Language Preorder as a Precongruence*

Wan Fokkink
University of Wales Swansea
Department of Computer Science
Singleton Park, Swansea SA2 8PP, Wales
w.j.fokkink@swan.ac.uk

Abstract

Groote and Vaandrager introduced the tyft format, which is a congruence format for strong bisimulation equivalence. This article proposes additional syntactic requirements on the tyft format, extended with predicates, to obtain a precongruence format for language preorder.

1 Introduction

Structural operational semantics [20] is a popular method to provide formal languages, process algebras, and specification languages with an interpretation. It is based on the use of transition systems. Given a set of states, the transitions between these states are obtained inductively from a transition system specification (TSS), which consists of transition rules. Validity of the (positive) premises of such a rule, under a certain substitution, implies validity of the conclusion of this rule under the same substitution. This article considers transition systems in which states are the closed terms generated by a single-sorted first-order signature, and transitions are supplied with labels.

Labelled transition systems can be distinguished by a wide range of behavioural equivalences [12], one of the coarsest of which is language equivalence. Two processes are language equivalent if they can terminate successfully with exactly the same sequences of actions. Language equivalence underlies the algebra of regular expressions initiated by Kleene [16]; see for instance [10, 8].

In general, the language equivalence induced by a TSS is not a congruence, i.e., the equivalence class of a term $f(p_1, ..., p_n)$ need not be determined by the equivalence classes of its arguments $p_1, ..., p_n$. Congruence is an important property, for instance, to fit the equivalence into an axiomatic framework. Congruence proofs in operational semantics tend to be long and technical, and are therefore often omitted. Congruence formats have been developed for a number of behavioural equivalences, firstly to avoid repetitive congruence proofs, secondly to explore the limits of sensible TSS definitions. Groote and Vaandrager [15] introduced a congruence format for bisimulation equivalence called tyft/tyxt, supplied with a well-foundedness criterion. Fokkink and van Glabbeek [9] showed that this well-foundedness criterion can be

*Sponsored in part by a grant from The Nuffield Foundation.

omitted. Baeten and Verhoef [3] extended tyft/tyxt with predicates, to obtain the path format.

This article introduces a syntactic format for TSSs, and it is shown that the language equivalence induced by a TSS in this format is always a congruence. The congruence format for language equivalence, called L cool, consists of the path format together with a number of additional syntactic requirements. We distinguish between so-called 'tame' and 'wild' arguments of function symbols, and put restrictions on occurrences of 'dangerous' variables in transition rules. A number of counter-examples is given to show that all the syntactic restrictions of the L cool format are essential. Furthermore, it is explained how the L cool requirements can be verified efficiently by finding a suitable tame/wild labelling. We focus on language *preorder* (instead of equivalence), because this yields a more general precongruence result, and simplifies proofs.

Regular expressions are constructed from atomic constants, alternative composition, sequential composition, and the Kleene star. The axiomatizability of regular expressions modulo language equivalence has been investigated in several articles. For example, Salomaa [21] and Kozen [17] presented complete axiomatizations. The fact that its operators preserve language equivalence is folklore; see for example [21, Theo. 1] and [1, Prop. 3.2]. We show as an example that the transition rules for regular expressions are L cool, so that the congruence property follows automatically. Furthermore, we show that the transition rules for ACP [5] and for recursion [11] are L cool.

In contrast with positive premises, negative premises of a transition rule have to be invalid in order to imply the conclusion of the rule, under a certain substitution. The path format for bisimulation equivalence has been extended with negative premises; see [14, 22, 7]. However, negative premises do not combine well with language equivalence. For instance, the transition rules for the priority operator [2], which include negative premises, do not preserve language equivalence. We show that a general congruence format for language equivalence cannot include negative premises.

Bloom [6] suggested congruence formats RBB cool and RWB cool for rooted branching and weak equivalence, respectively. Van Glabbeek [13] sketched congruence formats for ready simulation, ready trace, failure trace, and trace equivalence. The L cool format is incomparable with each of these formats.

2 Preliminaries

2.1 Signature

Definition 2.1 *A* (single-sorted) signature Σ *consists of*

- *an infinite set* $\mathcal{V}$ *of variables* $x, y, z, ...;$
- *a set* $\mathcal{F}$ *of function symbols* $f, g, h, ...,$ *disjoint from* $\mathcal{V}$*, where each function symbol* f *has an arity* $ar(f)$*.*

A function symbol of arity zero is called a constant*.*

Definition 2.2 *The collection* $\mathbb{T}(\Sigma)$ *of* (open) terms $r, s, t, u, v, w, ...$ *over* Σ *is defined as the least set that satisfies:*

- *each variable from* $\mathcal{V}$ *is in* $\mathbb{T}(\Sigma)$,

- *if* $t_1, ..., t_{ar(f)} \in \mathbb{T}(\Sigma)$, *then* $f(t_1, ..., t_{ar(f)}) \in \mathbb{T}(\Sigma)$.

$\mathrm{var}(t)$ *denotes the collection of variables in term* t. *A term is* closed *if it does not contain any variables. The set of closed terms is denoted by* $\mathcal{T}(\Sigma)$.

2.2 Transition System Specifications

We assume a signature Σ together with a set of labels.

Definition 2.3 *For each label* ℓ, $\xrightarrow{\ell}$ *denotes a binary relation On terms, and* $\xrightarrow{\ell}\surd$ *denotes a unary predicate on terms.*

For $t, t' \in \mathbb{T}(\Sigma)$, *expressions* $t \xrightarrow{\ell} t'$ *and* $t \xrightarrow{\ell} \surd$ *are called* transitions. *A transition is* closed *if it involves closed terms only.*

Intuitively, $t \xrightarrow{\ell} t'$ expresses that term t can evolve into term t' by the execution of ℓ, while $t \xrightarrow{\ell} \surd$ expresses that term t can terminate successfully by the execution of ℓ.

Definition 2.4 *A* (transition) rule ρ *is an expression of the form* H/c, *with* H *a collection of transitions, called the* premises *(or* hypotheses*) of* ρ, *and* c *a transition, called the* conclusion *of* ρ. *The collection of variables in* ρ *is denoted by* $\mathrm{var}(\rho)$. *The left- and right-hand side of the conclusion of* ρ *are called the* source *and the* target *of* ρ, *respectively.*

A transition system specification *(TSS) is a collection of transition rules.*

A *substitution* is a mapping $\sigma : \mathcal{V} \to \mathbb{T}(\Sigma)$. Substitutions extend to mappings from terms to terms in the standard way. Substitutions apply to transitions and rules as expected. A transition rule can be derived from substitution instances of transition rules in a TSS as follows.

