Theoretical Computer Science 118 (1993) 263–299
Elsevier

Transition system specifications with negative premises

Jan Friso Groote*

Department of Software Technology, CWI, P.O. Box 4079, 1009 AB Amsterdam, The Netherlands

Communicated by M. Wirsing
Received February 1991
Revised November 1991

Abstract

Groote, J.F., Transition system specifications with negative premises, Theoretical Computer Science 118 (1993) 263–299.

In this article the general approach to Plotkin-style operational semantics of Groote and Vaandrager (1989) is extended to transition system specifications (TSSs) with rules that may contain negative premises. Two problems arise: firstly the rules may be inconsistent, and secondly it is not obvious how a TSS determines a transition relation. We present a general method, based on the stratification technique in logic programming, to prove consistency of a set of rules and we show how a specific transition relation can be associated with a TSS in a natural way. Then a special format for the rules, the *ntyft/ntyxt* format, is defined. It is shown that for this format three important theorems hold. The first theorem says that bisimulation is a congruence if all operators are defined using this format. The second theorem states that, under certain restrictions, a TSS in *ntyft* format can be added conservatively to a TSS in pure *ntyft/ntyxt* format. Finally, it is shown that the trace congruence for image-finite processes induced by the pure *ntyft/ntyxt* format is precisely bisimulation equivalence.

1. Introduction

In recent years, many process calculi, programming languages and specification languages are provided with an operational semantics in Plotkin style [26, 27]. We mention CCS [20, 22], SCCS [21], ACP [14], MEIJE [4], Esterel [9], LOTOS [17] and Ada [3].

Correspondence to: J.F. Groote, Department of Philosophy, Utrecht University, Heidelberglaan 8, 3584 CS Utrecht, The Netherlands. Email: jfg@phil.ruu.nl.

*The research of the author was partly supported by RACE project no. 1046, Specification and Programming Environment for Communication Software (SPECS). This document does not necessarily reflect the view of the SPECS project.

In [15] an operational semantics in Plotkin style is defined by a TSS (transition system specification). Basically, a TSS consists of three components. The first component is a *signature* defining the language elements. All terms over this signature will be referred to as (*process*) *terms* or *processes*. The second component of a TSS is a set of *actions* or *labels* representing the different activities that process terms may do. The last component is a set of *rules* that define how processes can perform certain activities depending on the *presence* of specific actions in other processes. In [15] the possibility to perform activity based on the *absence* of actions is not considered.

But in many cases it is convenient to have this possibility. For instance, a deadlock detector $D(p)$ of a process p can naturally be specified as follows: if p can do *no* action then $D(p)$ may signal deadlock. We find deadlock detectors described in this way in [18, 25].

Deadlock detection is also used in sequencing processes. If in $p.q$ (process p sequenced with q) p *cannot* do anything, q may start. See, for instance, [23, 10], where it is observed that sequencing can only be defined using negative premises.

Negative conditions are also useful to describe priorities. Suppose θ is a unary operator that blocks all actions which do not have the highest priority. An operational description of $\theta(p)$ could be that it can only perform action a if it *cannot* perform any activity with higher priority. Descriptions of priorities with negative premises can be found in [6, 13, 15].

Another area where negative conditions can be fruitfully applied is the area of (semi-) synchronous parallel operators. Suppose a sender wants to send data to a receiver. If the receiver is willing to accept the data then data transfer will take place. If the receiver is *not* willing to accept the data then the sender may not be blocked and data may, for instance, disappear. This can conveniently be described using negative premises. Pnueli [28] defines an operator in this way. Also the *put* and *get* primitives of Bergstra [8] can be defined using negative premises.

Often, negative premises can be avoided. Using additional labels, function names and rules, an operational semantics can be given with only postive premises. But then there are many auxiliary transitions that do not correspond to *positive* activity. Moreover, definitions of operational semantics become more complex than necessary. This means that an important property of operational semantics in Plotkin style, namely simplicity, is violated.

For these reasons, we believe that it is useful to investigate how one can deal with negative premises in TSSs.

A format of rules that allows negative premises is the GSOS format of Bloom et al. [10]. All operators mentioned above can be defined in this format. The GSOS format, however, is incompatible with the (pure) *tyft/tyxt* format [15], that allows *lookahead* and no negative premises. Many useful operators definable in the *tyft/tyxt* format cannot be defined using the GSOS format. Their relations are described by the black arrows in Fig. 1. The *positive* GSOS format is the most general format that is below both the *tyft/tyxt* format and the GSOS format. Below the positive GSOS format we find the de Simone format [30], which was already defined by R. de Simone in 1984.

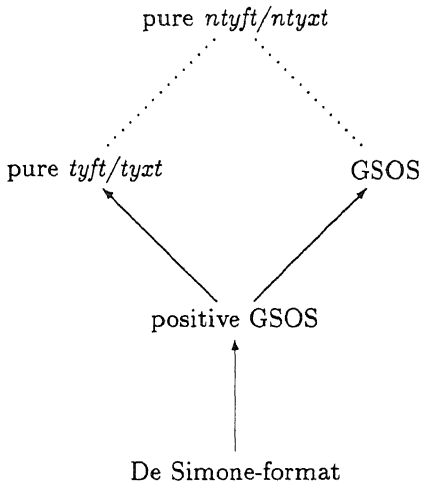

Fig. 1. Pure *ntyft/ntyxt* extends both GSOS and pure *tyft/tyxt*.

The de Simone format is powerful enough to define all the usual operators of CCS, SCCS, ACP and MEIJE. All formats will be explained more precisely in the last section of this article.

The natural question arises whether a format exists that is more general than both the (pure) *tyft/tyxt* format and the GSOS format. An obvious candidate for such a format is obtained by adding negative premises to the *tyft/tyxt* format, getting the *ntyft/ntyxt* format. The *n* in the name of the format is added to indicate the possible presence of negative premises. We arrive at the situation depicted by the dotted lines in Fig. 1.

Two problems arise when rules can be in pure *ntyft/ntyxt* format:

- It is possible to give an inconsistent set of rules. This occurs if one can deduce, using the rules, that a process can perform an action if and only if it cannot do so. In this case the rules do not define an operational semantics.
- Even if the rules are consistent, it is not immediately obvious how these rules determine an operational semantics. The normal notion of provability of transitions where the rules in a TSS are used as inference rules is not satisfactory.

We deal with the first problem by formulating a method of checking whether a transition relation is consistent. This method is based on the *stratifications* [2, 29] that are used in logic programming. The other problem is solved by formulating an explicit definition of the transition relation.

Furthermore, general properties of the *ntyft/ntyxt* format are studied. It is shown that bisimulation is a congruence for this format. Then in Section 5 we define the *sum* of two TSSs and we prove a theorem stating very general conditions under which a TSS can be added *conservatively* to another TSS.

In [15] the completed-trace congruences induced by the pure *tyft/tyxt* format and the GSOS format are characterized. It is interesting to know the impact of the more

powerful testing capabilities of the pure *ntyft/ntyxt* format. Surprisingly, it turns out that the (completed) trace congruence induced by the pure *ntyft/ntyxt* format is exactly strong bisimulation. This is shown by a small test system that provides an alternative for the test systems of [1, 10]. We do not need the *global testing* operators like the ones used in these articles. The combination of *copying*, *lookahead* and negative premises turns out to be powerful enough.

Recently, when applying the stratification technique, we ran into the problem that it is not always satisfactory to find a transition relation for a set of rules. In [12] this problem is analysed in depth. In that paper a general criterion has been given for a TSS to be meaningful. We consider various techniques to prove that this criterion holds for a TSS. Among these is stratification and a stronger technique, called reduction. Furthermore, we reconsider the congruence theorem for the *ntyft/ntyxt* format and the conservativity theorems of this paper in a more extended setting, and we study the relation with complete axiomatisations.

2. Transition system specifications and stratifications

This section describes a TSS as a general framework for defining an operational semantics in Plotkin style. A condition is developed that guarantees the existence of transition relations agreeing with a TSS. This condition is comparable to local stratification as used in logic programming. Next we define which transition relation is associated with a TSS. Finally, some remarks are made about a class of TSSs which determine a transition relation in a unique way. We start off by defining the basic notations that are used throughout the paper. We assume the presence of an infinite set V of *variables* with typical elements $x, y, z \ldots$

Definition 2.1. A (*single-sorted*) *signature* is a structure $\Sigma=(F, r)$, where
- F is a set of *function names* disjoint with V,
- $r: F \to \mathbb{N}$ is a *rank function* which gives the arity of a function name; if $f \in F$ and $r(f)=0$ then f is called a *constant name*.

Let $W \subseteq V$ be a set of variables. The set of Σ-*terms* over W, notation $T(\Sigma, W)$, is the least set satisfying:
- $W \subseteq T(\Sigma, W)$,
- if $f \in F$ and $t_1, \ldots, t_{r(f)} \in T(\Sigma, W)$, then $f(t_1, \ldots, t_{r(f)}) \in T(\Sigma, W)$.

$T(\Sigma, \emptyset)$ is abbreviated as $T(\Sigma)$; elements from $T(\Sigma)$ are called *ground* or *closed* terms. $\mathbb{T}(\Sigma)$ is used to abbreviate $T(\Sigma, V)$, the set of *open terms*. Clearly, $T(\Sigma) \subset \mathbb{T}(\Sigma)$. $Var(t) \subseteq V$ is the set of variables in a term $t \in \mathbb{T}(\Sigma)$. A *substitution* σ is a mapping in $V \to \mathbb{T}(\Sigma)$. A substitution σ is extended to a mapping $\sigma: \mathbb{T}(\Sigma) \to \mathbb{T}(\Sigma)$ in a standard way by the following definition:

$$\sigma(f(t_1, \ldots, t_{r(f)})) = f(\sigma(t_1), \ldots, \sigma(t_{r(f)})) \quad \text{for } f \in F \text{ and } t_1, \ldots, t_{r(f)} \in \mathbb{T}(\Sigma).$$

A substitution is *ground* if it maps all variables onto ground terms.

Definition 2.2. A *TSS* (*transition system specification*) is a triple $P=(\Sigma, A, R)$, with $\Sigma=(F,r)$ a signature, A a set of labels and R a set of rules of the form

$$\frac{\{t_k \xrightarrow{a_k} t'_k \mid k\in K\}\cup\{t_l \xrightarrow{b_l}\!\!\!\!\!\!/\;\;\mid l\in L\}}{t\xrightarrow{a} t'},$$

with K, L index sets, $t_k, t'_k, t_l, t, t'\in\mathbb{T}(\Sigma)$, $a_k, b_l, a\in A$ ($k\in K$, $l\in L$). An expression of the form $t\xrightarrow{a}t'$ is called a (*positive*) *literal*. Here t is called the *source* and t' the *target* of the literal. $t\xrightarrow{a}\!\!\!\!\!\!/\;\;$ is called a *negative literal*. ϕ, ψ, χ are used to range over literals. The literals above the line are called the *premises* and the literal below the line is called the *conclusion*. A rule is called an *axiom* if its set of premises is empty. An axiom

$$\frac{\emptyset}{t\xrightarrow{a}t'}$$

is often written as $t\xrightarrow{a}t'$. The notions "substitution", "Var" and "ground" extend to literals and rules as expected.

Note that this definition differs from the definition of a TSS in [15] because it allows an infinite number of premises and premises may now be negative. The purpose of a TSS is to define a *transition relation* $\rightarrow\subseteq Tr(\Sigma, A)=T(\Sigma)\times A\times T(\Sigma)$. A transition relation states under what actions ground terms over the signature can evolve into one another. This expresses the operational behaviour of these terms. Elements (t, a, t') of a transition relation are written as $t\xrightarrow{a}t'$. We say that a positive literal ψ *holds* in $\rightarrow$, notation $\rightarrow\models\psi$, if $\psi\in\rightarrow$. A negative literal $t\xrightarrow{a}\!\!\!\!\!\!/\;\;$ *holds* in $\rightarrow$, notation $\rightarrow\models t\xrightarrow{a}\!\!\!\!\!\!/\;\;$, if, for no $t'\in T(\Sigma)$, $t\xrightarrow{a}t'\in\rightarrow$.

For TSSs without negative premises the notion of a transition relation that must be associated with it is rather straightforward. All literals that can be proved by a well-founded proof tree, where the rules of the TSS P are used as inference rules, are in the transition relation associated with P. For TSSs with negative premises, these proof trees cannot be used. They can only show the presence of a transition. But, in order to prove the premises of inference rules with negative premises, the absence of a transition must be proved also. This incompatibility is not easily overcome. In fact, it is not very obvious which transition relation should be associated with such a TSS. In [10] Bloom et al. require that a transition relation *agrees with* a TSS. In terms of logic programming, this means that the transition relation is a supported model of the TSS.

Definition 2.3. Let $P=(\Sigma, A, R)$ be a TSS. Let $\rightarrow \subseteq Tr(\Sigma, A)$ be a transition relation. $\rightarrow$ *agrees with* P iff

$$\psi \in \rightarrow \Leftrightarrow \exists \frac{\{\chi_k \mid k \in K\}}{\chi} \in R \text{ and } \exists \sigma: V \rightarrow T(\Sigma) \text{ such that } \sigma(\chi)=\psi$$

$$\text{and } \forall k \in K: \rightarrow \models \sigma(\chi_k).$$

Unfortunately, for a given TSS P, it is not guaranteed that a transition relation that agrees with P exists and, if it exists, it need not be unique. We give three examples illustrating these points. The last example already occurred in [10].

Example 2.4. It is possible to give a TSS P such that no transition relation agrees with it. Let P consist of one constant f, one label a and the rule

$$\frac{f \overset{a}{\not\rightarrow}}{f \overset{a}{\rightarrow} f}.$$

For any transition relation $\rightarrow$ that agrees with P, $f \overset{a}{\rightarrow} f \in \rightarrow$ iff $f \overset{a}{\rightarrow} f \notin \rightarrow$. Clearly, such a transition relation does not exist.

Example 2.5. This example shows that if a transition relation that agrees with a TSS exists, it need not be unique. Take, for example, a TSS with the only rule

$$\frac{f \overset{a}{\rightarrow} f}{f \overset{a}{\rightarrow} f}.$$

Both the empty transition relation and the transition relation $\{f \overset{a}{\rightarrow} f\}$ agree with this TSS.

Example 2.6. If we only use variables in the premises, we can still have an inconsistency. Suppose we have a TSS which consists of constants a and δ and two unary function names f and g. Furthermore, we have exactly one label a and the following rules:

$$\frac{x \overset{a}{\rightarrow} y \qquad y \overset{a}{\rightarrow} z}{f(x) \overset{a}{\rightarrow} \delta},$$

$$\frac{x \overset{a}{\not\rightarrow}}{g(x) \overset{a}{\rightarrow} \delta},$$

$$a \overset{a}{\rightarrow} g(f(a)).$$

No transition relation agrees with this TSS since, if it would exist, we would have that $f(a) \xrightarrow{a} \delta$ is an element of this relation iff it is not.

