A Syntactic Commutativity Format for SOS

MohammadReza Mousavi, Michel Reniers and Jan Friso Groote

Department of Computer Science,
Eindhoven University of Technology,
Post Box 513, NL-5600MB, Eindhoven,
The Netherlands
{m.r.mousavi, m.a.reniers, j.f.groote}@tue.nl

Abstract

Considering operators defined using Structural Operational Semantics (SOS), commutativity axioms are intuitive properties that hold for many of them. Proving this intuition is usually a laborious task, requiring several pages of boring and standard proof. To save this effort, we propose a syntactic SOS format which guarantees commutativity for a set of composition operators.

Key words: Formal Semantics, Structural Operational Semantics (SOS), Standard SOS Formats, Commutativity.

1 Introduction

Structural Operational Semantics [10] has become a de facto standard in defining operational semantics for specification and programming languages. Hence, developing meta-theorems for SOS specifications can be beneficial to a large community of researchers in different areas of computer science and can save them a lot of repetitive effort in proving theorems about their theories. Congruence formats for different notions of equality are the best known examples of such meta-theorems (see [2] for an overview) which guarantee a particular notion of equality to be a congruence provided that the SOS rules in the specification conform to a certain syntactic format. Deriving algebraic axioms for SOS rules in [1, 4] are other examples in this direction which try to generate a set of sound and complete axioms for a given operational semantics in a syntactic format. Although both [1] and [4] generate, among others commutativity axioms, they assume the existence of a number of standard constants and operators in the signature.

In this paper, we aim at developing a meta-theorem for deriving commutativity axioms for certain operators in an SOS specification. Our format does not assume the presence of any special operator and builds upon a general congruence format, namely tyft [8]. The ultimate goal of this line of research is to develop the necessary theoretical background for a tool-set that can assist specifiers in developing Structural Operational Semantics for their languages, by proving different properties for the developed languages automatically.

The rest of this paper is organized as follows. In Section 2, we start by presenting some preliminary notions about (Structural) Operational Semantics, congruence, standard congruence formats and commutativity. Then, in Section 3 we give our proposal for a syntactic format for commutativity called comm-tyft (for *comm*utative *tyft*). Section 4 addresses possible extensions of this format by adding tyxt rules, predicates and negative premises to the format (thus, achieving the expressivity of PANTH [12]). Finally, Section 5 summarizes the results and presents concluding remarks.

2 Preliminaries

2.1 Transition System Specification

Definition 1 (Signature and Terms) We assume an infinite set of variables V (with typical members x, y, x', y', x_0, $y_0 \ldots$). A *signature* Σ is a set of function symbols (operators) with fixed arity. Functions with zero arity are called constants. A term $t \in T(\Sigma)$ is defined inductively as follows: a variable $x \in V$ is a term, if $t_0, \ldots, t_{n-1}$ are terms then for all $f \in \Sigma$ with arity n, $f(t_0, \ldots, t_{n-1})$ is a term, as well (i.e., constants are indeed terms). Terms are typically denoted by $t, t', t_i, t'_i, \ldots$. All terms are considered open terms. Closed terms $C(\Sigma)$ are defined as expected and typically denoted by $p, q, p', q', p_0, q_0, p'_0, q'_0 \ldots$. A substitution σ replaces a variable in a term with another term. The set of variables appearing in term t is denoted by $vars(t)$.

Definition 2 (Transition System Specification (TSS)) A *transition system specification* is a tuple (Σ, L, Rel, D) where Σ is a signature, L is a set of labels (with typical members $l, l', l_0, \ldots$), Rel is a set of transition relation symbols, and D is a set of deduction rules, where for all $r \in Rel$, $l \in L$ and $s, s' \in T(\Sigma)$ we define that $(s, s') \in \stackrel{l}{\rightarrow}_r$ is a formula. A deduction rule $dr \in D$, is defined as a tuple (H, c) where H is a set of formulae and c is a formula. The formula c is called the *conclusion* and the formulae from H are called *premises*.

Notions of open and closed extend to formulae as expected. Also, the concept of substitution is lifted to formulae and sets of formulae in the natural way (i.e., a substitution applied to a formula, applies to both terms). A formula $(s, s') \in \stackrel{l}{\rightarrow}_r$ is denoted by the more intuitive notation $s \stackrel{l}{\rightarrow}_r s'$, as well. We refer to s as the source and to s' as the target of the transition.

A deduction rule (H, c) is mostly denoted by $\dfrac{H}{c}$.

Definition 3 A *proof* of a closed formula ϕ is a well-founded upwardly branching tree of which the nodes are labelled by closed formulae such that

- the root node is labelled by ϕ, and
- if ψ is the label of a node and $\{\psi_i \mid i \in I\}$ is the set of labels of the nodes directly above this node, then there are a deduction rule $\dfrac{\{\chi_i \mid i \in I\}}{\chi}$ and a substitution σ such that $\sigma(\chi) = \psi$, and for all $i \in I$, $\sigma(\chi_i) = \psi_i$.

2.2 Bisimulation, Congruence and Standard Formats

Definition 4 (Bisimulation [9]) A relation $R \subseteq C(\Sigma) \times C(\Sigma)$ is a *bisimulation* relation if and only if $\forall_{p,q,l,r}\ (p, q) \in R \Rightarrow$

1. $\forall_{p'}\ p \stackrel{l}{\rightarrow}_r p' \Rightarrow \exists_{q'}\ q \stackrel{l}{\rightarrow}_r q'\ \wedge (p', q') \in R$;
2. $\forall_{q'}\ q \stackrel{l}{\rightarrow}_r q' \Rightarrow \exists_{p'}\ p \stackrel{l}{\rightarrow}_r p'\ \wedge (p', q') \in R$.

Two closed terms p and q are *bisimilar*, denoted by $p \leftrightarrow q$, if and only if there exists a bisimulation relation R such that $(p, q) \in R$.

