}(\Sigma)$ is deterministic for label l if the transition relation associated with $\mathcal{T}$ is deterministic for label l when restricted to the set of closed terms that are reachable from p.

As with most semantic properties of languages, the determinism of a transition relation is undecidable.

Theorem 9. *Given a finite transition system specification (Σ, L, D), a term $p \in \mathbb{C}(\Sigma)$ and a label l, the problem of determining whether $p \in \mathbb{C}(\Sigma)$ is deterministic for label l is undecidable.*

Proof. See the Appendix. □

In the light of undecidability results like the one above, it is interesting to isolate some conditions on the deduction rules in a transition system specification that are sufficient to guarantee the determinism of certain transition relations.

Before defining a format for determinism, we need two auxiliary definitions. The first one is the definition of source-dependent variables, which we borrow from [22] with minor additions.

Definition 10 (*Source Dependency*)**.** For a deduction rule, we define the set of *source-dependent* variables as the smallest set that contains

1. all variables appearing in the source of the conclusion, and
2. all variables that appear in the target of a premise where all variables in the source of that premise are source dependent.

For a source-dependent variable v, let $\mathcal{R}$ be the collection of transition relations appearing in a set of premises needed to show source dependency through condition 2. We say that v is source dependent via the relations in $\mathcal{R}$.

We define the *source distance* of a source-dependent variable as the least number of applications of item 2 in Definition 10 needed to show its source dependency. A variable in the source of the conclusion is thus of source distance 0.

Note that, for a source-dependent variable, the set $\mathcal{R}$ is not necessarily unique. For example, in the rule

$$\frac{y \xrightarrow{l_1} y' \quad x \xrightarrow{l_2} z \quad z \xrightarrow{l_3} y'}{f(x, y) \xrightarrow{l} y'},$$

the variable y' is source dependent via both the set $\{\xrightarrow{l_1}\}$ and the set $\{\xrightarrow{l_2}, \xrightarrow{l_3}\}$.

The second auxiliary definition needed for our determinism format is the definition of determinism-respecting substitutions.

Definition 11 (*Determinism-Respecting Substitutions*)**.** A pair (σ, σ') of substitutions is *determinism respecting* with respect to a pair of sets of formulae (Φ, Φ') and a set of labels L when, for all two positive formulae $s \xrightarrow{l} s' \in \Phi$ and $t \xrightarrow{l} t' \in \Phi'$ such that $l \in L$, it holds that $\sigma(s) \equiv \sigma'(t)$ only if $\sigma(s') \equiv \sigma'(t')$.

Definition 12 (*Determinism Format*)**.** A TSS $\mathcal{T}$ is in the determinism format with respect to a set of labels L when, for each $l \in L$, the following conditions hold.

1. In each deduction rule $\frac{\Phi}{t \xrightarrow{l} t'}$, each variable $v \in vars(t')$ is source dependent via a subset of $\{\xrightarrow{l'} \mid l' \in L\}$.
2. For each pair of distinct deduction rules $\frac{\Phi_0}{t_0 \xrightarrow{l} t_0'}$ and $\frac{\Phi_1}{t_1 \xrightarrow{l} t_1'}$, and for each determinism-respecting pair of substitutions (σ, σ') with respect to (Φ_0, Φ_1) and L such that $\sigma(t_0) \equiv \sigma'(t_1)$, it holds that either $\sigma(t_0') \equiv \sigma'(t_1')$ or $\sigma(\Phi_0)$ contradicts $\sigma'(\Phi_1)$.

As the proof of Theorem 14 to follow will make clear, the first condition in the definition above ensures that each rule in a TSS in the determinism format, with some $l \in L$ as the label of its conclusion, can be used to prove at most one outgoing transition for each closed term. The second requirement guarantees that no pair of different rules can be used to prove two distinct l-labelled transitions for any closed term.

Remark 13. Usually, the term "format" refers to a template for deduction rules that is defined using purely syntactic conditions. The latter requirement in Definition 12 is not syntactic since it refers to a condition that needs to be checked for each determinism-respecting pair of substitutions. However, rather than coining a new word for the requirements in Definition 12, we decided to stretch the use of the term "format" and to refer to the notion defined there as "determinism format".

Theorem 14. *Consider a TSS with (C, U) as its least three-valued stable model and a subset L of its labels. If the TSS is in the determinism format with respect to L, then C is deterministic for each $l \in L$.*

Proof. Let $\mathcal{T}$ be a TSS with (C, U) as its least three-valued stable model. Instead of proving that C is deterministic for each $l \in L$, we establish the following more general result. We prove that, for each $l \in L$, if $p \xrightarrow{l} p' \in C \cup U$ and $p \xrightarrow{l} p'' \in C$, then $p' \equiv p''$.

Since $p \xrightarrow{l} p' \in C \cup U$, there exists a provable transition rule such that $\mathcal{T} \vdash \frac{N}{p \xrightarrow{l} p'}$, for some set N of negative formulae with $C \models N$. We show the claim by an induction on the proof structure for the transition rule $\frac{N}{p \xrightarrow{l} p'}$. Consider the last deduction rule r and closed substitution σ used in the proof structure for $\frac{N}{p \xrightarrow{l} p'}$.

Since $p \xrightarrow{l} p'' \in C$, there also exists a proof structure such that $\mathcal{T} \vdash \frac{N'}{p \xrightarrow{l} p''}$ for some set N' of negative formulae with $C \cup U \models N'$. Again, consider the last deduction rule r' and closed substitution σ' used in the proof structure for $\mathcal{T} \vdash \frac{N'}{p \xrightarrow{l} p''}$.

We first consider the case when r and r' are the same rule, say $\frac{\Phi}{t \xrightarrow{l} t'}$. Obviously $\sigma(t) \equiv \sigma'(t)$, since both must be equal to p. Since $\sigma(t')$ and $\sigma'(t')$ are equal to p' and p'', respectively, we need to show that $\sigma(t') \equiv \sigma'(t')$.

For each variable v that is source dependent via a subset of $\{\xrightarrow{l} \mid l \in L\}$, we proceed with another induction on the source distance of v to prove that $\sigma(v) \equiv \sigma'(v)$. If we show this claim, then it follows that $\sigma(t') \equiv \sigma'(t')$ since all variables in t' are source dependent by the first condition of our rule format.

We consider the two possible reasons for v being source dependent.

1. Assume that v appears in t. In this case, $\sigma(v) \equiv \sigma'(v)$ since $\sigma(t) \equiv \sigma'(t)$.
2. Assume that v appears in the target of some premise $t_i \xrightarrow{l_i} t_i' \in \Phi$, where $l_i \in L$ and all variables in t_i are source dependent via a subset of $\{\xrightarrow{l} \mid l \in L\}$. Each variable $w \in vars(t_i)$ has a source distance smaller than that of v. Therefore, the induction hypothesis (on the source distance of variables) applies and we have that $\sigma(w) \equiv \sigma'(w)$. This means that $\sigma(t_i) \equiv \sigma'(t_i)$. This allows us to apply the induction hypothesis on the proof structure, since $\frac{N}{\sigma(t_i) \xrightarrow{l_i} t_i'}$ has a proof structure that is smaller than the one for $\frac{N}{p \xrightarrow{l} p'}$, to conclude that $\sigma(t_i') \equiv \sigma'(t_i')$. Since v appears in t_i', it follows that $\sigma(v) \equiv \sigma'(v)$.

In either case, σ and σ' agree on the value of v. Since this holds for all variables of t', we reach the conclusion we seek, namely that $\sigma(t') \equiv \sigma'(t')$.

We now consider the case where the rules r and r' are distinct. Let Φ and Φ' be the sets of premises of r and r', respectively. We first show that (σ, σ') is determinism respecting with respect to (Φ, Φ') and L.

Assume, towards a contradiction, that our claim concerning determinism-respecting substitutions does not hold. Then, for some $l \in L$, there exist two positive formulae $s_i \xrightarrow{l} s_i'$ and $t_i \xrightarrow{l} t_i'$ among the premises of r and r', respectively, such that $\sigma(s_i) \equiv \sigma'(t_i)$, but it does not hold that $\sigma(s_i') \equiv \sigma'(t_i')$. Since $s_i \xrightarrow{l} s_i'$ is a premise of r, the transition $\sigma(s_i \xrightarrow{l} s_i')$ is

contained in $C \cup U$ and has a smaller proof structure than the one justifying that $p \xrightarrow{l} p' \in C \cup U$. Following a similar reasoning, $\sigma'(t_i \xrightarrow{l} t_i') \in C$. But the induction hypothesis (on the proof structure) applies, and hence we have $\sigma(s_i') \equiv \sigma'(t_i')$, which contradicts our earlier assumption that $\sigma(s_i') \equiv \sigma'(t_i')$ does not hold. Hence, we conclude that (σ, σ') is determinism respecting with respect to (Φ, Φ') and L.

Since we have shown that (σ, σ') is determinism respecting, it then follows from the second condition of the determinism format that either $\sigma(conc(r)) \equiv \sigma'(conc(r'))$, which was to be shown, or there exist premises $\phi_i \equiv s_i \xrightarrow{l_i} s_i'$ in one deduction rule and $\phi_i' \equiv t_i \not\xrightarrow{l_i}$ in the other deduction rule such that $\sigma(\phi_i)$ contradicts $\sigma'(\phi_i')$. We show that the latter possibility leads to a contradiction, thus completing the proof. Since $\sigma(\phi_i)$ contradicts $\sigma'(\phi_i')$, we have that $\sigma(s_i) \equiv \sigma'(t_i)$. We distinguish the following two cases based on the status of the positive and negative contradicting premises with respect to r and r'.