Definition 2.5 *A* proof *from a TSS* T *of a transition rule* H/c *consists of an upwardly branching tree in which all upward paths are finite, where the nodes of the tree are labelled by transitions such that:*

- *the root has label* c;
- *if some node has label* d, *and* G *is the set of labels of nodes directly above this node, then*
  - *either* G/d *is a substitution instance of a rule in* T;
  - *or* $G = \emptyset$ *and* $d \in H$.

We write $T \vdash H/c$.

The transition relation induced by a TSS consists of all the closed transitions that are provable from the TSS. Each proof from a TSS of a closed transition can be transformed into a proof in which all the labels are closed transitions, simply by substituting arbitrary closed terms for the variables in the labels of the original proof (cf. [15, Lem. 3.3]).

2.3 Language Preorder

We assume a transition relation.

Definition 2.6 *A sequence of labels $\ell_1 \cdots \ell_n$ is a* terminating trace *of a closed term t iff $t \xrightarrow{\ell_1} t_1 \xrightarrow{\ell_2} \cdots \xrightarrow{\ell_{n-1}} t_{n-1} \xrightarrow{\ell_n} \surd$. The set of terminating traces of a closed term t is denoted by $L(t)$. For closed terms t, s we write $t \sqsubseteq_L s$ iff $L(t) \subseteq L(s)$.*

Definition 2.7 *The preorder $\sqsubseteq_L$ is a* precongruence *if $t_i \sqsubseteq_L s_i$ for $i = 1, ..., ar(f)$ implies $f(t_1, ..., t_{ar(f)}) \sqsubseteq_L f(s_1, ..., s_{ar(f)})$.*

In general, the language preorder induced by a TSS is *not* a precongruence, as is shown by several examples in Section 3.3.

Language equivalence $\simeq_L$ is the kernel of language preorder: $t \simeq_L s$ iff $t \sqsubseteq_L s$ and $s \sqsubseteq_L t$. It is easy to see that if $\sqsubseteq_L$ is a precongruence, then its kernel is a congruence, in the sense that $t_i \simeq_L s_i$ for $i = 1, ..., ar(f)$ implies $f(t_1, ..., t_{ar(f)}) \simeq_L f(s_1, ..., s_{ar(f)})$.

2.4 Path Rules

Groote and Vaandrager [15] introduced a congruence format for strong bisimulation equivalence [19], called the *tyft/tyxt* format. We follow the approach of Baeten and Verhoef [3], who extended the tyft/tyxt format with predicates such as $\xrightarrow{a} \surd$, to obtain the *path* format. We present the definition of the path format with the extra restriction that the source is not allowed to be a single variable (i.e., we only consider the tyft component of the path format.)

Definition 2.8 *A transition rule is a* path *rule if it is of the form*

$$\frac{\{s_i \xrightarrow{a_i} \surd \mid i \in I\} \cup \{t_j \xrightarrow{b_j} y_j \mid j \in J\}}{f(x_1, ..., x_{ar(f)}) \xrightarrow{c} \surd}$$

or

$$\frac{\{s_i \xrightarrow{a_i} \surd \mid i \in I\} \cup \{t_j \xrightarrow{b_j} y_j \mid j \in J\}}{f(x_1, ..., x_{ar(f)}) \xrightarrow{c} t}$$

where the x_k and the y_j are distinct variables.

A TSS is said to be in path *format if it consists of path rules only.*

The syntactic restrictions of path are essential for congruence formats, which can be seen for instance from the convincing examples that are provided in [15]. In the next section we provide additional requirements, to obtain a precongruence format for language preorder.

The following notion for transition rules originates from [15].

Definition 2.9 *The dependency graph of a set C of transitions is a directed graph, with the collection of variables $\mathcal{V}$ as vertices, and with as edges*

$\{\langle x, y\rangle \mid x$ *and* y *occur in the left- and right-hand side of some* $c \in C$*, respectively*$\}$.

3 A Precongruence Format for Language Preorder

3.1 The L Cool Format

Some of the terminology that is introduced in this section is inspired by [6]. We assume a signature Σ, together with a labelling Λ which labels the arguments of function symbols to be either *tame* or *wild*.

Definition 3.1 *The variables in a path rule ρ that occur in a wild argument of the source, or as the right-hand side of a premise, are called* dangerous *for ρ.*

A context is said to be w-nested if the context symbol occurs inside a nested string of wild arguments.

Definition 3.2 *The collection of* w-nested *contexts is defined inductively by:*

1. *$[]$ is w-nested;*
2. *if $C[]$ is w-nested, and argument i of function symbol f is wild, then*

$$f(t_1, ..., t_{i-1}, C[], t_{i+1}, ..., t_{ar(f)})$$

is w-nested.

Definition 3.3 *A path rule ρ is* L cool*, with respect to a tame/wild labelling of arguments of function symbols, if it satisfies the following syntactic restrictions.*

- *Each dangerous variable for ρ occurs exactly once either as the left-hand side of a premise or at a w-nested position in the target.*
- *The dependency graph (see Definition 2.9) of the premises of ρ does not contain an infinite forward chain of edges.*

A TSS is L cool *if there exists a tame/wild labelling of arguments of function symbols such that all the transition rules in T are L cool.*

Theorem 3.4 *If a TSS is L cool, then the language preorder that it induces is a precongruence.*

The intuition behind the L cool format is as follows. Assume a premise $t \xrightarrow{a} y$ of a transition rule ρ in an L cool TSS T, and let $\sigma(t) \sqsubseteq_L \sigma'(t)$. If $T \vdash \sigma(t) \xrightarrow{a} \sigma(y)$, so that the premise $t \xrightarrow{a} y$ can be satisfied, then we want that $T \vdash \sigma'(t) \xrightarrow{a} t'$ for some closed term t'. Since $\sigma(t) \sqsubseteq_L \sigma'(t)$, this follows if $L(\sigma(y)) \neq \emptyset$. Therefore, we test for one terminating trace of $\sigma(y)$; we do not test for more than one such trace, because such a test on branching behaviour would not agree with language preorder. The test for $T \vdash \sigma(y) \xrightarrow{b} \surd$ or $T \vdash \sigma(y) \xrightarrow{b} \sigma(z)$ (as the start of a terminating trace of $\sigma(y)$) can be performed by a premise $y \xrightarrow{b} \surd$ or $y \xrightarrow{b} z$ of ρ, respectively. In the latter case we proceed to test for a terminating trace of $\sigma(z)$. Alternatively, the test for a terminating trace of $\sigma(y)$ can be deferred to another rule, by incorporating y in the target of ρ. Arguments of function symbols in the target that contain y as a subterm are marked 'wild', to ensure that the test for a terminating trace of $\sigma(y)$ is continued by some other rule. The non-existence of infinite forward paths in dependency graphs of premises ensures that this test will eventually terminate successfully.

