Centrum voor Wiskunde en Informatica

Centre for Mathematics and Computer Science

J.F. Groote

Transition system specifications with negative premises

Computer Science/Department of Software Technology Report CS-R8950 December

1989

Centrum voor Wiskunde en Informatica
Centre for Mathematics and Computer Science

J.F. Groote

Transition system specifications with negative premises

Computer Science/Department of Software Technology Report CS-R8950 December

The Centre for Mathematics and Computer Science is a research institute of the Stichting Mathematisch Centrum, which was founded on February 11, 1946, as a nonprofit institution aiming at the promotion of mathematics, computer science, and their applications. It is sponsored by the Dutch Government through the Netherlands Organization for the Advancement of Research (N.W.O.).

Transition System Specifications with Negative Premises

Jan Friso Groote
Centre for Mathematics and Computer Science
P.O. Box 4079, 1009 AB Amsterdam, The Netherlands

In this article the general approach to Plotkin style operational semantics of [12] is extended to Transition System Specifications (TSS's) 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.

Key Words and Phrases: Structured Operational Semantics (SOS), stratification, negative conditions, TSS's, compositionality, labeled transition systems, bisimulation, congruence, *ntyft/ntyxt*-format, modularity of transition system specifications, full abstraction.
1985 Mathematical Subject Classification: 68Q05, 68Q55.
1980 Mathematical Subject Classification: 68B10
1982 CR Categories: D.3.1, F.1.1, F.3.2, F.4.3.
Note: The research of the author was partly supported by RACE project no. 1046, Specification and Programming Environment for Communication Software (SPECS).

1. Introduction

In recent years, many process calculi, programming languages and specification languages are provided with an operational semantics in Plotkin style [22, 23]. We mention CCS [16, 18], SCCS [17], ACP [11], MEIJE [4], Esterel [8], LOTOS [13] and Ada [3].

In [12] an operational semantics in Plotkin style is defined by a TSS (Transition System Specification). Basically, a TSS consists of three components. A *signature* defines 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 of process terms. 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 [12] 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 any more then $D(p)$ may signal deadlock. We find deadlock detectors described in this way in [14, 21]. Deadlock detection also is used in sequencing processes. If in $p \cdot q$ (process p sequenced with q) p *cannot* do anything, q may start. See for instance [19] or [9], 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

Report CS-R8950
Centre for Mathematics and Computer Science
P.O. Box 4079, 1009 AB Amsterdam, The Netherlands

can only perform action *a* if it *cannot* perform any activity with higher priority. Descriptions of priorities with negative premises can be found in [10, 12].

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 [24] defines an operator in this way. Also the *put* and *get* primitives of BERGSTRA [7] 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 positive premises. But then there are many auxiliary transitions that do not correspond to *positive* activity of the system that is modeled or specified. 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 TSS's.

A format of rules that allows negative premises is the GSOS-format of BLOOM, ISTRAIL & MEYER [9]. All operators mentioned above can be defined in this format. The GSOS-format, however, is incompatible with the (pure) *tyft*/*tyxt*-format [12] that allows *lookahead* and no negative premises. Many useful operators definable in the *tyft*/*tyxt*-format cannot be defined using the GSOS format.

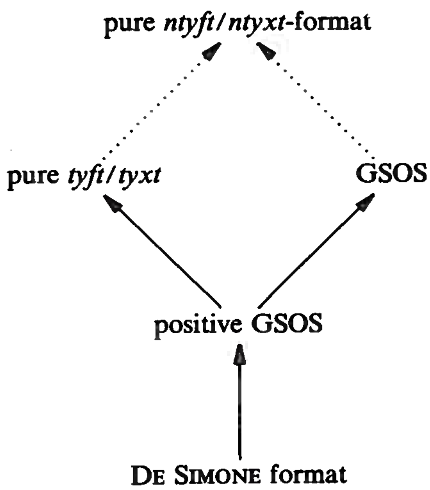

FIGURE 1

The situation is described by the black arrows in figure 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 [26] which was already defined by R. de Simone in 1984. 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 pure *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 arrows in figure 1.

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

1. It is possible to give an inconsistent set of rules. This means that one can derive with 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.

2. 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 an easy method of checking whether a transition relation is consistent. This method is based on the *local stratifications* [2,25] that are used in logic programming. The other problem is solved by formulating an explicit definition of the transition relation. We argue that our choice is a very natural one.

Furthermore, general properties of the *ntyft/ntyxt*-format are studied. It is shown that bisimulation is a congruence for this format. Then, in section 6 we define the *sum* of two TSS's and we prove a theorem stating the most general conditions under which a TSS can be added *conservatively* to another TSS.

In [12] 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] and [9]. 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.

