# Ntyft/ntyxt Rules Reduce to Ntree Rules

Wan Fokkink*
*Department of Computer Science*
*CWI*
*Amsterdam, The Netherlands*
wan@cwi.nl

Rob van Glabbeek†
*Computer Science Department*
*Stanford University*
*Stanford, CA 94305, USA*
rvg@cs.stanford.edu

Groote and Vaandrager introduced the *tyft/tyxt format* for Transition System Specifications (TSSs), and established that for each TSS in this format that is *well-founded*, the bisimulation equivalence it induces is a congruence. In this paper, we construct for each TSS in tyft/tyxt format an equivalent TSS that consists of *tree rules* only. As a corollary we can give an affirmative answer to an open question, namely whether the well-foundedness condition in the congruence theorem for tyft/tyxt can be dropped. These results extend to tyft/tyxt with negative premises and predicates.

## 1 Introduction

A current method to provide process algebras and specification languages with an operational semantics is based on the use of transition systems, advocated by Plotkin [15]. Given a set of states, the transitions between these states are obtained inductively from a Transition System Specification (TSS), which consists of transition rules. Such a rule, together with a number of transitions, may imply the validity of another transition.

We will consider a specific type of transition systems, in which states are the closed terms generated by a single-sorted signature, and transitions are supplied with labels. A great deal of the operational semantics of formal languages in Plotkin style that have been defined over the years, are within the scope of this format.

To distinguish such labelled transition systems, many different equivalences have been defined, the finest of which is the strong bisimulation equivalence of Park [14]. In general, this equivalence is not a congruence, i.e. the equivalence class of a term $f(p_1, ..., p_m)$ modulo strong bisimulation is not always determined by the equivalence classes of the terms $p_i$. However, congruence is an essential property, for instance, to fit the equivalence into an axiomatic framework.

Several formats have been developed which ensure that the bisimulation equivalence induced by a TSS in such a format is always a congruence. A first proposal was made by De Simone [16], which was generalized by Bloom, Istrail and Meyer [3] to the GSOS format. Next, Groote and Vaandrager [12] introduced the tyft/tyxt format, and proved a congruence theorem for TSSs in this format that satisfy a well-foundedness criterion.

*Current affiliation: Department of Philosophy, Utrecht University, Utrecht, The Netherlands.
†This work was supported by ONR under grant number N00014-92-J-1974.

Up to now, it has been an open question whether or not well-foundedness is an essential ingredient of this congruence theorem. The requirement popped up in the proof, but no counter-example was found to show that the theorem breaks down if well-foundedness were omitted from it. In this paper, we prove that the congruence theorem does hold for general TSSs in tyft/tyxt format, i.e. that the requirement of well-foundedness can be omitted.

In fact, we will establish a stronger result, namely that for each TSS in tyft/tyxt format, there is an equivalent TSS which consists of 'tree rules' only. A tree rule is a well-founded rule of the form

$$\frac{\{z_i \xrightarrow{a_i} y_i \mid i \in I\}}{f(x_1, ..., x_m) \xrightarrow{a} t}$$

where the $y_i$ and the $x_j$ are distinct variables and are the only variables that occur in the rule, the $z_i$ are variables, $f$ is a function symbol, and $t$ is any term. Using terminology from [12], we can say that a tree rule is a pure xyft rule. Since tree rules are well-founded, the reduction of tyft/tyxt rules to tree rules immediately implies that the congruence theorem concerning the tyft/tyxt format can do without well-foundedness.

A major advantage of the main theorem is that it facilitates reasoning about the tyft/tyxt format. Because often it is much easier to prove a theorem for TSSs in tree format than for TSSs in tyft/tyxt format. For example, this is the case with the congruence theorem itself. Another striking example consists of Theorems 8.6.6 and 8.9.1 in [12]. With our result at hand, the complicated proof of the second theorem can be skipped, because now the second theorem follows from the first one.

Furthermore, the removal of well-foundedness from the congruence theorem for tyft/tyxt increases the convenience of applying this theorem, since the user no longer has to recall and check the complicated well-foundedness criterion.

The main result of this paper was obtained independently by the authors in [9] and [5]. Our present proof improves the ones envisioned in [9] and given in [5]. It makes heavy use of a standard result from unification theory, which says that for each set of equations that is unifiable, there exists an idempotent most general unifier. In unification theory, this result is proved for finite sets of equations, and for substitutions that have a finite domain. However, we will need the result in a setting which does not satisfy these finiteness constraints. A proof of the unification result in the infinite case can be found in [6]. Here we prove the special case of this result that is needed for our main theorem.

Groote [11] added negative premises to tyft/tyxt, resulting in the ntyft/ntyxt format (that also generalizes the GSOS format of [3]), and proved that the congruence theorem extends to certain well-founded TSSs in ntyft/ntyxt format. We will show that the reduction of tyft/tyxt rules to tree rules can be lifted to the positive part of rules in ntyft/ntyxt format, but a simple example learns that this reduction cannot be applied to the negative premises. Again, we will find that the congruence theorem concerning the ntyft/ntyxt format can do without well-foundedness.

Verhoef [17] defined the panth format, which adds predicates to ntyft/ntyxt, and proved that the congruence theorem holds for well-founded TSSs in panth format. We will show that our results extend to the panth format too.

**Acknowledgments.** Catuscia Palamidessi and Fer-Jan de Vries noted the link with unification. Frits Vaandrager and Chris Verhoef provided useful comments.

## 2 Preliminaries

This section contains the basic definitions.

### 2.1 The signature

In the sequel we assume the existence of an infinite set of variables $V$.

**Definition 2.1** *A (single-sorted) signature $\Sigma$ consists of a set of function symbols, disjoint with $V$, together with their arities.*

*The collection $\mathbb{T}(\Sigma)$ of* (open) terms *over $\Sigma$ is defined as the least set satisfying:*

- *each variable from $V$ is in $\mathbb{T}(\Sigma)$,*
- *if $f \in \Sigma$ has arity $n$, and $t_1, ..., t_n \in \mathbb{T}(\Sigma)$, then $f(t_1, ..., t_n) \in \mathbb{T}(\Sigma)$.*

*A term is called* closed *if it does not contain any variables.*

In the sequel we assume a fixed signature $\Sigma$.

A *substitution* is a mapping $\sigma : V \to \mathbb{T}(\Sigma)$. Each substitution is extended to a mapping from terms to terms in the standard way. As usual, $\sigma\rho$ denotes the composition of the substitutions $\sigma$ and $\rho$, in which $\rho$ is applied first.

### 2.2 Transition system specifications

In the sequel we assume the existence of a set of labels $A$.