The intuitions above are captured by the counter-examples in Section 3.3. A formal proof of Theorem 3.4 is presented in Section 3.4.

3.2 Construction of Tame/Wild Labels

Assuming that a TSS T consists of a finite number of rules, which each have finitely many premises, it is easy to verify whether the rules in T are path. Moreover, given a tame/wild labelling of arguments of function symbols, it is easy to determine the dangerous variables of each rule in T, and to check whether each rule satisfies the restrictions as imposed by the L cool format in Definition 3.3. The crux in determining whether T is L cool is to find a suitable tame/wild labelling of arguments of function symbols. Assuming that the collection $\mathcal{F}$ of function symbols is finite, there exists an efficient procedure to compute a tame/wild labelling Λ such that (T, Λ) is L cool if and only if there exists a labelling Λ' such that (T, Λ') is L cool.

> Procedure "Compute Wild Labels for $(\mathcal{F}, ar)$ and T"
>
> The red/green directed graph G consists of vertices $\langle f, k\rangle$ for $f \in \mathcal{F}$ and $1 \leq k \leq ar(f)$. There is an edge from $\langle f, k\rangle$ to $\langle g, l\rangle$ in G if and only if there is a transition rule in T with its conclusion of the form
>
> $$f(x_1, ..., x_{ar(f)}) \xrightarrow{a} C[g(t_1, ..., t_{l-1}, C'[x_k], t_{l+1}, ..., t_{ar(g)})].$$
>
> A vertex $\langle g, l\rangle$ is red if and only if there is a transition rule in T with its target of the form
>
> $$C[g(t_1, ..., t_{l-1}, C'[y], t_{l+1}, ..., t_{ar(g)})]$$
>
> whereby y is the right-hand side of a premise of this rule. All other vertices in G are coloured green.
>
> The procedure colours green vertices in G red as follows. If a vertex $\langle f, k\rangle$ is red, and there exists an edge in G from $\langle f, k\rangle$ to a green vertex $\langle g, l\rangle$, then $\langle g, l\rangle$ is coloured red.
>
> The procedure terminates if none of the green vertices can be coloured red anymore, at which it outputs the red/green directed graph.

Λ labels an argument k of a function symbol f 'wild' if and only if the vertex $\langle f, k\rangle$ in the output graph of the procedure above is red.

3.3 Counter-Examples

We give a string of examples of TSSs in the path format (see Definition 2.8), to show that all syntactic restrictions of the L cool format are essential. In the first three counter-examples, a and b are constants, f is a function symbol of arity one with a tame argument, and the set of labels of transitions is $\{a, d\}$.

Negative premises of the form $t \stackrel{a}{\not\rightarrow}$ in a transition rule express that the conclusion of the rule only holds if t can*not* do any a-transitions. See [14, 22, 7] how to give meaning to TSSs with negative premises, and how the congruence format for bisimulation equivalence extends to such a setting. It is well-known that it is hazardous to combine negative premises and language preorder. For example, the priority operator [2], which is defined by means of negative premises, does not preserve language preorder. We give a simpler counter-example, showing that one cannot hope to extend the L cool format with negative premises.

Example 3.5 *Consider the TSS*

$$a \xrightarrow{a} a \qquad\qquad \frac{x \overset{a}{\not\rightarrow}}{f(x) \xrightarrow{d} \surd}$$

Then $b \sqsubseteq_L a$, *because* $L(b) = L(a) = \emptyset$. *However,* $f(b) \not\sqsubseteq_L f(a)$, *because* $L(f(b)) = \{d\}$ *and* $L(f(a)) = \emptyset$.

The following counter-example shows that the L cool format cannot allow an infinite forward chain of edges in the dependency graph of the premises of a transition rule.

Example 3.6 *Consider the TSS*

$$a \xrightarrow{a} a \qquad\qquad \frac{x_i \xrightarrow{a} x_{i+1} \quad i = 0, 1, 2, ...}{f(x_0) \xrightarrow{d} \surd}$$

Note that the edges $\langle x_i, x_{i+1} \rangle$ *for* $i = 0, 1, 2, ...$ *in the dependency graph of the premises of the second transition rule form an infinite forward chain.*

$a \sqsubseteq_L b$, *because* $L(a) = L(b) = \emptyset$. *However,* $f(a) \not\sqsubseteq_L f(b)$, *because* $L(f(a)) = \{d\}$ *and* $L(f(b)) = \emptyset$.

The following counter-example shows that the L cool format must enforce that each dangerous variable occurs as the left-hand side of a premise or in the target.

Example 3.7 *Consider the TSS*

$$a \xrightarrow{a} a \qquad\qquad \frac{x \xrightarrow{a} y}{f(x) \xrightarrow{d} \surd}$$

Note that in the second transition rule, the dangerous variable y *does not occur as the left-hand side of the premise, nor in the target.*

$a \sqsubseteq_L b$, *because* $L(a) = L(b) = \emptyset$. *However,* $f(a) \not\sqsubseteq_L f(b)$, *because* $L(f(a)) = \{d\}$ *and* $L(f(b)) = \emptyset$.

$$\ell \xrightarrow{\ell} \surd$$

$$\frac{x_1 \xrightarrow{\ell} \surd}{x_1 + x_2 \xrightarrow{\ell} \surd} \qquad \frac{x_1 \xrightarrow{\ell} y}{x_1 + x_2 \xrightarrow{\ell} y} \qquad \frac{x_2 \xrightarrow{\ell} \surd}{x_1 + x_2 \xrightarrow{\ell} \surd} \qquad \frac{x_2 \xrightarrow{\ell} y}{x_1 + x_2 \xrightarrow{\ell} y}$$

$$\frac{x_1 \xrightarrow{\ell} \surd}{x_1 \cdot x_2 \xrightarrow{\ell} x_2} \qquad \frac{x_1 \xrightarrow{\ell} y}{x_1 \cdot x_2 \xrightarrow{\ell} y \cdot x_2}$$

Table 1: Transition Rules for BPA

The remaining examples assume basic process algebra (BPA) [4], which is built from a collection of constants, called atomic actions, together with two function

symbols of arity two: the alternative composition $s+t$ executes either s or t, and the sequential composition $s \cdot t$ executes first s and then t. The atomic actions are also used as labels of transitions. These intuitions are made precise in the operational semantics of BPA, which is presented in Table 1, whereby the ℓ ranges over the set of atomic actions. Note that its transition rules are all in the path format (see Definition 2.8). The procedure in Section 3.2 calculates the following tame/wild labelling: the first argument of sequential composition is wild, and both arguments of alternative composition and the second argument of sequential composition are tame. The TSS in Table 1 is L cool with respect to this tame/wild labelling:

- in the third and fifth rule, the dangerous variable y is the target, and does not occur in the left-hand side of the premise;
- in the sixth and seventh rule, the dangerous variable x_1 is the left-hand side of the premise, and does not occur in target;
- in the seventh rule, the dangerous variable y occurs in a wild argument of the target, and not in the left-hand side of the premise.