1. Assume that the positive formula is a premise of r. Then, $\sigma(s_i \xrightarrow{l_i} s_i') \in C \cup U$, but, from $C \cup U \models N'$ and $\sigma'(\phi_i') \in N'$, it follows that, for no p'', we have that $\sigma(s_i) \equiv \sigma'(t_i) \xrightarrow{l_i} p'' \in C \cup U$, thus reaching a contradiction.
2. Assume that the positive formula is a premise of r'. Then $\sigma'(s_i \xrightarrow{l_i} s_i') \in C$, but, from $C \models N$ and $\sigma(\phi_i') \in N$, it follows that, for no p_1, we have that $\sigma(t_i) \equiv \sigma'(s_i) \xrightarrow{l_i} p_1 \in C$, again reaching a contradiction. □

For a TSS in the determinism format with (C, U) as its least three-valued stable model, U and thus $C \cup U$ need not be deterministic. The following counter-example illustrates this phenomenon.

Example 15. Consider the TSS given by the following deduction rules.

$$\frac{a \xrightarrow{l} a}{a \xrightarrow{l} b} \qquad \frac{a \not\xrightarrow{l}}{a \xrightarrow{l} a}.$$

The above-given TSS is in the determinism format since $a \xrightarrow{l} a$ and $a \not\xrightarrow{l}$ contradict each other (under any substitution). Its least three-valued stable model is, however, $(\emptyset, \{a \xrightarrow{l} a, a \xrightarrow{l} b\})$ and $\{a \xrightarrow{l} a, a \xrightarrow{l} b\}$ is not deterministic.

Example 16. The conditions in Definition 12 are not necessary to ensure determinism. For example, consider the TSS with constant a and rule $\frac{}{x \xrightarrow{a} y}$. The transition relation $\xrightarrow{a}$ is obviously deterministic, but the variable y is not source dependent in the rule $\frac{}{x \xrightarrow{a} y}$. However, as the following two examples show, relaxing the conditions in Definition 12 may jeopardize the determinism.

To see the need for condition 1, consider the TSS with constant 0 and unary function symbol f with rule $\frac{}{f(x) \xrightarrow{a} y}$. This TSS satisfies condition 2 in Definition 12 vacuously, but the transition relation $\xrightarrow{a}$ it determines is not deterministic since, for instance, $f(0) \xrightarrow{a} p$ holds for each closed term p. Note that the variable y is not source dependent in $\frac{}{f(x) \xrightarrow{a} y}$.

The need for condition 2 is exemplified by the classic non-deterministic choice operator, given by the following deduction rules.

$$\frac{x_0 \xrightarrow{a} x_0'}{x_0 + x_1 \xrightarrow{a} x_0'} \qquad \frac{x_1 \xrightarrow{a} x_1'}{x_0 + x_1 \xrightarrow{a} x_1'}.$$

The rules for this operator satisfy condition 1, but not condition 2. The transition relations defined by those rules are non-deterministic except for "trivial TSSs" including this operator.

Corollary 17. *Consider a complete TSS with L as a subset of its labels. If the TSS is in the determinism format with respect to L, then its defined transition relation is deterministic for each $l \in L$.*

Condition 2 in Definition 12 may seem difficult to verify, since it requires checks for all possible (determinism-respecting) substitutions. However, in practical cases, to be quoted in the remainder of this paper, variable names are chosen in such a way that condition 2 can be checked syntactically. For example, consider the following two deduction rules.

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

If in both deduction rules $f(x, y)$ (or symmetrically $f(y, x)$) was used, it could have been easily seen from the syntax of the rules that the premises of one deduction rule always (under all pairs of substitutions agreeing on the value of x) contradict the premises of the other deduction rule and, hence, condition 2 is trivially satisfied. Based on this observation, we next present a rule format whose conditions have a purely syntactic form and that is sufficiently powerful to handle all the examples we discuss in Section 3.3. (Note that, for the examples in Section 3.3, checking the conditions of Definition 12 is not too hard either.)

3.2. The syntactic determinism format

In order to derive a syntactic rule format for determinism, we limit the syntactic structure of the rules to the following, very common, subset of normalized TSSs.

Definition 18 (*Normalized TSSs*)**.** A TSS is normalized with respect to L if

1. each deduction rule is f-defining for some function symbol f,
2. for each deduction rule of the form

$$(r)\ \frac{\Phi_r}{f(\vec{s}) \xrightarrow{l} s'},$$

each variable $v \in vars(s')$ is source dependent in r via some subset of $\{\xrightarrow{l'} \mid l' \in L\}$, and
3. for each label $l \in L$, each function symbol f and each pair of deduction rules of the form

$$(r)\ \frac{\Phi_r}{f(\vec{s}) \xrightarrow{l} s'} \qquad (r')\ \frac{\Phi_{r'}}{f(\vec{t}) \xrightarrow{l} t'}$$

the following conditions are satisfied:
   (a) the sources of the conclusions coincide, i.e., $f(\vec{s}) \equiv f(\vec{t})$,
   (b) for each variable $v \in vars(r) \cap vars(r')$ there is a set of formulae in $\Phi_r \cap \Phi_{r'}$ proving its source dependency (both in r and r') via some subset of $\{\xrightarrow{l'} \mid l' \in L\}$.

The second and third conditions in Definition 19 guarantee that the syntactic equivalence of relevant terms (the target of the conclusion or the premises negating each other) will lead to syntactically equivalent closed terms under all determinism-respecting pairs of substitutions.

The reader can check that all the examples quoted from the literature in Section 3.3 are indeed normalized TSSs.

Definition 19 (*Syntactic Determinism Format*)**.** A normalized TSS is in the (syntactic) determinism format with respect to L when, for each two deduction rules $\frac{\Phi_0}{f(\vec{s}) \xrightarrow{l} s'}$ and $\frac{\Phi_1}{f(\vec{s}) \xrightarrow{l} s''}$ with $l \in L$, it holds that $s' \equiv s''$ or Φ_0 contradicts Φ_1.

Unlike the semantic condition of determinism, it is decidable whether a finite TSS is in the syntactic determinism format with respect to a set of labels L.

Theorem 20. *The problem of deciding whether a finite TSS is in the syntactic determinism format with respect to a set of labels is decidable.*

Proof. Assume that we are given a finite TSS $\mathcal{T}$ and a finite subset of its labels L. Observe that, given two finite sets of transition formulae Φ_0 and Φ_1, it is decidable whether Φ_0 contradicts Φ_1. The condition in Definition 19 can therefore be effectively checked because the set of deduction rules in $\mathcal{T}$ is finite, and the set of premises of each rule in $\mathcal{T}$ is also finite. The conditions in Definition 18 can also be checked effectively because the number of rules is finite and so is the set of premises of each rule in $\mathcal{T}$. □

The following theorem states that, for normalized TSSs, Definition 19 implies Definition 12.

Theorem 21. *Each normalized TSS in the syntactic determinism format with respect to L is also in the determinism format with respect to L.*

Proof. Let $\mathcal{T}$ be a normalized TSS in the syntactic determinism format with respect to L. Condition 1 of Definition 12 is satisfied since $\mathcal{T}$ is normalized. (To see this, consider the first two conditions in Definition 18.)

To prove condition 2 of Definition 12, let $r = \frac{\Phi_0}{t_0 \xrightarrow{l} t_0'}$ and $r' = \frac{\Phi_1}{t_1 \xrightarrow{l} t_1'}$ be distinct rules of $\mathcal{T}$ and (σ, σ') be a determinism-respecting pair of substitutions with respect to (Φ_0, Φ_1) and L such that $\sigma(t_0) \equiv \sigma'(t_1)$. Since $\mathcal{T}$ is normalized, both r and r' are f-defining for some function symbol f, i.e., $t_0 = f(\vec{s})$ and $t_1 = f(\vec{t})$. Furthermore, since $f(\vec{s}) \equiv f(\vec{t})$, we have that σ and σ' agree on all variables appearing in $f(\vec{s}) \equiv f(\vec{t})$.

In order to prove the theorem, it suffices to show only the following claim.

Claim: $\sigma(v) \equiv \sigma'(v)$ for each $v \in vars(r) \cap vars(r')$.

Indeed, using the above claim, we can prove the theorem as follows. Definition 19 yields that either $t_0' \equiv t_1'$ or Φ_0 contradicts Φ_1. If $t_0' \equiv t_1'$, then variables in $vars(t_0') = vars(t_1')$ are all source dependent via transitions in L that are common to both Φ_0 and Φ_1 (by condition 3 of Definition 18). By the above-mentioned claim, $\sigma(t_0') \equiv \sigma'(t_1')$ and condition 2 of Definition 12 follows, which was to be shown. If Φ_0 contradicts Φ_1, then assume that the premises negating each other are $\phi_j \equiv s_j \xrightarrow{l_j} s_j'$ and $\phi_{j'} \equiv t_{j'} \xrightarrow{l_j}$ and it holds that $s_j \equiv t_{j'}$. All variables in $t_{j'} \equiv s_j$ are source dependent via transitions in L

(by condition 3 of Definition 18). It follows from the claim that $\sigma(s_j) \equiv \sigma'(t_{j'})$ and thus $\sigma(\phi_j)$ contradicts $\sigma'(\phi_{j'})$, which again implies condition 2 of Definition 12.

We now proceed to prove the claim. For each variable $v \in vars(r) \cap vars(r')$, we define its *common source distance* to be the source distance of v when only taking the formulae in $\Phi_0 \cap \Phi_1$ into account. Note that such a common source distance exists since, by condition 3 of Definition 18, all $v \in vars(r) \cap vars(r')$ are source dependent via a subset of $\{\xrightarrow{l} \mid l \in L\}$ included in the collection of transition relations used in $\Phi_0 \cap \Phi_1$.