**Definition 2.2** *For each label $a$, the expression $\xrightarrow{a}$ denotes a binary relation on terms. A pair $t \xrightarrow{a} t'$ is called a* transition. *A transition is* closed *if it involves closed terms only.*

**Definition 2.3** *A* (transition) rule *$r$ is an expression of the form $H/c$, with $H$ a collection of transitions, called the* premises *(or the* hypotheses*), of $r$, and $c$ a transition, called the* conclusion *of $r$. In the sequel, $concl(r)$ will denote the conclusion of the rule $r$.*

*A* Transition System Specification (TSS) *is a collection of transition rules.*

*A TSS is* small *if for each of its rules, the cardinality of its collection of premises does not exceed the cardinality of the set $V$ of variables.*

The notion of substitution extends to transitions and rules as expected.

**Definition 2.4** *A* proof structure *is a tuple $(B, r, \phi)$, where*

- *$B$ is a collection of transition rules which do not have any variables in common,*
- *$r \in B$,*
- *$\phi$ is an injective mapping from $B\backslash\{r\}$ to the collection of premises of rules in $B$, such that each chain $b_0, b_1, b_2, ...$ in $B$ with $\phi(b_{i+1})$ a premise of $b_i$ is finite.*

*In the sequel, $top(B, r, \phi)$ will denote the collection of premises of rules in $B$ that are outside the image of $\phi$.*

*Write $(B', r', \phi') < (B, r, \phi)$ iff $B' \subset B$, $\phi' = \phi\restriction(B'\backslash\{r'\})$, $top(B', r', \phi') \subseteq top(B, r, \phi)$ and there is a chain $r = b_0, b_1, ..., b_n = r'$ with $n > 0$ and $\phi(b_{i+1})$ a premise of $b_i$.*

Note that $<$ is a partial well-order, i.e. any chain $(B_0, r_0, \phi_0) > (B_1, r_1, \phi_1) > (B_2, r_2, \phi_2) > \cdots$ is finite. Hence we may apply induction w.r.t. $<$.

**Definition 2.5** *A substitution $\sigma$* matches *with a proof structure $(B, r, \phi)$ if $\sigma(concl(b)) = \sigma(\phi(b))$ for each $b \in B \backslash \{r\}$.*

*A rule $H/c$ is* provable from *a small TSS $R$ if $c \in H$ or there exists a proof structure $(B, r, \phi)$ where each rule in $B$ is in $R$ modulo $\alpha$-conversion (bijective renaming of variables), and a substitution $\sigma$ that matches with $(B, r, \phi)$, such that $\sigma(top(B, r, \phi)) \subseteq H$ and $\sigma(concl(r)) = c$.*

**Example 2.6 (A fragment of CCS with replication operator).** *Let $\mathcal{A}$ be a set of* names. *The set $\bar{\mathcal{A}}$ of* co-names *is given by $\bar{\mathcal{A}} = \{\bar{a} \mid a \in \mathcal{A}\}$, and $\mathcal{L} = \mathcal{A} \cup \bar{\mathcal{A}}$ is the set of* visible actions. *The function $\bar{\cdot}$ is extended to $\mathcal{L}$ by declaring $\bar{\bar{a}} = a$. Furthermore $A = L \cup \{\tau\}$ is the set of* actions. *Note that $\bar{\tau}$ is undefined. The language CCS has a constant $0$, a unary operator $a$ for $a \in A$, binary operators $+$ and $|$, and a few constructs that are omitted here. In addition we consider the unary* replication operator !. *The transition system specification* CCS! *is given by the transition rules below. These rules are actually schemata, where $a$ ranges over $A$.*

$$
\begin{array}{|cccc|}
\hline
ax \xrightarrow{a} x & \dfrac{x \xrightarrow{a} x'}{x + y \xrightarrow{a} x'} & \dfrac{y \xrightarrow{a} y'}{x + y \xrightarrow{a} y'} & \\[3ex]
\dfrac{x \xrightarrow{a} x'}{x \mid y \xrightarrow{a} x' \mid y} & \dfrac{x \xrightarrow{a} x',\ y \xrightarrow{\bar{a}} y'}{x \mid y \xrightarrow{\tau} x' \mid y'} & \dfrac{y \xrightarrow{a} y'}{x \mid y \xrightarrow{a} x \mid y'} & \dfrac{!x \mid x \xrightarrow{a} x'}{!x \xrightarrow{a} x'} \\
\hline
\end{array}
$$

*Here follows an example of a proof structure $(B, r, \phi)$, together with a matching substitution $\sigma$. The rule on the bottom is $r$, and $\phi$ is indicated by the arrows. $Top(B, r, \phi) = \{w \xrightarrow{\bar{a}} w'\}$.*

$$
\begin{array}{ccc}
ap \xrightarrow{a} p & & \\
\downarrow & & \\
\dfrac{q \xrightarrow{a} q'}{q + r \xrightarrow{a} q'} & & \\
\downarrow & & \\
\dfrac{t \xrightarrow{a} t'}{s \mid t \xrightarrow{a} s \mid t'} & & \\
\downarrow & & \\
\dfrac{!u \mid u \xrightarrow{a} u'}{!u \xrightarrow{a} u'} & & \dfrac{x \xrightarrow{\bar{a}} x'}{y + x \xrightarrow{\bar{a}} x'} \\
\searrow & & \swarrow \\
& \dfrac{v \xrightarrow{a} v' \quad w \xrightarrow{\bar{a}} w'}{v \mid w \xrightarrow{\tau} v' \mid w'} & \\
& \downarrow & \\
& \dfrac{!z \mid z \xrightarrow{\tau} z'}{!z \xrightarrow{\tau} z'} &
\end{array}
\qquad
\begin{array}{lcl}
\sigma(p) & = & 0 \\
\sigma(q) & = & a0 \\
\sigma(q') & = & 0 \\
\sigma(r) & = & x \\
\sigma(s) & = & !(a0 + x) \\
\sigma(t) & = & a0 + x \\
\sigma(t') & = & 0 \\
\sigma(u) & = & a0 + x \\
\sigma(u') & = & !(a0 + x) \mid 0 \\
\sigma(v) & = & !(a0 + x) \\
\sigma(v') & = & !(a0 + x) \mid 0 \\
\sigma(w) & = & a0 + x \\
\sigma(w') & = & x' \\
\sigma(x) & = & x \\
\sigma(x') & = & x' \\
\sigma(y) & = & a0 \\
\sigma(z) & = & a0 + x \\
\sigma(z') & = & !(a0 + x) \mid 0 \mid x'
\end{array}
$$

*This structure and $\sigma$ demonstrate that the rule* $\dfrac{x \xrightarrow{\bar{a}} x'}{!(a0 + x) \xrightarrow{\tau} !(a0 + x) \mid 0 \mid x'}$ *is provable from* CCS!.