We assume three constants a, b, and c, a function symbol f of arity one with a tame argument, and a set of labels $\{a, b, c, d\}$. It is easy to see that $a \cdot (b+c) \simeq_L (a \cdot b)+(a \cdot c)$, because in BPA both terms have terminating traces ab and ac.

The following counter-example shows that the L cool format cannot allow a dangerous variable to be the left-hand side of two premises.

Example 3.8 *Suppose that we extend BPA with the transition rule*

$$\frac{x \xrightarrow{a} y \quad y \xrightarrow{b} \surd \quad y \xrightarrow{c} \surd}{f(x) \xrightarrow{d} \surd}$$

Note that the dangerous variable y is the left-hand side of two premises.

$f(a \cdot (b+c)) \not\simeq_L f((a \cdot b)+(a \cdot c))$, because $L(f(a \cdot (b+c))) = \{d\}$ and $L(f((a \cdot b)+(a \cdot c))) = \emptyset$.

The following counter-example shows that the L cool format cannot allow multiple occurrences of the same dangerous variable in the target.

Example 3.9 *Let g be a function symbol of arity two with both arguments wild. Suppose that we extend BPA with two transition rules*

$$\frac{x \xrightarrow{a} y}{f(x) \xrightarrow{d} g(y, y)} \qquad \frac{x_1 \xrightarrow{b} \surd \quad x_2 \xrightarrow{c} \surd}{g(x_1, x_2) \xrightarrow{d} \surd}$$

Note that in the first transition rule, the dangerous variable y occurs twice in the target.

$f(a \cdot (b+c)) \not\simeq_L f((a \cdot b)+(a \cdot c))$, because $L(f(a \cdot (b+c))) = \{dd\}$ and $L(f((a \cdot b)+(a \cdot c))) = \emptyset$.

The following counter-example shows that the L cool format cannot allow a dangerous variable to occur both as the left-hand side of a premise and in the target.

Example 3.10 *Let g be a function symbol of arity one with a wild argument. Suppose that we extend BPA with the two transition rules*

$$\frac{x \xrightarrow{a} y \quad y \xrightarrow{b} \surd}{f(x) \xrightarrow{d} g(y)} \qquad \frac{x \xrightarrow{c} \surd}{g(x) \xrightarrow{d} \surd}$$

Note that in the first transition rule, the dangerous variable y occurs as the left-hand side of a premise and in the target.

$f(a\cdot(b+c)) \not\trianglerighteq_L f((a\cdot b)+(a\cdot c))$, because $L(f(a\cdot(b+c))) = \{dd\}$ and $L(f((a\cdot b)+(a\cdot c))) = \emptyset$.

The counter-examples above are based on violation of the L cool format by dangerous variables that occur as the right-hand side of a premise. Similar counter-examples exist using dangerous variables that occur in a wild argument of the source. The basic idea is to add to the examples above a function symbol h with a wild argument, together with a transition rule

$$\frac{x \xrightarrow{a} y}{f(x) \xrightarrow{d} h(y)}$$

The wild argument of h can be used to violate the L cool format. For instance, Example 3.8 takes the following form.

Example 3.11 *Suppose that we extend BPA with the transition rules*

$$\frac{x \xrightarrow{a} y}{f(x) \xrightarrow{d} h(y)} \qquad \frac{x \xrightarrow{b} \surd \quad x \xrightarrow{c} \surd}{h(x) \xrightarrow{d} \surd}$$

Note that in the second rule, the dangerous variable x is the left-hand side of two premises.

$f(a\cdot(b+c)) \not\trianglerighteq_L f((a\cdot b)+(a\cdot c))$, because $L(f(a\cdot(b+c))) = \{dd\}$ and $L(f((a\cdot b)+(a\cdot c))) = \emptyset$.

Finally, if in Example 3.11 the argument of h were defined to be tame, then the variable x in the second rule would no longer be dangerous, but the dangerous variable y in the first rule would occur in a tame argument of the target. This shows that the L cool format can only allow dangerous variables to occur at w-nested positions in the target.

3.4 Proof of the Precongruence Theorem

The following notions for transition rules originate from [15].

Definition 3.12 *A transition rule is* well-founded *if the dependency graph (see Definition 2.9) of its set of premises does not contain an infinite backward chain of edges.*

A variable in a transition rule is free *if it does not occur in the right-hand sides of the premises nor in the source of the rule.*

A transition rule is pure *if it is well-founded and does not contain free variables. A TSS is pure if all its rules are.*

Lemma 3.13 *For each L cool TSS T there exists a pure L cool TSS T' that induces the same transition relation.*

Proof. Let T' be the collection of pure L cool transition rules that are provable from T. Clearly, each closed transition that is provable from T' is provable from T. We show that the converse also holds.

Consider a closed transition $t \xrightarrow{a} t'$ in the transition relation induced by T. (The case $T \vdash t \xrightarrow{a} \surd$ can be handled in a similar fashion and is therefore omitted.) We show that $T' \vdash t \xrightarrow{a} t'$, using ordinal induction with respect to the length α of the shortest proof for $t \xrightarrow{a} t'$ from T.

Since $T \vdash t \xrightarrow{a} t'$, there exist a rule ρ in T and a substitution $\sigma : \mathcal{V} \to \mathcal{T}(\Sigma)$ such that the σ-instances of the premises of ρ can be derived from T by a proof shorter than α, and the σ-instance of the conclusion of ρ yields $t \xrightarrow{a} t'$. Let ρ be of the form

$$\frac{\{s_i \xrightarrow{a_i} \surd \mid i \in I\} \cup \{t_j \xrightarrow{b_j} y_j \mid j \in J\}}{f(x_1, ..., x_{ar(f)}) \xrightarrow{a} r}$$

By ordinal induction we may assume that $T' \vdash \sigma(s_i) \xrightarrow{a_i} \surd$ for $i \in I$ and $T' \vdash \sigma(t_j) \xrightarrow{b_j} \sigma(y_j)$ for $j \in J$.

We construct from ρ a rule ρ' in T' as follows. For each $j \in J$, if the dependency graph of the premises of ρ contains an infinite backward path from y_j, then we replace all occurrences of y_j in ρ by $\sigma(y_j)$. In this case the L cool format ensures that $t_j = y_{j'}$ for some $j' \in J$, and that the dependency graph of the premises of ρ contains an infinite backward path from $y_{j'}$, so then t_j is replaced by $\sigma(t_j)$. Moreover, we replace free variables z in ρ by $\sigma(z)$. The resulting rule ρ' is a substitution instance of ρ, so it is provable from T. We remove each premise $\sigma(t_j) \xrightarrow{b_j} \sigma(y_j)$ from ρ'. Since these transitions are provable from T, the resulting rule ρ'' is provable from T as well.