We prove the claim by an induction on the common source distance of $v \in vars(r) \cap vars(r')$. If $v \in vars(f(\vec{s}))$, then we know that $\sigma(v) \equiv \sigma'(v)$ (since $t_0 \equiv t_1$ and $\sigma(t_0) \equiv \sigma'(t_1)$). Otherwise, since v is source dependent in r via transitions with labels in L, there is a positive premise $u \xrightarrow{l} u'$ in Φ_0 with $l \in L$ such that $v \in vars(u')$ and all variables in u are source dependent with a shorter common source distance than that for v. Furthermore, since v appears in both rules, i.e., $v \in vars(r) \cap vars(r')$, this premise also appears in Φ_1 according to condition 3 of Definition 18, and thus $vars(u) \subseteq vars(r) \cap vars(r')$. By the induction hypothesis, we have that $\sigma(u) \equiv \sigma'(u)$, and since (σ, σ') is determinism respecting with respect to (Φ_0, Φ_1) and L, we know that $\sigma(u') \equiv \sigma'(u')$. Specifically, the substitutions must agree on the value of v, i.e., $\sigma(v) \equiv \sigma'(v)$ as desired. □

The following statement is thus a corollary of Theorems 21 and 14.

Corollary 22. *Consider a normalized TSS with (C, U) as its least three-valued stable model and a subset L of its labels. If the TSS is in the (syntactic) determinism format with respect to L (according to Definition 19), then C is deterministic with respect to any $l \in L$.*

It is natural to ask oneself whether the syntactic determinism format can be easily generalized. The following examples show that relaxing the restrictions of that format may jeopardize Theorem 21. (Examples showing the need for the first two conditions in Definition 18 were already discussed in Example 16.)

In order to see that condition 3a in Definition 18 is necessary, consider a TSS with constants a and b, and a binary function symbol f with the following rules.

$$\frac{}{f(x_0, x_1) \xrightarrow{a} x_0} \qquad \frac{}{f(x_0, y_1) \xrightarrow{a} y_1}.$$

Each of these rules is f-defining and each variable occurring in them is source dependent via the empty set of transition relations. Moreover, the condition in Definition 19 is vacuously met since the sources of the two deduction rules are different. It is easy to see that $f(a, b) \xrightarrow{a} a$ and $f(a, b) \xrightarrow{a} b$. Therefore the transition relation $\xrightarrow{a}$ is not deterministic.

In order to see that condition 3b in Definition 18 is necessary, consider a TSS with constants 0, a and a^2, and a binary function symbol f with the following rules.

$$\frac{}{a \xrightarrow{a} 0} \qquad \frac{}{a^2 \xrightarrow{a} a} \qquad \frac{x_0 \xrightarrow{a} y}{f(x_0, x_1) \xrightarrow{a} y} \qquad \frac{x_1 \xrightarrow{a} y}{f(x_0, x_1) \xrightarrow{a} y}.$$

This TSS meets all the conditions in Definitions 18 and 19, apart from condition 3b. It is easy to see that $f(a, a^2) \xrightarrow{a} 0$ and $f(a, a^2) \xrightarrow{a} a$. Therefore the transition relation $\xrightarrow{a}$ is not deterministic.

The need for the condition in Definition 19 is exemplified by the classic non-deterministic choice operator discussed in Example 43 to follow. The rules for this operator satisfy the conditions in Definition 18, but not the one in Definition 19. As remarked upon already in Example 16, the transition relations defined by those rules are non-deterministic except for "trivial TSSs" including this operator.

Remark 23. The reader might wonder whether for each TSS there is a normalized one that defines the same transition relation. This is false. As an example, consider the TSS with constants a and b, and a unary function symbol with rules

$$\frac{}{f(a) \xrightarrow{a} a} \qquad \frac{}{f(b) \xrightarrow{a} a}.$$

One can convince oneself that any normalized TSS over this signature defining the transition relation $\{f(a) \xrightarrow{a} a, f(b) \xrightarrow{a} a\}$ would have to contain a rule of the form

$$\frac{N}{f(x) \xrightarrow{a} t},$$

for some collection of negative formulae N. We may assume, without loss of generality, that the above rule proves a transition rule with conclusion $f(a) \xrightarrow{a} a$. From this fact, it is not too hard to see that such a rule could also be instantiated to prove either $f(b) \xrightarrow{a} b$ (if t is a variable) or $f(f(a)) \xrightarrow{a} a$ (if $t \equiv a$), contradicting the assumption that the normalized TSS defines the transition relation $\{f(a) \xrightarrow{a} a, f(b) \xrightarrow{a} a\}$.

Although the syntactic determinism format presented in Definition 19 is more straightforward to check than the format presented in Definition 12, the format in Definition 12 is interesting for the following reasons:

1. it is indeed more general than the syntactic determinism format,
2. it demonstrates and justifies the way we arrived at the syntactic format, and hence is interesting for pedagogical reasons, and
3. the correctness proof for the syntactic format relies conveniently on the one for the more semantic format from Definition 12 (although a direct proof is certainly possible).

3.3. Examples

In this section, we present some examples of various TSSs from the literature and apply our (syntactic) determinism format to them. Some of the examples we discuss below are based on TSSs with predicates. The extension of our formats with predicates is straightforward and we discuss it briefly below.

Definition 24 (*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 and P their label. 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, contradiction, consistency and three-valued stable models are then naturally extended by adopting the more general notion of formulae. In particular, the formulae $P\ t$ and $\neg P\ t$ contradict each other. The label of a deduction rule is either the label of the transition formula or of the predicate formula in its conclusion.

Definitions 12 and 19 apply unchanged to a setting with predicates, as do Theorems 14 and 21.

Example 25 (*Conjunctive Nondeterministic Processes*)**.** In their paper [16], Hennessy and Plotkin define a language, called conjunctive nondeterministic processes, for studying logical characterizations of processes. The signature of the language consists of a constant 0, a unary action prefixing operator '$a.\_$' for each $a \in A$, and a binary conjunctive nondeterminism operator '$\vee$'. The operational semantics of this language is defined by the following deduction rules.

$$\frac{}{0\ \mathrm{can}_a} \qquad \frac{}{a.x\ \mathrm{can}_a} \qquad \frac{x\ \mathrm{can}_a}{x \vee y\ \mathrm{can}_a} \qquad \frac{y\ \mathrm{can}_a}{x \vee y\ \mathrm{can}_a}$$

$$\frac{}{0\ \mathrm{after}_a\ 0} \qquad \frac{}{a.x\ \mathrm{after}_a\ x} \qquad \frac{}{a.x\ \mathrm{after}_b\ 0}\ a \neq b$$

$$\frac{x\ \mathrm{after}_a\ x' \quad y\ \mathrm{after}_a\ y'}{x \vee y\ \mathrm{after}_a\ x' \vee y'}.$$

The above TSS is in the (syntactic) determinism format with respect to A. Hence, we can conclude that the transition relations after_a are deterministic.

Example 26 (*Delayed Choice*)**.** The second example we discuss is a subset of the process algebra $\mathrm{BPA}_{\delta\epsilon} + \mathrm{DC}$ [5], i.e., Basic Process Algebra with deadlock and empty process extended with delayed choice. First we restrict attention to the fragment of this process algebra without non-deterministic choice '$+$' and with action prefix '$a.\_$' instead of general sequential composition '$\cdot$'. This altered process algebra has the following deduction rules, where a ranges over the set of actions A:

$$\frac{}{\epsilon\downarrow} \qquad \frac{}{a.x \xrightarrow{a} x} \qquad \frac{x\downarrow}{x \mp y\downarrow} \qquad \frac{y\downarrow}{x \mp y\downarrow}$$

$$\frac{x \xrightarrow{a} x' \quad y \xrightarrow{a} y'}{x \mp y \xrightarrow{a} x' \mp y'} \qquad \frac{x \xrightarrow{a} x' \quad y \overset{a}{\not\to}}{x \mp y \xrightarrow{a} x'} \qquad \frac{x \overset{a}{\not\to} \quad y \xrightarrow{a} y'}{x \mp y \xrightarrow{a} y'}.$$

In the above specification, predicate $p\downarrow$ denotes the possibility of termination for process p. The intuition behind the delayed choice operator, denoted by '$\_ \mp \_$', is that the choice between two components is only resolved when one performs an action that the other cannot perform. When both components can perform an action, the delayed choice between them remains unresolved and the two components synchronize on the common action. This transition system specification is in the (syntactic) determinism format with respect to A.

The addition of non-deterministic choice '$+$' or sequential composition '$\cdot$' results in deduction rules that do not satisfy the determinism format. For example, the addition of sequential composition comes with the following deduction rules:

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

The sets of premises of these rules do not contradict each other. The extended TSS is indeed non-deterministic since, for example, $(\epsilon \mp (a.\epsilon)) \cdot (a.\epsilon) \xrightarrow{a} \epsilon$ and $(\epsilon \mp (a.\epsilon)) \cdot (a.\epsilon) \xrightarrow{a} \epsilon \cdot (a.\epsilon)$.