Definition 5 (Congruence) A relation $R \subseteq T(\Sigma) \times T(\Sigma)$ is a congruence relation with respect to an n-ary function symbol $f \in \Sigma$ if and only if for all terms $p_i, q_i \in T(\Sigma)$, if $(p_i, q_i) \in R$ $(0 \leq i < n)$ then $(f(p_0, \ldots, p_{n-1}), f(q_0, \ldots, q_{n-1})) \in R$. Furthermore, R is called a *congruence* for a transition system specification if and only if it is a congruence with respect to all function symbols of the signature. The congruence closure of a relation R is the minimal congruence relation containing R.

Definition 6 (Tyft Format [8]) A deduction rule is in tyft format if and only if it has the following shape.

$$\frac{\{t_i \xrightarrow{l_i}_{r_i} y_i | i \in I\}}{f(x_0, \ldots, x_{n-1}) \xrightarrow{l}_r t}$$

where x_i and y_i are all distinct variables, f is an n-ary function symbol from the signature (e.g., sequential composition, parallel composition, etc.), I is a (possibly infinite) set of indices and t and t_i's are arbitrary terms. We call such a deduction rule an *f-defining* rule. A transition system specification is in tyft format if and only if all its deduction rules are.

For a deduction rule, consider a directed graph in which the nodes are the premises of the rule and edges denote dependencies between two premises, i.e., when a source of a premise uses a variable that appears in the target of the other, there is an edge from the latter to the former. We call such a graph the *variable dependency graph* and for simplicity of our proofs, we assume its acyclicity (i.e., well-foundedness of premises with respect to the variable dependency ordering). However, we believe that following the result of [5], this requirement can be relaxed in our setting, as well.

2.3 Commutativity

In this section, we define the notions of commutativity and the closure of commutativity under context. Here, we assume commutativity for all arguments of the operator. It is straightforward to restrict the results of this paper to prove commutativity for a subset of arguments.

Definition 7 (Commutativity) An n-ary function symbol $f \in \Sigma$ is called commutative on open (closed) terms with respect to relation $R \subseteq T(\Sigma) \times T(\Sigma)$ if and only if for all terms $t_i \in T(\Sigma)$ ($t_i \in C(\Sigma)$) such that $0 \leq i < n$ and for all j and k such that $0 \leq j < k < n$, $(f(t_0, \ldots, t_{n-1}), f(t_0, \ldots, t_k, \ldots, t_j, \ldots, t_{n-1})) \in R$. Function f is called commutative (in our setting) if and only if it is commutative on closed terms with respect to bisimilarity.

To account for swapping of arguments of a commutative operator under arbitrary context, we define the notion of commutative congruence as follows.

Definition 8 (Commutative Congruence) Consider a set of function symbols $COMM \subseteq \Sigma$. The commutative congruence relation $\sim_{cc}$ (w.r.t. $COMM$ and Σ) is the minimal relation satisfying the following requirements:

1. $\sim_{cc}$ is reflexive;
2. $\forall_{p_i, q_i (0 \leq i < n)} p_i \sim_{cc} q_i \Rightarrow$
$$\begin{cases} f(p_0, \ldots, p_{n-1}) \sim_{cc} f(q_0, \ldots, q_{n-1}) \\ g(p_0, \ldots, p_k, \ldots, p_j, \ldots, p_{n-1}) \sim_{cc} g(q_0, \ldots, q_{n-1}) \\ (\forall_{j,k} 0 \leq j < k < n \text{ and for arbitrary } n\text{-ary} \\ \text{ function symbols } f \in \Sigma \text{ and } g \in COMM) \end{cases}$$

It can easily be seen that $\sim_{cc}$ is an equivalence relation and thus, it partitions the terms into equivalence classes $[t]_{cc}$. Two formulae $t_i \xrightarrow{l}_r t'_i$ and $t_j \xrightarrow{l}_r t'_j$ are called CC-equal if and only if $t_i \sim_{cc} t_j$ and $t'_i = t'_j$ (for technical reasons, the definition is asymmetric with respect to the source and target of the transition). By abusing the same notation, we denote the set of CC-equal formulae of $t_i \xrightarrow{l}_r t'_i$ by $[t_i \xrightarrow{l}_r t'_i]_{cc}$ and for a set H of formulae we write $[H]_{cc}$ for $\bigcup_{h \in H} [h]_{cc}$.

As it appears from its name, there is more to Definition 8 than only commutativity. It also exploits congruence to switch arguments of commutative operators in $COMM$ in an arbitrary context. The reason for adding congruence to commutativity is to make our commutativity format as general as possible. By taking the minimal reflexive and commutative relation (on open terms) instead of $\sim_{cc}$, one can obtain a simpler commutativity format compared to what we define in the

remainder (and a simpler proof for it). This simpler format works equally well for the examples that we have checked so far. However, we prefer the existing formulation for its generality in order to cope with more possible applications in the future. The following example illustrates Definition 8.

Example 1 Suppose that signature Σ contains a binary operator $||$ and $||\in COMM$; according to Definition 8, for arbitrary terms $t_0, t_1, t_2 \in T(\Sigma)$, we have $[(t_0 \,||\, (t_1 \,||\, t_2)]_{cc} = \{t_0 \,||\, (t_1 \,||\, t_2)$, $t_0 \,||\, (t_2 \,||\, t_1)$, $(t_1 \,||\, t_2) \,||\, t_0$, $(t_2 \,||\, t_1) \,||\, t_0\}$. Furthermore, we have $(x_2 \,||\, x_1) \,||\, x_0 \stackrel{l}{\rightarrow}_r y_0 \in [x_0 \,||\, (x_1 \,||\, x_2) \stackrel{l}{\rightarrow}_r y_0]_{cc}$.