In this section we will develop a condition on TSSs which guarantees the existence of transition relations that agree with them. The idea is that a transition relation is constructed in a stepwise manner. Whenever it is assumed that some literal does not exist in a transition relation, it must be guaranteed that there is no way to derive the opposite from this assumption. It can be visualised how literals can be derived from each other in a *literal dependency graph* of a TSS $P=(\Sigma, A, R)$. In this graph it is recorded by directed edges how literals depend on each other. An edge from literal ϕ to ψ is labelled by "p" to express that ψ is the conclusion and ϕ a positive premise of $\sigma(r)$ for some ground substitution σ and rule $r \in R$. An edge from $t \xrightarrow{a} t'$ to ψ is labelled with "n" if ψ is the conclusion of $\sigma(r)$ and $t \xrightarrow{a}$ is a negative premise. If there is a cycle in the literal dependency graph with a negative edge then one may derive – from the assumption that, for any t'', literal $t \xrightarrow{a} t''$ is not an element of a transition relation $\rightarrow$ agreeing with P – that $t \xrightarrow{a} t'$ must be an element of $\rightarrow$, which is a contradiction. As an example, a part of the literal dependency graph of Example 2.6 is depicted in Fig. 2.

Definition 2.7. Let $P=(\Sigma, A, R)$ be a TSS. The *(labelled) literal dependency graph* (LDG) G related to P has as nodes the literals in $Tr(\Sigma, A)$ and as labels p and n. The edges of G are given by the triples

- $\langle \sigma(\phi), p, \sigma(\psi) \rangle$, where σ is a ground substitution such that there is a rule $r \in R$ with a positive premise ϕ and a conclusion ψ,

combined with

- $\langle \phi, n, \sigma(\psi) \rangle$, where σ is a ground substitution such that there is a rule $r \in R$ with a negative premise $t \xrightarrow{b}$ and a conclusion ψ such that, for some $t' \in T(\Sigma)$, $\sigma(t \xrightarrow{b} t') = \phi$.

If there is a path between two literals ϕ and ψ of which all edges are labelled with p, it is said that there is a *positive dependency* between ϕ and ψ. If this path contains at least one edge with label n, we say that ψ *depends negatively* on ϕ.

In the next definition the notion of a *stratifiable* TSS is introduced. It will be shown that for stratifiable TSSs there exists a transition relation that agrees with it. As the adjective *stratifiable* suggests, it is possible to make a "stratification". This will be shown later.

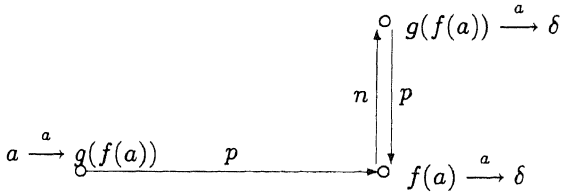

Fig. 2. The LDG belonging to Example 2.6.

Definition 2.8. Let P be a TSS. P is *stratifiable* iff there is no node in the literal dependency graph G of P, such that a path ending in this node contains an infinite number of negative edges.

The following definition assigns an ordinal to each positive literal ϕ. This ordinal represents the number of negative edges in paths ending in ϕ.

Definition 2.9. Let P be a stratifiable TSS with a literal dependency graph G. Nodes that have no incoming paths containing a negative edge are called *LDG basic nodes*. Furthermore, ρ is the equivalence relation between literals such that $\phi \rho \psi$ iff $\phi \equiv \psi$ or there is a path in G from ϕ to ψ and vice versa. Note that if $\phi \rho \psi$ then ϕ is an LDG basic node iff ψ is an LDG basic node. Define $rank_P$ on the equivalence classes of $Tr(\Sigma, A)/\rho$ as follows:
- $rank_P(\phi/\rho)=0$ if ϕ is an LDG basic node;
- $rank_P(\phi/\rho)=\sup(\{rank_P(\psi/\rho)+1 \mid (\psi, n, \chi)$ is an edge in G and $\chi \in \phi/\rho\} \cup \{rank_P(\psi/\rho) \mid (\psi, p, \chi)$ is an edge in $G, \chi \in \phi/\rho$ and $\psi \notin \phi/\rho\})$ otherwise.

Here sup (X) gives the least ordinal $\geqslant$ all elements in the set X. Define $rank_P(\phi)=rank_P(\phi/\rho)$.

Example 2.10. Here we give an example of a TSS P for which the $rank_P$ function uses infinite ordinals. Take the TSS P with one constant f and with natural numbers as labels. Take as rules

$$\frac{f \overset{n}{\not\to}}{f \overset{n+2}{\longrightarrow} f}, \quad n \geqslant 0,$$

$$\frac{f \overset{n}{\not\to}}{f \overset{0}{\to} f} \quad \text{for } n \text{ odd.}$$

$rank_P: Tr(\Sigma, A) \to \omega \cdot 2$ is defined by $rank_P(f \overset{n}{\to} f)=(n-1)/2$ for n odd and $rank_P(f \overset{a}{\to} f)=\omega+n/2$ for n even.

Checking whether or not a literal dependency graph contains cycles with negative edges is laborious and, therefore, not very useful for checking the consistency of a set of rules. The literal dependency graph can be used more fruitfully to construct examples showing that a given TSS is inconsistent. *Local stratifications* [2, 29] provide a more useful technique to show consistency. A stratification of a TSS is given by the following definition.

Definition 2.11. Let $P=(\Sigma, A, R)$ be a TSS. A function $S: Tr(\Sigma, A)\to\alpha$, where α is an ordinal, is called a *stratification* of P iff, for every rule

$$\frac{\{t_k \xrightarrow{a_k} t'_k \mid k\in K\}\cup\{t_l \overset{b_l}{\not\to} \mid l\in L\}}{t\xrightarrow{a} t'}\in R$$

and every substitution $\sigma: V\to T(\Sigma)$, it holds that:

$$\text{for all } k\in K,\quad S(\sigma(t_k \xrightarrow{a_k} t'_k))\leqslant S(\sigma(t\xrightarrow{a} t')),$$

$$\text{for all } l\in L \text{ and } t'_l\in T(\Sigma),\quad S(\sigma(t_l \xrightarrow{b_l} t'_l))<S(\sigma(t\xrightarrow{a} t')).$$

If P has a stratification, we say that P is *stratified*. For $\beta<\alpha$, $S_\beta=\{\phi \mid S(\phi)=\beta\}$ is called a *stratum*. If all literals with the same label are in the same stratum then we speak of a *label-independent* stratification. In the same way, we speak of a *source-independent* and a *target-independent* stratification.

Lemma 2.12. *Let $P=(\Sigma, A, R)$ be a TSS. P is stratifiable iff P is stratified.*

Proof. $\Rightarrow$: As P is stratifiable, the function $rank_P: Tr(\Sigma, R)\to\alpha$ for some ordinal α is defined. It is easy to check that $rank_P$ is a stratification of P.

$\Leftarrow$: Suppose P is stratified by a stratification $S: Tr(\Sigma, A)\to\alpha$. Construct the literal dependency graph G of P. By transfinite induction on β, it is shown that if $S(\phi)=\beta$ then there is no path ending in ϕ in the literal dependency graph, containing an infinite number of negative edges. Suppose the induction holds for all $\beta'<\beta$, $S(\phi)=\beta$ and there is a path ending in ϕ labelled with an infinite number on n's. Then this means that there is a tail of the path

$$\ldots\psi, \phi_n \ldots \phi_2, \phi_1, \phi$$

such that ϕ depends positively on ϕ_1, ϕ_1 depends positively on ϕ_2, etc., while ϕ_n is the first literal that depends negatively on a literal ψ. Hence, $S(\psi)<S(\phi)=\beta$. Using the induction hypothesis, there is no path labelled with an infinite number of n's ending in ψ. But this contradicts the assumption that there was one from ϕ. □

As remarked in Example 2.5, there is not always one unique transition relation that agrees with P. Therefore, we define, given a TSS P with a stratification S, a relation $\to_{P,S}$, which we call the transition relation *associated with* P (and *based on* S). The construction of the transition relation $\to_{P,S}$ from a transition system specification is as follows: a literal ϕ with $S(\phi)=0$ is in $\to_{P,S}$ if it can be "derived" using rules of P which do not have negative premises in the ordinary sense. We now know which literals ϕ with $S(\phi)=0$ are not in $\to_{P,S}$. We use this information to "derive" the literals ϕ with $S(\phi)=1$ which are in $\to_{P,S}$. In this way, we can continue for all strata.

The transition relation associated with P has two nice properties. When we have a TSS P without negative premises, then the transition relation associated with P coincides exactly with the transition relation containing all provable literals [15]. Moreover, the transition relation $\rightarrow_{P,S}$ is independent of the stratification S. This last statement is proved in Lemma 2.16.

First the *degree*(r) of a rule r in a TSS is defined. It is a cardinal that is greater than the number of positive premises in r. Moreover, it is regular. This means that if an ordinal $\alpha_\phi < degree(r)$ is assigned to each positive premise ϕ of r, then there is still some ordinal β such that $\alpha_\phi < \beta < degree(r)$ for all premises ϕ. If r has a finite number of premises, then $degree(r) = \omega$. *degree* is introduced to avoid taking the union over the class of all ordinals in Definition 2.14. In the proof of Theorem 2.15 the regularity of *degree*(r) is crucial.

Definition 2.13. Let $P = (\Sigma, A, R)$ be a TSS. Let $r \in R$ be a rule in R. *degree*(r) is the smallest regular cardinal greater than $|K|$, where K is the index set of positive premises of r. *degree*(P) is the smallest regular cardinal such that $degree(P) \geqslant degree(r)$ for each $r \in R$.

Definition 2.14. Let $P = (\Sigma, A, R)$ be a TSS with a stratification $S: Tr(\Sigma, A) \rightarrow \alpha$ for some ordinal α. The transition relation $\rightarrow_{P,S}$ *associated with* P (and *based on* S) is defined as

$$\rightarrow_{P,S} = \bigcup_{0 \leqslant i < \alpha} \rightarrow_i^P,$$

where transition relations $\rightarrow_i^P \subseteq Tr(\Sigma, A)$ $(0 \leqslant i < \alpha)$, $\rightarrow_{ij}^P \subseteq Tr(\Sigma, A)$ $(0 \leqslant i < \alpha,\ 0 \leqslant j < degree(P))$ are inductively defined by

$$\rightarrow_i^P = \bigcup_{0 \leqslant j < degree(P)} \rightarrow_{ij}^P \quad \text{for } 0 \leqslant i < \alpha,$$

$$\rightarrow_{ij}^P = \{\phi \mid S(\phi) = i,$$

$$\exists \frac{\{\chi_k \mid k \in K\}}{\chi} \in R,\ \exists \sigma: V \rightarrow T(\Sigma):$$

$$\sigma(\chi) = \phi \text{ and } \forall k \in K\ [\chi_k \text{ is positive} \Rightarrow \bigcup_{0 \leqslant j' < j} \rightarrow_{ij'}^P \cup \bigcup_{0 \leqslant i' < i} \rightarrow_{i'}^P \models \sigma(\chi_k)]$$

and

$$[\chi_k \text{ is negative} \Rightarrow \bigcup_{0 \leqslant i' < i} \rightarrow_{i'}^P \models \sigma(\chi_k)]\}$$

for $0 \leqslant i < \alpha$ and $0 \leqslant j < degree(P)$.

Theorem 2.15. *Let $P = (\Sigma, A, R)$ be a TSS with stratification $S: Tr(\Sigma, A) \rightarrow \alpha$ for some ordinal α. Then there is a transition relation, namely $\rightarrow_{P,S}$, that agrees with P.*

Proof. We show that $\rightarrow_{P,S}$ agrees with P:

$\Rightarrow$: Suppose that, for a rule

$$r=\frac{\{t_k \xrightarrow{a_k} t'_k \mid k\in K\}\cup\{t_l \xrightarrow{a_l}\!\!\!\!\!\!/\;\;\mid l\in L\}}{t \xrightarrow{a} t'}\in R$$

and a ground substitution σ, all premises hold in $\rightarrow_{P,S}$. Define $\beta=S(\sigma(t \xrightarrow{a} t'))$. For a negative premise $t_l \xrightarrow{a_l}\!\!\!\!\!\!/\;\;$, it trivially holds that, for every $t''\in T(\Sigma)$, $t_l \xrightarrow{a_l} t''\notin\bigcup_{0\leqslant i<\beta}\rightarrow_i^P$. For a positive premise $t_k \xrightarrow{a_k} t'_k$, it holds that either $\sigma(t_k \xrightarrow{a_k} t'_k)\in\bigcup_{0\leqslant i<\beta}\rightarrow_i^P$ or $\sigma(t_k \xrightarrow{a_k} t'_k)\in\rightarrow_\beta^P$. Consider the set $T=\{j \mid j<degree(P)$ and, for some $k\in K$, j is the smallest ordinal such that $\sigma(t_k \xrightarrow{a_k} t'_k)\in\rightarrow_{\beta j}^P\}$. $|T|\leqslant|K|<degree(P)$. As $degree(P)$ is a regular cardinal, there is some $0\leqslant j'\leqslant degree(P)$ such that $j''<j'<degree(P)$ for every $j''\in T$. Hence, for this j', $\sigma(t \xrightarrow{a} t')\in\rightarrow_{\beta j'}^P$ by definition. Hence, $\sigma(t \xrightarrow{a} t')\in\rightarrow_{P,S}$.

$\Leftarrow$: Suppose $\psi\in\rightarrow_{P,S}$. Then, for some $0\leqslant i<\alpha$, $0\leqslant j<degree(P)$, $\psi\in\rightarrow_{ij}^P$. According to the definition of $\rightarrow_{P,S}$, this means that there is a ground substitution σ and a rule

$$r=\frac{\{\chi_k \mid k\in K\}}{\chi}\in R$$

such that $\sigma(\chi)=\psi$ and, if χ_k is positive, $\sigma(\chi_k)\in\bigcup_{0\leqslant j'<j}\rightarrow_{ij'}^P\cup\bigcup_{0\leqslant i'<i}\rightarrow_{i'}^P$. But then $\sigma(\chi_k)\in\rightarrow_{P,S}$. If $\chi_k\equiv t \xrightarrow{a}\!\!\!\!\!\!/\;\;$ then for every $t'\in T(\Sigma)$: $\sigma(t \xrightarrow{a} t')\notin\bigcup_{0\leqslant i'<i}\rightarrow_{i'}^P$. Due to the stratification, $S(\sigma(t \xrightarrow{a} t'))<i$. Hence, $\sigma(t \xrightarrow{a} t')\notin\rightarrow_{i'}^P$ for $i'\geqslant i$ and, therefore, $\sigma(t \xrightarrow{a} t')\notin\rightarrow_{P,S}$. So, all premises of $\sigma(r)$ hold in $\rightarrow_{P,S}$. □

We show here that the particular stratification used in the construction of $\rightarrow_{P,S}$ is not of any importance.

Lemma 2.16. *Let P be a TSS which is stratified S and S'. The transition relation associated with P and based on S is equal to the transition relation associated with P and based on S'.*

Proof. Assume $P=(\Sigma, A, R)$. Suppose $\rightarrow_{P,S}\neq\rightarrow_{P,S'}$. This means that there is some ϕ such that either $\phi\in\rightarrow_{P,S}-\rightarrow_{P,S'}$ or $\phi\in\rightarrow_{P,S'}-\rightarrow_{P,S}$. Assume that ϕ is minimal with respect to S, i.e. $S(\phi)\leqslant S(\psi)$ for all $\psi\in(\rightarrow_{P,S}-\rightarrow_{P,S'})\cup(\rightarrow_{P,S'}-\rightarrow_{P,S})$. Define $i=S(\phi)$.