Example 27 (*Time Transitions I*)**.** This example deals with the Algebra of Timed Processes, ATP, of Nicollin and Sifakis [24]. In the TSS given below, we specify the time transitions (denoted by label χ) of delayable deadlock 'δ', non-deterministic choice '$\_ \oplus \_$', unit-delay operator '$\lfloor \_ \rfloor \_$' and parallel composition '$\_ \parallel \_$'.

$$\frac{}{\delta \xrightarrow{\chi} \delta} \qquad \frac{x \xrightarrow{\chi} x' \quad y \xrightarrow{\chi} y'}{x \oplus y \xrightarrow{\chi} x' \oplus y'} \qquad \frac{}{\lfloor x \rfloor(y) \xrightarrow{\chi} y} \qquad \frac{x \xrightarrow{\chi} x' \quad y \xrightarrow{\chi} y'}{x \parallel y \xrightarrow{\chi} x' \parallel y'}.$$

These deduction rules all trivially satisfy the determinism format for time transitions since the sources of conclusions of different deduction rules cannot be unified. Also the additional operators involving time, namely, the delay operator '$\lfloor \_ \rfloor^d \_$', execution delay operator '$\lceil \_ \rceil^d \_$' and unbounded start delay operator '$\lfloor \_ \rfloor^\omega$', satisfy the determinism format for time transitions. The deduction rules are given below, for $d \geq 1$:

$$\frac{}{\lfloor x \rfloor^1(y) \xrightarrow{\chi} y} \qquad \frac{x \xrightarrow{\chi} x'}{\lfloor x \rfloor^{d+1}(y) \xrightarrow{\chi} \lfloor x' \rfloor^d(y)} \qquad \frac{x \xrightarrow{\chi}}{\lfloor x \rfloor^{d+1}(y) \xrightarrow{\chi} \lfloor x \rfloor^d(y)}$$

$$\frac{x \xrightarrow{\chi} x'}{\lfloor x \rfloor^\omega \xrightarrow{\chi} \lfloor x' \rfloor^\omega} \qquad \frac{x \xrightarrow{\chi}}{\lfloor x \rfloor^\omega \xrightarrow{\chi} \lfloor x \rfloor^\omega}$$

$$\frac{x \xrightarrow{\chi} x'}{\lceil x \rceil^1(y) \xrightarrow{\chi} y} \qquad \frac{x \xrightarrow{\chi} x'}{\lceil x \rceil^{d+1}(y) \xrightarrow{\chi} \lceil x' \rceil^d(y)}.$$

Example 28 (*Time Transitions II*)**.** Most of the timed process algebras that originate from the Algebra of Communicating Processes (ACP) from [8,7], such as those reported in [6], have a deterministic time transition relation as well.

In the TSS given below, the unary time unit delay operator is denoted by 'σ_{rel}', nondeterministic choice is denoted by '$+$', and sequential composition is denoted by '$\cdot$'. The deduction rules for the time transition relation for this process algebra are the following:

$$\frac{}{\sigma_{\mathrm{rel}}(x) \xrightarrow{1} x} \qquad \frac{x \xrightarrow{1} x' \quad y \xrightarrow{1} y'}{x + y \xrightarrow{1} x' + y'} \qquad \frac{x \xrightarrow{1} x' \quad y \not\xrightarrow{1}}{x + y \xrightarrow{1} x'} \qquad \frac{x \not\xrightarrow{1} \quad y \xrightarrow{1} y'}{x + y \xrightarrow{1} y'}$$

$$\frac{x \xrightarrow{1} x' \quad x \not\downarrow}{x \cdot y \xrightarrow{1} x' \cdot y} \qquad \frac{x \xrightarrow{1} x' \quad y \not\xrightarrow{1}}{x \cdot y \xrightarrow{1} x' \cdot y}$$

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

Note that here we have an example of deduction rules, the first two deduction rules for time transitions of a sequential composition, for which the premises do not contradict each other. Still these deduction rules satisfy the determinism format since the targets of the conclusions are identical. In the syntactically richer framework of [29], where arbitrary first-order logic formulae over transitions are allowed, those two deduction rules are presented by a single rule with premise $x \xrightarrow{1} x' \wedge (x \not\downarrow \vee y \not\xrightarrow{1})$.

Sometimes such timed process algebras have an operator for specifying an arbitrary delay, denoted by 'σ^*_{rel}', with the following deduction rules.

$$\frac{x \not\xrightarrow{1}}{\sigma^*_{\mathrm{rel}}(x) \xrightarrow{1} \sigma^*_{\mathrm{rel}}(x)} \qquad \frac{x \xrightarrow{1} x'}{\sigma^*_{\mathrm{rel}}(x) \xrightarrow{1} x' + \sigma^*_{\mathrm{rel}}(x)}.$$

The premises of these rules contradict each other and so the TSS also satisfies the conditions of our (syntactic) determinism format.

3.4. Other forms of determinism

One may also wish to consider stronger forms of determinism than the one considered so far in this section, which is the standard one in the concurrency-theory literature and underlies results such as those presented in [12,28]. In what follows we consider two variations on Definition 8 and show how the rule format we presented in this section can be easily modified to guarantee them.

Definition 29 (*Strong Determinism*)**.** A transition relation T is called *strongly deterministic* when, if $p \xrightarrow{l} p' \in T$ and $p \xrightarrow{l'} p'' \in T$, then $p' \equiv p''$.

So, in a strongly deterministic transition relation, all the outgoing transitions from a closed term have the same target. One can easily modify the formats in Definitions 12 and 19 to guarantee strong determinism as follows. We first present a modification of the format offered in Definition 12. In defining a format for strong determinism, the following variation on Definition 11 will be useful.

Definition 30. A pair of substitutions (σ, σ') is *strong-determinism respecting* with respect to a pair of sets of formulae (Φ, Φ') when, for all two positive formulae $s \xrightarrow{l} s' \in \Phi$ and $t \xrightarrow{l'} t' \in \Phi'$, it holds that $\sigma(s) \equiv \sigma'(t)$ only if $\sigma(s') \equiv \sigma'(t')$.

Definition 31 (*Strong-Determinism Format*)**.** We say that a TSS is in the strong-determinism format if the following conditions hold.

1. In each deduction rule $\frac{\Phi}{t \xrightarrow{l} t'}$, each variable $v \in vars(t')$ is source dependent.
2. For each pair of distinct deduction rules $\frac{\Phi_0}{t_0 \xrightarrow{l} t_0'}$ and $\frac{\Phi_1}{t_1 \xrightarrow{l'} t_1'}$ and for each strong-determinism-respecting pair of substitutions (σ, σ') with respect to (Φ_0, Φ_1) such that $\sigma(t_0) \equiv \sigma'(t_1)$, it holds that either $\sigma(t_0') \equiv \sigma'(t_1')$ or $\sigma(\Phi_0)$ contradicts $\sigma'(\Phi_1)$.

By essentially replaying the proof of Theorem 14, we can show the following result to the effect that the conditions in the definition of the strong-determinism format are sufficient to guarantee strong determinism of the induced transition relation.

Theorem 32. *Consider a TSS with (C, U) as its least three-valued stable model. If the TSS is in the strong-determinism format, then C is strongly deterministic.*

Following the earlier developments in this section, we next present a rule format, whose conditions have a purely syntactic form and are based on a variation on those in Definitions 18 and 19, that guarantees strong determinism of the induced transition relation.

Definition 33 (*Syntactic Strong-Determinism Format*)**.** We say that a TSS is strongly normalized when

1. each deduction rule is f-defining for some function symbol f,
2. for each deduction rule of the form
$$(r)\ \frac{\Phi_r}{f(\vec{s}) \xrightarrow{l} s'},$$
each variable $v \in vars(s')$ is source dependent in r, and
3. for each function symbol f and each pair of deduction rules of the form
$$(r)\ \frac{\Phi_r}{f(\vec{s}) \xrightarrow{l} s'} \qquad (r')\ \frac{\Phi_{r'}}{f(\vec{t}) \xrightarrow{l'} t'}$$
the following conditions are satisfied:
   (a) the sources of the conclusions coincide, i.e., $f(\vec{s}) \equiv f(\vec{t})$,
   (b) for each variable $v \in vars(r) \cap vars(r')$ there is a set of formulae in $\Phi_r \cap \Phi_{r'}$ proving its source dependency (both in r and r').

A strongly normalized TSS is in the (syntactic) strong-determinism format when, for each two deduction rules $\frac{\Phi_0}{f(\vec{s}) \xrightarrow{l} s'}$ and $\frac{\Phi_1}{f(\vec{s}) \xrightarrow{l'} s''}$, it holds that $s' \equiv s''$ or Φ_0 contradicts Φ_1.

By mimicking the proof of Theorem 21, we can now show the following analogue of that result, which implies, together with Theorem 32, that the syntactic strong determinism format guarantees strong determinism of the induced transition relation.

Theorem 34. *Each TSS in the syntactic strong-determinism format is also in the strong-determinism format.*

It follows immediately from Definition 29 that each strongly deterministic transition relation is also deterministic with respect to the set of labels of a TSS. We now introduce a further strengthening of the notion of (strong) determinism.

Definition 35 (*Functional Transition Relation*)**.** A transition relation T is called *functional* when, if $p \xrightarrow{l} p' \in T$ and $p \xrightarrow{l'} p'' \in T$, then $l = l'$ and $p' \equiv p''$.

In a functional transition relation each closed term affords at most one transition. Below, we limit ourselves to presenting a modification of the format offered in Definition 12 that guarantees that the induced transition relation is functional.

Definition 36 (*Functional Determinism Format*)**.** A TSS $\mathcal{T}$ is in the functional determinism format if the following conditions hold.