ACKNOWLEDGEMENTS. I thank Frits Vaandrager, Jos Baeten and Fer-Jan de Vries for their helpful comments.

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 TSS's 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 $x,y,z \cdots$.

2.1. DEFINITION. A *(single sorted) signature* is a structure $\Sigma=(F,r)$ where:

- F is a set of *function names* disjoint with V,
- $r: F \rightarrow \mathbb{N}$ 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 $f(t_1, \ldots, t_{r(f)}) \in T(\Sigma, W)$.

$T(\Sigma, \varnothing)$ is abbreviated by $T(\Sigma)$; elements from $T(\Sigma)$ are called *closed* or *ground (process) 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 $V \rightarrow \mathbb{T}(\Sigma)$. A substitution σ is extended to a mapping $\sigma: \mathbb{T}(\Sigma) \rightarrow \mathbb{T}(\Sigma)$ in a standard way by the following definition:

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

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

2.2. DEFINITION. 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:

$$\frac{\{t_k \xrightarrow{a_k} t_k' \mid k\in K\} \cup \{t_l \stackrel{b_l}{\not\rightarrow} \mid l\in L\}}{t \xrightarrow{a} t'}$$

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. $t\stackrel{a}{\not\rightarrow}$ 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 $\frac{\varnothing}{t\xrightarrow{a}t'}$ is often written as $t\xrightarrow{a}t'$. The notions 'substitution', 'Var' and 'closed' extend to literals and rules as expected.

Note that this definition differs from the definition of a TSS in [12] because it allows an infinite number of premises and some premises may now be negative. The purpose of a TSS is to define a *transition relation* $\longrightarrow\subseteq Tr(\Sigma,A)=T(\Sigma)\times A\times T(\Sigma)$. A transition relation states under what actions closed terms over the signature can evolve into one another. This expresses the operational behavior 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 $\longrightarrow$, notation $\longrightarrow\models\psi$, if $\psi\in\longrightarrow$. A negative literal $t\stackrel{a}{\not\rightarrow}$ *holds* in $\longrightarrow$, notation $\longrightarrow\models t\stackrel{a}{\not\rightarrow}$, if for no $t'\in T(\Sigma)$ $t\xrightarrow{a}t'\in\longrightarrow$.

2.3. For TSS's 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 finite proof tree where the rules of the TSS P are used as inference rules, are in the transition relation associated with P. For TSS's with negative premises these proof trees cannot be used. It is not so obvious which transition relation should be associated with a TSS. In [9] BLOOM, ISTRAIL and MEYER require that a transition relation *agrees with* a TSS. We think that this should at least be the case. We repeat their definition here, using our own notation.

2.3.1. DEFINITION. Let $P=(\Sigma,A,R)$ be a TSS. Let $\longrightarrow\subseteq Tr(\Sigma,A)$ be a transition relation. $\longrightarrow$ *agrees with P* if:

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

$$\sigma(\chi)=\psi \text{ and } \forall k\in K:\ \longrightarrow\models\sigma(\chi_k).$$

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 [9].

2.3.2. EXAMPLE. It is possible to give a TSS P that does not define a transition relation. Let P consist of one constant f, one label a and one rule

$$\frac{f\stackrel{a}{\not\rightarrow}}{f\xrightarrow{a}f}.$$

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

2.3.3. Example. 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 as only rule:

$$\frac{f \xrightarrow{a} f}{f \xrightarrow{a} f}.$$

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

2.3.4. Example. 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:

$$\frac{x \xrightarrow{a} y \quad y \xrightarrow{a} z}{f(x) \xrightarrow{a} \delta},$$

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

$$a \xrightarrow{a} g(f(a)).$$

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.

2.4. In this section we will develop a condition on TSS's 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 visualized 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 labeled by 'p' to express that ψ is the conclusion and ϕ a positive premise of $\sigma(r)$ for some closed substitution σ and rule $r \in R$. An edge from $t \xrightarrow{a} t'$ to ψ is labeled with 'n' if ψ is the conclusion of $\sigma(r)$ and $t \stackrel{a}{\not\rightarrow}$ 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 $\longrightarrow$ agreeing with P, that $t \xrightarrow{a} t'$ must be an element of $\longrightarrow$, which is a contradiction. As an example a part of the literal dependency graph of example 2.3.4 is depicted in figure 2.

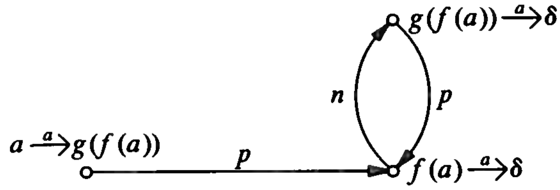

Figure 2

2.4.1. Definition. Let $P=(\Sigma,A,R)$ be a TSS. The *(labeled) 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 closed 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 closed substitution such that there is a rule $r \in R$ with a negative premise $t \stackrel{b}{\not\rightarrow}$ 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 labeled 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 TSS's 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.