- Suppose $\phi\in\rightarrow_{P,S}-\rightarrow_{P,S'}$. Then $\phi\in\rightarrow_{ij}^P$ for some $0\leqslant j<degree(P)$ (see Definition 2.2). Assume that ϕ is minimal with respect to $\rightarrow_{ij}^P$, i.e. for all ψ with $S(\psi)=i$ and $\psi\in\rightarrow_{P,S}-\rightarrow_{P,S'}$: $\psi\notin\rightarrow_{ij'}^P$, with $j'<j$.

  As $\rightarrow_{P,S}$ agrees with P, there is a ground instantiated rule $\sigma(r)$ with conclusion ϕ and premises χ_k $(k\in K)$ such that $\rightarrow_{P,S}\models\chi_k$. As $\phi\notin\rightarrow_{P,S'}$, it cannot be that all premises χ_k $(k\in K)$ hold in $\rightarrow_{P,S'}$. Hence, $\rightarrow_{P,S'}\not\models\chi_{k'}$ for some $k'\in K$. If $\chi_{k'}$ is a positive

literal then $\chi_{k'} \in \bigcup_{0 \leq j'' < j} \rightarrow^P_{ij''} \cup \bigcup_{0 \leq i'' < i} \rightarrow^P_{i''}$ and $\chi_{k'} \notin \rightarrow_{P,S'}$. But this contradicts one of the assumptions that ϕ is minimal.

If $\chi_{k'} \equiv t \overset{a_l}{\not\rightarrow}$ then, for some $t' \in T(\Sigma)$, $t \overset{a}{\rightarrow} t' \in \rightarrow_{P,S'} - \rightarrow_{P,S}$ and $S(t \overset{a}{\rightarrow} t') < i$. But this contradicts the minimality assumption with respect to S.

- Considering $\phi \in \rightarrow_{P,S'} - \rightarrow_{P,S}$ leads to a contradiction in almost the same way as the former case. □

This last lemma allows us to drop the stratification as a subscript in the transition relation $\rightarrow_{P,S}$ associated with a stratifiable TSS P. Further, it provides the following technique to give an operational semantics in Plotkin style when there are negative premises around: define a TSS P and prove with a convenient stratification that P is stratifiable. Then P alone determines the transition relation $\rightarrow_P$ associated with P.

In the remainder of this section we show that if we strengthen the requirements on stratifications, then the transition relation that agrees with P is unique.

Definition 2.17. Let $P = (\Sigma, A, R)$ be a TSS and let $S: Tr(\Sigma, A) \rightarrow \alpha$ for some ordinal α be a stratification of P. S is a *strict stratification of* P if, for every rule

$$r = \frac{\{t_k \overset{a_k}{\rightarrow} t'_k \mid k \in K\} \cup \{t_l \overset{a_l}{\not\rightarrow} \mid l \in L\}}{t \overset{a}{\rightarrow} t'} \in R$$

and every substitution σ, $\sigma(t \overset{a}{\rightarrow} t')$ is in a strictly higher stratum than $\sigma(t_k \overset{a_k}{\rightarrow} t'_k)$ $(k \in K)$ and $\sigma(t_l \overset{a_l}{\rightarrow} t'')$ for $l \in L$ and any $t'' \in T(\Sigma)$. In this case we call P *strictly stratifiable*.

If P is strictly stratifiable then this is equivalent to stating that the literal dependency graph of P contains no infinite path ending in some literal ϕ.

Theorem 2.18. *Let P be a strictly stratifiable TSS. Then the transition relation that is associated with P is the unique relation that agrees with P.*

Proof. Let $P = (\Sigma, A, R)$. Suppose $\rightarrow_1$ is a transition relation that agrees with P. P has a strict stratifications $S: T(\Sigma) \rightarrow \alpha$ for some ordinal α. Let $\rightarrow_{P,S}$ be the transition relation that is associated with P. Assume, in order to generate a contradiction, that $\rightarrow_{P,S} \neq \rightarrow_1$. This implies that there is some literal ϕ such that $\phi \in \rightarrow_{P,S} - \rightarrow_1$ or $\phi \in \rightarrow_1 - \rightarrow_{P,S}$. Assume, furthermore, that ϕ is minimal, i.e. for all $\psi \in (\rightarrow_{P,S} - \rightarrow_1) \cup (\rightarrow_1 - \rightarrow_{P,S})$: $S(\phi) \leqslant S(\psi)$. We consider only one case, namely $\phi \in \rightarrow_{P,S} - \rightarrow_1$. The case where $\phi \in \rightarrow_1 - \rightarrow_{P,S}$ goes in exactly the same way. As $\rightarrow_{P,S}$ agrees with P, there is a rule

$$\frac{\{\chi_k \mid k \in K\}}{\chi} \in R$$

and a substitution $\sigma: V \to T(\Sigma)$ such that $\phi = \sigma(\chi)$, $\to_{P,S} \models \sigma(\chi_k)$ for all $k \in K$. Then, for some $k' \in K$, $\to_1 \not\models \sigma(\chi_{k'})$ because otherwise, as $\to_1$ agrees with P, $\phi \in \to_1$, contradicting the assumption.

If $\sigma(\chi_{k'})$ is a positive literal, then $\sigma(\chi'_k) \in \to_{P,S}$, $\sigma(\chi_{k'}) \notin \to_1$ and $S(\chi_{k'}) < S(\phi)$. This contradicts the minimality of ϕ. If $\sigma(\chi_{k'}) \equiv t \overset{a}{\not\to}$ then, for some $t' \in T(\Sigma)$, $t \overset{a}{\to} t' \in \to_1$, but $t \overset{a}{\to} t' \notin \to_{P,S}$ and $S(t \overset{a}{\to} t') < S(\phi)$. This contradicts the minimality of ϕ as well. □

3. Examples showing the use of stratifications

The techniques of the previous section are introduced for showing that specifications using negative premises define a transition relation in a neat way. Here two examples illustrate the use of these techniques.

Example 3.1. Here the GSOS format is defined. It differs slightly from the GSOS format as given by Bloom et al. [10] because we do not consider a special rule for guarded recursion. Suppose we have a TSS P with signature $\Sigma = (F, r)$, labels A and rules of the form

$$\frac{\{x_k \xrightarrow{a_{kl}} y_{kl} \mid k \in K_1,\ l \in L_1\} \cup \{x_k \overset{a_{kl}}{\not\to} \mid k \in K_2,\ l \in L_2\}}{f(x_1, \ldots, x_{r(f)}) \overset{a}{\to} t},$$

with $f \in F$, $x_1, \ldots, x_{r(f)}, y_{kl}$ pairwise different variables, $K_1, K_2 \subseteq \{1, \ldots, r(f)\}$, L_1, L_2 finite disjoint index sets and $t \in \mathbb{T}(\Sigma)$. There is a unique transition relation that agrees with the rules. This can be seen by giving the strict stratification $S: Tr(\Sigma, A) \to \omega$:

$$S(t \overset{a}{\to} t') = n \quad \text{if } t \text{ contains } n \text{ function names.}$$

S is strict as the source in the conclusion of any rule contains more function names than any source in the premises.

Example 3.2. In [7] a priority operator is defined on process graphs. In [15] an operational definition is given to the priority operator using rules with negative premises. However, the combination of unguarded recursion, the priority operator and renaming [5] gives rise to inconsistencies. Here we show that simple conditions on either the relabelling operator or recursion can circumvent this problem.

We base this example on the rules for $\mathrm{BPA}^{\varepsilon}_{\delta}$ as given in [15] (see rules 1–6 in Table 1). The TSS $P_{\mathrm{prio}} = (\Sigma_{\mathrm{prio}}, A_{\mathrm{prio}}, R_{\mathrm{prio}})$, with $\Sigma_{\mathrm{prio}} = (F_{\mathrm{prio}}, r_{\mathrm{prio}})$, contains constant names a for all $a \in Act$, where Act is a given set of atomic actions. We suppose that there is a "backwardly" well-founded ordering $<$ on Act, which is used to construct a stratification. The signature also contains constant names ε for the *empty process*, and δ representing *inaction*, resembling *NIL* in CCS [20].

Table 1
BPA^{ρ}_{δ} with renaming and priorities

(1)	$a \xrightarrow{a} \varepsilon, \quad a \neq \surd$	(2)	$\varepsilon \overset{\backprime}{\to} \delta$
(3)	$\dfrac{x \xrightarrow{a} x'}{x+y \xrightarrow{a} x'}$	(4)	$\dfrac{y \xrightarrow{a} y'}{x+y \xrightarrow{a} y'}$
(5)	$\dfrac{x \xrightarrow{a} x'}{x \cdot y \xrightarrow{a} x' \cdot y}, \quad a \neq \surd$	(6)	$\dfrac{x \overset{\backprime}{\to} x' \quad y \xrightarrow{a} y'}{x \cdot y \xrightarrow{a} y'}$
(7)	$\dfrac{x \xrightarrow{a} x'}{\rho_f(x) \xrightarrow{f(a)} \rho_f(x')}, \quad a \neq \surd$	(8)	$\dfrac{x \overset{\backprime}{\to} x'}{\rho_f(x) \overset{\backprime}{\to} \rho_f(x')}$
(9)	$\dfrac{x \xrightarrow{a} x' \quad \forall b > a \; x \not\xrightarrow{b}}{\theta(x) \xrightarrow{a} \theta(x')}, \quad a, b \neq \surd$	(10)	$\dfrac{x \overset{\backprime}{\to} x'}{\theta(x) \overset{\backprime}{\to} \theta(x')}$
(11)	$\dfrac{t \xrightarrow{a} x'}{X_t \xrightarrow{a} x'}$ for $X_t \Leftarrow t \in E$		

There is a unary function name θ, the *priority operator*. If x can perform several actions, say $x \xrightarrow{a} x'$ and $x \xrightarrow{b} x''$, then $\theta(x)$ allows only those transitions which are the highest in the ordering $<$. So, if $a > b$ then $\theta(x) \xrightarrow{a} \theta(x')$ is an allowed transition while $\theta(x) \xrightarrow{b} \theta(x'')$ is not possible. We have another unary function name ρ_f, the *renaming operator*. f is a renaming function from Act to Act. $\rho_f(x)$ renames the labels of the transitions of x by f. There are two binary operators. *Sequential composition* is denoted by $\cdot$ (this symbol is usually omitted). *Alternative composition* is denoted by $+$.

For recursion it is assumed that there is some given set Ξ with *process names*. Each name in Ξ is a constant in the signature. E is a set of *process declarations* of the form $X \Leftarrow t_X$ for all process names $X \in \Xi$ ($t_X \in T(\Sigma_{\text{prio}})$). In $X \Leftarrow t_X$ for all process names $X \in \Xi$ ($t_X \in T(\Sigma_{\text{prio}})$). In $X \Leftarrow t_X$, t_X is the *body* of process name X.

The labels in A_{prio} are given by $Act_{\backprime}$ ($= Act \cup \{\surd\}$). $\surd$ is an auxiliary symbol that is introduced to represent termination of a process. The rules are given in Table 1. Here a, b range over $Act_{\backprime}$. In rule 9 of Table 1 we use the abbreviation $\forall b > a \; x \not\xrightarrow{b}$ in the premises. It means that for all $b > a$ there is a premise $x \not\xrightarrow{b}$. As an infinite number of negative premises are allowed in the premises of a rule, rule scheme 9 generates proper rules. With these rules we have the following inconsistency (cf. [6]). Define

$$X \Leftarrow \theta(\rho_f(X) + b),$$

with $f(b) = a, f(a) = c, f(d) = d$ for all $d \in Act - \{a, b\}$ and $a > b$. Now $X \xrightarrow{b} \varepsilon$ iff $X \not\xrightarrow{b}$.

As a first solution for this problem, we consider renaming functions satisfying the requirement that if $a>b$ then not $f(b)=a$ for all $a,b\in Act$, i.e. we may not rename actions to ones with higher priority. It is now easy to see that a transition relation associated with P_{prio} exists using the following stratification of P_{prio}. Define $rk(a)$ for all $a\in A_{\text{prio}}$ by

$$rk(a)=\sup(\{rk(b)+1 \mid a<b\}) \quad \text{for } a\in Act,$$

where $\sup(\emptyset)=0$ and $rk(\surd)=0$. Define $S: Tr(\Sigma_{\text{prio}}, A_{\text{prio}})\to\alpha$ for some ordinal α by

$$S(t \xrightarrow{a} t')=rk(a)$$

(it is straightforward to check that S is a stratification of P_{prio}).

Another solution is to disallow that the priority operator appears in the body of a process name. In this case a stratification can be given by

$$S(t \xrightarrow{a} t')=n,$$

where n is the total number of occurrences of θ's in t.

A last possibility is obtained by disallowing unguarded recursion in the bodies of process definitions. A stratification can now be constructed as follows: Suppose one has a literal $t \xrightarrow{a} t'$. Let n be the number of θ's in t. Moreover, let m be the number of the θ's in the bodies t'' of all process names X'' ($X'' \Leftarrow t_{X''}\in E$) that occur unguarded in t. Then we define a stratification $S: Tr(\Sigma_{\text{prio}}, A_{\text{prio}})\to\omega$ by $S(t \xrightarrow{a} t')=n+m$. One can check that S is a stratification of P_{prio}.

4. The *ntyft/ntyxt* format and the congruence theorem

Often, one considers bisimulation equivalence as the finest extensional equivalence that one wants to impose. If bisimulation is not a congruence then one can distinguish bisimilar processes by putting them in appropriate contexts. Therefore, it is a nice property of a format of rules if it guarantees that all operators defined by this format respect bisimulation.

The notion of strong bisimulation equivalence as defined below is from Park [24].

Definition 4.1. Let $P=(\Sigma, A, R)$ be a stratifiable TSS. A relation $R\subseteq T(\Sigma)\times T(\Sigma)$ is a (*strong*) (*P*-) *bisimulation relation* if it satisfies the following:

(1) Whenever $t\,R\,u$ and $t \xrightarrow{a}_P t'$, then, for some $u'\in T(\Sigma)$, we have $u \xrightarrow{a}_P u'$ and $t'\,R\,u'$.

(2) Conversely, whenever $t\,R\,u$ and $u \xrightarrow{a}_P u'$ then, for some $t'\in T(\Sigma)$, we have $t \xrightarrow{a}_P t'$ and $t'\,R\,u'$.

We say that two terms $t,t'\in T(\Sigma)$ are (*P*-)*bisimilar*, notation $t\leftrightarrows_P t'$, iff there is a P-bisimulation relation R such that $t\,R\,t'$. We write $t\leftrightarrows t'$ if P is clear from the context. Note that $\leftrightarrows_P$ is an equivalence relation.

In the setting of [15], where TSSs without negative premises are considered, the *tyft/tyxt* format is a very general format for which bisimulation is a congruence.