1. In each deduction rule $\frac{\Phi}{t \xrightarrow{l} t'}$, each variable $v \in vars(t')$ is source dependent.
2. For each pair of distinct deduction rules $\frac{\Phi_0}{t_0 \xrightarrow{l} t'_0}$ and $\frac{\Phi_1}{t_1 \xrightarrow{l'} t'_1}$ and for each pair of substitutions (σ, σ') such that $\sigma(t_0) \equiv \sigma'(t_1)$, it holds that
   (a) either $l = l'$ and $\sigma(t'_0) \equiv \sigma'(t'_1)$
   (b) or $\sigma(\Phi_0)$ contradicts $\sigma'(\Phi_1)$.

The proof of Theorem 14 can be re-used to show the following result. Namely, the conditions in the definition of the functional determinism format indeed guarantee that the induced transition relation is functional.

Theorem 37. *Consider a TSS with (C, U) as its least three-valued stable model. If the TSS is in the functional determinism format, then C is functional.*

We have not made any attempt to maximize the level of generality of the above rule formats. Our aim was rather to show how to obtain rule formats for other forms of determinism one might be interested in guaranteeing as variations of the ones we considered in this section.

4. Idempotence

Our agenda in this section is to present a rule format that guarantees the idempotence of certain binary operators. In the definition of our rule format, we rely implicitly on the work presented in the previous section. Indeed, as Definition 40 and the examples to follow will make clear, a widely applicable rule format for idempotence makes an essential use of the determinism of certain transition relations.

4.1. Format

For the sake of clarity, we begin by defining formally the notion of idempotence for a binary operator.

Definition 38 (*Idempotence*)**.** A binary operator $f \in \Sigma$ is *idempotent with respect to an equivalence* $\sim$ on closed terms if and only if, for each $p \in \mathbb{C}(\Sigma)$, it holds that $f(p, p) \sim p$.

Idempotence is defined with respect to a notion of behavioral equivalence. There are various notions of behavioral equivalence defined in the literature, which are, by and large, weaker than bisimilarity defined below. Thus, to be as general as possible, we prove our idempotence result for all notions that include, i.e., are weaker than, bisimilarity.

Definition 39 (*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 well known that bisimilarity is indeed an equivalence. (See, for instance, [19] for a textbook proof of this fact.) 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 $\sigma : V \to \mathbb{C}(\Sigma)$. In the remainder of this paper, we restrict our attention to the notions of equivalence on *closed terms* that include strong bisimilarity. However, all our results carry over (without any change) to the notions on *open terms* that include strong bisimilarity on open terms in the above sense. Indeed, if $f(x, x) \leftrightarrow x$ and $\leftrightarrow$ is included in $\sim$, then $f(x, x) \sim x$ also holds.

Definition 40 (*The Idempotence Rule Format*)**.** Let $\gamma : L \times L \to L$ be a partial function such that $\gamma(l_0, l_1) \in \{l_0, l_1\}$ if it is defined. We define the following two rule forms.

1_l. *Choice rules.* A choice rule is a rule of the following form.

$$\frac{\{x_i \xrightarrow{l} t\} \cup \Phi}{f(x_0, x_1) \xrightarrow{l} t}, \quad i \in \{0, 1\}$$

$2_{l_0,l_1}$. *Communication rules.* A communication rule is a rule of the following form, where $\gamma(l_0, l_1)$ is defined.

$$\frac{\{x_0 \xrightarrow{l_0} t_0,\ x_1 \xrightarrow{l_1} t_1\} \cup \Phi}{f(x_0, x_1) \xrightarrow{\gamma(l_0,l_1)} f(t_0, t_1)}, \quad t_0 \equiv t_1 \text{ or } (l_0 = l_1 \text{ and } \xrightarrow{l_0} \text{ is deterministic}).$$

In each case, Φ can be an arbitrary, possibly empty, set of (positive or negative) formulae.

In addition, we define the starred version of each form, 1_l^* and $2^*_{l_0,l_1}$.

1_l^*. *Choice rules.*

$$\frac{\{x_i \xrightarrow{l} x_i'\}}{f(x_0, x_1) \xrightarrow{l} x_i'}, \quad i \in \{0, 1\}$$

$2^*_{l_0,l_1}$. *Communication rules.*

$$\frac{\{x_0 \xrightarrow{l_0} x_0',\ x_1 \xrightarrow{l_1} x_1'\}}{f(x_0, x_1) \xrightarrow{\gamma(l_0,l_1)} f(x_0', x_1')}, \quad x_0' \equiv x_1' \text{ or } (l_0 = l_1 \text{ and } \xrightarrow{l_0} \text{ is deterministic}).$$

As above, in a communication rule of the form $2^*_{l_0,l_1}$, we assume that $\gamma(l_0, l_1)$ is defined.

A TSS is in *idempotence format with respect to a binary operator* f if each deduction rule is g-defining for some operator g, and each f-defining rule is of the forms 1_l or $2_{l_0,l_1}$, for some $l, l_0, l_1 \in L$, and for each label $l \in L$ there exists at least one rule of the forms 1_l^* or $2^*_{l,l}$.

We should note that the starred versions of the forms are special cases of their unstarred counterparts; for example a rule which has form 1_l^* also has form 1_l.

Intuitively, the presence of rules of the form 1_l^* or $2^*_{l,l}$ for each label l guarantees that, for each closed term p, the term $f(p, p)$ can mimic the behaviour of p. Conversely, the constraint that each rule for f is of the forms 1_l or $2_{l_0,l_1}$, for some $l, l_0, l_1 \in L$, ensures that transitions from $f(p, p)$ can be simulated by transitions from p.

Theorem 41. *Assume that a TSS is complete and is in the idempotence format with respect to a binary operator* f. *Then,* f *is idempotent with respect to any equivalence* $\sim$ *such that* $\underline{\leftrightarrow} \subseteq \sim$.

Proof. First define the relation $\simeq_f \subseteq \mathbb{C}(\Sigma) \times \mathbb{C}(\Sigma)$ as follows.

$$\simeq_f = \{(p, p), (p, f(p, p)), (f(p, p), p) \mid p \in \mathbb{C}(\Sigma)\}.$$

To prove the theorem, it suffices to show that $\simeq_f$ is a bisimulation relation. If it is, then $f(p, p) \underline{\leftrightarrow} p$ for any closed term p and, since $\underline{\leftrightarrow} \subseteq \sim$, we obtain the theorem.

Let (C, U) be the least three-valued stable model for the TSS under consideration. First, consider a closed term p such that $p \xrightarrow{l} p' \in C$ for some l and p'. (Note that $U = \emptyset$ since the TSS is complete.) Next, we argue that $f(p, p) \xrightarrow{l} p''$ for some p'' such that $p' \simeq_f p''$. Since $p \xrightarrow{l} p' \in C$, there exists a provable transition rule of the form $\frac{N}{p \xrightarrow{l} p'}$ for some set of negative formulae N such that $C \models N$. (In particular, that means that $p \overset{l}{\nrightarrow} \notin N$.) In this case we make use of the requirement that there exists at least one rule of a starred form for label l. If there exists a rule of the form 1_l^*, i.e.,

$$\frac{x_i \xrightarrow{l} x'}{f(x_0, x_1) \xrightarrow{l} x'}, \quad i \in \{0, 1\}$$

then we can instantiate it, using the transition $p \xrightarrow{l} p'$ as premise, to prove that $f(p, p) \xrightarrow{l} p' \in C$. In particular, it does not matter whether $i = 0$ or $i = 1$. Since $\simeq_f$ is reflexive, $p' \simeq_f p'$ holds. If there exists a rule of the form $2^*_{l,l}$, we observe that $\gamma(l, l) = l$, so the transition rule becomes

$$\frac{x_0 \xrightarrow{l} x_0' \quad x_1 \xrightarrow{l} x_1'}{f(x_0, x_1) \xrightarrow{l} f(x_0', x_1')},$$

where $x_0' \equiv x_1'$ or $\xrightarrow{l}$ is deterministic. Now we can use the existence of $p \xrightarrow{l} p'$ to satisfy both premises and obtain that $f(p, p) \xrightarrow{l} f(p', p')$. By the definition of $\simeq_f$, we also have that $p' \simeq_f f(p', p')$. In either case, if $p \xrightarrow{l} p' \in C$, then there exists a p'' such that $f(p, p) \xrightarrow{l} p'' \in C$ and $p' \simeq_f p''$.

Now assume that $f(p, p) \xrightarrow{k} p' \in C$. Then there exists a provable transition rule $\frac{N}{f(p,p) \xrightarrow{k} p'}$ for some set of negative formulae N such that $C \models N$. Since each rule is g-defining for some g and all rules for f are either of the form 1_l or $2_{l_0,l_1}$, this provable transition rule must be based on a rule of those forms. We analyze each possibility separately, showing that in each case $p \xrightarrow{k} p''$ for some p'' such that $p' \simeq_f p''$.

If the rule is based on a rule of form 1_l, its positive premises must also be provable. In particular, it must hold that $p \xrightarrow{k} p' \in C$, since both x_0 and x_1 in the rule are instantiated to p. The other premises are of no consequence to this conclusion and, again, we observe that $p' \simeq_f p'$.

Now consider the case where the transition is a consequence of a rule of the form $2_{l_0,l_1}$. If $t_0 \equiv t_1$, say both are instantiated to p'', we must consider two cases, namely $k = l_0$ and $k = l_1$. If $k = l_0$, then the first premise of the rule actually states that $p \xrightarrow{k} p''$. If $k = l_1$, then the second premise similarly states that $p \xrightarrow{k} p''$. In either case, we note that $p' \equiv f(p'', p'')$ must hold and, again by the definition of $\simeq_f$, we have that $f(p'', p'') \simeq_f p''$.