The following lemma shows that $\sim_{cc}$ is preserved under substitutions that respect $\sim_{cc}$.

Lemma 1 If $t \sim_{cc} t'$ (with respect to $COMM$), and for all $x \in vars(t)$, $\sigma(x) \sim_{cc} \sigma'(x)$, then $\sigma(t) \sim_{cc} \sigma'(t')$.

Proof By an induction on the structure of $\sim_{cc}$:

- If the pair (t, t') is in $\sim_{cc}$ due to reflexivity, then the lemma holds due to a straightforward induction on the size of t: If t and t' are both a constant or a variable, then the lemma holds vacuously. Otherwise, if $t = t' = f(t_0, \ldots, t_{n-1})$, then it follows from the induction hypothesis that $\sigma(t_i) \sim_{cc} \sigma'(t_i)$ and since $\sim_{cc}$ is closed under congruence, we have: $f(\sigma(t_0), \ldots, \sigma(t_{n-1})) \sim_{cc} f(\sigma'(t_0), \ldots, \sigma'(t_{n-1}))$ and hence $\sigma(t) \sim_{cc} \sigma'(t)$.
- If $p = f(t_0, \ldots, t_{n-1})$ and $t' = f(t'_0, \ldots, t'_{n-1})$, where $t_i \sim_{cc} t'_i$, for all $0 \leq i < n$, then according to the induction hypothesis, we have $\sigma(t_i) \sim_{cc} \sigma'(t'_i)$ and it follows from the same constraint that $f(\sigma(t_0), \ldots, \sigma(t_{n-1})) \sim_{cc} f(\sigma'(t'_0), \ldots, \sigma'(t'_{n-1}))$. Hence, $\sigma(t) = f(\sigma(t_0), \ldots, \sigma(t_{n-1})) \sim_{cc} f(\sigma'(t'_0), \ldots, \sigma'(t'_{n-1})) = \sigma'(t')$.
- If $p = f(t_0, \ldots, t_j, \ldots, t_k, \ldots, t_{n-1})$ and $t' = f(t'_0, \ldots, t'_{n-1})$, where $0 \leq j < k < n$ and $t_i \sim_{cc} t'_i$, for all $0 \leq i < n$, then according to the induction hypothesis, we have $\sigma(t_i) \sim_{cc} \sigma'(t'_i)$ and it follows from the same constraint that $f(\sigma(t_0), \ldots, \sigma(t_j), \ldots, \sigma(t_k), \ldots, \sigma(t_{n-1})) \sim_{cc} f(\sigma'(t'_0), \ldots, \sigma'(t'_{n-1}))$. Hence, $\sigma(t) \sim_{cc} \sigma'(t')$.

$\boxtimes$

3 Standard Format for Commutativity

We set our starting point from a standard congruence format (namely, tyft format) because in parts of our proofs, congruence is an essential ingredient.

Definition 9 (Comm-tyft) A transition system specification is in comm-tyft format with respect to a set of function symbols $COMM \subseteq \Sigma$ if all its deduction rules are in tyft format and for every f-defining rule with $f \in COMM$ of the following form

$$\textbf{(d)} \frac{\{t_i \stackrel{l_i}{\rightarrow}_{r_i} y_i | i \in I\}}{f(x_0, \ldots, x_{n-1}) \stackrel{l}{\rightarrow}_r t}$$

for which we denote the set of premises of **(d)** by H and the conclusion by c, and for all $0 \leq j < k < n$, there exists a deduction rule **(d')** of the following form in the transition system specification

$$\textbf{(d')} \frac{\{t'_j \stackrel{l'_j}{\rightarrow}_{r'_j} y'_j | j \in J\}}{f(x'_0, \ldots, x'_{n-1}) \stackrel{l}{\rightarrow}_r t'}$$

and a bijective mapping (substitution) $\hbar$ on variables such that

- $\hbar(x'_i) = x_i$ for $0 \leq i < n$, $i \neq j$ and $i \neq k$,
- $\hbar(x'_j) = x_k$ and $\hbar(x'_k) = x_j$,
- $\hbar(t') \sim_{cc} t$,
- $\forall_{j \in J}\ \hbar(t'_j \xrightarrow{l'_j}_{r'_j} y'_j) \in [H]_{cc} \cup \{c\}$.

Deduction rule **(d')** is called the *commutative mirror* of **(d)** (on arguments j and k).

To put it informally, the role of substitution $\hbar$ in this definition is to account for the single swapping in the source of the conclusion and a possible isomorphic renaming of variables. Thus, the above format requires that for each f-defining rule, there exists a commutative mirror which firstly, has the same source of the conclusion as the original deduction rule with one swapping in the arguments, secondly, has the same source of the premises and target of the conclusion as the original rule up to arbitrary swapping of the arguments of commutative function symbols, and finally, may have the conclusion of the original rule as one of its premises. Next, we state that comm-tyft format indeed induces commutativity for the set of operators under consideration.

Theorem 1 (Commutativity for comm-tyft) If a transition system specification is in comm-tyft with respect to a set of operators $COMM$, then all operators in $COMM$ are commutative.

Proof Let R be the relation $\sim_{cc}$ restricted to closed terms. By definition, this relation contains all the desired pairs of terms (of the form $(f(p_0, \ldots, p_{n-1}),\ f(p_0, \ldots, p_k, \ldots, p_j, \ldots,\ p_{n-1})))$. Then, it only remains to prove that R is a bisimulation relation.

Consider an arbitrary pair $(p, q) \in R$. Suppose that $p \xrightarrow{l}_r p'$ for some r, l, p'. We have to prove the existence of a q' such that $q \xrightarrow{l}_r q'$ and $(p', q') \in R$ (and vice versa, the proof of which we omit due to symmetry). We start with a case distinction using the structure of R in Definition 8.