We say that a transition $t \xrightarrow{a} t'$ is provable from $R$, if the rule with no premises and conclusion $t \xrightarrow{a} t'$ is provable from $R$. The *transition relation* $\longrightarrow_R$ *determined by a TSS* $R$ is the set of all closed transitions provable from $R$.

**Definition 2.7** *Two TSSs are* transition equivalent *if they determine the same transition relation.*

Our notion of provability is chosen in such a way that we can easily obtain our main result. In order to show that it coincides with the notions of provability found elsewhere in the literature, we need the following definition.

**Definition 2.8** *The* provable closure *of a TSS $R$ is the smallest set $R^{\vdash}$ of rules such that*

- *if $c \in H$ then $H/c \in R^{\vdash}$, and*
- *if $K/c \in R$ and $H/\sigma(d) \in R^{\vdash}$ for $d \in K$ and some substitution $\sigma$, then $H/\sigma(c) \in R^{\vdash}$.*

For notions of provability found elsewhere in the literature (e.g. [2, 4, 5, 9, 10, 11, 12, 17]) the following proposition is easily obtained. By establishing the same for our notion, it follows that it coincides with the others. The proposition only holds for small TSSs, but this restriction will turn out to be inessential for our main result. Moreover, every TSS can be made 'small' by adding sufficiently many variables.

**Proposition 2.9** *A rule $H/c$ is provable from a small TSS $R$ iff it belongs to $R^{\vdash}$.*

**Proof.** "Only if": The case $c \in H$ is trivial. The other case is established by induction on the partial well-order $<$ between proof-structures. Let $H/\sigma(c)$ be provable from $R$ by means of a proof structure $(B, K/c, \phi)$ and a matching substitution $\sigma$. Assume that any formula provable by means of a smaller proof structure belongs to $R^{\vdash}$. Then $H/\sigma(d) \in R^{\vdash}$ for any $d \in K$. It follows that $H/\sigma(c) \in R^{\vdash}$.

"If": By induction on the construction of $R^{\vdash}$. The induction base, $c \in H$, is again trivial. Now suppose $K/c \in R$ and $H/\rho(d)$ is provable from $R$ for $d \in K$ and some substitution $\rho$. Let $(B_d, r_d, \phi_d)$ be proof structures with matching substitutions $\sigma_d$ that establish $H/\rho(d)$ for $d \in K$. Since there exist at least as many variables as there are premises in $K$, the variables in these proof structures can be renamed to become all different, and different from the ones in $K/c$, and a substitution $\sigma$ can be constructed that matches with each of these proof structures so as to yield the corresponding rule, and equals $\rho$ on the variables in $K/c$. Now $(\bigcup_{d \in K} B_d \cup \{K/c\}, K/c, \bigcup_{d \in K} \phi_d \cup \phi')$, where $\phi'$ is the function that sends $r_d$ to $d$ for $d \in K$, is a proof structure that matches with $\sigma$, yielding $H/\rho(c)$. $\square$

The proof of the following lemma is straightforward and left to the reader.

**Lemma 2.10** *If all the rules in a TSS $S$ are provable from a TSS $R$, then all the rules that are provable from $S$ are also provable from $R$.*

### 2.3 Strong bisimulation

**Definition 2.11** *Assume a TSS $R$. Two closed terms $p_0, q_0$ are $R$-*bisimilar*, notation $p_0 \leftrightarrow_R q_0$, if there exists a symmetric binary relation $\mathcal{B}$ on closed terms such that*

- $p_0 \mathcal{B} q_0$,
- *if $p\mathcal{B}q$ and $p \xrightarrow{a}_R p'$, then there is a closed term $q'$ such that $q \xrightarrow{a}_R q'$ and $p'\mathcal{B}q'$.*

## 2.4 The tyft/tyxt format

In general, bisimulation equivalence it is not a *congruence*, i.e. it may be the case that $p_i \underline{\leftrightarrow}_R q_i$ for $i = 1, ..., n$, but $f(p_1, ..., p_n)$ and $f(q_1, ..., q_n)$ are not $R$-bisimilar. Therefore, Groote and Vaandrager [12] have introduced the *tyft/tyxt format*. If a TSS is in this format, and if it satisfies a well-foundedness criterion, then the bisimulation it induces is a congruence.

**Definition 2.12** *A transition rule is a* tyft *rule if it is of the form*

$$\frac{\{t_i \xrightarrow{a_i} y_i \mid i \in I\}}{f(x_1, ..., x_m) \xrightarrow{a} t}$$

*where the $x_k$ and the $y_i$ are distinct variables (and $I$ is some, not necessarily finite, index set). Similarly, a* tyxt *rule is of the form*

$$\frac{\{t_i \xrightarrow{a_i} y_i \mid i \in I\}}{x \xrightarrow{a} t}$$

*where $x$ and the $y_i$ are distinct variables. A TSS is said to be in* tyft/tyxt *format if it consists of tyft and tyxt rules only.*

The TSS CCS! from Example 2.6 is in tyft/tyxt format. All its rules are tyft rules. Note that any TSS in tyft/tyxt format is 'small' in the sense of Definition 2.2.

**Definition 2.13** *Assume a set $\{t_i \xrightarrow{a_i} t_i' \mid i \in I\}$ of transitions. Its 'dependency graph' is a directed graph, with the collection of variables $V$ as vertices, and with as edges the collection*

$$\{\langle x, y\rangle \mid x \textit{ and } y \textit{ occur in } t_i \textit{ and } t_i' \textit{ respectively, for some } i \in I\}.$$

*A set of transitions is called* well-founded *if any backward chain of edges in its dependency graph is finite. A transition rule is well-founded if its collection of premises is so, and a TSS is well-founded if all its rules are so.*

**Example 2.14** *Examples of sets of transitions that are not well-founded are:*

- $\{y \xrightarrow{a} y\}$,
- $\{y_1 \xrightarrow{a} y_2,\ y_2 \xrightarrow{b} y_1\}$,
- $\{y_{i+1} \xrightarrow{a} y_i \mid i = 0, 1, 2, ...\}$.

The following congruence theorem originates from [12].

**Theorem 2.15** *If a TSS $R$ is well-founded and in tyft/tyxt format, then $\underline{\leftrightarrow}_R$ is a congruence.*

In Section 4 we will see that the requirement of well-foundedness in this theorem can be dropped.

## 3 Unification

A standard result from logic programming says that if a finite collection $E$ of equations between terms is *unifiable*, then there exists a *unifier* $\rho'$ for $E$ such that each unifier for $E$ is also a unifier for $\rho'$. This result follows from the well-known Martelli-Montanari algorithm [13]. See [1] for the basic definitions and for an introduction to the field of logic programming and unification.