If, however, $t_0 \not\equiv t_1$, the side condition requires that $l_0 = l_1 = k$, which also implies that $\gamma(l_0, l_1) = l_0 = k$, and that the transition relation $\xrightarrow{l_0}$ is deterministic. In this case it is easy to see that the right-hand sides of the first two premises, namely t_0 and t_1, evaluate to the same closed term in the proof structure, say p''. The conclusion then states that $k = l_0$ and $p' \equiv f(p'', p'')$. It must thus hold that $p \xrightarrow{k} p'' \in C$ and $f(p'', p'') \simeq_f p''$, as before.

From this we obtain that if $f(p, p) \xrightarrow{k} p' \in C$ then there exists a p'' such that $p \xrightarrow{k} p'' \in C$ and $p' \simeq_f p''$. Thus, $\simeq_f$ is a bisimulation, as required. □

4.2. Relaxing the restrictions

In this section we consider the constraints of the idempotence rule format and show that they cannot be dropped without jeopardizing the meta-theorem. We remark at the outset, however, that the requirement that all deduction rules be g-defining for some g is not strictly necessary in order to prove Theorem 41. Its presence simplifies our technical developments and does not reduce the applicability of our results. Indeed, all of the examples of use of our rule format for idempotence we are aware of use only g-defining rules.

First of all we note that, in rule form 1_l, it is necessary that the label of the premise matches the label of the conclusion. If it does not, in general, we cannot prove that $f(p, p)$ simulates p or vice versa. This requirement can be stated more generally for both rule forms in Definition 40; the label of the conclusion must be among the labels of the premises. The requirement that $\gamma(l, l') \in \{l, l'\}$ exists to ensure this constraint for form $2_{l,l'}$. A simple synchronization rule provides a counter-example that shows why this restriction is needed. Consider the following TSS with constants 0, τ, a and $\bar{a}$ and two binary operators $+$ and $\|$:

$$\frac{}{\alpha \xrightarrow{\alpha} 0} \qquad \frac{x \xrightarrow{\alpha} x'}{x + y \xrightarrow{\alpha} x'} \qquad \frac{y \xrightarrow{\alpha} y'}{x + y \xrightarrow{\alpha} y'} \qquad \frac{x \xrightarrow{a} x' \quad y \xrightarrow{\bar{a}} y'}{x \parallel y \xrightarrow{\tau} x' \parallel y'}$$

where α is τ, a or $\bar{a}$. Here it is easy to see that, although $(a + \bar{a}) \parallel (a + \bar{a})$ has an outgoing τ-transition, $a + \bar{a}$ does not afford such a transition.

The condition that for each l at least one rule of the forms 1^*_l or $2^*_{l,l}$ must exist comprises a few constraints on the rule format. First of all, it says there must be at least one f-defining rule. If not, it is easy to see that there could exist a process p where $f(p, p)$ deadlocks (since there are no f-defining rules) but p does not. It also states that there must be at least one rule in the starred form, where the targets are restricted to variables. To motivate these constraints, consider the following TSS.

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

The processes a and $f(a, a)$ are not bisimilar as the former can perform an a-transition but the latter is stuck. The starred forms also require that Φ is empty, i.e., there is no testing. This is necessary in the proof because, in the presence of extra premises, we cannot in general instantiate such a rule to show that $f(p, p)$ simulates p. Finally, the condition requires that, if we rely on a rule of the form $2^*_{l,l'}$ and $t_0 \not\equiv t_1$, then the labels l and l' in the premises of the rule must coincide. To see why, consider a TSS containing a *left synchronize* operator $\lfloor\rfloor$—that is, one that synchronizes a step from each operand but uses the label of the left one. Here we let $\alpha \in \{a, \bar{a}\}$.

$$\frac{}{\alpha \xrightarrow{\alpha} 0} \qquad \frac{x \xrightarrow{\alpha} x'}{x + y \xrightarrow{\alpha} x'} \qquad \frac{y \xrightarrow{\alpha} y'}{x + y \xrightarrow{\alpha} y'} \qquad \frac{x \xrightarrow{a} x' \quad y \xrightarrow{\bar{a}} y'}{x \lfloor\rfloor y \xrightarrow{a} x' \lfloor\rfloor y'}.$$

In this TSS the processes $(a + \bar{a})$ and $(a + \bar{a}) \lfloor\rfloor (a + \bar{a})$ are not bisimilar since the latter does not afford an $\bar{a}$-transition whereas the former does.

For rules of form $2_{l,l'}$, we require that either $t_0 \equiv t_1$, or that the mentioned labels are the same and the associated transition relation is deterministic. This requirement is necessary in the proof to ensure that the target of the conclusion fits our definition of $\simeq_f$, i.e., the operator is applied to two identical terms. Consider the following TSS where $\alpha \in \{a, b\}$.

$$\frac{}{a \xrightarrow{a} a} \qquad \frac{}{a \xrightarrow{a} b} \qquad \frac{}{b \xrightarrow{b} b} \qquad \frac{x \xrightarrow{\alpha} x' \quad y \xrightarrow{\alpha} y'}{x \mid y \xrightarrow{\alpha} x' \mid y'}.$$

For the operator |, this violates the condition $t_0 \equiv t_1$ (note that $\xrightarrow{a}$ is not deterministic). We observe that $a \mid a \xrightarrow{a} a \mid b$. The only possibility for a to simulate this a-transition is either with $a \xrightarrow{a} a$ or with $a \xrightarrow{a} b$. However, neither a nor b can be bisimilar to $a \mid b$, because both a and b have outgoing transitions while $a \mid b$ is stuck. Therefore a and $a \mid a$ cannot be bisimilar. If $t_0 \not\equiv t_1$, we must require that the labels match, $l_0 = l_1$, and that $\xrightarrow{l_0}$ is deterministic. We require the labels to match because, if they do not, then given only $p \xrightarrow{l} p'$ it is, in general, impossible to prove that $f(p, p)$ can simulate it using only a $2^*_{l,l'}$ rule. For example, consider the following TSS.

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

Then $f(a, a)$ does not afford an a-labelled transition, unlike a. Therefore f is not idempotent.

The determinacy of the transition with label $l_0 = l_1$ is necessary when proving that transitions from $f(p, p)$ can, in general, be simulated by p; if we assume that $f(p, p) \xrightarrow{l} p'$, then we must be able to conclude that p' has the shape $f(p'', p'')$ for some p'', in order to meet the bisimulation condition for $\simeq_f$. As another example of the use of determinism in rule $2^*_{l,l'}$, consider the standard choice operator $+$ and prefixing operator . of Milner's Calculus of Communicating Systems (CCS) with the | operator from the last example, with $\alpha \in \{a, b, c\}$.

$$\frac{}{\alpha \xrightarrow{\alpha} 0} \qquad \frac{}{\alpha.x \xrightarrow{\alpha} x} \qquad \frac{x \xrightarrow{\alpha} x'}{x + y \xrightarrow{\alpha} x'} \qquad \frac{y \xrightarrow{\alpha} y'}{x + y \xrightarrow{\alpha} y'} \qquad \frac{x \xrightarrow{\alpha} x' \quad y \xrightarrow{\alpha} y'}{x \mid y \xrightarrow{\alpha} x' \mid y'}.$$

If we let $p = a.b + a.c$, then $p \mid p \xrightarrow{a} b \mid c$ and $b \mid c$ is stuck. However, p cannot simulate this transition with respect to $\simeq_f$. Indeed, p and $p \mid p$ are not bisimilar.

4.3. Predicates

There are many examples of TSSs where predicates are used. The definitions presented in Sections 2 and 4 can be easily adapted to deal with predicates as well. For example, the notion of bisimulation (Definition 39), and thus the notion of idempotence (Definition 38), are extended naturally to the setting with predicates, by requiring that, for each two closed terms and each predicate, one term satisfies the predicate if and only if the other one satisfies the predicate. To extend the idempotence rule format to a setting with predicates, the following types of rules for predicates are introduced:

3_P. *Choice rules for predicates.*

$$\frac{\{Px_i\} \cup \Phi}{Pf(x_0, x_1)}, \quad i \in \{0, 1\}$$

4_P. *Synchronization rules for predicates.*

$$\frac{\{Px_0,\ Px_1\} \cup \Phi}{Pf(x_0, x_1)}.$$

As before, we define the starred version of these forms, 3^*_P and 4^*_P.

3^*_P. *Choice rules for predicates.*

$$\frac{\{Px_i\}}{Pf(x_0, x_1)}, \quad i \in \{0, 1\}$$

4^*_P. *Synchronization rules for predicates.*

$$\frac{\{Px_0,\ Px_1\}}{Pf(x_0, x_1)}.$$

With these additional definitions, the idempotence format is defined as follows.
A TSS with predicates is in *idempotence format with respect to a binary operator* f if

- each rule is g-defining for some operator g, and
- each f-defining rule, i.e., a deduction rule with f appearing in the source of the conclusion, is of one the forms 1_l, $2_{l_0,l_1}$, 3_P or 4_P, for some $l, l_0, l_1 \in L$ or for some predicate symbol P. Moreover, for each $l \in L$, there exists at least one f-defining rule of the forms 1^*_l or $2^*_{l,l}$, and for each predicate symbol P there is an f-defining rule of the form 3^*_P or 4^*_P.

A simple modification of the proof of Theorem 41 yields the following result stating the correctness of the idempotence format in a setting with predicates.

Theorem 42. *Assume that a TSS with predicates is complete and is in the idempotence format with respect to a binary operator f. Then, f is idempotent with respect to any equivalence $\sim$ such that $\leftrightarrow \subseteq \sim$.*

4.4. Examples

In this section, we present a number of examples from the literature that witness the applicability of the idempotence format.