1. For the reflexivity part of R, the theorem holds trivially.
2. For the congruence part, the theorem follows due to a similar construction as that of [8] since comm-tyft is a restriction of tyft format. We quote the following Lemma from [8] which is a necessary ingredient of our proof.

   Lemma 2 Consider a relation $R \subseteq T(\Sigma) \times T(\Sigma)$ which is closed under congruence. If for substitutions σ and σ' and a term $t \in T(\Sigma)$, it holds that $\forall_{x \in vars(t)}(\sigma(x), \sigma'(x)) \in R$, then $(\sigma(t), \sigma'(t)) \in R$.

   Since (p, q) is in R due to congruence, we have that $p = f(p_0, \ldots, p_n)$, $q = f(q_0, \ldots, q_n)$, for some n-ary $f \in \Sigma$ and $(p_i, q_i) \in R$ $(0 \leq i < n)$. We proceed with an induction on the depth of the proof for transition $p \xrightarrow{l}_r p'$.

   If the transition has a proof of depth one, then there is a rule **(d)** of the following form:

   $$\textbf{(d)}\ \frac{}{f(x_0, \ldots, x_{n-1}) \xrightarrow{l}_r t}$$

   and a substitution σ such that $\sigma(x_i) = p_i$ $(0 \leq i < n)$ and $\sigma(t) = p'$. By defining a new substitution σ' as:

   $$\sigma'(x) := \begin{cases} q_i & \text{if } x = x_i (0 \leq i < n) \\ \sigma(x) & \text{otherwise} \end{cases}$$

   we have a proof for $f(q_0, \ldots q_n) \xrightarrow{l}_r \sigma'(t)$ using **(d)** and σ'. It follows from the definition of σ' that $\forall_{x \in V}(\sigma(x), \sigma'(x)) \in R$ and thus according to Lemma 2, $(\sigma(t), \sigma'(t)) \in R$ and this concludes the induction basis.

For the induction hypothesis, suppose that transition $p \xrightarrow{l}_r p'$ has a proof of depth n due to the following rule (as the root of its proof tree):

$$\textbf{(d)}\frac{\{t_i \xrightarrow{l_i}_{r_i} y_i | i \in I\}}{f(x_0, \ldots, x_{n-1}) \xrightarrow{l}_r t}$$

and a substitution σ such that $\sigma(x_i) = p_i$ $(0 \leq i < n)$ and $\sigma(t) = p'$. Let $X := \{x_i | 0 \leq i < n\}$ and $Y := \{y_i | i \in I\}$. Our aim is to define a new substitution σ' in such a way that $\forall_{x \in V}(\sigma(x), \sigma'(x)) \in R$. We start with the following partial definition:

$$\sigma'(x) := \begin{cases} q_i & \text{if } x = x_i \ \ (0 \leq i < n) \\ \sigma(x) & \text{if } x \in V \setminus (X \cup Y) \end{cases}$$

Hitherto, we already have that $\forall_{x \in V \setminus Y}(\sigma(x), \sigma'(x)) \in R$. It remains to complete the definition of σ' for variables in Y in an appropriate way. Take a premise of which σ' is defined on all variables in the source (such a premise should exist due to our well-foundedness assumption about premises, see Definition 6). Suppose that one such a premise is $t_i \xrightarrow{l_i}_{r_i} y_i$, for some $i \in I$. For all $x \in vars(t_i)$, σ' is already defined in appropriate way, thus $(\sigma(x), \sigma'(x)) \in R$. It follows from Lemma 1 that $(\sigma(t_i), \sigma'(t_i)) \in R$. Transition $\sigma(t_i) \xrightarrow{l_i}_{r_i} \sigma(y_i)$ has a proof of depth $n-1$ or less and since $(\sigma(t_i), \sigma'(t_i)) \in R$, it follows from the induction hypothesis that there exists a p'_i such that $\sigma'(t_i) \xrightarrow{l}_r p'_i$ and $(\sigma(y_i), p'_i) \in R$. We define $\sigma'(y_i) := p'_i$ and hence it holds that $(\sigma(y_i), \sigma'(y_i)) \in R$. This way, we complete the proof for $\sigma'(t_i \xrightarrow{l_i}_{r_i} y_i)$. Defining σ' for each y_i inductively, in this manner, concludes the proof since rule **(d)** together with σ' gives us a proof for $q \xrightarrow{l}_r \sigma'(t)$ and according to the construction of σ' (which preserves the property $\forall_{x \in V}$, $(\sigma(x), \sigma'(x)) \in R$), it follows from Lemma 2 that $(\sigma(t), \sigma'(t)) \in R$.

3. It remains to prove the theorem for the case where $p = f(p_0, \ldots, p_{n-1})$ and $q = f(q_0, \ldots, q_k, \ldots, q_j, \ldots, q_{n-1})$ for $f \in COMM$ and some $0 \leq j < k < n$ such that for all $0 \leq i < n$, $(p_i, q_i) \in R$.

   We prove this by an induction on the depth of the proof for $p \xrightarrow{l}_r p'$. For the induction basis, suppose that p can make a transition with a proof of depth 1. Then, this transition is due to an f-defining rule of the following form:

$$\textbf{(d)}\frac{}{f(x_0, \ldots, x_{n-1}) \xrightarrow{l}_r t}$$

   and a substitution σ such that $\sigma(x_i) = p_i$ and $\sigma(t) = p'$. According to Definition 9, another rule exists in the transition system specification of the following form:

$$\textbf{(d')}\frac{H}{f(x'_0, \ldots, x'_{n-1}) \xrightarrow{l}_r t'}$$

   and a mapping $\hbar$ such that

   - $\hbar(x'_i) = x_i$ for $0 \leq i < n$ and $i \neq j$ and $i \neq k$
   - $\hbar(x'_j) = x_k$ and $\hbar(x'_k) = x_j$
   - $\hbar(t') \sim_{cc} t$
   - $\forall_{h \in H}\ \hbar(h) \in \{f(x_0, \ldots, x_{n-1}) \xrightarrow{l}_r t\}$