In Fokkink [6], this theorem is generalized to the case where $E$ may be infinite. The first property in Lemma 3.2, which will be vital in the proof of the main theorem, is a corollary of this unification result. However, we present a full proof of the lemma, because we will need two extra properties of the unifier $\rho'$, which follow most easily from its construction. Also, the proof of this lemma is much simpler than the proof of the stronger unification result in [6].

**Definition 3.1** *A substitution $\sigma$ is a* unifier *for a substitution $\rho$ if $\sigma\rho = \sigma$. In this case, $\rho$ is called* unifiable*.*

**Lemma 3.2** *If a substitution $\rho$ is unifiable, then there exists a unifier $\rho'$ for $\rho$ with the following properties:*

1. *Each unifier for $\rho$ is also a unifier for $\rho'$.*
2. *If $\rho(x) = x$, then $\rho'(x) = x$.*
3. *If $\rho^n(x)$ is a variable for all $n \geq 0$, then $\rho'(x)$ is a variable.*

**Proof.** Let $W$ denote the collection of variables $x$ for which $\rho^n(x)$ is a variable for all $n \geq 0$. First, we define the restriction $\rho'_0$ of $\rho'$ to $W$.

Define a binary relation $\sim$ on $W$ by $x \sim x'$ if $\rho^m(x) = \rho^n(x')$ for certain $m$ and $n$. Note that $\sim$ is an equivalence relation. Under $\rho'_0$, we contract the elements of each equivalence class $C \subseteq W$ to one variable from this class as follows.

- If $\rho(x_0) = x_0$ for some $x_0 \in C$, then for all $x \in C$ $\rho^n(x) = x_0$ for some $n$. This implies $\rho(x) \neq x$ for $x \in C \backslash \{x_0\}$, so $x_0$ is determined uniquely. Put $\rho'_0(x) = x_0$ for $x \in C$.
- If $\rho(x) \neq x$ for all $x \in C$, then just pick some $x_0 \in C$ and put $\rho'_0(x) = x_0$ for $x \in C$.

Put $\rho'_0(y) = y$ for $y \notin W$.

We construct $\rho'(y)$ as follows. By assumption, $\rho$ allows a unifier $\sigma$. Since $\sigma\rho = \sigma$, it follows that $\sigma\rho^n = \sigma$ for $n \geq 0$. Clearly, the *size* of each $\rho^n(y)$ (that is, the number of function symbols it contains) is smaller or equal than the size of $\sigma\rho^n(y) = \sigma(y)$. Moreover, each term $\rho^{n+1}(y)$ has at least the size of $\rho^n(y)$. Since the sizes of the $\rho^n(y)$ cannot grow beyond the size of $\sigma(y)$, it follows that from a certain natural number $N(y)$ onwards, the terms $\rho^n(y)$ all have the same size. Hence, for $n \geq N(y)$, $\rho^{n+1}(y)$ is obtained from $\rho^n(y)$ by replacing variables by variables. This means that all variables in $\rho^{N(y)}(y)$ are in $W$. Put

$$\rho'(y) = \rho'_0\rho^{N(y)}(y).$$

Note that $N(x) = 0$ if $x \in W$, so $\rho'$ equals $\rho'_0$ on $W$. We check the required properties for $\rho'$.

- $\rho'$ is a unifier for $\rho$.

  First, consider a variable $x \in W$. Since $\rho(x) \sim x$, and $\rho'_0$ contracts variables in the same equivalence class, we have $\rho'_0\rho(x) = \rho'_0(x)$. Since $\rho'$ equals $\rho'_0$ on $W$, this implies $\rho'\rho(x) = \rho'(x)$.

  Next, consider a variable $y \notin W$. Then clearly $N(y) = N(\rho(y)) + 1$, so

  $$\rho'\rho(y) = \rho'_0\rho^{N(\rho(y))}\rho(y) = \rho'_0\rho^{N(y)}(y) = \rho'(y).$$

- Each unifier $\sigma$ for $\rho$ is a unifier for $\rho'$.

  First, consider a variable $x \in W$. Since $\rho'_0(x) \sim x$, there are $m$ and $n$ such that $\rho^m\rho'_0(x) = \rho^n(x)$. After applying $\sigma$ to both sides we get $\sigma\rho'_0(x) = \sigma(x)$. Since $\rho'_0(y) = y$ for variables $y \notin W$, it follows that $\sigma\rho'_0 = \sigma$.

  So for each variable $y$ we have

  $$\sigma\rho'(y) = \sigma\rho'_0\rho^{N(y)}(y) = \sigma\rho^{N(y)}(y) = \sigma(y).$$

- If $\rho(x) = x$, then $\rho'(x) = x$.

  Clearly $x \in W$, so $\rho'(x) = \rho'_0(x)$. Since $\rho(x) = x$, the construction of $\rho'_0$ ensures that $\rho'_0(x) = x$.

- If $\rho^n(x)$ is a variable for all $n \geq 0$, then $\rho'(x)$ is a variable.

  By definition $x \in W$, so $\rho'(x) = \rho'_0(x)$. From the construction of $\rho'_0$ it follows that its image contains variables only. $\square$

# 4 Tyft/Tyxt Reduces to Tree

This section contains the proof of the main theorem, which says that for each TSS in tyft/tyxt format there exists a transition equivalent TSS in the more restrictive tree format.

## 4.1 Tyft/tyxt reduces to tyft

The following lemma from [12] indicates that we can refrain from tyxt rules.

**Lemma 4.1** *Each TSS $R$ in tyft/tyxt format is transition equivalent to a TSS in tyft format.*

**Proof.** Replace each tyxt rule $r$ in $R$ by a collection of tyft rules $\{r_f | f \in \Sigma\}$, where each $r_f$ is obtained by substituting $f(x_1, ..., x_n)$ for $x$ in $r$, with $x_1, ..., x_n$ variables that do not yet occur in $r$. Let $R'$ denote the collection of tyft rules that is thus obtained. Clearly, for each proof from $R$ of a certain closed transition, there is a proof from $R'$ of the same transition, and vice versa. Hence, $R$ and $R'$ are transition equivalent. $\square$

### 4.2 Tyft reduces to xyft

**Definition 4.2** *A transition rule is said to be a* xytt *rule if the terms at both sides of its premises are all single variables.*

**Definition 4.3** *A transition rule is called* xyft *if it is both tyft and xytt.*

In this section, we show that each TSS in tyft format is xytt equivalent to a TSS in xyft format, where xytt equivalence is a stronger equivalence notion than transition equivalence.