Since ρ is L cool, it is not hard to see that the same holds for ρ'' (owing to the fact that $x_1, ..., x_{ar(f)}$ have not been replaced by their σ-instances). Furthermore, ρ'' is well-founded and does not contain free variables, so it is pure. Hence, $\rho'' \in T'$. Since the σ-instances of the premises of ρ'' are provable from T', and since the σ-instance of the conclusion of ρ'' yields $t \xrightarrow{a} t'$, it follows that $T' \vdash t \xrightarrow{a} t'$. $\Box$

Now we are in a position to prove the main theorem.

Proof of Theorem 3.4. We prove that the language preorder induced by an L cool TSS T is a precongruence. According to Lemma 3.13 we can assume that T is pure. Let R be the least relation on $\mathcal{T}(\Sigma) \times \mathcal{P}(\mathcal{T}(\Sigma))$ that satisfies:

1. if for each $\tau \in L(t)$ there is an $s \in S$ such that $\tau \in L(s)$, then tRS;
2. if $t_k R S_k$ for $k = 1, ..., ar(f)$ and $|S_k| = 1$ for tame arguments k of f, then
$$f(t_1, ..., t_{ar(f)}) R \{f(s_1, ..., s_{ar(f)}) \mid s_k \in S_k \text{ for } k = 1, ..., ar(f)\};$$
3. if tRS_0 and $S_0 \subseteq S_1$, then tRS_1.

We want to show that R is fully defined by the first option in its definition, because this will imply that $\sqsubseteq_L$ is a precongruence.

We proceed to prove two statements. Note that statement B depends on the fact that the second argument of relation R is a *set* of terms.

A. If tRS and $T \vdash t \xrightarrow{a} \surd$, then there is an $s \in S$ such that $T \vdash s \xrightarrow{a} \surd$.

B. If tRS and $T \vdash t \xrightarrow{a} t'$, then $t'R\{s' \mid \exists s \in S\,(T \vdash s \xrightarrow{a} s')\}$.

We focus on proving statement B; statement A can be proved in a similar fashion.

It is easy to see that statement B holds if tRS satisfies the first option in the definition of R. Namely, if $\tau \in L(t')$ then $a\tau \in L(t)$ (because $T \vdash t \xrightarrow{a} t'$). So if tRS satisfies the first option in the definition of R, then $a\tau \in L(s)$ for some $s \in S$, or in other words, $\tau \in L(s')$ where $T \vdash s \xrightarrow{a} s'$. Hence, by the first option in the definition of R, $t'R\{s' \mid \exists s \in S\,(T \vdash s \xrightarrow{a} s')\}$.

Furthermore, it is easy to see that statement B holds if tRS satisfies the third option in the definition of R. Namely, suppose that tRS_0 and $S_0 \subseteq S_1$. We can assume (by induction) that we already proved statement B for tRS_0, that is, $t'R\{s' \mid \exists s \in S_0\,(T \vdash s \xrightarrow{a} s')\}$. So, since $S_0 \subseteq S_1$, the third option in the definition of R yields $t'R\{s' \mid \exists s \in S_1\,(T \vdash s \xrightarrow{a} s')\}$.

We focus on the case where tRS satisfies the second option in the definition of R. Summarizing, let t_kRS_k for $k = 1, ..., ar(f)$ and $|S_k| = 1$ for tame arguments k of f; we show that if $T \vdash f(t_1, ..., t_{ar(f)}) \xrightarrow{a} t'$, then

$$t'R\{s' \mid \exists s_1 \in S_1 \cdots \exists s_{ar(f)} \in S_{ar(f)}\,(T \vdash f(s_1, ..., s_{ar(f)}) \xrightarrow{a} s')\}.$$

Proof. We apply ordinal induction on the length α of a shortest proof of $f(t_1, ..., t_{ar(f)}) \xrightarrow{a} t'$ from T. In other words, it is assumed that statement A and statement B have already been proved for closed transitions $t \xrightarrow{a} \surd$ and $t \xrightarrow{a} t'$ that allow a proof from T that is shorter than α, respectively.

There exists a rule ρ in T and a substitution $\sigma : \mathcal{V} \to \mathcal{T}(\Sigma)$ such that the σ-instances of the premises of ρ are provable from T by a proof shorter than α, and the σ-instance of the conclusion of ρ yields $f(t_1, ..., t_{ar(f)}) \xrightarrow{a} t'$. Since ρ is path, it is of the form

$$\frac{\{u_i \xrightarrow{a_i} \surd \mid i \in I\} \cup \{v_j \xrightarrow{b_j} y_j \mid j \in J\}}{f(x_1, ..., x_{ar(f)}) \xrightarrow{a} r}$$

with $\sigma(x_k) = t_k$ for $k = 1, ..., ar(f)$ and $\sigma(r) = t'$.

Define a mapping $\Psi : \mathrm{var}(\rho) \to \mathcal{P}(\mathcal{T}(\Sigma))$ by:

- $\Psi(x_k) = S_k$ for $k = 1, ..., ar(f)$;
- for $j \in J$, if $\Psi(z)$ is defined for all $z \in \mathrm{var}(v_j)$, then

$$\Psi(y_j) = \{v' \mid \exists \psi\,(\psi(z) \in \Psi(z) \text{ for all } z \in \mathrm{var}(v_j), \text{ and } T \vdash \psi(v_j) \xrightarrow{b_j} v')\}.$$

Since ρ is pure, $\Psi(z)$ is thus defined for all $z \in \mathrm{var}(\rho)$, and since ρ is path, this definition is unambiguous. For terms t with $\mathrm{var}(t) \subseteq \mathrm{var}(\rho)$ we use the abbreviation

$$\Psi(t) \stackrel{\triangle}{=} \{\psi(t) \mid \psi(z) \in \Psi(z) \text{ for all } z \in \mathrm{var}(t)\}.$$

Lemma 3.14 *If $t \in \mathbb{T}(\Sigma)$ with $\mathrm{var}(t) \subseteq \mathrm{var}(\rho)$, and*

1. *$\sigma(z)R\Psi(z)$ for all $z \in \mathrm{var}(t)$; and*

2. if $|\Psi(z)| \neq 1$ for some $z \in \mathrm{var}(t)$, then z only occurs at w-nested positions in t;

then $\sigma(t) R \Psi(t)$.