Example 43. A most prominent example of an idempotent operator is non-deterministic choice, denoted by '$+$'. It typically has the following deduction rules, where a ranges over the collection of labels.

$$\frac{x_0 \xrightarrow{a} x_0'}{x_0 + x_1 \xrightarrow{a} x_0'} \qquad \frac{x_1 \xrightarrow{a} x_1'}{x_0 + x_1 \xrightarrow{a} x_1'}.$$

Clearly, these are in the idempotence format with respect to $+$.

Example 44 (*External Choice*)**.** The well-known external choice operator '$\Box$' from Hoare's Communicating Sequential Processes (CSP) [17] has the following deduction rules.

$$\frac{x_0 \xrightarrow{a} x_0'}{x_0 \,\Box\, x_1 \xrightarrow{a} x_0'} \qquad \frac{x_1 \xrightarrow{a} x_1'}{x_0 \,\Box\, x_1 \xrightarrow{a} x_1'}$$
$$\frac{x_0 \xrightarrow{\tau} x_0'}{x_0 \,\Box\, x_1 \xrightarrow{\tau} x_0' \,\Box\, x_1} \qquad \frac{x_1 \xrightarrow{\tau} x_1'}{x_0 \,\Box\, x_1 \xrightarrow{\tau} x_0 \,\Box\, x_1'}.$$

Note that the third and fourth deduction rule are not instances of any of the allowed types of deduction rules. Therefore, no conclusion about the validity of idempotence can be drawn from our format. In this case this does not point to a limitation of our format, because this operator is not idempotent in strong bisimulation semantics as observed in, e.g., [11].

Example 45 (*Strong Time-Deterministic Choice*)**.** The choice operator that is used in the timed process algebra ATP [24] has the following deduction rules.

$$\frac{x_0 \xrightarrow{a} x_0'}{x_0 \oplus x_1 \xrightarrow{a} x_0'} \qquad \frac{x_1 \xrightarrow{a} x_1'}{x_0 \oplus x_1 \xrightarrow{a} x_1'} \qquad \frac{x_0 \xrightarrow{\chi} x_0' \quad x_1 \xrightarrow{\chi} x_1'}{x_0 \oplus x_1 \xrightarrow{\chi} x_0' \oplus x_1'}.$$

The idempotence of this operator follows from our format since the last rule for '$\oplus$' fits the form $2^*_{\chi,\chi}$ because, as we remarked in Example 27, the transition relation $\xrightarrow{\chi}$ is deterministic.

Example 46 (*Weak Time-Deterministic Choice*)**.** The version of the choice operator '$+$' that is used in most ACP-style timed process algebras has the following deduction rules.

$$\frac{x_0 \xrightarrow{a} x_0'}{x_0 + x_1 \xrightarrow{a} x_0'} \qquad \frac{x_1 \xrightarrow{a} x_1'}{x_0 + x_1 \xrightarrow{a} x_1'}$$
$$\frac{x_0 \xrightarrow{1} x_0' \quad x_1 \xrightarrow{1} x_1'}{x_0 + x_1 \xrightarrow{1} x_0' + x_1'} \qquad \frac{x_0 \xrightarrow{1} x_0' \quad x_1 \not\xrightarrow{1}}{x_0 + x_1 \xrightarrow{1} x_0'} \qquad \frac{x_0 \not\xrightarrow{1} \quad x_1 \xrightarrow{1} x_1'}{x_0 + x_1 \xrightarrow{1} x_1'}.$$

The third deduction rule is of the form $2^*_{1,1}$ (since the transition relation $\xrightarrow{1}$ is deterministic, as remarked in Example 28). The others are of forms 1^*_a and 1_1. This operator is idempotent, and this follows from our Theorem 41.

Example 47 (*Conjunctive Nondeterminism*)**.** The operator '$\vee$' as defined in Example 25 by means of the deduction rules

$$\frac{x \text{ can}_a}{x \vee y \text{ can}_a} \qquad \frac{y \text{ can}_a}{x \vee y \text{ can}_a} \qquad \frac{x \text{ after}_a\ x' \quad y \text{ after}_a\ y'}{x \vee y \text{ after}_a\ x' \vee y'}$$

satisfies the idempotence format (extended to a setting with predicates). The first two deduction rules are of the form $3^*_{\text{can}_a}$ and the last one is of the form $2^*_{a,a}$. (Here we have used the fact that the transition relations after_a are deterministic, as concluded in Example 25.)

Example 48 (*Delayed Choice*)**.** Delayed choice can be concluded to be idempotent in the restricted setting without '+' and '·' by using the idempotence format and the fact that in this restricted setting the transition relations $\xrightarrow{a}$ are deterministic. (See Example 26.)

$$\frac{x\downarrow}{x \mp y\downarrow} \qquad \frac{y\downarrow}{x \mp y\downarrow}$$

$$\frac{x \xrightarrow{a} x' \quad y \xrightarrow{a} y'}{x \mp y \xrightarrow{a} x' \mp y'} \qquad \frac{x \xrightarrow{a} x' \quad y \overset{a}{\not\to}}{x \mp y \xrightarrow{a} x'} \qquad \frac{x \overset{a}{\not\to} \quad y \xrightarrow{a} y'}{x \mp y \xrightarrow{a} y'}.$$

The first two deduction rules are of form $3^*_{\downarrow}$, the third one is a $2^*_{a,a}$ rule, and the others are 1_a rules. Note that for any label a and for the predicate $\downarrow$ a starred rule is present.

For the extensions discussed in Example 26, idempotence cannot be established using our rule format since the transition relations are no longer deterministic. In fact, delayed choice is not idempotent in those cases.

As witnessed by the examples discussed in this section, our format for idempotence is widely applicable. Indeed, it covers all the practical cases from the literature that we have discovered so far, which is an indication of its expressiveness and relevance. However, the constraints of this format can be slightly generalized to cater for more possible applications in the future, such as the (artificial) example presented below.

Example 49. Consider a TSS with constant a^ω, and with binary operations f and g with the following rules.

$$\frac{}{a^\omega \xrightarrow{a} a^\omega} \qquad \frac{x_0 \xrightarrow{a} x'_0, \ x_1 \xrightarrow{a} x'_1}{f(x_0, x_1) \xrightarrow{a} g(x'_0, x'_1)} \qquad \frac{x_0 \xrightarrow{a} x'_0, \ x_1 \xrightarrow{a} x'_1}{g(x_0, x_1) \xrightarrow{a} f(x'_0, x'_1)}.$$

It is not hard to see that both f and g are idempotent. Indeed, the transition relation $\xrightarrow{a}$ is deterministic and the symmetric closure of the relation

$$\{(f(p, p), p), (g(p, p), p) \mid p \text{ a closed term}\}$$

is a bisimulation. However, the TSS is neither in the idempotence format with respect to f nor in the idempotence format with respect to g, in the sense of Definition 40. Indeed, neither the rule for f nor the rule for g is of the form $2_{a,a}$ because the targets of their conclusions do not have the required form.

Note that f is idempotent because so is g, and vice versa.

The above example points to a (mostly theoretical) limitation of the format we proposed in Definition 40. Indeed, in order for an operation f to be idempotent, it is not necessary that the targets of conclusions of rules of the form $2_{l_0,l_1}$ have f as head operator. As in the above example, in rules of that type, it would be enough to have a target of the conclusion of the form $g(t_0, t_1)$, where g is itself an operator whose idempotence can be shown using the format. In other words, an operation f is guaranteed to be idempotent if

- its rules satisfy the constraints in Definition 40, but
- the targets of conclusions of rules of the form $2_{l_0,l_1}$ have the form $g(t_0, t_1)$, where g is itself guaranteed to be idempotent.

By considering the largest set of binary operators that satisfy the (generalized) constraints quoted above, one obtains a more general format that can easily deal with Example 49. The proof of correctness for the generalized format is almost identical to the proof of Theorem 41. Namely, assume that I is the largest set of binary operators satisfying the generalized constraints, given above. We claim that each operator $f \in I$ is idempotent with respect to any relation $\sim$ that includes bisimilarity. To prove our claim, it suffices to show that the relation $\simeq_I \subseteq \mathbb{C}(\Sigma) \times \mathbb{C}(\Sigma)$ defined below is a bisimulation relation.

$$\simeq_I = \{(p, p), (p, g(p, p)), (g(p, p), p) \mid p \in \mathbb{C}(\Sigma), g \in I\}.$$

Indeed, if it is, then $f(p, p) \leftrightarrows p$ for any closed term p and $f \in I$. Therefore, $f(p, p) \sim p$ for any closed term p and each relation $\sim$ such that $\leftrightarrows \subseteq \sim$, which establishes our claim.

Proving that $\simeq_I$ is a bisimulation relation is done following the lines of the proof of Theorem 41.

As we mentioned earlier, we are not aware of any operation from the literature whose idempotence cannot be established using the format in Definition 40, which is easier to apply and to check than its generalization. This is the reason why we have presented first the simpler, but widely applicable, format. We cannot rule out, however, that practical examples that can only be handled using the generalized format we just offered may appear in the future.

5. Conclusions

In this paper, we have presented rule formats guaranteeing the determinism of certain transitions and the idempotence of binary operators. Our rule formats cover all practical cases of determinism and idempotence that we have thus far encountered in the literature.

We plan to extend our rule formats with the addition of data/store. Such an extension would, for instance, allow us to account for the determinism of the time transition relations in the hybrid process algebra presented in [9] and to deal with process calculi with data and (fragments of) programming languages.

Also, it is interesting to study the addition of structural congruences pertaining to idempotence to the TSSs in our idempotence format.

Last, but not least, we think that it would be worthwhile to investigate the robustness of the properties established using our syntactic rule formats with respect to taking disjoint extensions of languages in the sense of [2].