**Definition 4.4** *Two TSSs are* xytt equivalent *if exactly the same xytt rules are provable from both.*

**Theorem 4.5** *Each TSS $R$ in tyft format is xytt equivalent to a TSS in xyft format.*

**Proof.** We shall prove $R$ xytt equivalent to the TSS $S$ of xyft rules that are provable from $R$. Since all rules in $S$ are provable from $R$, Lemma 2.10 yields that the xytt rules provable from $S$ are provable from $R$. We show that the converse is also true, i.e. that each xytt rule $H/c$ provable from $R$ is provable from $S$. We apply induction on the partial well-order $<$ between proof structures, so suppose that $(B, r, \phi)$ derives $H/c$ from $R$, and the case has been proved for xytt rules that are derivable from $R$ by means of a proof structure smaller than $(B, r, \phi)$.

Since $(B, r, \phi)$ is a proof structure for $H/c$, there exists a substitution $\sigma$ that matches with $(B, r, \phi)$ such that $\sigma(top(B, r, \phi)) \subseteq H$ and $\sigma(concl(r)) = c$. From $(B, r, \phi)$ we construct recursively a sub-structure $(B', r, \phi')$ which is a proof structure for a rule $s \in S$. In parallel, we construct a partial substitution $\rho$ which is unified by $\sigma$ in the sense that $\sigma(\rho(x)) = \sigma(x)$ for those variables $x$ for which $\rho$ has been defined.

- $r \in B'$,
- if $b \in B \backslash \{r\}$, and if $\phi(b)$ is a premise $t \xrightarrow{a} y$ of a rule in $B'$ such that for some $k \geq 0$:
  1. $\rho^i(t)$ is defined for $i = 0, ..., k$,
  2. $\rho^i(t)$ is a variable for $i = 0, ..., k-1$,
  3. $\rho^k(t)$ is of the form $f(t_1, ..., t_n)$,

  then $b \in B'$.

  Since $\sigma$ matches with $(B, r, \phi)$, we have $\sigma(concl(b)) = \sigma(t \xrightarrow{a} y)$. By assumption, $\sigma$ is a unifier for the partially defined $\rho$, so $\sigma(t) = \sigma\rho^k(t) = \sigma(f(t_1, ..., t_n))$. Hence, $concl(b)$ is of the form $f(x_1, ..., x_n) \xrightarrow{a} u$, with $\sigma(x_j) = \sigma(t_j)$ for $j = 1, ..., n$ and $\sigma(u) = \sigma(y)$. Define $\rho(x_j) = t_j$ for $j = 1, ..., n$ and $\rho(y) = u$. Note that $\sigma$ is a unifier for the extended $\rho$.

In order to extend $\rho$ to a full substitution, we define $\rho(x) = x$ for all variables $x$ for which $\rho$ has not yet been defined. Finally, $\phi'$ is the restriction of $\phi$ to $B' \backslash \{r\}$.

Since $\sigma$ is a unifier for $\rho$, Lemma 3.2 indicates the existence of a unifier $\rho'$ for $\rho$ with the following properties.

1. $\sigma\rho' = \sigma$.
2. If $\rho(x) = x$, then $\rho'(x) = x$.

3. If $\rho^k(x)$ is a variable for all $k \geq 0$, then $\rho'(x)$ is a variable.

Consider the rule $b$ in the construction of $B'$ and $\rho$. Recall that its conclusion is of the form $f(x_1,...,x_n) \xrightarrow{a} u$ and $\phi'(b) = t \xrightarrow{a} y$, where $\rho^k(t) = f(t_1,...,t_n) = \rho(f(x_1,...,x_n))$ and $\rho(y) = u$. Since $\rho'$ is a unifier for $\rho$, it follows that

$$\rho'(\phi'(b)) = \rho'(t \xrightarrow{a} y) = \rho'(\rho^k(t) \xrightarrow{a} \rho(y)) = \rho'(\rho(f(x_1,...,x_n)) \xrightarrow{a} u) = \rho'(concl(b)).$$

So $\rho'$ matches with $(B', r, \phi')$. Hence the rule $s = \rho'(top(B', r, \phi')/concl(r))$ is provable from $R$.

We show that $s$ is xyft. From the construction of $\rho$ it follows that its *domain* (i.e. the variables $x$ for which $\rho(x) \neq x$) consists of two kinds of variables:

1. variables that occur at the left-hand side of the conclusion of rules in $B' \backslash \{r\}$,

2. variables that occur at the right-hand side of premises in the range of $\phi'$.

Hence, if $g(x_1,...,x_m) \xrightarrow{b} t$ is the conclusion of $r$, then $\rho(x_j) = x_j$ for $j = 1,...,m$. Now property 2 of $\rho'$ yields $\rho'(x_j) = x_j$ for $j = 1,...,m$, so the conclusion $\rho'(g(x_1,...,x_m) \xrightarrow{b} t)$ of $s$ is of the form $g(x_1,...,x_m) \xrightarrow{b} \rho'(t)$.

The premises of $s$ are in $\rho'(top(B', r, \phi'))$, so they are of the form $\rho'(t \xrightarrow{a} y)$ where $t \xrightarrow{a} y$ is a premise of a rule in $B'$ outside the range of $\phi'$. Hence $y$ is not in the domain of $\rho$, i.e. $\rho(y) = y$, so property 2 of $\rho'$ yields $\rho'(y) = y$. Moreover, as in a proof structure no two rules have variables in common, all variables $y$ at the right-hand side of these premises and $x_1,...,x_m$ are distinct. In order to show that $\rho'(t)$ is a variable, we distinguish two cases.

1. $t \xrightarrow{a} y \in top(B, r, \phi)$.

   Then $\sigma(t \xrightarrow{a} y) \in H$, so $\sigma(t)$ is a variable. As $\sigma\rho'(t) = \sigma(t)$, also $\rho'(t)$ is a variable.

2. $t \xrightarrow{a} y \notin top(B, r, \phi)$.

   Then $\phi(b) = t \xrightarrow{a} y$ for some $b \in B$. Since $t \xrightarrow{a} y$ is outside the range of $\phi'$, it follows that $b \notin B'$. Hence the inductive construction of $B'$ and $\rho$ implies that $\rho^k(t)$ is a variable for $k \geq 0$. So property 3 of $\rho'$ yields that $\rho'(t)$ is a variable.

Hence, $s$ is xyft.