   We define a new substitution σ' as follows:

$$\sigma'(x) := \begin{cases} q_i & \text{if } x = x'_i \wedge 0 \leq i < n \wedge i \neq j \wedge i \neq k \\ q_j & \text{if } x = x'_k \\ q_k & \text{if } x = x'_j \\ \sigma(\hbar(x)) & \text{otherwise} \end{cases}$$

It immediately follows from the above definition that for all $x \in V$, $((\sigma \circ \hbar)(x), \sigma'(x)) \in R$ (where $\circ$ denotes composition of mappings). It also follows from the definition of σ' that $\sigma'(f(x'_0, \ldots x'_{n-1})) = q$. The last constraint on comm-tyft format implies that H is such that $\forall_{h \in H} \hbar(h) \in \{p \xrightarrow{l} t\}$. Otherwise said, H is either the empty set or $H = \{\hbar^{-1}(f(x_0, \ldots, x_{n-1}) \xrightarrow{l}_r t)\}$. If H is empty, then we already have a proof for $q \xrightarrow{l}_r \sigma'(t')$ using deduction rule **(d')** and substitution σ'. Also, if $H = \{\hbar^{-1}(f(x_0, \ldots, x_{n-1}) \xrightarrow{l}_r t)\}$ the proof for $q \xrightarrow{l}_r \sigma'(t')$ is complete since we already have a proof for $\sigma'(\hbar^{-1}(f(x_0, \ldots, x_{n-1}) \xrightarrow{l}_r t))$, using rule **(d)** and substitution $\sigma' \circ \hbar^{-1}$. Since $\hbar(t') \sim_{cc} t$ (thus, $\hbar^{-1}(t) \sim_{cc} t'$), and for all $x \in V$, $(\sigma \circ \hbar)(x) \sim_{cc} \sigma'(x)$, it follows from Lemma 1 that $\sigma(t) \sim_{cc} ((\sigma' \circ \hbar) \circ \hbar^{-1})(t')$. Hence, we have $(\sigma(t), \sigma'(t')) \in R$ (both $\sigma(t)$ and $\sigma(t')$ are closed terms and R is $\sim_{cc}$ restricted to closed terms). This concludes the induction basis.

For the induction step, suppose that the theorem holds for all transitions with proofs of depth less than n. Then, consider a transition $p \xrightarrow{l}_r p'$ with a proof of depth n. This transition should be due to an f-defining rule **(d)** of the following form:

$$\textbf{(d)} \frac{\{t_i \xrightarrow{l_i}_{r_i} y_i | i \in I\}}{f(x_0, \ldots, x_{n-1}) \xrightarrow{l}_r t}$$

and a substitution σ such that $\sigma(x_i) = p_i$ and $\sigma(t) = p'$. We refer to the set of premises of **(d)** as H and its conclusion as c. According to the Definition 9, there exists another rule **(d')** in the transition system specification of the following form:

$$\textbf{(d')} \frac{\{t'_j \xrightarrow{l'_j}_{r'_j} y'_j | j \in J\}}{f(x'_0, \ldots, x'_{n-1}) \xrightarrow{l}_r t'}$$

and a mapping $\hbar$ of variables such that

- $\hbar(x'_i) = x_i$ for $0 \leq i < n$ and $i \neq j$ and $i \neq k$
- $\hbar(x'_j) = x_k$ and $\hbar(x'_k) = x_j$
- $\hbar(t') \sim_{cc} t$
- $\forall_{j \in J}\ \hbar(t'_j \xrightarrow{l'_j}_{r'_j} y'_j) \in [H]_{cc} \cup \{c\}$

Let $X' = \{x'_i | 0 \leq i < n\}$ and $Y' = \{y'_j | j \in J\}$ be the set of variables in the source of the conclusion and the targets of the premises of deduction rule **(d)**, respectively. We aim at defining a new substitution σ' such that for all variables $x \in V$, $((\sigma \circ \hbar)(x), \sigma'(x)) \in R$. Similar to the induction basis, we start with the following (partial) definition for σ':

$$\sigma'(x) := \begin{cases} q_i & \text{if } x = x'_i \wedge 0 \leq i < n \wedge i \neq j \wedge i \neq k \\ q_j & \text{if } x = x'_k \\ q_k & \text{if } x = x'_j \\ (\sigma \circ \hbar^{-1})(x) & \text{if } x \in V \setminus (X' \cup Y') \end{cases}$$

To this end, we have $\sigma'(f(x'_0, \ldots x'_{n-1})) = q$ and σ' is defined on all variables, except for those in Y' and it satisfies $\forall_{x \in V \setminus Y'}\ ((\sigma \circ \hbar)(x), \sigma'(x)) \in R$. For the variables in Y', we complete the definition of σ' by taking a premise of which σ' is defined on all variables in the source and defining its target (such a premise should exist due to our well-foundedness assumption about premises, see Definition 6). Suppose that one of such premises is $t'_j \xrightarrow{l'_j}_{r'_j} y'_j$, for some $j \in J$. It follows from the last requirement on $\hbar$ in Definition 8 that either $\hbar(t'_j \xrightarrow{l'_j}_{r'_j} y'_j) = f(x_0, \ldots, x_{n-1}) \xrightarrow{l}_r t$ meaning that $t'_j = \hbar^{-1}(f(x_0, \ldots, x_{n-1}))$, $y'_j = \hbar^{-1}(t)$,

or there exists an $i \in I$ such that $\hbar(t'_j \xrightarrow{l'_j}_{r'_j} y'_j) = t_i \xrightarrow{l_i}_{r_i} y_i$, which means that $t'_j \sim_{cc} \hbar^{-1}(t_i)$ and $y'_j = \hbar^{-1}(y_i)$.