Definition 4.2 (*tyft/tyxt format*; Groote and Vaandrager [15]). Let $\Sigma=(F,r)$ be a signature. Let $P=(\Sigma,A,R)$ be a TSS. A rule $r\in R$ is in *tyft format* if it has the form

$$\frac{\{t_k \xrightarrow{a_k} y_k \mid k\in K\}}{f(x_1,\ldots,x_{r(f)}) \xrightarrow{a} t},$$

with K an index set, y_k, x_i $(1\leqslant i\leqslant r(f))$ all different variables, $a_k, a\in A$, $f\in F$ and $t_k, t\in\mathbb{T}(\Sigma)$. A rule $r\in R$ is in *tyxt format* if it fits

$$\frac{\{t_k \xrightarrow{a_k} y_k \mid k\in K\}}{x \xrightarrow{a} t},$$

with K an index set, y_k, x all different variables, $a_k, a\in A$, $t_k, t\in\mathbb{T}(\Sigma)$. P is in *tyft/tyxt format* if all its rules are either in *tyft* or in *tyxt* format.

A distinctive feature of the *tyft/tyxt* format is that it allows *lookahead*. This means that a variable on the right-hand side of an arrow in the premises can be used again in the premises. An example of lookahead is given by rule 2 in Table 2, where the variable y' is used again in $B^{n-1}(x',y')$.

In [15] the generality of the *tyft/tyxt* format has been shown by counter examples. For instance, the following example shows that the format cannot be extended by allowing more than one function symbol on the left-hand side of the arrow in the conclusion. There are similar examples for all other general extensions of the format (see [15]).

Example 4.3. Consider a TSS with a unary function symbol f, a constant c, one label a and the rule $f(c)\xrightarrow{a}f(c)$. Now $f(f(c))$ is bisimilar to c because both cannot perform any action. But $f(f(f(c)))$ is not bisimilar to $f(c)$ because $f(f(f(c)))$ cannot perform any step, while $f(c)$ can do an a-action. So, bisimulation is not a congruence.

Now we introduce the *ntyft/ntyxt* format as the most general extension of the *tyft/tyxt* format with negative premises such that, for operators defined in this format, bisimulation is again a congruence.

Definition 4.4 (*ntyft/ntyxt format*). Let $\Sigma=(F,r)$ be a signature. Let $P=(\Sigma,A,R)$ be a stratifiable TSS. A rule $r\in R$ is in *ntyft format* if it has the form

$$\frac{\{t_k \xrightarrow{a_k} y_k \mid k\in K\}\cup\{t_l \xrightarrow{b_l} \mid l\in L\}}{f(x_1,\ldots,x_{r(f)}) \xrightarrow{a} t},$$

with K, L index sets, y_k, x_i $(1 \leqslant i \leqslant r(f))$ all different variables, $a_k, b_l, a \in A$, $f \in F$ and $t_k, t_l, t \in \mathbb{T}(\Sigma)$. A rule $r \in R$ is in *ntyxt format* if it fits

$$\frac{\{t_k \xrightarrow{a_k} y_k \mid k \in K\} \cup \{t_l \xrightarrow{b_l} \mid l \in L\}}{x \xrightarrow{a} t},$$

with K, L index sets, y_k, x all different variables, $a_k, b_l, a \in A$ and $t_k, t_l, t \in \mathbb{T}(\Sigma)$. P is in *ntyft format* if all its rules are in *ntyft* format, and P is in *ntyft/ntyxt format* if all its rules are either in *ntyft* or in *ntyxt* format.

In the remainder of this section we show that the congruence theorem holds for the *ntyft/ntyxt* format. In order to do so, we need a similar well-foundedness restriction on the premises of the rules as was necessary to prove the congruence theorem for the *tyft/tyxt* format. It is an open question whether both congruence theorems can be proved without this restriction.

Definition 4.5 (*well founded*). Let $P = (\Sigma, A, R)$ be a TSS. Let $W = \{t_k \xrightarrow{a_k} t'_k \mid k \in K\} \subseteq \mathbb{T}(\Sigma) \times A \times \mathbb{T}(\Sigma)$ be a set of positive literals over Σ and A. The *variable dependency graph* (VDG) of W is a directed (unlabelled) graph with:
- Nodes: $\bigcup_{k \in K} Var(t_k \xrightarrow{a_k} t'_k)$,
- Edges: $\{\langle x, y \rangle \mid x \in Var(t_k),\ y \in Var(t'_k)$ for some $k \in K\}$.

W is called *well founded* if any backward chain of edges in the variable dependency graph is finite. A rule is called *well founded* if its set of positive premises is well founded. A TSS is called *well founded* if its rules are well founded.

Note that it is not useful to include negative premises in this definition as they do not have a target and, therefore, do not determine values of variables.

Example 4.6. The variable dependency graph of $\{f(x', y_1) \xrightarrow{a} y_2,\ g(x, y_2) \xrightarrow{a} y_1\}$ is given in Fig. 3. The set of rules is not well founded because the graph contains a cycle.

Example 4.7. Consider the variable dependency graph G of $\{x_{n+1} \xrightarrow{a_n} x_n \mid n \in \mathbb{N}\}$. G is not well founded because, for any variable x_i $(i \in \mathbb{N})$ that acts as a node in G, there is an infinite path ending in this node. A part of G is depicted in Fig. 4.

The following lemma says that, for well-founded TSSs in *ntyft/ntyxt* format, it is sufficient to consider only target-independent stratifications.

Lemma 4.8. *Let P be a well-founded stratifiable TSS in ntyft/ntxft format. Then P has a target-independent stratification.*

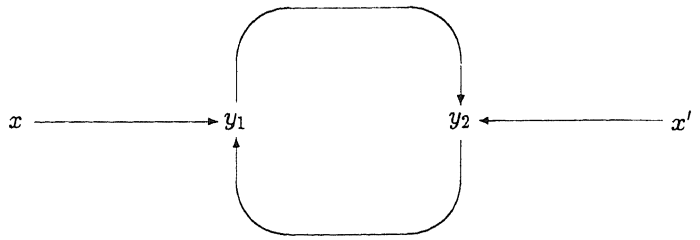

Fig. 3. A VDG with a cycle.

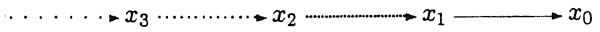

Fig. 4. A VDG that is not well founded.

Proof. Let $P=(\Sigma, A, R)$, with $\Sigma=(F, r)$, and let $S: Tr(\Sigma, A)\to\alpha$ for some ordinal α be a stratification of P. Define a mapping $S': Tr(\Sigma, A)\to\alpha+1$ by

$$S'(t \xrightarrow{a} t')=\sup(\{S(t \xrightarrow{a} u)+1 \mid u\in T(\Sigma)\}).$$

We show that S' is a stratification of P. As S' is clearly target-independent, this is sufficient to finish the proof.

Consider a rule in *ntyxt* format (the argument for a rule in *ntyft* format is exactly the same),

$$r=\frac{\{t_k \xrightarrow{a_k} y_k \mid k\in K\}\cup\{t_l \overset{t_{lv}}{\not\longrightarrow} \mid l\in L\}}{x \xrightarrow{a} t}\in R,$$

and some ground substitution σ. For each positive premise $t_k \xrightarrow{a} y_k$ we have that, for each term $u\in T(\Sigma)$,

$$\begin{aligned} S(\sigma(t_k) \xrightarrow{a} u)) &= S(\sigma'(t_k \xrightarrow{a_k} y_k)) \\ &\leqslant S(\sigma'(x \xrightarrow{a} t)) \\ &= S(\sigma(x) \xrightarrow{a} \sigma'(t)), \end{aligned} \tag{1}$$

where σ' is a ground substitution defined by

$$\sigma'(z)=\begin{cases}\sigma(z) & \text{if } z\not\equiv y_k,\\ u & \text{if } z\equiv y_k.\end{cases}$$

As P is well founded and in *ntyft/ntyxt* format, $\sigma'(t_k)=\sigma(t_k)$ and $\sigma'(x)=\sigma(x)$.

Now it is easy to see that S' is a stratification:

$$S'(\sigma(t_k \xrightarrow{a_k} y_k)) = \sup(\{S(\sigma(t_k) \xrightarrow{a_k} u) + 1 \mid u \in T(\Sigma)\})$$

$$\overset{(1)}{\leqslant} \sup(\{S(\sigma(x) \xrightarrow{a} u') + 1 \mid u' \in T(\Sigma)\})$$

$$= S'(\sigma(x \xrightarrow{a} t))$$

and

$$S'(\sigma(t_l) \xrightarrow{a_l} u)) = \sup(\{S(\sigma(t_l) \xrightarrow{a_l} u') + 1 \mid u' \in T(\Sigma)\})$$

$$\leqslant S(\sigma(x \xrightarrow{a} t))$$

$$< \sup(\{S(\sigma(x) \xrightarrow{a} u'') + 1 \mid u'' \in T(\Sigma)\})$$

$$= S'(\sigma(x \xrightarrow{a} t)). \qquad \square$$

Definition 4.9. Let W be a set of positive literals which is well founded and let G be the variable dependency graph of W. Let $Var(W)$ be the set of variables occurring in literals in W. Define for each $x \in Var(W)$: $n_{\mathrm{VDG}}(x) = \sup(\{n_{\mathrm{VDG}}(y) + 1 \mid \langle y, x \rangle \text{ is an edge of } G\})$ $(\sup(\emptyset) = 0)$.

If W is a set of positive premises of a rule in *ntyft/ntyxt* format then $n_{\mathrm{VDG}}(x) \in \mathbb{N}$ for each $x \in Var(W)$; every variable y_k only occurs once on the right-hand side of a positive literal in the premises. As the term t_k is finite, it contains only a finite number of variables x. Therefore, the set $U = \{n_{\mathrm{VDG}}(x) + 1 \mid \langle x, y_k \rangle \text{ is an edge of } G\}$ is finite. Hence, $n_{\mathrm{VDG}}(y_k) = \sup(U)$ is a natural number.

Definition 4.10. Two stratifiable TSSs $P = (\Sigma, A, R)$ and $P' = (\Sigma', A', R')$ are *transition-equivalent if* $\Sigma = \Sigma'$, $A = A'$ and $\rightarrow_P = \rightarrow_{P'}$.

Lemma 4.11. *Let $P = (\Sigma, A, R)$ be a stratifiable TSS in ntyft/ntyxt format. Then there is a stratifiable TSS $P' = (\Sigma, A, R')$ in ntyft format that is transition-equivalent with P.*

Proof. Let $\Sigma=(F, rank)$. Let R' contain every rule $r\in R$ that is in *ntyft* format together with the rules $\sigma_f(r)$ for every rule $r\in R$ in *ntyxt* format and every function name $f\in F$, where σ_f is defined as

$\sigma_f(x)=f(z_1, \ldots, z_{rank(f)})$ if x is the source in the conclusion of r; $z_1, \ldots, z_{rank(f)}$ are variables that do not occur in r,

$\sigma_f(x)=x$ otherwise.

Note that R' is in *ntyft* format. As P is stratifiable, there is a stratification $S: Tr(\Sigma, A)\to\alpha$ of P. It is not hard to see that this stratification is also a stratification for P'. It is enough to show that $\to_{P,S}=\to_{P',S}$. In order to see this, we only need to prove that $\to_{ij}^{P}=\to_{ij}^{P'}$ for all $0\leqslant i<\alpha$, $0\leqslant i<degree(P)$. This will be done by induction on i and, within this induction, by induction on j.

$\subseteq$: Suppose $\phi\in\to_{ij}^{P}$ for some i and j. According to the definition of $\to_{ij}^{P}$, this means that there is a ground instantiated rule $\sigma(r)$, with conclusion ϕ and premises χ_k $(k\in K)$, such that $\bigcup_{0\leqslant j'<j}\to_{ij'}^{P}\cup\bigcup_{0\leqslant i'<i}\to_{i'}^{P'}\models\chi_k$. If χ_k is positive then, inductively, $\bigcup_{0\leqslant j'<j}\to_{ij'}^{P'}\cup\bigcup_{0\leqslant i'<i}\to_{i'}^{P'}\models\chi_k$. If $\chi_k\equiv t\overset{a}{\not\to}$ then for all $t'\in T(\Sigma)$: $t\overset{a}{\to}t'\notin\bigcup_{0\leqslant i'<i}\to_{i'}^{P}$ and, therefore, $t\overset{a}{\to}t'\notin\bigcup_{0\leqslant i'<i}\to_{i'}^{P'}$. Hence, in both cases $\bigcup_{0\leqslant j'<j}\to_{ij'}^{P'}\cup\bigcup_{0\leqslant i'<i}\to_{i'}^{P'}\models\chi_k$ for all $k\in K$. If r is an *ntyft* rule, one can apply $\sigma(r)$ again to obtain $\phi\in\to_{ij}^{P'}$. If r is in *ntyxt* format and the left-hand side of ϕ is $f(t_1, \ldots, t_{rank(f)})$, apply the instantiated rule $\sigma'(\sigma_f(r))$, where $\sigma'(x)=t_k$ for $x=z_k$ $(1\leqslant k\leqslant rank(f))$ and $\sigma'(x)=\sigma(x)$ otherwise. Hence, $\phi\in\to_{ij}^{P'}$.

$\supseteq$: The reverse implication can be shown in the same way. □

Definition 4.12. Let $P=(\Sigma, A, R)$ be a TSS. Let $r\in R$ be a rule. A variable x is called *free* in r if it occurs in r but not in the source of the conclusion or in the target of a positive premise. The rule r is called *pure* if it is well founded and does not contain free variables. P is called *pure* if all rules in R are pure.

Lemma 4.13. *Let $P=(\Sigma, A, R)$ be a stratifiable and well-founded TSS in ntyft/ntyxt* format. Then there is a stratifiable TSS $P'=(\Sigma, A, R')$ in pure *ntyft/ntyxt* format which is transition-equivalent with P. If P is in *ntyft* format then P' is in pure *ntyft* format.

Proof. R' contains a rule $\sigma(r)$ for every rule $r\in R$ and substitution σ satisfying

$\sigma(x)=t\in T(\Sigma)$ if x is free in r,

$\sigma(x)=x$ otherwise.

Note that P' constructed in this way is pure; if P is in *ntyft* format then P' is also in *ntyft* format and any stratification for P is also a stratification for P'. The remainder of the proof proceeds in the same way as the proof of Lemma 4.11. □

Next we state the congruence theorem.

Theorem 4.14. *Let P be a well-founded stratifiable TSS in ntyft/ntyxt format. Then $\leftrightarrows_P$ is a congruence relation.*

Proof. This proof closely resembles the proof of a similar theorem in [15]. Assume $P=(\Sigma, A, R_0)$, with $\Sigma=(F,r)$. According to Lemmas 4.11 and 4.13, we may assume that P is in pure *ntyft* format. As P is stratifiable, there is a target-independent stratification $S: Tr(\Sigma, A)\to\alpha$ for some ordinal α of P. Furthermore, there is a transition relation $\to_P$ associated with P. We must show that for all $f\in F, u_1, \ldots, u_{r(f)}$, $v_1, \ldots, v_{r(f)}\in T(\Sigma)$:

$$\forall 1\leqslant k\leqslant r(f), \quad u_k \leftrightarrows_P v_k \Rightarrow f(u_1, \ldots, u_{r(f)}) \leftrightarrows_P f(v_1, \ldots, v_{r(f)}).$$

In order to do so, we define a relation $R\subseteq T(\Sigma)\times T(\Sigma)$ as the minimal relation satisfying

(1) $\leftrightarrows_P \subseteq R$,

and, for all function names $f\in F$,

(2) $\forall 1\leqslant k\leqslant r(f)$, $u_k R v_k \Rightarrow f(u_1, \ldots, u_{r(f)}) R f(v_1, \ldots, v_{r(f)})$.