Since $s$ is provable from $R$ and xyft, by definition $s \in S$. For $c' \in \sigma(top(B', r, \phi'))$, the xytt rule $H/c'$ is provable from $R$ by means of a strictly smaller sub-structure of $(B, r, \phi)$, so by induction such rules $H/c'$ are provable from $S$. Since $\sigma(s) = \sigma\rho'(top(B', r, \phi')/concl(r)) = \sigma(top(B', r, \phi'))/c$ it follows from Proposition 2.9 that $H/c$ is provable from $S$. □

**Example 4.6** *Applying this construction to the proof structure $(B, r, \phi)$ of Example 2.6 gives rise to the sub-structure $(B', r, \phi')$ displayed below, together with the (partial) substitution $\rho$. Applying the construction in the proof of the unification lemma to $\rho$ gives the substitution $\rho'$ (with $\rho'(x) = x$*

*for variables $x$ not explicitly mentioned).*

$$\frac{t \xrightarrow{a} t'}{s \mid t \xrightarrow{a} s \mid t'} \qquad \begin{array}{lcl} \rho(v) & = & !z \\ \rho(w) & = & z \\ \rho(z') & = & v' \mid w' \end{array} \qquad \begin{array}{lcl} \rho'(v) & = & !z \\ \rho'(w) & = & z \\ \rho'(z') & = & !z \mid t' \mid w' \end{array}$$

$$\frac{!u \mid u \xrightarrow{a} u'}{!u \xrightarrow{a} u'} \qquad \begin{array}{lcl} \rho(u) & = & z \\ \rho(v') & = & u' \end{array} \qquad \begin{array}{lcl} \rho'(u) & = & z \\ \rho'(v') & = & !z \mid t' \end{array}$$

$$\frac{v \xrightarrow{a} v' \quad w \xrightarrow{\bar{a}} w'}{v \mid w \xrightarrow{\tau} v' \mid w'} \qquad \begin{array}{lcl} \rho(s) & = & !u \\ \rho(t) & = & u \\ \rho(u') & = & s \mid t' \end{array} \qquad \begin{array}{lcl} \rho'(s) & = & !z \\ \rho'(t) & = & z \\ \rho'(u') & = & !z \mid t' \end{array}$$

$$\frac{!z \mid z \xrightarrow{\tau} z'}{!z \xrightarrow{\tau} z'}$$

*The resulting xyft rule $s$ is* $\dfrac{z \xrightarrow{a} t' \quad z \xrightarrow{\bar{a}} w'}{!z \xrightarrow{\tau} !z \mid t' \mid w'}$.

Although according to Theorem 4.5 the tyft/tyxt format reduces to the more restrictive xyft format, this is by no means an argument to abandon the tyft/tyxt format. Because a simple TSS in tyft/tyxt format may take a much more complicated form if it is described in xyft format. This is demonstrated by the following example.

**Example 4.7** *Assume two functions $a, b$ of arity zero, a function $f$ of arity one, and a label $l$. Consider the following TSS in tyft format.*

$$a \xrightarrow{l} a \qquad\qquad \frac{a \xrightarrow{l} y}{a \xrightarrow{l} f(y)}$$

*In order to describe this TSS in xyft format, we need an infinite number of rules: $a \xrightarrow{l} f^n(a)$ for $n = 0, 1, 2, ...$ (The auxiliary function symbol $b$ is present to avoid that the TSS can be described by the single rule $a \xrightarrow{l} x$.)*

## 4.3 Xyft reduces to tree

The following terminology originates from [12].

**Definition 4.8** *A variable is called* free *in a rule if it does not occur at the right-hand side of the premises, nor at the left-hand side of the conclusion of the rule. A rule is called* pure *if it is well-founded and does not contain any free variables. A* tree *rule is a pure xyft rule.*

**Theorem 4.9** *Each TSS $R$ in xyft format is transition equivalent to a TSS in tree format.*

**Proof.** We prove $R$ transition equivalent with the TSS $S$ of tree rules that can be proved from $R$. Since all rules in $S$ can be proved from $R$, Lemma 2.10 implies that each transition provable from $S$ is also provable from $R$. We check the converse, namely that a closed transition $p \xrightarrow{a} p'$ provable from $R$ is provable from $S$.

Since $p \xrightarrow{a} p'$ is provable from $R$, there exist a rule $r \in R$ and a substitution $\sigma$ such that the premises of $r$ under $\sigma$ are provable from $R$ and the conclusion of $r$ under $\sigma$ yields $p \xrightarrow{a} p'$. Let $r$ be of the form

$$\frac{\{z_i \xrightarrow{a_i} y_i \mid i \in I\}}{f(x_1, ..., x_m) \xrightarrow{a} t}$$

Using induction, we may assume that $\sigma(z_i \xrightarrow{a_i} y_i)$ is provable from $S$ for $i \in I$.

We construct from $r$ a rule $r'$ in $S$ as follows. If there is no backward path in the dependency graph of $r$ from a vertex $y_i$ to a vertex $x_j$, then replace the variables $z_i$ and $y_i$ in $r$ by $\sigma(z_i)$ and $\sigma(y_i)$ respectively. Moreover, replace free variables $z$ in $t$ by $\sigma(z)$. As $p \xrightarrow{a} p$‘ is a closed transition, $\sigma(z)$ does not contain any variables. The resulting rule $r''$ is a substitution instance of $r$, so $r''$ is provable from $R$. Remove each premise $\sigma(z_i \xrightarrow{a_i} y_i)$ from $r''$. Since those transitions are provable from $R$, the resulting rule $r'$ is provable from $R$ as well.

Clearly, $r'$ is xyft and without free variables. Moreover, $r'$ is well-founded, because for each premise $z_i \xrightarrow{a_i} y_i$ in $r'$, the (only) backward path from the vertex $y_i$ in the dependency graph of $r'$ terminates at a vertex $x_j$. Hence, $r'$ is a tree rule, so $r' \in S$. Since the premises of $r'$ under $\sigma$ are provable from $S$, and since the conclusion of $r'$ under $\sigma$ yields $p \xrightarrow{a} p'$, Proposition 2.9 implies that $p \xrightarrow{a} p'$ is provable from $S$. $\square$

So, we have found that for each TSS in tyft/tyxt format there exists a transition equivalent TSS in tree format. Since tree rules are well-founded tyft rules, this result implies that the congruence theorem for tyft/tyxt can do without well-foundedness.

**Corollary 4.10** *If a TSS $R$ is in tyft/tyxt format, then $\underline{\leftrightarrow}_R$ is a congruence.*

# 5 Extensions to Other Formats

## 5.1 The ntyft/ntyxt format

Groote [11] extended the tyft/tyxt format to the *ntyft/ntyxt* format, which as extra feature allows transition rules to contain negative premises, i.e. expressions of the form $t \stackrel{a}{\not\longrightarrow}$. In a setting with negative premises, the definition of the transition relation determined by a TSS has to be adapted. Certain TSSs may fail to determine a transition relation at all, for instance due to rules such as

$$\frac{t \stackrel{a}{\not\longrightarrow}}{t \xrightarrow{a} t'}$$

One of the most general ways to associate transitions to TSSs with negative premises is through the notion of a *stability*, which was introduced by Gelfond and Lifschitz [8] in logic programming. The transition relation determined by a TSS is then its *unique stable transition relation* if such exists. Bol and Groote [4], who adapted this notion for TSSs, showed that there exist TSSs in ntyft/ntyxt format with a unique stable transition relation for which bisimulation is not a congruence. However, they found a subclass of such TSSs for which it is. They defined a (somewhat complicated) notion of *reduction* of TSSs, inspired by the work of Van Gelder, Ross and Schlipf [7] in logic programming, and proved a congruence theorem for well-founded TSSs in the ntyft/ntyxt format that are *positive* (that is without negative premises) after applying reduction. The transition relation associated to a TSS that is positive after reduction consist of the closed transitions that are provable from the reduced TSS. This is then the unique stable transition relation of the TSS.