In the former case, we have $\sigma'(t'_j) = (\sigma' \circ \hbar^{-1})(f(x_0, \ldots, x_n)) = \sigma'(f(x'_0, \ldots, x'_j, \ldots, x'_k, \ldots, x'_n))$ $= f(q_0, \ldots, q_{n-1})$. Since we know that $(p_i, q_i) \in R$ $(0 \leq i < n)$, and $f(p_0, \ldots, p_{n-1}) \xrightarrow{l}_r p'$ it follows from the congruence part of this proof that there exists a q' such that $\sigma'(t'_j) \xrightarrow{l}_r q'$ and $(p', q') \in R$. Then, we define $\sigma'(y'_j) := q'$ and we fulfill the requirement that $((\sigma \circ \hbar)(y'_j), \sigma'$ $(y'_j)) \in R$ (since $\sigma(y_i) = p'$ and $y_i = \hbar(y'_j)$).

In the latter case, since $\hbar^{-1}(t_i) = t'_j$ (thus $\hbar^{-1}(t_i) \sim_{cc} t'_j$) and for all $x \in vars(t'_j)$, $(\sigma \circ \hbar)(x) \sim_{cc} \sigma'(x)$, it follows from Lemma 1 that $(\sigma \circ \hbar)(\hbar^{-1}(t_i)) \sim_{cc} \sigma'(t'_j)$ and thus $(\sigma(t_i), \sigma'(t'_j)) \in R$. Transition $\sigma(t_i) \xrightarrow{l_i}_{r_i} \sigma(y_i)$ has a proof of depth $n-1$ or less and since $(\sigma(t_i), \sigma'(t'_j)) \in R$, it follows from the induction hypothesis that there exists a p'_j such that $\sigma'(t'_j) \xrightarrow{l}_r p'_j$ and $(\sigma(y_i), p'_j) \in R$. We define $\sigma'(y'_j) := p'_j$ and hence it holds that $((\sigma \circ \hbar)(y'_j), \sigma'(y'_j)) \in R$. This way, we complete the proof for $\sigma'(t'_j \xrightarrow{l}_r y'_j)$. Defining σ' for each y'_j inductively, in this manner, concludes the proof since rule **(d')** together with σ' gives us a proof for $q \xrightarrow{l}_r \sigma'(t')$ and according to the construction of σ' (which preserves the property $\forall_{x \in V}$, $(\sigma \circ \hbar)(x) \sim_{cc} \sigma'(x)$ and since $\hbar^{-1}(t) \sim_{cc} t'$, it follows from Lemma 1 that $(\sigma \circ \hbar)(\hbar^{-1}(t)) \sim_{cc} (\sigma \circ \hbar)(t')$ and hence $(\sigma(t), \sigma'(t')) \in R$.

$\boxtimes$

In the remainder, we illustrate the idea behind our format by a few examples from the area of process algebra.

Example 2 (Commutativity of Parallel Composition: Standard Rules) Consider the following transition system specification for a parallel composition operator:

$$\textbf{(p0)}\frac{x_0 \xrightarrow{l}_{r_0} y_0}{x_0 \mid\mid x_1 \xrightarrow{l}_{r_0} y_0 \mid\mid x_1} \qquad \textbf{(p1)}\frac{x_1 \xrightarrow{l}_{r_0} y_1}{x_0 \mid\mid x_1 \xrightarrow{l}_{r_0} x_0 \mid\mid y_1}$$

$$\textbf{(p2)}\frac{x_0 \xrightarrow{l_0}_{r_0} y_0 \quad x_1 \xrightarrow{l_1}_{r_0} y_1}{x_0 \mid\mid x_1 \xrightarrow{l}_{r_0} y_0 \mid\mid y_1} \quad comm(l_0, l_1) = l$$

If the synchronization function *comm* is commutative, then the above TSS is in comm-tyft w.r.t. the singleton set $COMM = \{||\}$. This can be seen as follows. All deduction rules are obviously in tyft format. Deduction rule **(p1)** is the commutative mirror of **(p0)** (and vice versa) by using the mapping $\hbar$ defined by $\hbar(x_0) = x_1$, $\hbar(x_1) = x_0$, and $\hbar(y_1) = y_0$ and for the sake of symmetry $\hbar(y_0) = y_1$. Observe that $\hbar(x_0 \mid\mid y_1) = x_1 \mid\mid y_0 \sim_{cc} y_0 \mid\mid x_1$ and $\hbar(x_1 \xrightarrow{l}_{r_0} y_1) = x_0 \xrightarrow{l}_{r_0} y_0 \sim_{cc}$ $x_0 \xrightarrow{l}_{r_0} y_0$. Similarly, deduction rule **(p0)** is the commutative mirror of **(p1)**, using the inverse of $\hbar$ (which is $\hbar$ itself) as a variable mapping. Finally, deduction rule **(p2)** is the commutative mirror of itself by using the mapping $\hbar$ such that $\hbar(x_0) = x_1$, $\hbar(x_1) = x_0$, $\hbar(y_0) = y_1$ and $\hbar(y_1) = y_0$. It is not obvious that this mapping satisfies the requirements due to the fact that the premises have different labels. In this case this is not a problem as the function *comm* is commutative. In fact, the instance of this deduction rule where l_0 and l_1 are instantiated by labels a and b matches with an instance where these are instantiated by b and a respectively.