For the relation R, we have the following useful fact.

Fact 1. *Let $t\in\mathbb{T}(\Sigma)$ and let $\sigma, \sigma': V\to T(\Sigma)$ be substitutions such that for all x in $Var(t)$: $\sigma(x) R \sigma'(x)$. Then $\sigma(t) R \sigma'(t)$.*

Proof of Fact 1. Straightforward induction on the structure of t. □

Proof of Theorem 4.14 (*continued*). If we show that R is a bisimulation relation, then it immediately follows that $R=\leftrightarrows_P$ and, consequently, that $\leftrightarrows_P$ is a congruence relation. In order to see that R is a bisimulation relation, we must check that R has the transfer property: if $u R v$ and $u \xrightarrow{a} u'$ then there is a v' with $v \xrightarrow{a} v'$ and $u' R v'$ and vice versa. If $u\leftrightarrows_P v$ then this is trivial. So, suppose $u=f(u_1, \ldots, u_{r(f)})$, $v=f(v_1, \ldots, v_{r(f)})$ and $u_k R v_k$ for $1\leqslant k\leqslant r(f)$. We are ready if we have shown (by induction on β) that the following holds for all β:

If $\mathscr{L}(f(u_1, \ldots, u_{r(f)}), a)+\mathscr{L}(f(v_1, \ldots, v_{r(f)}), a)=\beta$ then

- *$f(u_1, \ldots, u_{r(f)}) \xrightarrow{a} u'\in\to_P$ and $u_k R v_k$ for $1\leqslant k\leqslant r(f)$ implies $\exists v'$ s.t. $f(v_1, \ldots, v_{r(f)}) \xrightarrow{a} v'\in\to_P$ and $u' R v'$,*
- *vice versa.*

Here we define $\mathscr{L}(t,a)=S(t,a,t')$ for some $t'\in T(\Sigma)$. The definition of $\mathscr{L}(t,a)$ is correct because S is target-independent. As the induction hypothesis is symmetric, we need only check one half of it. Suppose the induction hypothesis holds for all $\beta'<\beta$. The validity of the induction hypothesis for β follows immediately if the following fact

holds for all $1 \leqslant i < \alpha$ and $1 \leqslant j \leqslant degree(P)$:

$$\text{If } \mathscr{L}(f(u_1, \ldots, u_{r(f)}), a) + \mathscr{L}(f(v_1, \ldots, v_{r(f)}), a) = \beta,$$

$$f(u_1, \ldots, u_{r(f)}) \xrightarrow{a} u' \in \rightarrow_{ij}^{P} \text{ and } u_k R v_k \quad \text{for } 1 \leqslant k \leqslant r(f),$$

$$\text{then } \exists v' \text{ s.t. } f(v_1, \ldots, v_{r(f)}) \xrightarrow{a} v' \in \rightarrow_P \text{ and } u' R v'.$$

We prove this statement by induction on i and, within that, by induction on j. So, suppose the second induction hypothesis holds for $i' < i$ or for $i' = i$ if $j' < j$. Assume $\mathscr{L}(u, a) + \mathscr{L}(v, a) = \beta$ and $u \xrightarrow{a} u' \in \rightarrow_{ij}^{P}$. As $\rightarrow_P$ agrees with P, there is a rule

$$r = \frac{\{t_k \xrightarrow{a_k} y_k \mid k \in K\} \cup \{t_l \xrightarrow{a_l} \mid l \in L\}}{f(x_1, \ldots, x_{r(f)}) \xrightarrow{a} t} \in R_0$$

and a substitution σ such that

- $\sigma(f(x_1, \ldots, x_{r(f)})) = u$,
- $\sigma(x_i) = u_i$ for $1 \leqslant i \leqslant r(f)$,
- $\sigma(t) = u'$,
- $\rightarrow_P \models \sigma(t_k \xrightarrow{a_k} y_k)$ and $\rightarrow_P \models \sigma(t_l) \xrightarrow{a_l}$.

We will use rule r again in order to show that, for some v', $v \xrightarrow{a} v' \in \rightarrow_P$ and $u' R v'$. Consider the VDG G of the positive premises of r. With induction on n, we show that the following fact holds for all n.

Fact 2. There is a ground substitution σ' such that, for any $x \in nodes(G)$, with $n_{\mathrm{VDG}}(x) < n$, $\sigma(x) R \sigma'(x)$; if $x = y_k$ for some $k \in K$ then $\sigma'(t_k \xrightarrow{a} y_k) \in \rightarrow_P$ and if $x = x_i$ then $\sigma'(x_i) = v_i$.

Proof of Fact 2. Suppose x is a node of G, with $n_{\mathrm{VDG}}(x) = n$, and the claim holds for $n' < n$. As r is pure, there are two cases:

- $x = x_i$ $(1 \leqslant i \leqslant r(f))$. In this case the claim holds for n as $\sigma(x) = u_i R v_i = \sigma'(x)$.
- $x = y_k$ $(k \in K)$ and $t_k \xrightarrow{a} y_k$ is a premise of r. By induction, it holds that there is a ground substitution σ' such that for all $y \in Var(t_k)$: $\sigma(y) R \sigma'(y)$. By Fact 1, $\sigma(t_k) R \sigma'(t_k)$. Now distinguish between the two cases:
  (1) $\sigma(t_k) \leftrightarrows_P \sigma'(t_k)$. In this case there is a $w \in T(\Sigma)$ such that $\sigma'(t_k) \xrightarrow{a_k} w \in \rightarrow_P$ and $\sigma(y_k) R w$.
  (2) There is a function name g in F and there are terms $w_{k'}$, $w'_{k'}$ for $1 \leqslant k' \leqslant r(g)$ such that

$$\sigma(t_k) = g(w_1, \ldots, w_{r(g)}),$$

$$\sigma'(t_k) = g(w'_1, \ldots, w'_{r(g)}),$$

$$w_j R w'_j \quad \text{for } 1 \leqslant j \leqslant r(g).$$

Furthermore, we know that $\mathscr{L}(\sigma(t_k), a_k) + \mathscr{L}(\sigma'(t_k), a_k) \leqslant \mathscr{L}(u, a) + \mathscr{L}(v, a)$. Also $\sigma(t_k \xrightarrow{a_k} y_k) \in \bigcup_{i' < i} \rightarrow_{i'}^{P} \cup \bigcup_{j' < j} \rightarrow_{ij'}^{P}$. Now we can apply the first or second induction hypothesis, which gives that there is a w such that $g(w'_1, \ldots, w'_{r(g)}) \xrightarrow{a_k} w \in \rightarrow_P$ and $\sigma(y_k) R w$.

So, for any x with $n_{\mathrm{VDG}}(x) = n$, we can find a w_x such that $\sigma(x) R w_x$. Define a ground substitution σ'' such that $\sigma''(x') = \sigma'(x')$ if $n_{\mathrm{VDG}}(x') \neq n$ and $\sigma''(x') = w_{x'}$ if $n_{\mathrm{VDG}}(x') = n$. Clearly, all inductive properties hold for σ''. □

Proof of Theorem 4.14 (*conclusion*). Now the proof of the theorem can be finished. For all positive premises ϕ of r, it follows that we can prove that $\sigma'(\phi) \in \rightarrow_P$ for some ground substitution σ' satisfying the properties of Fact 2. We show that, for each negative premise $t_l \overset{a_{ij}}{\not\rightarrow}$ also hold in $\rightarrow_P$. We know, using Fact 1, that $\sigma(t_l) R \sigma'(t_l)$ because $\sigma(x) R \sigma'(x)$ for all variables x in t_l. By definition of R, there are two possibilities:

- $\sigma(t_l) \leftrightarrows_P \sigma'(t_l)$. In this case $\sigma'(t_l) \overset{a_{ij}}{\not\rightarrow}$ clearly holds in $\rightarrow_P$.
- $\sigma(t_l) = g(w_1, \ldots, w_{r(g)})$ and $\sigma'(t_l) = g(w'_1, \ldots, w'_{r(g)})$, $g \in F$ and $w_i R w'_i$ $(1 \leqslant i \leqslant r(g))$. In order to arrive at a contradiction, we assume that, for some $w \in T(\Sigma)$, $\sigma'(t_l) \xrightarrow{a_k} w$. Clearly, $\mathscr{L}(\sigma(t_l), a_l) + \mathscr{L}(\sigma'(t_l), a_l) < \mathscr{L}(u, a) + \mathscr{L}(v, a)$. So, by applying the first induction hypothesis, we know that $\exists w'$ s.t. $\sigma(t_l) \xrightarrow{a_l} w'$. But this contradicts that $\sigma(t_l) \overset{a_{ij}}{\not\rightarrow}$ holds in $\rightarrow_P$. So, for every negative premise $t_l \overset{a_{ij}}{\not\rightarrow}$ of r: $\rightarrow_P \models \sigma'(t_l) \overset{a_{ij}}{\not\rightarrow}$.

Now, as all premises of $\sigma'(r)$ hold, we may conclude that $\sigma'(f(x_1, \ldots, x_{r(f)}) \xrightarrow{a} t) \in \rightarrow_P$. Define $v' = \sigma'(t)$. For all $x \in Var(t)$: $\sigma(x) R \sigma'(x)$. By an application of Fact 1, it follows that $\sigma(t) R \sigma'(t)$ or, equivalently, $u' R v'$. This completes the induction step for the second induction hypothesis. □

5. Modular properties of TSSs

Sometimes one wants to extend a TSS with new function and constants. Therefore, the *sum* of two TSSs is introduced [15]. The combination of two TSSs P_0 and P_1 is denoted by $P_0 \oplus P_1$, where we generally assume that P_1 is the extension of P_0. With negative premises, care is needed to guarantee that $P_0 \oplus P_1$ still defines a transition relation.

If P_1 is added to P_0 ($P_0 = (\Sigma_0, A_0, R_0)$), it would be nice if all literals with source $t \in T(\Sigma_0)$ in $\rightarrow_{P_0 \oplus P_1}$ are exactly the literals in $\rightarrow_{P_0}$. In this case we say that $P_0 \oplus P_1$ is a *conservative extension* of P_0.

Definition 5.1. Let $\Sigma_i = (F_i, r_i)$ $(i = 0, 1)$ be two signatures such that $f \in F_0 \cap F_1 \Rightarrow r_0(f) = r_1(f)$. The *sum* of Σ_0 and Σ_1, notation $\Sigma_0 \oplus \Sigma_1$, is the signature

$$\Sigma_0 \oplus \Sigma_1 = (F_0 \cup F_1, \lambda f. \text{ if } f \in F_0 \text{ then } r_0(f) \text{ else } r_1(f)).$$

Definition 5.2. Let $P_i=(\Sigma_i, A_i, R_i)$ $(i=0, 1)$ be two TSSs with $\Sigma_0 \oplus \Sigma_1$ defined. The *sum* of P_0 and P_1, notation $P_0 \oplus P_1$, is the TSS

$$P_0 \oplus P_1 = (\Sigma_0 \oplus \Sigma_1, A_0 \cup A_1, R_0 \cup R_1).$$

Definition 5.3. Let $P_i=(\Sigma_i, A_i, R_i)$ $(i=0, 1)$ be two TSSs with $P=P_0 \oplus P_1$ defined. Let $P=(\Sigma, A, R)$. We say that P is a *conservative extension* of P_0 and that P_1 *can be added conservatively* to P_0 if $P_0 \oplus P_1$ is stratifiable and for all $t \in T(\Sigma_0)$, $a \in A$ and $t' \in T(\Sigma)$:

$$t \xrightarrow{a} t' \in \rightarrow_P \Leftrightarrow t \xrightarrow{a} t' \in \rightarrow_{P_0}.$$

Remark 5.4. If $P_0 \oplus P_1 = (\Sigma, A, R)$ is a conservative extension of $P_0=(\Sigma_0, A_0, R_0)$, then it follows immediately that for all $t, u \in T(\Sigma_0)$: $t \leftrightarrows_{P_0} u \Leftrightarrow t \leftrightarrows_{P_0 \oplus P_1} u$.

The following theorem gives the conditions under which a TSS P_1 can be added conservatively to P_2. The theorem is the same as the one that holds for TSSs without negative premises [15], except for the constraint that $P_0 \oplus P_1$ is stratifiable. By an example, it will be shown that this condition is necessary. That the other conditions cannot be weakened is shown in [15].

Theorem 5.5. *Let $P_0=(\Sigma_0, A_0, R_0)$ be a TSS in pure ntyft/ntyxt format and let $P_1=(\Sigma_1, A_1, R_1)$ be a TSS in ntyft format such that there is no rule in R_1 containing a function name from Σ_0 in the source of its conclusion. Let $P=P_0 \oplus P_1$ be defined and stratifiable. Then P_1 can be added conservatively to P_0.*

Proof. Let $P=(\Sigma, A, R)$. As P is stratifiable, there is a stratification $S: Tr(\Sigma, A) \rightarrow \alpha$ for some ordinal α for P. Define $S^0: Tr(\Sigma_0, A_0) \rightarrow \alpha$ by $S^0(\phi)=S(\phi)$. It is not hard to check that S^0 is a stratification of P_0. Hence, $\rightarrow_P$ and $\rightarrow_{P_0}$ are the transition relations associated with P and P_0, respectively.

It is sufficient to prove that

$$t \in T(\Sigma_0),\ a \in A_0,\ t \xrightarrow{a} t' \in \rightarrow_P \Leftrightarrow t \xrightarrow{a} t' \in \rightarrow_{P_0},\ t' \in T(\Sigma_0).$$

This is done by induction on the ordinal β $(0 \leqslant \beta < \alpha)$, with $S(t \xrightarrow{a} t') = S^0(t \xrightarrow{a} t') = \beta$.

Assume that the induction hypothesis holds for all $\beta' < \beta$.

$\Rightarrow$: Suppose $t \xrightarrow{a} t' \in \rightarrow^P_{\beta j}$ for some j. Here $\rightarrow^P_{\beta j}$ is the relation from Definition 2.14 to construct $\rightarrow_P$. By induction on j, it is shown that

$$t \in T(\Sigma_0),\ a \in A_0,\ t \xrightarrow{a}{}^P_{\beta j} t' \Rightarrow t \xrightarrow{a} t' \in \rightarrow_{P_0},\ t' \in T(\Sigma_0).$$

As $\rightarrow_P$ agrees with P, there is a rule $r \in R$ with conclusion $u \xrightarrow{a} u'$ and a substitution $\sigma: V \rightarrow T(\Sigma)$ such that $\sigma(u)=t$, $\sigma(u')=t'$, $r \notin R_1$ as all rules in R_1 are in *ntyft* format, containing function names not occurring in Σ_0 on the left-hand side of their conclusions. So, $r \in R_0$. In the remainder we will only deal with the case that r is in *ntyft*

format. The case that r is in *ntyxt* format goes in the same way. So, assume r is equal to $(u=f(x_1,\ldots,x_{r(f)}))$:

$$\frac{\{s_k \xrightarrow{a_k} y_k \mid k\in K\}\cup\{u_l \xrightarrow{b_l}\!\!\!\!\!\!/\ \mid l\in L\}}{f(x_1,\ldots,x_{r(f)}) \xrightarrow{a} u'}.$$

Now we use induction on $n_{\mathrm{VDG}}(x)$ of the variable dependency graph G of the premises of r to prove that for all $x\in Var(r)$: $\sigma(x)\in\Sigma_0$ and if $x=y_k$ $(k\in K)$ then $\sigma(s_k \xrightarrow{a_k} y_k)\in\rightarrow_{P_0}$. Suppose $n_{\mathrm{VDG}}(x)=n\in\mathbb{N}$. As P_0 is pure, we distinguish two cases:

- $x=x_i$ $(1\leqslant i\leqslant r(f))$. As $t\in T(\Sigma_0)$, $\sigma(x)\in T(\Sigma_0)$.
- $x=y_k$ $(k\in K)$ and $s_k \xrightarrow{a_k} y_k$ is a positive premise of r. By induction, we know that, for all $y\in Var(s_k)$, $\sigma(y)\in T(\Sigma_0)$. As $r\in R_0$, $\sigma(s_k)\in T(\Sigma_0)$. By induction and $\sigma(s_k \xrightarrow{a_k} y_k)\in\rightarrow_{P_0\oplus P_1}$, we can derive $\sigma(s_k \xrightarrow{a_k} y_k)\in\rightarrow_{P_0}$ and $\sigma(y_k)\in T(\Sigma_0)$.