Earlier, Groote [11] had adapted the concept of *stratification*—also found in logic programming, see Apt [1]—to transition system specifications, and showed how a *stratified* TSS determines a transition relation. He also proved that bisimulation equivalence is a congruence for well-founded stratified TSSs in the ntyft/ntyxt format. A TSS that is stratified is surely positive after reduction, and the transition relation determined by the method of stratification is the same as the one determined by the method of reduction. Thus we have a hierarchy of properties

$$\textit{positive} \;\Rightarrow\; \textit{stratified} \;\Rightarrow\; \textit{positive after reduction} \;\Rightarrow\; \textit{has unique transition relation.}$$

The reverse of these inclusions does not hold.

In Van Glabbeek [10] the notion of a *complete* TSS is proposed, which is equivalent to *positive after reduction*. For this purpose, the notion of provability is extended in order to allow the derivation of negative transitions. Then, a TSS is said to be *complete* if for each closed transition $p \xrightarrow{a} p'$, the TSS can prove either $p \xrightarrow{a} p'$ or its negation $p \stackrel{a}{\not\longrightarrow} p'$. In the same paper it is also argued that the unique stable transition relation of an incomplete TSS is not always convincing as the determined transition relation. If for any reason a transition relation needs to be associated to arbitrary TSSs, it is suggested to take the set of closed transitions $p \xrightarrow{a} p'$ that are *irrefutable*, in the sense that $p \stackrel{a}{\not\longrightarrow} p'$ is not provable using the extended concept of provability. Although this method yields the 'right' transition relation for complete TSSs, in the case of incomplete TSSs with a unique stable transition relation it may yield a different—and equally unconvincing—result as the method of stability. The transition relation associated to incomplete TSSs usually has very unpleasant properties. In particular, the congruence result for TSSs in ntyft/ntyxt format does not extend to such TSSs [10]. The following proposition, taken from [10], gives a sufficient condition for two TSSs to be transition equivalent according to each of the methods stability, completeness (=reduction) and irrefutability.

**Proposition 5.1** *Let $R$ and $R'$ be TSSs such that $R \vdash N/c \Leftrightarrow R' \vdash N/c$ for any closed transition rule $N/c$ with only negative premises. Here $\vdash$ denotes provability in the sense of Section 2. Then*

- *$R$ has a unique stable transition relation iff $R'$ has, and in that case these relations coincide;*
- *$R$ is complete iff $R'$ is, and in that case they determine the same transition relation;*
- *and the transitions irrefutable from $R$ are the same as the ones irrefutable from $R'$.*

Thus without committing ourselves on their precise meaning, we can extend our results to TSSs with negative premises by strengthening the requirement of transition equivalence to provability of the same closed transition rules without positive premises. All definitions, lemmas and propositions of Section 2 generalize straightforwardly to TSSs with negative premises, except that a rule is now called well-founded if its collection of *positive* premises is so.

**Definition 5.2** *A* xyntt *rule is an xytt rule enriched with arbitrary negative premises $t \stackrel{a}{\not\longrightarrow}$. A transition rule is called* xynft *if it is both ntyft and xyntt. It is an* ntree *rule if it moreover is pure.*

Without any further complications, we can repeat the construction from the previous section to show that each complete TSS in ntyft/ntyxt format is transition equivalent—it proves the same closed rules without positive premises—to a complete TSS in the ntree format.

Again, TSSs in the latter format are well-founded, so as a corollary we see that the well-foundedness condition in the congruence theorem for the ntyft/ntyxt format can be dropped.

**Corollary 5.3** *If a complete TSS $R$ is in ntyft/ntyxt format, then $\leftrightarrow_R$ is a congruence.*

We show that in general, terms in negative premises cannot be reduced to variables. The *simple negative tree format* allows complete TSSs which consist of pure and well-founded ntyft/ntyxt rules, where the variables of all the premises (so also of the negative premises) are variables. We present a complete TSS in ntyft/ntyxt format for which there does not exist a transition equivalent TSS in simple negative tree format.

Our counter-example is presented in the setting of the process algebra basic CCS. This formalism assumes a constant 0, a binary function alternative composition $x + y$, and unary functions prefix sequential composition $ax$, where $a$ ranges over an alphabet $A$. Basic CCS assumes relations $\xrightarrow{a}$ for $a \in A$, and its operational semantics is defined in Example 2.6.

Add two functions $f$ and $g$ with arity one to the signature of basic CCS, and extend the operational semantics by the following transition rules, to obtain the TSS $R$.

$$\frac{x \xrightarrow{a} y_1 \quad y_1 \xrightarrow{a} y_2}{g(x) \xrightarrow{a} 0} \qquad\qquad \frac{g(x) \stackrel{a}{\not\longrightarrow}}{f(x) \xrightarrow{a} 0}$$

The TSS $R$ is complete and in ntyft/ntyxt format. The premise $g(x) \stackrel{a}{\not\longrightarrow}$ cannot be reduced. An obvious attempt to delete this negative premise would be to replace the second rule by the following two rules.

$$\frac{x \stackrel{a}{\not\longrightarrow}}{f(x) \xrightarrow{a} 0} \qquad\qquad \frac{x \xrightarrow{a} y \quad y \stackrel{a}{\not\longrightarrow}}{f(x) \xrightarrow{a} 0}$$

However, this adapted TSS is not transition equivalent to $R$. For example, $f(aa0 + a0) \xrightarrow{a} 0$ holds in the new TSS, but not in $R$.

In order to provide a rigorous argument that $R$ does not reduce to a TSS in simple negative tree format, we need the following lemma. First note that a TSS $T$ in simple negative tree format is always stratified and hence complete [10], so that there is no ambiguity about the associated transition relation. The latter can thus be taken to be the set of closed transitions that are provable from $T$ in the extended sense of [10]. This is the concept of provability used below.