Example 3 (Commutativity of Parallel Composition: Non-Standard Rules) For the reason of finite axiomatization, parallel composition may be defined in terms of auxiliary operators, namely the left merge $\between$ and the communication merge $|$. Although the left merge is not a commutative operator, comm-tyft is still applicable for the following transition system specification of parallel composition, if one takes $COMM = \{||, |\}$.

$$\textbf{(p0)}\frac{x_0 \between x_1 \xrightarrow{l}_r y_0}{x_0 \between x_1 \xrightarrow{l}_r y_0} \qquad \textbf{(p1)}\frac{x_1 \between x_0 \xrightarrow{l}_r y_0}{x_0 \between x_1 \xrightarrow{l}_r y_0} \qquad \textbf{(p2)}\frac{x_0 \mid x_1 \xrightarrow{l}_r y_0}{x_0 \between x_1 \xrightarrow{l}_r y_0}$$

$$\textbf{(lp0)}\frac{x_0 \xrightarrow{l}_r y_0}{x_0 \between x_1 \xrightarrow{l}_r y_0 \between x_1} \qquad \textbf{(cp0)}\frac{x_0 \xrightarrow{l_0}_r y_0 \quad x_1 \xrightarrow{l_1}_r y_1}{x_0 \mid x_1 \xrightarrow{l}_r y_0 \between y_1} \quad comm(l_0, l_1) = l$$

Similar to the previous case, in the above specification **(p0)** and **(p1)** are commutative mirrors of each other. But for **(p2)** to be the commutative mirror of itself, we need commutativity of communication merge operator $\mid$. Hence, we have to check the rule **(cp0)**, as well. Rule **(cp0)** is the commutative mirror of itself, if *comm* is a commutative function. For the other remaining rule, since $\between$ is not a commutative operator, **(lp0)** only needs to be in tyft format which is indeed the case.

Example 4 (Commutativity of Nondeterministic Choice: Standard) Consider the following transition system specification for a nondeterministic choice operator:

$$\textbf{(c0)}\frac{x_0 \xrightarrow{l}_r y_0}{x_0 + x_1 \xrightarrow{l}_r y_0} \qquad \textbf{(c1)}\frac{x_1 \xrightarrow{l}_r y_1}{x_0 + x_1 \xrightarrow{l}_r y_1}$$

Then, following Theorem 1, we can derive that nondeterministic choice is a commutative operator: Both of **(c0)** and **(c1)** are in tyft format and each is the commutative mirror of the other under the mapping $\hbar(x_0) = x_1$, $\hbar(x_1) = x_0$, $\hbar(y_1) = y_0$ and $\hbar(y_0) = y_1$.

The following example illustrates that allowing a premise to be mapped to the conclusion of another rule can be useful.

Example 5 (Commutativity of Nondeterministic Choice: Non-Standard) As an alternative to the transition system specification from the previous example, nondeterministic choice can also be defined by means of the following transition system specification.

$$\textbf{(c0)}\frac{x_0 \xrightarrow{l}_r y_0}{x_0 + x_1 \xrightarrow{l}_r y_0} \qquad \textbf{(c1)}\frac{x_1 + x_0 \xrightarrow{l}_r y_0}{x_0 + x_1 \xrightarrow{l}_r y_0}$$

Deduction rule **(c0)** can not be matched by deduction rule **(c0)** itself. Deduction rule **(c0)** is matched by deduction rule **(c1)** using the mapping $\hbar(x_0) = x_1$, $\hbar(x_1) = x_0$, and $\hbar(y_0) = y_0$. Observe that if we would not have allowed the premise of **(c1)** to match the conclusion of **(c0)**, then it would have been impossible to match **(c0)**. Hence, Theorem 1 could not have been applied here, although nondeterministic choice as specified by this transition system specification is commutative.

Rule **(c1)** is the SOS characterization of commutativity and henceforth one may expect that by adding this rule for a specific operator to any TSS, one should be able to make it commutative. It can be easily shown, using comm-tyft format that this is indeed true. For any (binary) operator f, such a rule can play the role of a commutative mirror of all f-defining rules in the TSS by taking a mapping similar to the one above and hence making the TSS conform to the comm-tyft format.

The following example illustrates that we cannot relax the last constraint of the comm-tyft format to include CC-variants of the conclusion in the premises of the mirror rule.

Example 6 Suppose that we slightly relax the last constraint on the comm-tyft format to the following:

$$\forall_{h \in H} \hbar(h) \in [H \cup \{c\}]_{cc}$$

Then, the commutativity result is jeopardized. The following counter-example shows this fact:

$$\textbf{(a)}\frac{}{a \xrightarrow{a}_r b}$$

$$\textbf{(d)}\frac{x_0 \xrightarrow{l}_r y_0}{x_0 + x_1 \xrightarrow{l}_r y_0} \qquad \textbf{(d')}\frac{x_0 + x_1 \xrightarrow{l}_r y_0}{x_0 + x_1 \xrightarrow{l}_r y_0}$$

Suppose that the signature contains constants a and b and a binary function symbol $+$. The above transition system specification is in the relaxed version of comm-tyft format with respect to the set $COMM = \{+\}$; all rules are in tyft format and **(d')** is the commutative mirror of **(d)** and itself under $\hbar(x_0) = x_1$, $\hbar(x_1) = x_0$ and $\hbar(y_0) = y_0$. However, it does not hold that $a + b \leftrightarrow b + a$ since $a + b$ can make a transition due to **(d)** while $b + a$ deadlocks.

4 Possible Extensions

4.1 Tyxt Rules

In [8], tyft-tyxt format is introduced. So far, in the comm-tyft format, we have only allowed for tyft part. Rules in tyxt format are of the following form:

$$\frac{\{t_i \xrightarrow{l_i}_{r_i} y_i | i \in I\}}{x \xrightarrow{l}_r t}$$

Interesting enough, we can extend our comm-tyft format to allow for arbitrary rules in tyxt format. In [8], it has been already shown that tyxt format reduces to tyft format by copying the rule for all $f \in \Sigma$ and substituting variable x by $f(x_0, \ldots, x_{n-1})$ where n is the arity of operator f. So, tyxt rules do not harm the congruence property. They do not harm commutativity, either, since after such a copying procedure, each copied tyxt rule (for operators in the set $COMM$) is the commutative mirror of itself by taking the mapping $\hbar(x_i) = x_j$ and $\hbar(x_j) = x_i$ for any two arbitrary $0 \leq i < j < n$ and $\hbar(x) = x$ for all other variables.