As a consequence of this inductive proof, it holds for all positive premises ϕ of r that $\sigma(\phi)\in\rightarrow_{P_0}$. For a negative premise $u_l \xrightarrow{b_l}\!\!\!\!\!\!/$, we assume, in order to generate a contradiction that $\exists u_l'\in T(\Sigma_0)$, $\sigma(u_l \xrightarrow{b_l} u_l')\in\rightarrow_{P_0}$. As $\sigma(u_l \xrightarrow{b_l} u_l')$ is in a strictly lower stratum than $t \xrightarrow{a} t'$ in S^0, it follows, by induction, that $\sigma(u_l \xrightarrow{b_l} u_l')\in\rightarrow_P$. This contradicts $\sigma(u_l)\xrightarrow{b_l}\!\!\!\!\!\!/$.

As $\rightarrow_{P_0}$ agrees with P_0 and all premises of $\sigma(r)$ hold in $\rightarrow_{P_0}$, it follows that $\sigma(u \xrightarrow{a} u')$ also holds in $\rightarrow_{P_0}$. As for all variables in $Var(r)$, $\sigma(r)\in T(\Sigma_0)$, it also holds that $\sigma(u')\in T(\Sigma_0)$.

$\Leftarrow$: This case has the same structure as the proof of $\Rightarrow$. Take as an intermediate induction hypothesis

$$t \xrightarrow{a} t' \in \rightarrow_{P_0} \;\Rightarrow\; t \xrightarrow{a} t' \in \rightarrow_P.$$

We skip the details but we remark that induction on n_{VDG} is not necessary. From the induction hypothesis it follows that

$$t \xrightarrow{a} t' \in \rightarrow_{P_0} \;\Rightarrow\; t \xrightarrow{a} t' \in \rightarrow_P, \quad t\in T(\Sigma_0),\ a\in A_0.$$

After the combination of this result with $\Rightarrow$ the outermost induction step is proved. From this the theorem follows immediately. □

In the remainder of this section we study how we can combine stratifications of two stratifiable TSSs P_0 and P_1 to a stratification of $P_0\oplus P_1$. The following examples show that, in general, the sum of two stratifiable TSSs is not stratifiable.

Example 5.6. This example shows that, under certain circumstances, it can even be dangerous to extend the signature of a TSS. Let P_0 be a TSS with unary function name f, a label a and a rule

$$\frac{f(x) \overset{a}{\not\rightarrow}}{f(x) \overset{a}{\rightarrow} f(x)}.$$

This TSS is stratifiable as there are no ground instances of literals. Adding a TSS P_1 that only contains the single constant c already leads to an inconsistency. If $\rightarrow$ is a relation that agrees with $P_0 \oplus P_1$ then $\rightarrow \models f(c) \overset{a}{\rightarrow} f(c)$ iff $\rightarrow \models f(c) \overset{a}{\not\rightarrow}$.

Example 5.7. This is a less trivial example that shows a problem that can occur when stratifying the sum of stratifiable TSSs. Let P_0 consist of a unary function name g, a constant δ, labels a, b and a rule

$$\frac{x \overset{b}{\not\rightarrow}}{g(x) \overset{a}{\rightarrow} \delta}.$$

P_1 consists of unary function names g and f, constant δ, labels a, b and a rule

$$\frac{g(f(x)) \overset{a}{\rightarrow} y}{f(x) \overset{b}{\rightarrow} \delta}.$$

Both P_0 and P_1 have an associated transition relation. $P_0 \oplus P_1$, however, makes it possible to show that $f(\delta) \overset{b}{\rightarrow} \delta$ iff $f(\delta) \overset{b}{\not\rightarrow}$ for any transition relation $\rightarrow$ agreeing with $P_0 \oplus P_1$. In Fig. 5 the dependency graph of $P_0 \oplus P_1$ is drawn. The negative edge comes from P_0 and the positive edge from P_1, together constituting a cycle with a negative edge.

Checking the stratifiability of the sum of two stratifiable TSSs can be done by giving a stratification for $P_0 \oplus P_1$. Sometimes the following theorem is helpful.

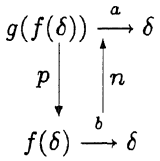

Fig. 5. The LDG belonging to Example 5.7.

Theorem 5.8. *Let* $\Sigma_0=(F_0,\bar{r}_0)$ *and* $\Sigma_1=(F_1,\bar{r}_1)$ *be signatures such that for some constants* a_0,a_1: $a_0\in F_0$ *and* $a_1\in F_1$. *Let* $P_0=(\Sigma_0,A_0,R_0)$, $P_1=(\Sigma_1,A_1,R_1)$ *be stratifiable TSSs. Let* $\Sigma_0\oplus\Sigma_1$ *be defined. If, for all ground substitutions* σ_0 *and* σ_1 *and rules* $r_0\in R_0$ *and* $r_1\in R_1$,

- ϕ *is the conclusion of* r_1,
- ψ *is a positive premise of* r_0, *or* $\psi=t\xrightarrow{a}t'$ *and* $t\overset{a}{\not\to}$ *is a negative premise of* r_0 *and*
- $\sigma_0(\psi)\neq\sigma_1(\phi)$,

then $P_0\oplus P_1$ *is a stratifiable TSS.*

Proof. Assume that P_0 has stratification $S^0: Tr(\Sigma_0,A_0)\to\alpha_0$ and that P_1 has stratification $S^1: Tr(\Sigma_1,A_1)\to\alpha_1$. Construct a stratification S for $P_0\oplus P_1$ as follows: define $U\subseteq Tr(\Sigma_0\oplus\Sigma_1,A_0\cup A_1)$ as the set of all literals that fit a premise of a rule $r_0\in R_0$. If literal $\phi\in U$ then construct a literal $\bar{\phi}$ be replacing all subterms $f(\bar{u})$, for $f\in F_1$, in ϕ by a_0. As the label of ϕ is in A_0, $\bar{\phi}\in Tr(\Sigma_0,A_0)$ and, thus, $\bar{\phi}$ occurs in a stratum β in S^0. Define $S(\phi)=\beta$.

Assume $\phi\notin U$. If the label of ϕ is not in A_1 then $S(\phi)=\alpha_0$. If the label of ϕ is in A_1 then construct $\bar{\phi}$ from ϕ by replacing every subterm $f(\bar{u})$ in ϕ, with $f\in\Sigma_0$, by a_1. Now $\bar{\phi}\in Tr(\Sigma_1,A_1)$. So, it must hold that $\bar{\phi}$ is in a stratum β in S^1. Define $S(\phi)=\alpha_0+\beta$. Now every literal $\phi\in Tr(\Sigma_0\oplus\Sigma_1,A_0\cup A_1)$ has a place in S.

We now check that S is a stratification of $P_0\oplus P_1$. Take a rule $r\in R_0\oplus R_1$. Suppose σ is a ground substitution and ψ is the conclusion, ϕ a positive premise (if present in $\sigma(r)$) and $t\overset{a}{\not\to}$ a negative premise (also if present) of $\sigma(r)$. We proceed by case analysis.

- $\psi\in U$. By the condition in this theorem, ψ is not an instance of a conclusion in a rule R_1 and, thus, $r\in R_0$. Hence, for all $t'\in T(\Sigma_0\oplus\Sigma_1)$: $\phi,t\xrightarrow{a}t'\in U$. ϕ,ψ and $t\xrightarrow{a}t'$ are related in S in the same way as $\bar{\phi},\bar{\psi}$ and $t\xrightarrow{a}t'$ are related in S^0. As $\bar{\phi},\bar{\psi}$ and $t\xrightarrow{a}t'$ are also instances of r for some σ', they satisfy the conditions for a proper stratification in S_0 and, therefore, ϕ,ψ and $t\xrightarrow{a}t'$ satisfy these conditions in S.
- $\psi\notin U$.
  - If ψ has a label $a\notin A_1$ then r cannot be a rule of R_1 and, so, $r\in R_0$. As ϕ and $t\xrightarrow{a}t'$ (for all t') are elements of U, ψ is in a strictly higher stratum than all its premises. Hence, r satisfies the stratification condition in this case.
  - If ψ has a label in A_1 then $\psi\in S_{\alpha_0+\beta}$ if $\bar{\psi}$ is in stratum S^1_β. If $\phi\in U$ then ϕ is in a strictly lower stratum than ψ and if $t\xrightarrow{a}t'\in U$ then $t\xrightarrow{a}t'$ is in a strictly lower stratum than ψ. If $\phi\notin U$ and $\bar{\phi}\in S^1_\gamma$, then $S(\phi)=\alpha_0+\gamma$ as the label of ϕ comes from A_1. If $t\xrightarrow{a}t'\notin U$ and $t\xrightarrow{a}t'\in S^1_{\gamma_{t'}}$ then $S(t\xrightarrow{a}t')=\alpha_0+\gamma_{t'}$ because $a\in A_1$. Now, as $\bar{\psi},\bar{\phi}$ and $t\xrightarrow{a}t'$ are all instances of r for some substitution σ', $\gamma\leqslant\beta$ and $\gamma_{t'}<\beta$. Hence, ψ is in an equal or higher stratum than ϕ in S and $t\xrightarrow{a}t'$ is in a strictly lower stratum than ψ. This shows that also in the last case the stratifiability condition for r is satisfied. □

6. The trace congruence generated by the *ntyft/ntyxt* format

In this section we show that if we define operators using the pure *ntyft/ntyxt* format, then for image-finite processes the trace congruence generated by this format is exactly (strong) bisimulation equivalence. First we give the definition of a trace congruence generated by a format and the definition of image-finite processes. In Fig. 6 we show how we will then prove our result. The arrows denote set inclusion and "IF" indicates that we need image finiteness.

Definition 6.1. Let $P=(\Sigma, A, R)$ be a stratifiable TSS and let $\rightarrow_P$ be the transition relation associated with P. Let $t\in T(\Sigma)$. A sequence $a_1 * \cdots * a_n \in A^*$ is a *(P-)trace* from t iff there are terms $t_1, \ldots, t_n \in T(\Sigma)$ for some $n\in\mathbb{N}$ such that $t \xrightarrow{a_1}_P t_1 \xrightarrow{a_2}_P \cdots \xrightarrow{a_n}_P t_n$. $Tr(t)$ is the set of all P-traces from t. Two process terms $t, t'\in T(\Sigma)$ are *trace-equivalent with respect to* P iff $Tr(t)=Tr(t')$. This is also denoted as $t \equiv^{\mathrm{T}}_P t'$.

Note that if two terms t and t' are bisimilar, then they are also trace-equivalent.

Definition 6.2. Let $\mathcal{F}$ be some format of TSS rules. Let $P=(\Sigma, A, R)$ be a stratifiable TSS in $\mathcal{F}$ format. Two terms $t, t'\in T(\Sigma)$ are *trace-congruent with respect to* $\mathcal{F}$ rules, notation $t \equiv^{\mathrm{T}}_{\mathcal{F}} t'$, iff for every TSS $P'=(\Sigma', A', R')$ in $\mathcal{F}$ format which can be added conservatively to P and for every $\Sigma \oplus \Sigma'$ context $C[\,]$: $C[t] \equiv^{\mathrm{T}}_{P\oplus P'} C[t']$.

Definition 6.3. Let $P=(\Sigma, A, R)$ be a stratifiable TSS. Let $\rightarrow_P$ be the transition relation associated with P. $\rightarrow_P$ is called *image-finite* iff, for all $t\in T(\Sigma)$ and $a\in A$, the set $\{u \mid t \xrightarrow{a}_P u\}$ is finite.

Definition 6.4. Let $P=(\Sigma, A, R)$ be a stratifiable TSS with associated transition relation $\rightarrow_P$. The relations $\leftrightarrows^n_P \subseteq T(\Sigma)\times T(\Sigma)$ for $n\in\mathbb{N}$ are inductively defined by

- $\leftrightarrows^0_P = T(\Sigma)\times T(\Sigma)$,
- $\leftrightarrows^{n+1}_P = \{(t,u) \mid$
  - $t \xrightarrow{a} t' \Rightarrow \exists u'$ s.t. $u \xrightarrow{a} u'$ and $t' \leftrightarrows^n_P u'$
  - $u \xrightarrow{a} u' \Rightarrow \exists t'$ s.t. $t \xrightarrow{a} t'$ and $t' \leftrightarrows^n_P u'\}$.

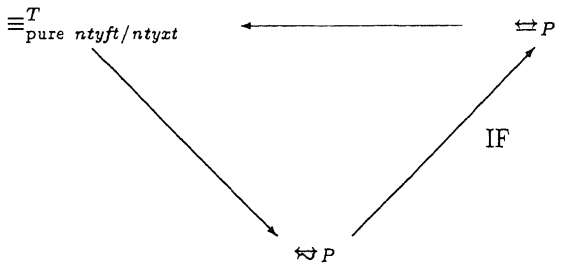

Fig. 6. Inclusions among several process equivalences.

Two process expressions $t, t' \in T(\Sigma)$ are *n-bounded bisimilar* (*for* P) iff $t \leftrightarrows_P^n t'$. Two terms $t, t' \in T(\Sigma)$ are *bounded bisimilar for* P, notation $t \leftrightarrows_P t'$, iff, for all $n \in \mathbb{N}$, $t \leftrightarrows_P^n t'$.

The following lemma gives a condition under which bounded bisimilar states are bisimilar.

Lemma 6.5. *Let* $P=(\Sigma, A, R)$ *be a stratifiable TSS such that* $\rightarrow_P$ *is image-finite. Let* $t, u \in T(\Sigma)$. *Then*

$$t \leftrightarrows_P u \Leftrightarrow t \leftrightarrow_P u.$$

Proof. $\Leftarrow$ is trivial. See [14] for $\Rightarrow$. □

We now give the basic definitions and lemmas to prove that $\equiv^{\mathrm{T}}_{\mathrm{pure\ ntyft/ntyxt}} \subseteq \leftrightarrows_P$. The main component is the following test system. We show that this test system is stratifiable and that it can test equality between n-bounded bisimilar processes.