**Lemma 5.4** *Let $T$ be a TSS in simple negative tree format and $p_0$ and $p_1$ closed terms, such that:*

1. *if $T$ proves $p_0 \xrightarrow{a} q$, then $T$ proves $p_1 \xrightarrow{a} q$,*
2. *if $T$ proves $p_0 \stackrel{a}{\not\longrightarrow}$, then $T$ proves $p_1 \stackrel{a}{\not\longrightarrow}$.*

*If $T$ proves $f(p_0) \xrightarrow{b} q$, then $T$ proves $f(p_1) \xrightarrow{b} q'$ for some $q'$.*

**Proof.** Let $f(p_0) \xrightarrow{b} q$ be provable from $T$. Then, by Proposition 14 in [10], there exists a rule $r \in T$ and a substitution $\sigma$, such that the premises of $r$ under $\sigma$ are provable from $T$ and the conclusion of $r$ under $\sigma$ yields $f(p_0) \xrightarrow{b} q$. Since $r$ is in ntyft format, it has a conclusion of the form $f(x) \xrightarrow{b} t$, where $\sigma(x) = p_0$ and $\sigma(t) = q$.

Define a substitution $\sigma'$ by $\sigma'(x) = p_1$, and $\sigma'(x) = \sigma(y)$ for $y \neq x$. Since $r$ is in simple negative tree format, and since the premises of $r$ under $\sigma$ are provable from $T$, properties 1,2 of the transition systems of $p_0$ and $p_1$ ensure that the premises of $r$ under $\sigma'$ are provable from $T$. So according to Proposition 13 in [10], the conclusion of $r$ under $\sigma'$, $f(p_1) \xrightarrow{b} \sigma'(t)$, is provable from $T$ as well. $\square$

Suppose that the TSS $R$ that was defined before is transition equivalent to a TSS $T$ in simple negative tree format. If $p_0 = a0$ and $p_1 = aa0+a0$, then it is easy to see that the two properties that were formulated in Lemma 5.4 are satisfied. On the other hand, $R$ (and so $T$) proves $f(a0) \xrightarrow{a} 0$ and $f(aa0+a0) \stackrel{a}{\not\longrightarrow}$. According to Lemma 5.4 this cannot be, so apparently $R$ cannot be transition equivalent to a TSS in simple negative tree format.

### 5.2 The panth format

Baeten and Verhoef [2] extended the tyft/tyxt format with predicates, i.e. not only relations $t \xrightarrow{a} t'$, but also predicates such as $t \xrightarrow{a} \surd$ are allowed to occur in transition rules. The definition of strong bisimulation, Definition 2.11, is adapted accordingly by adding a third condition:

– if $p\mathcal{B}q$ and $p \xrightarrow{a}_R \surd$, then $q \xrightarrow{a}_R \surd$.

Next, Verhoef [17] extended the resulting format with negative premises. A congruence theorem holds for well-founded complete TSSs that are in the so-called *panth* format, which is essentially the natural extension of ntyft/ntyxt with predicates.

Without any further complications, we can repeat the construction from the previous section to show that each complete TSS in panth format is transition equivalent to a complete TSS in an extension of the tree format, which allows rules to have premises of the form $z \xrightarrow{a} \surd$ and $t \stackrel{a}{\not\longrightarrow}$ and $t \stackrel{a}{\not\longrightarrow} \surd$, and a conclusion of the form $f(x_1, ..., x_m) \xrightarrow{a} \surd$. As a corollary, we see that the well-foundedness condition in the congruence theorem for the panth format can be dropped.

**Corollary 5.5** *If a complete TSS $R$ is in panth format, then $\underline{\leftrightarrow}_R$ is a congruence.*

## References


[1] K.R. Apt (1990): *Logic programming.* In J. van Leeuwen, editor: *Handbook of Theoretical Computer Science, Volume B, Formal Methods and Semantics*, Elsevier, pp. 493–574.

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

[3] B. Bloom, S. Istrail & A.R. Meyer (1988): *Bisimulation can't be traced: preliminary report.* In *Proceedings 15th ACM Symposium on Principles of Programming Languages*, San Diego, California, pp. 229–239. To appear in *Journal of the ACM.*

[4] R.N. Bol & J.F. Groote (1991): *The meaning of negative premises in transition system specifications.* In J. Leach Albert, B. Monien & M. Rodríguez Artalejo, editors: *Proceedings 18th International Colloquium on Automata, Languages and Programming (ICALP'91)*, Madrid, LNCS 510, Springer-Verlag, pp. 481–494.

[5] W.J. Fokkink (1994): *The tyft/tyxt format reduces to tree rules.* In M. Hagiya & J.C. Mitchell, editors: *Proceedings 2nd Symposium on Theoretical Aspects of Computer Software (TACS'94)*, Sendai, Japan, LNCS 789, Springer-Verlag, pp. 440–453.

[6] W.J. Fokkink (1994): *Idempotent most general unifiers for infinite sets.* Report CS-R9442, CWI, Amsterdam.

[7] A. van Gelder, K. Ross & J.S. Schlipf (1991): *The well-founded semantics for general logic programs*, JACM 38(3), pp. 620–650.

[8] M. Gelfond & V. Lifschitz (1988): *The stable model semantics for logic programming.* In R. Kowalski & K. Bowen, editors: *Proceedings 5th Conference on Logic Programming*, MIT press, pp. 1070–1080.

[9] R.J. van Glabbeek (1993): *Full abstraction in structural operational semantics (extended abstract).* In M. Nivat, C. Rattray, T. Rus & G. Scollo, editors: *Proceedings 3rd Conference on Algebraic Methodology and Software Technology (AMAST'93),* Twente, The Netherlands, Workshops in Computing, Springer-Verlag, pp. 77–84.

[10] R.J. van Glabbeek (1995): *The meaning of negative premises in transition system specifications II.* Available at `ftp://boole.stanford.edu/pub/DVI/negative.dvi.Z`.

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

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

[13] A. Martelli & U. Montanari (1982): *An efficient unification algorithm.* ACM Transactions on Programming Languages and Systems 4(2), pp. 258–282.

[14] D.M.R. Park (1981): *Concurrency and automata on infinite sequences.* In P. Deussen, editor: *5th GI Conference*, LNCS 104, Springer-Verlag, pp. 167–183.

[15] G.D. Plotkin (1981): *A structural approach to operational semantics.* Report DAIMI FN-19, Aarhus University.

[16] R. de Simone (1985): *Higher-level synchronising devices in* Meije-*SCCS.* Theoretical Computer Science 37, pp. 245–267.

[17] C. Verhoef (1994): *A congruence theorem for structured operational semantics with predicates and negative premises.* In B. Jonsson & J. Parrow, editors: *Proceedings 5th Conference on Concurrency Theory (CONCUR'94),* Uppsala, LNCS 836, Springer-Verlag, pp. 433–448.