4.2 Predicates and Negative Premises

Predicates are used to specify properties on process terms such as termination and divergence. By adding predicates to our set of formulae, the syntax of a deduction rule will be extended to the following shapes:

$$\frac{\{t_i \xrightarrow{l_i}_{r_i} y_i | i \in I\} \quad \{P_j(t_j) | j \in J\}}{f(x_0, \ldots, x_{n-1}) \xrightarrow{l}_r t} \qquad \frac{\{t_i \xrightarrow{l_i}_{r_i} y_i | i \in I\} \quad \{P_j(t_j) | j \in J\}}{P(f(x_0, \ldots, x_{n-1}))}$$

This rule format is called PATH (for *P*redicates *A*nd *T*yft-tyxt *H*ybrid format) and was introduced in [3]. There, it is also shown that the PATH format can be reduced to tyft format by introducing dummy transition relations for each predicate symbol, dummy fresh variables in the target of the predicate formulae in the premises and dummy fresh constants in the target of the predicate conclusions. The same trick works here, as well. Thus, to interpret our comm-tyft format in a setting with predicates, it suffices to consider predicates as sources of transitions. We can safely assume that the targets of such transitions satisfy our format (by taking the same dummy variables and constants in mirror rules).

In [7], it is shown that allowing for negative premises (of the form $t_i \nrightarrow$) to a transition system specification may endanger the well-definedness of the induced transition relation. Several approaches have been proposed to deal with this problem, of which [6] provides an overview. Stratification [7] is a measure defined on formulae which does not increase from conclusion to positive premises and decreases from conclusion to negative ones. If such a stratification exists, it has been shown in [7] that the induced transition relation is well-defined. Thus, by following

the same approach we may accommodate negative premises in our comm-tyft format, too. All in all, by allowing for tyxt rules, negative premises and predicates, we get the expressive power of PANTH (*P*redicates *A*nd *NT*yft-ntyxt *H*ybrid) format of [11].

5 Conclusion

In this paper, we defined a standard SOS format that guarantees a number of commutativity axioms to be sound with respect to strong bisimilarity. Our standard format for commutativity, called comm-tyft, calls for so-called commutative mirror rules for each deduction rule defining a commutative operator. These mirror rules account for arbitrary switching of arguments and thus their existence is a sufficient condition for commutativity. The proposed standard format can help as a theoretical background for part of a toolkit assisting specifiers in defining operational semantics and proving meta-properties about their defined languages. We have tried the same method for more complicated frequently occurring axioms, such as associativity. It turns out that the result is an abstract representation of the proof for those axioms and no such a straightforward format can be obtained as for the commutativity axiom.

References

[1] Luca Aceto, Bard Bloom, and Frits W. Vaandrager. Turning SOS rules into equations. *Information and Computation (I&C)*, 111:1–52, 1994.

[2] Luca Aceto, Wan J. Fokkink, and Chris Verhoef. Structural operational semantics. In Jan A. Bergstra, Alban Ponse, and Scott A. Smolka, editors, *Handbook of Process Algebra, Chapter 3*, pages 197–292. Elsevier Science, Dordrecht, The Netherlands, 2001.

[3] Jos C. M. Baeten and Chris Verhoef. A congruence theorem for structured operational semantics with predicates. In Eike Best, editor, *International Conference on Concurrency Theory (CONCUR'93)*, volume 715 of *Lecture Notes in Computer Science*, pages 477–492. Springer-Verlag, Berlin, Germany, 1993.

[4] Jos C.M. Baeten and Erik P. de Vink. Axiomatizing GSOS with termination. In Helmu Alt and Afonso Ferreira, editors, *Proceedings of 19th International Symposium on Theoretical Aspects of Computer Science (STACS'02)*, volume 2295 of *Lecture Notes in Computer Science*, pages 583–595. Springer-Verlag, Berlin, Germany, 2002.

[5] Wan J. Fokkink and Rob J. van Glabbeek. Ntyft/ntyxt rules reduce to ntree rules. *Information and Computation (I&C)*, 126(1):1–10, 1996.

[6] Rob van Glabbeek. The meaning of negative premises in transition system specifications II. *Journal of Logic and Algebraic Programming (JLAP)*, 60-61:229–258, 2004.

[7] Jan Friso Groote. Transition system specifications with negative premises. *Theoretical Computer Science (TCS)*, 118(2):263–299, 1993.

[8] Jan Friso Groote and Frits W. Vaandrager. Structured operational semantics and bisimulation as a congruence. *Information and Computation (I&C)*, 100:202–260, October 1992.

[9] David M.R. Park. Concurrency and automata on infinite sequences. In *Proceedings of 5th GI Conference*, volume 104 of *Lecture Notes in Coputer Science*, pages 167–183. Springer-Verlag, Berlin, Germany, 1981.

[10] Gordon D. Plotkin. A structural approach to operational semantics. *Journal of Logic and Algebraic Progamming (JLAP)*, 60:17–139, 2004. An old version of this aritcle appeared as Technical Report DAIMI FN-19 of Computer Science Department, Aarhus University, Denmark, 1981.

[11] Chris Verhoef. A general conservative extension theorem in process algebra. In *Proceedings of the IFIP TC2/WG2.1/WG2.2/WG2.3 Working Conference on Programming Concepts, Methods and Calculi*, 1994.

[12] Chris Verhoef. A congruence theorem for structured operational semantics with predicates and negative premises. *Nordic Journal of Computing*, 2(2):274–302, 1995.