Definition 6.6. Let $P=(\Sigma, A, R)$ be a TSS. The *bisimulation tester of* P, $P_{\mathrm{T}}=(\Sigma_{\mathrm{T}}, A_{\mathrm{T}}, R_{\mathrm{T}})$, is a TSS with signature $\Sigma_{\mathrm{T}}=(F_{\mathrm{T}}, r_{\mathrm{T}})$ containing binary function names B^n and Q_a^n for all $n \in \mathbb{N}$, $a \in A$ and a constant δ. The labels of P_{T} are $A_{\mathrm{T}}=A \cup \{ok, yes, no\}$. The rules in R_{T} are given in Table 2.

The rules in Table 2 are based on the following meaning of the transitions $\xrightarrow{yes}$, $\xrightarrow{no}$ and $\xrightarrow{ok}$:

- $B^n(x, y) \xrightarrow{yes} \delta$ if x and y are n-bounded bisimilar.
- $B^n(x, y) \xrightarrow{no} \delta$ $(n>0)$ if x can perform a step that cannot be done by y such that the results are $(n-1)$-bounded bisimilar.
- $Q_a^n(x, y) \xrightarrow{ok} \delta$ $(n>0)$ means that y can perform an a-step such that the result is $(n-1)$-bounded bisimilar with x.

Table 2
A bisimulation tester

$B^0(x, y) \xrightarrow{yes} \delta$		(1)
$\dfrac{y \xrightarrow{a} y' \qquad B^{n-1}(x', y') \xrightarrow{yes} z}{Q_a^n(x', y) \xrightarrow{ok} \delta}$	for $n>0$, $a \in A$	(2)
$\dfrac{x \xrightarrow{a} x' \qquad Q_a^n(x', y) \overset{ok}{\not\rightarrow}}{B^n(x, y) \xrightarrow{no} \delta}$	for $n>0$, $a \in A$	(3)
$\dfrac{B^n(x, y) \overset{no}{\not\rightarrow} \qquad B^n(y, x) \overset{no}{\not\rightarrow}}{B^n(x, y) \xrightarrow{yes} \delta}$	for $n>0$	(4)

The rules in Table 2 just encode n-bounded bisimilarity. The negative premises model the universal quantifiers in Definition 6.4.

Remark 6.7. The test system P_T is able to test equivalences between terms $t, u \in T(\Sigma)$. However, it cannot test processes over $T(\Sigma \oplus \Sigma_T)$. The reason for this is that in rules 2 and 3 of Table 2 $a \neq ok, yes, no$. If a would be allowed range over $A \cup \{ok, yes, no\}$, then it is impossible to give a stratification, as done in this paper.

Lemma 6.8. *Let $P=(\Sigma, A, R)$ be a TSS. Let P_T be the bisimulation tester of P. P_T is stratifiable.*

Proof. It is enough to show that P has a stratification. Construct a mapping $S: Tr(\Sigma_T, A_T) \to \omega$ such that

- for all $a \in A$ and $t, t' \in T(\Sigma_T)$, $S(t \xrightarrow{a} t') = 1$,
- for $n \in \mathbb{N}$ and $t, u, v \in T(\Sigma_T)$, $S(B^n(t,u) \xrightarrow{yes} v) = 2n+1$,
- for $n \in \mathbb{N} - \{0\}$, $a \in A_T$ and $t, u, v \in T(\Sigma_T)$, $S(Q_a^n(t,u) \xrightarrow{ok} v) = 2n-1$,
- for $n \in \mathbb{N} - \{0\}$ and $t, u, v \in T(\Sigma_T)$, $S(B^n(t,u) \xrightarrow{no} v) = 2n$.

It is straightforward to check that S is a stratification for P_T. □

Lemma 6.9. *Let $P=(\Sigma, A, R)$ be a stratifiable TSS in pure ntyft/ntyxt format containing at least one constant in its signature. Furthermore, A does not contain the labels ok, no, yes, and Σ does not contain function names B^n and Q_a^n for all $a \in A$, $n \in \mathbb{N}$. Let $t, u \in T(\Sigma)$; then*

$$B^n(t,u) \xrightarrow{yes} \delta \in \to_{P \oplus P_T} \Leftrightarrow t \leftrightarrows_P^n u.$$

Proof. As $yes, no, ok \notin A$, conclusions of rules in R_T never fit a premise of rules in R. Furthermore, P and P_T are stratifiable and contain at least one constant in their signatures. Hence, by Theorem 5.8, $P \oplus P_T$ is stratifiable. So, $P \oplus P_T$ has an associated transition relation $\to_{P \oplus P_T}$. As a consequence of Theorem 5.5, $P \oplus P_T$ is a conservative extension of P.

$\Rightarrow$ Use induction on n.

Base case. For $n=0$, $t \leftrightarrows_P^n u$ for any $t, u \in T(\Sigma)$. Hence, the theorem holds in this case.

Induction. We have to show that (1) if $B^{n+1}(t,u) \xrightarrow{yes} \delta \in \to_{P \oplus P_T}$ and $t \xrightarrow{a} t' \in \to_P$ then $\exists u'$ s.t. $u \xrightarrow{a} u' \in \to_P$ and $t' \leftrightarrows_P^n u'$, and vice versa, (2) if $u \xrightarrow{a} u' \in \to_P$ then $\exists t'$ s.t. $t \xrightarrow{a} t' \in \to_P$ and $t' \leftrightarrows_P^n u'$. As $B^{n+1}(t,u) \xrightarrow{yes} \delta \in \to_{P \oplus P_T}$ and $\to_{P \oplus P_T}$ agrees with $P \oplus P_T$, it must be the case that, using rule 4, $B^{n+1}(t,u) \xrightarrow{no}\!\!\!\!\!\!\rightarrow$ and $B^{n+1}(u,t) \xrightarrow{no}\!\!\!\!\!\!\rightarrow$ hold in $\to_{P \oplus P_T}$. Therefore, it cannot be the case that the premises of rule 3 all hold with $\sigma(x)=t$, $\sigma(y)=u$. But we know that $t \xrightarrow{a} t' \in \to_P$ and, by conservativity, also $t \xrightarrow{a} t' \in \to_{P \oplus P_T}$. Hence, for some v, $Q_a^{n+1}(t',u) \xrightarrow{ok} v \in \to_{P \oplus P_T}$. But then the premises of rule 2 must be

true with $\sigma(y)=u$ and $\sigma(x')=t'$. Hence, for some u', $u \xrightarrow{a} u' \in \rightarrow_{P \oplus P_T}$ and $B^n(t',u') \xrightarrow{yes} \delta \in \rightarrow_{P \oplus P_T}$. By conservativity, $u \xrightarrow{a} u' \in \rightarrow_P$. With the induction hypothesis, $t' \leftrightarrows_P^n u'$. We can show (2) in the same way. Hence, if $B^{n+1}(t,u) \xrightarrow{yes} \delta \in \rightarrow_{P \oplus P_T}$ then $t \leftrightarrows_P^{n+1} u$.

$\Leftarrow$ Again, we use induction on n.

Base case. If $n=0$, the theorem is trivial as $B^0(t,u) \xrightarrow{yes} \delta \in \rightarrow_{P \oplus P_T}$ for all $t,u \in T(\Sigma)$.

Induction. Suppose $t \leftrightarrows_P^{n+1} u$. We will show that $B^{n+1}(t,u) \xrightarrow{yes} \in \rightarrow_{P \oplus P_T}$. By rule 4, it is sufficient to show that $B^{n+1}(t,u) \overset{no}{\not\rightarrow}$ and $B^{n+1}(u,t) \overset{no}{\not\rightarrow}$ hold in $\rightarrow_{P \oplus P_T}$. This means that we have to show that rule 3 can never be applied, i.e. either (3) $t \xrightarrow{a} t'$ or $Q_a^{n+1}(t',u) \overset{ok}{\not\rightarrow}$ nor (4) $u \xrightarrow{a} u'$ or $Q_a^{n+1}(u',t) \overset{ok}{\not\rightarrow}$ for any $a \in A$ holds in $\rightarrow_{P \oplus P_T}$. Suppose, for some $a \in A$, $t \overset{a}{\not\rightarrow}$ holds in $\rightarrow_{P \oplus P_T}$. Then (3) trivially does not hold. Now suppose $t \xrightarrow{a} t' \in \rightarrow_{P \oplus P_T}$ for some t'. As P_T extends P conservatively, $t \xrightarrow{a} t' \in \rightarrow_P$. Then, using $t \leftrightarrows_P^{n+1} u$, $\exists u' \in T(\Sigma)\, u \xrightarrow{a} u' \in \rightarrow_P$ and $t' \leftrightarrows_P^n u'$. By conservativity, $u \xrightarrow{a} u' \in \rightarrow_{P \oplus P_T}$. Using the induction hypothesis, $B^n(t',u') \xrightarrow{yes} \delta \in \rightarrow_{P \oplus P_T}$. Applying rule 2 yields $Q_a^{n+1}(t',u) \xrightarrow{ok} \delta \in \rightarrow_{P \oplus P_T}$ and, hence, $Q_a^{n+1}(t',u) \overset{ok}{\not\rightarrow}$ does not hold in $\rightarrow_{P \oplus P_T}$. We can prove (4) in the same way. □

The following theorem relates all notions.

Theorem 6.10. *Let $P=(\Sigma, A, R)$ be a stratifiable TSS in pure ntyft/ntyxt format containing at least one constant in its signature. Furthermore, $\rightarrow_P$ is image-finite, A does not contain labels ok, no, yes and Σ does not contain function names B^n, Q_a^n for all $a \in A$, $n \in \mathbb{N}$:*

$$t \equiv^{\mathrm{T}}_{\text{pure ntyft/ntyxt}} u \Leftrightarrow t \leftrightarrows_P u \Leftrightarrow t \leftrightarrow_P u.$$

Proof. Suppose $t \leftrightarrow_P u$. Let $P'=(\Sigma', A', R')$ be a TSS in pure *ntyft/ntyxt* format such that $P \oplus P'$ is a conservative extension of P. Then $t \leftrightarrow_{P \oplus P'} u$. By the congruence theorem, for any $\Sigma \oplus \Sigma'$ context C, $C[t] \leftrightarrow_{P \oplus P'} C[u]$. Hence, $t \equiv^{\mathrm{T}}_{\text{pure ntyft/ntyxt}} u$.

Suppose $t \leftrightarrows_P u$. This means that, for some $n \in \mathbb{N}$, $t \not\leftrightarrows_P^n u$. Construct the context $B^n(t,[\,])$. Now, by Lemma 6.9, $B^n(t,u) \xrightarrow{yes} \not\rightarrow$ holds in $\rightarrow_{P \oplus P_T}$ while $B^n(t,t) \xrightarrow{yes} \delta \in \rightarrow_{P \oplus P_T}$. Hence, $t \not\equiv^{\mathrm{T}}_{\text{pure ntyft/ntyxt}} u$ or, in other words, $t \equiv^{\mathrm{T}}_{\text{pure ntyft/ntyxt}} u \Rightarrow t \leftrightarrows_P u$.

The last case $t \leftrightarrows_P u \Rightarrow t \leftrightarrow_P u$ follows directly from Lemma 6.5. □

The condition that $ok, no, yes \notin A$ and B^n, Q_a^n are not in Σ is not a real restriction. It can be circumvented by simply renaming labels and function names. The requirement that Σ contains at least one constant is also natural: without such a constant there are no terms $t \in T(\Sigma)$ and, hence, a bisimulation tester would not be useful.

The bisimulation tester uses an infinite number of function names. For every $n \in \mathbb{N}$ and $a \in A$, there are binary operators B^n and Q_a^n. It is natural to ask whether a test system with a finite number of binary operators can be formulated. Here such a test

system is given. This test system has an additional property that if the number of labels in a tested system is finite, then there are only a finite number of rules necessary.

Definition 6.11. Let $P=(\Sigma, A, R)$ be a TSS with a countable set of labels A. Assume that there is a function $n: A \to \mathbb{N}$ that gives a unique number for each label, satisfying that if, for $a \in A$, $n(a)=m>0$ then $\exists b \in A$ s.t. $n(b)=m-1$. The *finite bisimulation tester* $P_{FT}=(\Sigma_{FT}, A_{FT}, R_{FT})$ contains constants 0, 1 and δ, unary function names S and S_0, a ternary function name B and a quaternary function name Q. The labels in P_{FT} are given by $A_{FT}=A \cup \bar{A} \cup \{ok, yes, no, 0, 1\}$. Here $\bar{A}=\{\bar{a} \mid a \in A\}$. The definition of n is extended to $\bar{A}$ by $n(\bar{a})=n(a)$. The rules in R_{FT} are given in Table 3. Here $l, l', n, n', x, x', y, y'$ are variables. a ranges over A, and b, c range over $\bar{A}$. $S_0^{n(a)}(1)$ is an abbreviation for $n(a)$ applications of S_0 to 1.

The main difference between P_T and P_{FT} is that labels and numbers do not occur any more as subscripts and superscripts at Q and B, but they are coded by zeroes and successor functions and included in the list of arguments. We have the same results for

Table 3
A finite bisimulation tester

$$0 \xrightarrow{0} \delta \tag{1}$$

$$S(x) \xrightarrow{1} x \tag{2}$$

$$1 \xrightarrow{b} \delta \quad \text{for } n(b)=0 \tag{3}$$

$$\frac{x \xrightarrow{b} x'}{S_0(x) \xrightarrow{c} \delta} \quad \text{if } n(c)=n(b)+1 \tag{4}$$

$$\frac{n \xrightarrow{0} n'}{B(n,x,y) \xrightarrow{yes} \delta} \tag{5}$$

$$\frac{l \xrightarrow{a} l' \qquad y \xrightarrow{a} y' \qquad B(n,x',y') \xrightarrow{yes} z}{Q(n,l,x',y) \xrightarrow{ok} \delta} \quad \text{for } a \in A \tag{6}$$

$$\frac{n \xrightarrow{1} n' \qquad x \xrightarrow{a} x' \qquad Q(n', S_0^{n(a)}(1), x', y) \overset{ok}{\not\longrightarrow}}{B(n,x,y) \xrightarrow{no} \delta} \quad \text{for } n>0,\ a \in A \tag{7}$$

$$\frac{B(n,x,y) \overset{no}{\not\longrightarrow} \qquad B(n,y,x) \overset{no}{\not\longrightarrow}}{B(n,x,y) \xrightarrow{yes} \delta} \quad \text{for } n>0 \tag{8}$$

P_{FT} as for P_T. We only give here the main lemmas and we omit the proofs. With these results, it can be shown in exactly the same way as in the proof of Theorem 6.10 that P_{FT} is also powerful enough to distinguish between nonbisimilar processes.

Lemma 6.12. *Let $P=(\Sigma, A, R)$ be a TSS with a countable set of labels A. The finite bisimulation tester P_{FT} of P is stratifiable.*

Lemma 6.13. *Let $P=(\Sigma, A, R)$ be a stratifiable TSS in pure ntyft/ntyxt format with a countable set of labels A not containing labels yes, no, ok, 0, 1 and at least one constant in Σ. Function names $0, S, 1, S_0, B, Q$ must not occur in Σ. Let $t, u \in T(\Sigma)$. $S^n(0)$ is an abbreviation for n applications of S on 0. Then*

$$B(S^n(0), t, u) \xrightarrow{yes}_{P \oplus P_{FT}} \delta \iff t \leftrightarrows_P^n u.$$

Remark 6.14. There are two other bisimulation testers proposed in the literature [1, 11]. Both testers have an operational definition using "global testing", a feature to explore all possible outgoing transitions of a term. The definitions of these testers do not exploit lookahead, and negative premises are used only in a nonessential way.