Proof. We apply induction on the structure of t. If t is a variable, then $\sigma(t) R \Psi(t)$ follows immediately from the first requirement of the lemma.

Let $t = g(t_1, ..., t_{ar(g)})$. If $|\Psi(z)| \neq 1$ for some $z \in \mathrm{var}(t)$, then the second requirement of the lemma ensures that z only occurs at w-nested positions in the t_k. So induction yields $\sigma(t_k) R \Psi(t_k)$ for $k = 1, ..., ar(f)$. Furthermore, for tame arguments k of g, the second requirement of the lemma imposes that $|\Psi(t_k)| = 1$, because each variable in t_k occurs at a position in t that is not w-nested. Hence, by the second option in the definition of R we have $\sigma(t) R \Psi(t)$. □

Lemma 3.15 *$\sigma(z) R \Psi(z)$ for all $z \in \mathrm{var}(\rho)$.*

Proof. Since ρ is path and does not contain free variables, we can distinguish two cases for z.

1. $z = x_k$ for some $k \in \{1, ..., ar(f)\}$.

   $\sigma(x_k) = t_k$ and $\Psi(x_k) = S_k$, and by assumption $t_k R S_k$.

2. $z = y_j$ for some $j \in J$.

   Since ρ is well-founded, we can assume that we already proved $\sigma(z') R \Psi(z')$ for all $z' \in \mathrm{var}(v_j)$. Furthermore, if $|\Psi(z')| \neq 1$ for some $z' \in \mathrm{var}(v_j)$ then z' is a dangerous variable for ρ; then the L cool format enforces that $v_j = z'$, so that z' occurs at a w-nested position in v_j. Hence, Lemma 3.14 yields $\sigma(v_j) R \Psi(v_j)$.

   Since $\sigma(v_j) R \Psi(v_j)$, and there is a proof for $\sigma(v_j) \xrightarrow{b_j} \sigma(y_j)$ from T that is shorter than α, ordinal induction with respect to statement B and the definition of $\Psi(y_j)$ together yield $\sigma(y_j) R \Psi(y_j)$. □

Lemma 3.15 implies that $\sigma(z) R \Psi(z)$ for all $z \in \mathrm{var}(r)$. Furthermore, if $|\Psi(z)| \neq 1$ for some $z \in \mathrm{var}(r)$, then z is a dangerous variable for ρ, so the L cool format enforces that z only occurs at w-nested positions in r. Hence, since $\sigma(r) = t'$, Lemma 3.14 yields

$$t' R \Psi(r).$$

In view of the third option in the definition of R, it remains to prove that

$$\Psi(r) \subseteq \{s' \mid \exists s_1 \in S_1 \cdots \exists s_{ar(f)} \in S_{ar(f)} \, (T \vdash f(s_1, ..., s_{ar(f)}) \xrightarrow{a} s')\}.$$

In other words, for each mapping $\psi : \mathrm{var}(r) \to \mathcal{T}(\Sigma)$ with $\psi(z) \in \Psi(z)$ for $z \in \mathrm{var}(r)$, there should exist $s_k \in S_k$ for $k = 1, ..., ar(f)$ such that $T \vdash f(s_1, ..., s_{ar(f)}) \xrightarrow{a} \psi(r)$. We prove this by extending each such ψ to a mapping $\overline{\psi} : \mathrm{var}(\rho) \to \mathcal{T}(\Sigma)$, where $\overline{\psi}(z) \in \Psi(z)$ for $z \in \mathrm{var}(\rho)$, and the $\overline{\psi}$-instances of the premises of ρ are provable from T. (If r is a closed term, then we need to find one such $\overline{\psi}$.)

We define $\overline{\psi}$ as follows, whereby $\overline{\psi}(z) \in \Psi(z)$ for all $z \in \mathrm{var}(\rho)$.

- $\overline{\psi}(x_k) \in S_k$ for tame arguments k of f. (Since $|S_k| = 1$ for such k, this definition is unambiguous.)

- If z is a dangerous variable for ρ that occurs in r, then $\overline{\psi}(z) = \psi(z)$.

- Suppose that u_i is a dangerous variable for ρ, for some $i \in I$. According to Lemma 3.15, $\sigma(u_i)R\Psi(u_i)$. Since $\sigma(u_i) \xrightarrow{a_i} \surd$ can be derived from T by a proof that is shorter than α, by ordinal induction statement A yields that there is a $u \in \Psi(u_i)$ with $T \vdash u \xrightarrow{a_i} \surd$. Put $\overline{\psi}(u_i) = u$.

- Suppose that v_j is a dangerous variable for ρ, for some $j \in J$, Since the dependency graph of the premises of ρ does not contain an infinite forward path of edges, we may assume that $\overline{\psi}(y_j)$ has already been defined. Since $\overline{\psi}(y_j) \in \Psi(y_j)$, the definition of $\Psi(y_j)$ ensures that there exists a $v \in \Psi(v_j)$ with $T \vdash v \xrightarrow{b_j} \overline{\psi}(y_j)$. Put $\overline{\psi}(v_j) = v$.

Since ρ does not contain free variables, $\overline{\psi}(z)$ is defined for all $z \in \mathrm{var}(\rho)$. Moreover, the L cool format enforces that each dangerous variable for ρ occurs exactly once as the left-hand side of a premise of ρ or in r, so this definition is unambiguous. Since $\psi(x_k) \in \Psi(x_k) = S_k$ for (tame) arguments k of f, it follows that $\overline{\psi}(z) = \psi(z)$ for all $z \in \mathrm{var}(r)$.

The $\overline{\psi}$-instances of premises of ρ are all provable from T:

- If u_i or v_j is a dangerous variable for ρ, then $\overline{\psi}(u_i)$ or $\overline{\psi}(v_j)$ has been selected such that $T \vdash \overline{\psi}(u_i) \xrightarrow{a_i} \surd$ or $T \vdash \overline{\psi}(v_j) \xrightarrow{b_j} \overline{\psi}(y_j)$, respectively.

- If u_i is not a dangerous variable for ρ, then the L cool format enforces that u_i does not contain dangerous variables for ρ. Since $|\Psi(x_k)| = |S_k| = 1$ for tame arguments k of f, we have $|\Psi(u_i)| = 1$. So $\overline{\psi}(u_i) \in \Psi(u_i)$ implies $\Psi(u_i) = \{\overline{\psi}(u_i)\}$. According to Lemma 3.15 together with Lemma 3.14, $\sigma(u_i)R\Psi(u_i)$. The transition $\sigma(u_i) \xrightarrow{a_i} \surd$ can be derived from T by a proof shorter than α, so by ordinal induction statement A yields $T \vdash \overline{\psi}(u_i) \xrightarrow{a_i} \surd$.