Acknowledgements

We thank the anonymous reviewers for their very careful reading of the paper and for their constructive suggestions, which led to several improvements. Any remaining infelicity is solely our responsibility.

Appendix. Proof of Theorem 9

We show that the problem of deciding whether a universal two-counter machine diverges on input n reduces to the problem of determining whether some closed term U_n is deterministic for label a with respect to the transition relation associated with a complete, finite transition system specification. To this end, we exhibit a finite transition system specification with, for each n, a term U_n that behaves like a universal two-counter machine on input n and performs a-labelled transitions as it computes. The a-labelled transition relation will be deterministic, when restricted to the set of terms that are reachable from U_n, iff the universal two-counter machine diverges on input n.

Recall that a universal two-counter machine operates on two counters I and J. The machine has a sequence of labelled instructions $\ell_1, \ldots, \ell_k$, which can take one of the following forms:

- halt,
- inc X, where X is either I or J,
- dec X, where X is either I or J,
- goto ℓ_i, where $1 \leq i \leq k$, and
- if $X = 0$ then goto ℓ_i, where X is either I or J, and $1 \leq i \leq k$.

The meaning of those instructions is the expected one. On input n, the machine starts computing from instruction ℓ_1 with $I = n$ and $J = 0$. The computation terminates if at any point the distinguished instruction halt is reached. We can assume, without loss of generality, that the instruction labelled ℓ_k is the distinguished halt instruction.

We now construct a finite transition system specification $\mathcal{U}$ that can "simulate" the above-mentioned universal two-counter machine. The signature of $\mathcal{U}$ contains a constant z (representing the number zero), a unary prefix operation 's' (which will be used to implement the successor operation on the natural numbers), and binary operation symbols $\ell_1, \ldots, \ell_k$.

The behaviour of the constant z and of the prefixing operation s is described by the rules

$$\frac{}{z \xrightarrow{z} z} \qquad \frac{}{s.x \xrightarrow{s} x}.$$

If the ith instruction is the increment of a counter, say inc I, then ℓ_i has rule

$$\frac{}{\ell_i(x, y) \xrightarrow{a} \ell_{i+1}(s.x, y)}.$$

If the ith instruction is the decrement of a counter, say dec I, then ℓ_i has rules

$$\frac{x \xrightarrow{z} x'}{\ell_i(x, y) \xrightarrow{a} \ell_{i+1}(x', y)} \qquad \frac{x \xrightarrow{s} x'}{\ell_i(x, y) \xrightarrow{a} \ell_{i+1}(x', y)}.$$

If the ith instruction is an unconditional jump goto ℓ_j, where $1 \leq j \leq k$, then ℓ_i has rule

$$\frac{}{\ell_i(x, y) \xrightarrow{a} \ell_j(x, y)}.$$

If the ith instruction is a conditional jump, say if $I = 0$ then goto ℓ_j, where $1 \leq j \leq k$, then ℓ_i has rules

$$\frac{x \xrightarrow{z} x'}{\ell_i(x, y) \xrightarrow{a} \ell_j(x', y)} \qquad \frac{x \xrightarrow{s} x'}{\ell_i(x, y) \xrightarrow{a} \ell_{i+1}(s.x', y)}.$$

Finally, if the ith instruction is halt, then ℓ_i has rules

$$\frac{}{\ell_i(x, y) \xrightarrow{a} z} \qquad \frac{}{\ell_i(x, y) \xrightarrow{a} s.z}.$$

Define

$$U_n = \ell_1(\underbrace{s.s.\ldots.s}_{n \text{ times}}.z, z).$$

Then it is easy to see that the transition relation $\xrightarrow{a}$, when restricted to the set of terms that are reachable from U_n, is deterministic if, and only if, the universal two-counter machine does not terminate its computation on input $I = n$. This completes the proof.

References

[1] L. Aceto, A. Birgisson, A. Ingolfsdottir, M. Mousavi, M.A. Reniers, Rule formats for determinism and idempotence, in: Proceedings of the Third IPM International Conference on Fundamentals of Software Engineering, FSEN 2009, in: Lecture Notes in Computer Science, vol. 5961, Springer, 2010, pp. 146–161.
[2] L. Aceto, B. Bloom, F.W. Vaandrager, Turning SOS rules into equations, Information and Computation 111 (1994) 1–52.
[3] L. Aceto, W.J. Fokkink, C. Verhoef, Structural operational semantics, in: J.A. Bergstra, A. Ponse, S.A. Smolka (Eds.), Handbook of Process Algebra, Elsevier Science, Dordrecht, The Netherlands, 2001, pp. 197–292 (Chapter 3).
[4] L. Aceto, A. Ingolfsdottir, M. Mousavi, M.A. Reniers, A rule format for unit elements, in: Proceedings of the 36th International Conference on Current Trends in Theory and Practice of Computing, SOFSEM 2010, in: Lecture Notes in Computer Science, vol. 5901, Springer, 2010, pp. 141–152.
[5] J. Baeten, S. Mauw, Delayed choice: An operator for joining Message Sequence Charts, in: Proceedings of Formal Description Techniques, in: IFIP Conference Proceedings, vol. 6, Chapman & Hall, 1995, pp. 340–354.
[6] J. Baeten, C.A. Middelburg, Process Algebra with Timing, in: Monographs in Theoretical Computer Series: An EATCS Series, Springer, 2002.
[7] J. Baeten, W.P. Weijland, Process algebra, in: Cambridge Tracts in Theoretical Computer Science, vol. 18, Cambridge University Press, 1990.
[8] J.A. Bergstra, J.W. Klop, Process algebra for synchronous communication, Information and Control 60 (1–3) (1984) 109–137.
[9] J.A. Bergstra, C.A. Middelburg, Process algebra for hybrid systems, Theoretical Computer Science 335 (2–3) (2005) 215–280.
[10] S. Cranen, M. Mousavi, M.A. Reniers, A rule format for associativity, in: F. van Breugel, M. Chechik (Eds.), Proceedings of the 19th International Conference on Concurrency Theory, CONCUR'08, in: Lecture Notes in Computer Science, vol. 5201, Springer, 2008, pp. 447–461.
[11] P. D'Argenio, τ-angelic choice for process algebras (revised version), Tech. rep., LIFIA, Depto. de Informática, Fac. de Cs. Exactas, Universidad Nacional de La Plata, 1995.
[12] J. Engelfriet, Determinacy → (observation equivalence = trace equivalence), Theoretical Computer Science 36 (1985) 21–25.
[13] W.J. Fokkink, T.D. Vu, Structural operational semantics and bounded nondeterminism, Acta Informatica 39 (6–7) (2003) 501–516.
[14] J.F. Groote, Transition system specifications with negative premises, Theoretical Computer Science 118 (2) (1993) 263–299.
[15] J.F. Groote, F.W. Vaandrager, Structured operational semantics and bisimulation as a congruence, Information and Computation 100 (2) (1992) 202–260.
[16] M. Hennessy, G. Plotkin, Finite conjunctive nondeterminism, in: K. Voss, H.J. Genrich, G. Rozenberg (Eds.), Concurrency and Nets, Advances in Petri Nets, Springer, 1987, pp. 233–244.
[17] C.A.R. Hoare, Communicating Sequential Processes, Prentice Hall, 1985.
[18] R. Lanotte, S. Tini, Probabilistic congruence for semistochastic generative processes, in: V. Sassone (Ed.), Proceedings of the 8th International Conference on Foundations of Software Science and Computational Structures, FOSSACS'05, in: Lecture Notes in Computer Science, vol. 3441, Springer, 2005, pp. 63–78.
[19] A.R. Milner, Communication and Concurrency, Prentice Hall, 1989.
[20] M. Mousavi, I.C. Phillips, M. Reniers, I. Ulidowski, Semantics and expressiveness of ordered SOS, Information and Computation 207 (2) (2009) 85–119.
[21] M. Mousavi, M. Reniers, J.F. Groote, A syntactic commutativity format for SOS, Information Processing Letters 93 (2005) 217–223.
[22] M. Mousavi, M.A. Reniers, Orthogonal extensions in structural operational semantics, in: Proceedings of the 32nd International Colloquium on Automata, Languages and Programming, ICALP'05, in: Lecture Notes in Computer Science, vol. 3580, Springer, 2005, pp. 1214–1225.
[23] M. Mousavi, M.A. Reniers, J.F. Groote, SOS formats and meta-theory: 20 years after, Theoretical Computer Science 373 (3) (2007) 238–272.
[24] X. Nicollin, J. Sifakis, The algebra of timed processes ATP: theory and application, Information and Computation 114 (1) (1994) 131–178.
[25] G.D. Plotkin, A structural approach to operational semantics, Journal of Logic and Algebraic Programming 60–61 (2004) 17–139.
[26] S. Tini, Rule formats for compositional non-interference properties, Journal of Logic and Algebraic Programming 60–61 (2004) 353–400.
[27] I. Ulidowski, S. Yuen, Process languages with discrete relative time based on the ordered SOS format and rooted eager bisimulation, Journal of Logic and Algebraic Programming 60–61 (2004) 401–460.
[28] F.W. Vaandrager, Determinism → (event structure isomorphism = step sequence equivalence), Theoretical Computer Science 79 (2) (1991) 275–294.
[29] M. van Weerdenburg, M. Reniers, Structural operational semantics with first-order logic, Electronic Notes in Theoretical Computer Science 229 (4) (2009) 85–106.
[30] C. Verhoef, A congruence theorem for structured operational semantics with predicates and negative premises, Nordic Journal of Computing 2 (2) (1995) 274–302.