It is not surprising that the distinguishing power of the *ntyft/ntyxt* format is as strong as global testing. With "lookahead" we can model existential quantification. For instance, the premise of rule 2 in Table 4 can be read as: there exists a y' reachable via an a-step such that y' is $(n-1)$-bounded bisimilar to x'. By also using negation, via the negative premises, universal quantification, i.e. global testing, can be modelled.

There is a second major difference. In [1, 11] Hennessy–Milner formulas (HM-formulas, [16]) are used as an auxiliary device to construct the bisimulation testers. The following elementary fact about Hennessy–Milner logic is used [16]:

$$t \leftrightarrows_P u \text{ iff } \begin{cases} \text{for all HM-formulas } \phi, \\ t \models \phi \iff u \models \phi. \end{cases} \tag{2}$$

Table 4
A Hennessy–Milner formula tester

$$T_T(x) \xrightarrow{ok} \delta \qquad \frac{x \xrightarrow{a} x' \qquad T_\phi(x') \xrightarrow{ok} y}{T_{\langle a \rangle \phi}(x) \xrightarrow{ok} \delta}$$

$$\frac{T_\phi(x) \overset{ok}{\not\longrightarrow}}{T_{\neg\phi}(x) \xrightarrow{ok} \delta} \qquad \frac{T_{\phi_1}(x) \xrightarrow{ok} y_1 \qquad T_{\phi_2}(x) \xrightarrow{ok} y_2}{T_{\phi_1 \wedge \phi_2}(x) \xrightarrow{ok} \delta}$$

Slightly simplifying the results in [1, 11] one can say that a tester T_ϕ is defined, with ϕ an HM-formula, such that

$$T_\phi(t) \xrightarrow{a_1} \cdots \xrightarrow{a_n} \quad \text{iff } t \models \phi,$$

where $a_1 \ldots a_n$ is a particular sequence of actions. If two terms t, u are not bounded bisimilar, then, by (2), there is some HM-formula ϕ such that

$$T_\phi(t) \models \phi \quad \text{and} \quad T_\phi(u) \not\models \phi.$$

Hence, $T_\phi(t) \xrightarrow{a_1} \cdots \xrightarrow{a_n}$, but not $T_\phi(u) \xrightarrow{a_1} \cdots \xrightarrow{a_n}$, and, thus, t and u are not (completed) trace-congruent. Using the *ntyft/ntyxt* format, it is easy to define such an HM-tester P_{HM} (see Table 4 for its rules). The tester T_ϕ has the property that $T_\phi(t) \xrightarrow{ok} \delta$ iff $t \models \phi$. In the same way as the bisimulation tester P_{T} is added to a stratified TSS, P_{HM} can be added also. So, P_{HM} can also be used to distinguish between nonbisimilar processes. It may be worth noting that the HM-tester P_{HM} contains infinitely many function symbols. It is rather standard to reduce these to a finite number (see e.g. [11, 15] and the finite bisimulation tester in Table 3) and we leave this to the reader. In the bisimulation testers in [1, 11] there is no rule for negation, which cannot easily be dealt with in that setting, but, instead, rules are given for F, $\vee$ and $[a]$. Testing an HM-formula $[a]\phi$ is, of course, done by global testing. As is well known, the HM-formula $[a]\phi$ can be expressed using negation and $\langle a \rangle$ as follows:

$$[a]\phi = \neg\langle a \rangle \neg \phi.$$

It is illustrative to see how $\neg\langle a \rangle \neg \phi$ is tested using the following instantiations of the rules of our HM-formula tester:

$$\frac{T_\phi(x) \overset{ok}{\not\rightarrow}}{T_{\neg\phi}(x) \xrightarrow{ok} \delta},$$

$$\frac{x \xrightarrow{a} \qquad T_{\neg\phi}(x) \xrightarrow{ok} y}{T_{\langle a \rangle \neg\phi}(x) \xrightarrow{ok} \delta},$$

$$\frac{T_{\langle a \rangle \neg\phi}(x) \overset{ok}{\not\rightarrow}}{T_{\neg\langle a \rangle \neg\phi}(x) \xrightarrow{ok} \delta}.$$

Or, in other words, a term t satisfies $[a]\phi$ iff it is not the case that t can do an a-step, or, for every t' reachable via an a-step from t, ϕ does (not not) hold in t'.

7. An overview of trace and completed-trace congruences

There are nowadays several different formats of rules for describing a Plotkin-style operational semantics. All these formats induce their own trace and completed-trace congruences. In Table 5 we give an overview of the main results. We will not explicitly define all equivalence notions, but we will confine ourselves to giving references. The first column describes the different formats for the rules. The pure *ntyft/ntyxt* format is the most extensive. All other formats are restricted versions of the pure *ntyft/ntyxt* format. The pure *tyft/tyxt* format [15] can be obtained from the pure *ntyft/ntyxt* format by not allowing negative premises in the rules. The GSOS format [10] has been defined in Example 3.1. It is a simplification of the pure *ntyft* format in the sense that rules in the GSOS format only have conclusions of the form $f(x_1, \ldots, x_{r(f)}) \xrightarrow{a} t$ and premises of the form $x_i \xrightarrow{a_i} x_i'$ for $1 \leqslant i \leqslant r(f)$ and $x_j \xrightarrow{b_j}\!\!\!\!\!\!/\;\;$ for $1 \leqslant j \leqslant r(f)$. In Example 3.1 it has been shown that a TSS in GSOS format has a unique associated transition relation.

The positive GSOS format [15] is almost equal to the GSOS format, the only difference being that rules in the positive GSOS format do not have negative premises. A typical example of a rule in the positive GSOS format is

$$\frac{x \xrightarrow{a} x_1' \qquad x \xrightarrow{b} x_2'}{f(x) \xrightarrow{c} g(x, x_1', x_1')}.$$

One can clearly see that variables may be used more than once in the source of the premises or the target of the conclusion. This is called *copying* [1]. The positive GSOS format is not only more restricted than the GSOS format, but also every rule satisfying the positive GSOS format is in the pure *tyft/tyxt* format (see Fig. 1).

The oldest format is the de Simone format [30]. It is equal to the positive GSOS format except that it does not allow copying. Every variable on the left-hand side of the conclusion may only occur once on the right-hand side of the conclusion or on the left-hand side of a premise. Every variable on the right-hand side of a premise may appear only once on the right-hand side of the conclusion.

Table 5
An overview of (completed) trace congruences

	Trace congruence	Completed-trace congruence
de Simone format	Trace equivalence	Failure equivalence
Positive GSOS format	Simulation equivalence	2/3 Bisimulation
GSOS format	2/3 Bisimulation	2/3 Bisimulation
Pure *tyft/tyxt* format	Simulation equivalence	2-Nested simulation equivalence
Pure *ntyft/ntyxt* format	Bisimulation	Bisimulation

The second and third columns of Table 5 give the trace and completed-trace congruences belonging to these formats. The notion of completed-trace congruences was not yet defined.

Definition 7.1. Let $P=(\Sigma, A, R)$ be a TSS with associated transition relation $\rightarrow_P$. Let $t\in T(\Sigma)$. t is a *deadlocked process*, notation $t\not\rightarrow$, iff there are no $u\in(\Sigma)$ and $a\in A$ with $t \xrightarrow{a}_P u$. A sequence $a_1 * \cdots * a_n \in A^*$ is a *completed trace* of t iff there are process terms $t_1, \ldots, t_n \in T(\Sigma)$ such that $t \xrightarrow{a_1}_P t_1 \xrightarrow{a_2}_P \cdots \xrightarrow{a_n}_P t_n \not\rightarrow$. $CT(t)$ is the set of all completed traces of t. Two process terms $t, u\in T(\Sigma)$ are *completed-trace-equivalent for* P if $CT(t)=CT(u)$. This is denoted as $t \equiv_P^{CT} u$.

The notion of *completed-trace-congruence* can be obtained by replacing "trace" by "completed trace", $\equiv_{\mathcal{F}}^{T}$ by $\equiv_{\mathcal{F}}^{CT}$ and $\equiv_P^{T}$ by $\equiv_P^{CT}$ in Definition 6.2.

The trace and completed-trace congruences for the de Simone format follow directly from an important result of R. de Simone [30]: All operators definable in the de Simone format can also be defined using *architectural expressions* over MEIJE–SCCS. It is a well-known result that trace equivalence is a congruence in MEIJE–SCCS. From this it follows immediately that the trace congruence is trace equivalence. Furthermore, an established result is that the completed-trace congruence is failure trace equivalence. For all other results, we refer to [15], where all completed-trace congruences, except for the pure *ntyft/ntyxt* format, are given. The notion of 2/3-bisimulation was first mentioned in [19] and simulation equivalence and 2-nested simulation equivalence are defined in [15]. The trace congruences for positive GSOS and GSOS are not published anywhere. However, with the help of the lemmas in [15] one can prove the results. In [15] it is shown that the trace congruence for the pure *tyft/tyxt* format is simulation equivalence.

Acknowledgment

I thank Jos Baeten, Roland Bol, Joachim Parrow, Frits Vaandrager, Fer-Jan de Vries and the referees for their helpful comments.

References

[1] S. Abramsky, Observation equivalence as a testing equivalence, *Theoret. Comput. Sci.* **53** (1987) 225–241.
[2] K.R. Apt, Logic programming, in: J. van Leeuwen, ed., *Handbook of Theoretical Computer Science, Vol. B, Formal Models and Semantics*, Chapter 10 (North-Holland, Amsterdam, 1990) 495–574.
[3] E. Astesiano, C. Bendix Nielsen, N. Botta, A. Fantechi, A. Giovini, P. Inverardi, E. Karlsen, F. Mazzanti, G. Reggio and E. Zucca, *The Trial Definition of Ada, Deliverable 7 of the CEC MAP Project: The Draft Formal Definition of ANSI/MIL-STD 1815 Ada*, 1986.
[4] D. Austry and G. Boudol, Algèbre de processus et synchronisations, *Theoret. Comput. Sci.* **30** (1984) 91–131.
[5] J.C.M. Baeten and J.A. Bergstra, Global renaming operators in concrete process algebra, *Inform. and Comput.* **78** (1988) 205–245.

[6] J.C.M. Baeten and J.A. Bergstra, Processen en procesexpressies, *Informatie* **30** (1988) 177–248, in Dutch.

[7] J.C.M. Baeten, J.A. Bergstra and J.W. Klop, Syntax and defining equations for an interrupt mechanism in process algebra, *Fund. Inform.* **IX**(2) (1986) 127–168.

[8] J.A. Bergstra, Put and get, primitives for synchronous unreliable message passing, Logic Group Preprint Series No. 3, CIF, State University of Utrecht, 1985.

[9] G. Berry and G. Gonthier, The synchronous programming language ESTEREL: design, semantics, implementation, *Sci. Comput. Programming* **19** (1992) 83–152.

[10] B. Bloom, S. Istrail and A.R. Meyer, Bisimulation can't be traced: preliminary report, in: *Proc. 15th ACM Symp. on Principles of Programming Languages*, San Diego, California (1988) 229–239.

[11] B. Bloom, S. Istrail and A.R. Meyer, Bisimulation can't be traced. Tech. Report 90-1150, Department of Computer Science, Cornell University, Ithaca, 1990.

[12] R.N. Bol and J.F. Groote, The meaning of negative premises in transition system specifications, Report CS-R9054, CWI, Amsterdam, 1990; extended abstract in: J. Leach Albert et al., eds., *Proc. 18th ICALP* (1991) 481–494.

[13] R. Cleaveland and M. Hennessy, Priorities in process algebra, in: *Proc. 3rd Ann. Symp. on Logic in Computer Science*, Edinburgh (IEEE Computer Soc. Press, Silver Spring, MD, 1988) 193–202.

[14] R.J. van Glabbeek, Bounded nondeterminism and the approximation induction principle in process algebra, in: F.J. Brandenburg, G. Vidal-Naquet and M. Wirsing, eds., *Proc. STACS '87*, Lecture Notes in Computer Science, Vol. 247 (Springer, Berlin, 1987) 336–347.

[15] J.F. Groote ad F.W. Vaandrager, Structured operational semantics and bisimulation as a congruence (extended abstract), in: G. Ausiello, M. Dezani-Ciancaglini and S. Ronchi Della Rocca, eds., *Proc. 16th ICALP*, Stresa, Lecture Notes in Computer Science, Vol. 372 (Springer, Berlin, 1989); full version: *Inform. and Comput.* **100** (1992) 202–260.

[16] M. Hennessy and R. Milner, Algebraic laws for nondeterminism and concurrency, *J. ACM* **32** (1985) 137–161.

[17] ISO, Information processing systems – open systems interconnection – LOTOS – a formal description technique based on the temporal ordering of observational behaviour ISO/TC97/SC21/N DIS8807, 1987.

[18] K.G. Larsen, Modal specifications, Tech. Report R 89-09, Institute for Electronic Systems, The University of Aalborg, 1989.

[19] K.G. Larsen and A. Skou, Bisimulation through probabilistic testing, in: *Proc. 16th ACM Symp. on Principles of Programming Languages*, Austin, Texas (1989) 344–352.

[20] R. Milner, *A Calculus of Communicating Systems*, Lecture Notes in Computer Science, Vol. 92 (Springer, Berlin, 1980).

[21] R. Milner, Calculi for synchrony and asynchrony, *Theoret. Comput. Sci.* **25** (1983) 267–310.

[22] R. Milner, *Communication and Concurrency* (Prentice-Hall, Englewood Cliffs, NJ, 1989).

[23] F. Moller, Axioms for concurrency, Ph.D. Thesis, Report CST-59-89, Department of Computer Science, University of Edinburgh, 1989.

[24] D.M.R. Park, Concurrency and automata on infinite sequences, in: P. Deussen, ed., *Proc. 5th GI Conf.*, Lecture Notes in Computer Science, Vol. 104 (Springer, Berlin, 1981) 167–183.

[25] I.C.C. Phillips, CCS with broadcast stability, unpublished manuscript.

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

[27] G.D. Plotkin, An operational semantics for CSP, in: D. Bjørner, ed., *Proc. IFIP TC2 Working Conf. on Formal Description of Programming Concepts – II*, Garmisch (North-Holland, Amsterdam, 1983) 199–225.

[28] A. Pnueli, Linear and branching structures in the semantics and logics of reactive systems, in: W. Brauer, ed., *Proc. 12th ICALP*, Nafplion, Lecture Notes in Computer Science, Vol. 194 (Springer, Berlin, 1985) 15–32.

[29] T.C. Przymusinski, On the declarative semantics of deductive databases and logic programs, in: J. Minker, ed., *Foundations of Deductive Databases and Logic Programming* (Morgan Kaufmann, Los Altos, CA, 1987) 193–216.

[30] R. de Simone, Higher-level synchronising devices in MEIJE-SCCS, *Theoret. Comput. Sci.* **37** (1985) 245–267.