- If v_j is not a dangerous variable for ρ, then the L cool format enforces that v_j does not contain dangerous variables for ρ. Since $|\Psi(x_k)| = |S_k| = 1$ for tame arguments k of f, we have $|\Psi(v_j)| = 1$. So $\overline{\psi}(v_j) \in \Psi(v_j)$ implies $\Psi(v_j) = \{\overline{\psi}(v_j)\}$. Since $\overline{\psi}(y_j) \in \Psi(y_j)$, the definition of $\Psi(y_j)$ ensures that $T \vdash \overline{\psi}(v_j) \xrightarrow{b_j} \overline{\psi}(y_j)$.

Since the $\overline{\psi}$-instances of the premises of ρ are all provable from T, the $\overline{\psi}$-instance of the conclusion of ρ is also provable from T. That is, $T \vdash f(s_1, ..., s_{ar(f)}) \xrightarrow{a} \psi(r)$ for certain $s_k \in S_k$ for $k = 1, ..., ar(f)$. This holds for each $\psi : \mathrm{var}(r) \to \mathcal{T}(\Sigma)$ with $\psi(z) \in \Psi(z)$ for $z \in \mathrm{var}(r)$, so

$$\Psi(r) \subseteq \{s' \mid \exists s_1 \in S_1 \cdots \exists s_{ar(f)} \in S_{ar(f)} \, (T \vdash f(s_1, ..., s_{ar(f)}) \xrightarrow{a} s')\}.$$

Since $t'R\Psi(r)$, the third option in the definition of R yields

$$t'R\{s' \mid \exists s_1 \in S_1 \cdots \exists s_{ar(f)} \in S_{ar(f)} \, (T \vdash f(s_1, ..., s_{ar(f)}) \xrightarrow{a} s')\}.$$

This finishes the proof of statement B. □

We apply statements A and B to prove statement C, which says that R is fully defined by the first option in its definition.

C. If tRS, then for each $\tau \in L(t)$ there is an $s \in S$ such that $\tau \in L(s)$.

Proof. We apply induction on the length of τ.

- Suppose that $\tau = a$, so $T \vdash t \xrightarrow{a} \surd$. Since tRS, statement A yields that there is an $s \in S$ such that $T \vdash s \xrightarrow{a} \surd$. So $a \in L(s)$.

- Suppose that $\tau = a\tau'$, so $T \vdash t \xrightarrow{a} t'$ with $\tau' \in L(t')$. Since tRS, statement B yields that $t'R\{s' \mid \exists s \in S\,(T \vdash s \xrightarrow{a} s')\}$. Since $\tau' \in L(t')$, and τ' has shorter length than τ, induction yields that there exists an $s \in S$ such that $T \vdash s \xrightarrow{a} s'$ and $\tau' \in L(s')$. This implies that $a\tau' \in L(s)$. $\Box$

Finally, statement C implies that $\sqsubseteq_L$ is a precongruence. Namely, suppose that $t_k \sqsubseteq_L s_k$ for $k = 1, ..., ar(f)$. By the first option in the definition of R, $t_k R\{s_k\}$ for $k = 1, ..., ar(f)$, so by the second option in the definition of R, $f(t_1, ..., t_{ar(f)})R\{f(s_1, ..., s_{ar(f)})\}$. Hence, by statement C, $f(t_1, ..., t_{ar(f)}) \sqsubseteq_L f(s_1, ..., s_{ar(f)})$. $\Box$

4 Applications

4.1 Regular Expressions

Kleene [16] introduced the binary Kleene star s^*t, which from an operational point of view repeatedly executes s, until it executes t, after which it terminates. Regular expressions [10, 8] are built from the binary Kleene star together with the operators in BPA as described in Section 3.3: a set of atomic actions, alternative composition, and sequential composition. Regular expressions can also include two special constants 0 and 1, which we do not take into account for the sake of simplicity; the 1 would require a reformulation of the transition rules for BPA in Table 1. The behaviour of the binary Kleene star is captured by the four transition rules in Table 2. Recall that if we take the first argument of sequential composition to be wild, and both arguments of alternative composition and the second argument of sequential composition to be tame, then the transition rules for BPA are L cool. Moreover, if we take both arguments of the binary Kleene star to be tame, then the four transition rules in Table 2 are also L cool. The path restrictions are easily checked. In the second rule, the dangerous variable y occurs in a wild argument of the target, and not in the left-hand side of the premise. In the fourth rule, the dangerous variable y is the target, and does not occur in the left-hand side of the premise.

$$\frac{x_1 \xrightarrow{\ell} \surd}{x_1{}^*x_2 \xrightarrow{\ell} x_1{}^*x_2} \qquad \frac{x_1 \xrightarrow{\ell} y}{x_1{}^*x_2 \xrightarrow{\ell} y \cdot (x_1{}^*x_2)} \qquad \frac{x_2 \xrightarrow{\ell} \surd}{x_1{}^*x_2 \xrightarrow{\ell} \surd} \qquad \frac{x_2 \xrightarrow{\ell} y}{x_1{}^*x_2 \xrightarrow{\ell} y}$$

Table 2: Transition Rules for the Binary Kleene Star

Corollary 4.1 *Language preorder is a precongruence with respect to BPA*.*

4.2 Communication

Let A represent a collection of atomic actions. The merge $s \| t$ [18] executes its two arguments in parallel. If s or t can execute an action a, then $s \| t$ can also execute

a. Moreover, there is a communication function $\gamma : A \times A \to A$, and if s and t can execute actions a and b, respectively, then $s \| t$ can execute $\gamma(a, b)$. The special constant δ does not display any behaviour, and the encapsulation operator $\partial_H(t)$ for collections $H \subseteq A$ obstructs actions of t that are in H, by renaming them into δ. The precise behaviour of the merge and encapsulation is described by their transition rules in Table 3. These transition rules are all L cool, if we take both arguments of the merge and the argument of encapsulation to be wild. The path restrictions are easily checked. For each transition rule, its variables are all dangerous, and they occur either as the left-hand side of a premise, or as the target, or in a wild argument of the target.

$$\frac{x_1 \stackrel{\ell}{\to} \surd}{x_1 \| x_2 \stackrel{\ell}{\to} x_2} \qquad \frac{x_1 \stackrel{\ell}{\to} y}{x_1 \| x_2 \stackrel{\ell}{\to} y \| x_2} \qquad \frac{x_2 \stackrel{\ell}{\to} \surd}{x_1 \| x_2 \stackrel{\ell}{\to} x_1} \qquad \frac{x_2 \stackrel{\ell}{\to} y}{x_1 \| x_2 \stackrel{\ell}{\to} x_1 \| y}$$

$$\frac{x_1 \stackrel{\ell}{\to} \surd \;\; x_2 \stackrel{\ell'}{\to} \surd}{x_1 \| x_2 \stackrel{\gamma(\ell,\ell')}{\to} \surd} \qquad \frac{x_1 \stackrel{\ell}{\to} \surd \;\; x_2 \stackrel{\ell'}{\to} y}{x_1 \| x_2 \stackrel{\gamma(\ell,\ell')}{\to} y} \qquad \frac{x_1 \stackrel{\ell}{\to} y \;\; x_2 \stackrel{\ell'}{\to} \surd}{x_1 \| x_2 \stackrel{\gamma(\ell,\ell')}{\to} y} \qquad \frac{x_1 \stackrel{\ell}{\to} y_1 \;\; x_2 \stackrel{\ell'}{\to} y_2}{x_1 \| x_2 \stackrel{\gamma(\ell,\ell')}{\to} y_1 \| y_2}$$

$$\frac{x \stackrel{\ell}{\to} \surd \quad \ell \notin H}{\partial_H(x) \stackrel{\ell}{\to} \surd} \qquad \frac{x \stackrel{\ell}{\to} y \quad \ell \notin H}{\partial_H(x) \stackrel{\ell}{\to} \partial_H(y)}$$

Table 3: Transition Rules for ACP

The algebra of communicating processes (ACP) [5] is obtained by adding δ, the merge and encapsulation to BPA. In fact, ACP also incorporates two auxiliary operators left and communication merge, which enable to axiomatize the merge. We have left these operators out, in order to keep the example simple. Their transition rules, however, are also L cool.

Corollary 4.2 *Language preorder is a precongruence with respect to ACP.*

4.3 Recursion

Given a signature Σ, a recursive specification E is a finite set of equations $\{X_i = t_i \mid i = 1, ..., n\}$, where the X_i are recursion variables, and the t_i are in $\mathbb{T}(\Sigma)$, with possible occurrences of recursion variables. Intuitively, the syntactic construct $\langle X|E\rangle$ denotes a solution of X with respect to E. The precise meaning of this construct is given by the transition rules for recursion in Table 4, which originate from [11]. The expression E in these transition rules represents a recursive specification, which contains an equation $X = t$. Furthermore, $\langle t|E\rangle$ denotes the term t with occurrences of recursion variables Y replaced by $\langle Y|E\rangle$. If we consider the expressions $\langle X|E\rangle$ to be constants, then the two transition rules in Table 4 are L cool. The path restrictions are easily checked. In the second rule, the dangerous variable y is the target, and does not occur in the left-hand side of the premise.

Corollary 4.3 *Language preorder is a precongruence with respect to ACP with recursion.*

$$\frac{\langle t|E\rangle \xrightarrow{\ell} \surd}{\langle X|E\rangle \xrightarrow{\ell} \surd} \qquad \frac{\langle t|E\rangle \xrightarrow{\ell} y}{\langle X|E\rangle \xrightarrow{\ell} y}$$

Table 4: Transition Rules for Recursion

References

[1] L. Aceto, W.J. Fokkink and A. Ingólfsdóttir. A menagerie of non-finitely based process semantics over BPA*. *Mathematical Structures in Computer Science*, 8(3):193–230, 1998.

[2] J.C.M. Baeten, J.A. Bergstra and J.W. Klop. Syntax and defining equations for an interrupt mechanism in process algebra. *Fundamenta Informaticae*, 9(2):127–167, 1986.

[3] J.C.M. Baeten and C. Verhoef. A congruence theorem for structured operational semantics with predicates. In E. Best, ed., *Proceedings 4th Conference on Concurrency Theory (CONCUR'93)*, Hildesheim, LNCS 715, pp. 477–492. Springer, 1993.

[4] J.C.M. Baeten and W.P. Weijland. *Process Algebra*. Cambridge Tracts in Theoretical Computer Science 18, Cambridge University Press, 1990.

[5] J.A. Bergstra and J.W. Klop. Process algebra for synchronous communication. *Information and Control*, 60(1/3):109–137, 1984.

[6] B. Bloom. Structural operational semantics for weak bisimulations. *Theoretical Computer Science*, 146(1/2):25–68, 1995.

[7] R.N. Bol and J.F. Groote. The meaning of negative premises in transition system specifications. *Journal of the ACM*, 43(5):863–914, 1996.

[8] J.H. Conway. *Regular Algebra and Finite Machines*. Chapman and Hall, 1971.

[9] W.J. Fokkink and R.J. van Glabbeek. Ntyft/ntyxt rules reduce to ntree rules, *Information and Computation*, 126(1):1–10, 1996.

[10] A. Ginzburg. *Algebraic Theory of Automata*. Academic Press, 1968.

[11] R.J. van Glabbeek. Bounded nondeterminism and the approximation induction principle in process algebra. In F.J. Brandeburg, G. Vidal-Naquet and M. Wirsing, eds., *Proceedings 4th Symposium on Theoretical Aspects of Computer Science (STACS'87)*, Passau, LNCS 247, pp. 336–347. Springer, 1987.

[12] R.J. van Glabbeek. The linear time – branching time spectrum. In J.C.M. Baeten and J.W. Klop, eds., *Proceedings 1st Conference on Concurrency Theory (CONCUR'90)*, Amsterdam, LNCS 458, pp. 278–297. Springer, 1990.

[13] R.J. van Glabbeek. Full abstraction in structural operational semantics. In M. Nivat, C. Rattray, T. Rus and G. Scollo, eds., *Proceedings 3rd Conference on*

Algebraic Methodology and Software Technology (AMAST'93), Enschede, *Workshops in Computing*, pp. 77–84. Springer, 1993.

[14] J.F. Groote. Transition system specifications with negative premises. *Theoretical Computer Science*, 118(2):263–299, 1993.

[15] J.F. Groote and F.W. Vaandrager. Structured operational semantics and bisimulation as a congruence. *Information and Computation*, 100(2):202–260, 1992.

[16] S.C. Kleene. Representation of events in nerve nets and finite automata. In *Automata Studies*, pages 3–41. Princeton University Press, 1956.

[17] D. Kozen. A completeness theorem for Kleene algebras and the algebra of regular events. *Information and Computation*, 110(2):366–390, 1994.

[18] R. Milner. *A Calculus of Communicating Systems*. LNCS 92, Springer, 1980.

[19] D.M.R. Park. Concurrency and automata on infinite sequences. In P. Deussen, ed., *Proceedings 5th GI Conference*, Karlsruhe, LNCS 104, pp. 167–183. Springer, 1981.

[20] G.D. Plotkin. A structural approach to operational semantics. Report DAIMI FN-19, Aarhus University, 1981.

[21] A. Salomaa. Two complete axiom systems for the algebra of regular events. *Journal of the ACM*, 13(1):158–169, 1966.

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