2.4.2. DEFINITION. Let P be a TSS. P is *stratifiable* if 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 chains ending in ϕ.

2.4.3. DEFINITION. Let P be a stratifiable TSS with a literal dependency graph G. Nodes that have no incoming chains containing a negative edge are called *LDG basic nodes*. Furthermore, ρ is an 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 a LDG basic node iff ψ is a 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 a 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 X. Define $rank_P(\phi)=rank_P(\phi/\rho)$.

2.4.4. EXAMPLE. Here we will 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 as labels the natural numbers. Take as rules:

$$\frac{f \stackrel{n}{\not\rightarrow}}{f \stackrel{n+2}{\longrightarrow} f} \quad n \geqslant 0,$$

$$\frac{f \stackrel{n}{\not\rightarrow}}{f \stackrel{0}{\rightarrow} f} \quad \text{for } n \text{ odd.}$$

$rank_P: Tr(\Sigma,A) \rightarrow \omega \cdot 2$ is defined by $rank_P(f \stackrel{n}{\rightarrow} f)=(n+1)/2$ if n odd and $rank_P(f \stackrel{n}{\rightarrow} f)=\omega+n/2$ if n even.

Checking whether or not a literal dependency graph contains cycles with negative edges is laborious and therefore not very useful to check 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* [2,25] provide a more useful technique to show consistency. A stratification of a TSS is given by the following definition.

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

$$\frac{\{t_k \stackrel{a_k}{\rightarrow} t_k' \mid k \in K\} \cup \{t_l \stackrel{b_l}{\not\rightarrow} \mid l \in L\}}{t \stackrel{a}{\rightarrow} t'} \in R$$

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

for all $k \in K$: $S(\sigma(t_k \stackrel{a_k}{\rightarrow} t_k')) \leqslant S(\sigma(t \stackrel{a}{\rightarrow} t'))$

for all $l \in L$ and $t_l' \in T(\Sigma)$: $S(\sigma(t_l \stackrel{b_l}{\rightarrow} t_l')) < S(\sigma(t \stackrel{a}{\rightarrow} t'))$

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 will speak about an *equi-label* *stratification.* In the same way we will speak about an *equi-source* and an *equi-target stratification.*

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

PROOF. "$\Rightarrow$" As P is stratifiable, the function $rank_P:Tr(\Sigma,A)\rightarrow\alpha$ 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 $S:Tr(\Sigma,A)\rightarrow\alpha$. 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 chain ending in ϕ in the literal dependency graph, containing an infinite number of negative labels. Suppose the induction holds for all $\beta'<\beta$, $S(\phi)=\beta$ and there is a chain ending in ϕ labeled with an infinite number of n's in the literal dependency graph. Then this means that there is a tail of the chain

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

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 chain labeled with an infinite number of n's ending in ψ. But this contradicts the assumption that there was one from ϕ. □

2.5. As remarked in example 2.3.3 there is not always one unique transition relation that agrees with P. Therefore, we will define, given a TSS P with a stratification S, a relation $\longrightarrow_{P,S}$ which we call the transition relation *associated with P* (*based on S*). The construction of the transition relation $\longrightarrow_{P,S}$ from a transition system specification is as follows: a literal ϕ with $S(\phi)=0$ is in $\longrightarrow_{P,S}$ 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 $\longrightarrow_{P,S}$. We use this information to find which literals ϕ with $S(\phi)=1$ are in $\longrightarrow_{P,S}$. 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 exactly coincides with the transition relation containing all provable literals [12]. Moreover, the definition of $\longrightarrow_{P,S}$ is independent of the stratification S. This last statement is proved in lemma 2.5.4.

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 premisses 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 2.5.2. In theorem 2.5.3 the regularity of $degree(r)$ is crucial.

2.5.1. DEFINITION. 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 $r\in R$.

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

$$\longrightarrow_{P,S}=\bigcup_{0\leqslant i<\alpha}\longrightarrow_i^P.$$

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

$$\longrightarrow_i^P=\bigcup_{0\leqslant j<degree(P)}\longrightarrow_{ij}^P \text{ for } 1\leqslant i<\alpha$$

$$\longrightarrow_{ij}^P=\{\phi\,|\,S(\phi)=i,$$

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

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

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

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

2.5.3. THEOREM. *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 $\longrightarrow_{P,S}$, that agrees with P.*

PROOF. We will show that $\longrightarrow_{P,S}$ agrees with P:

$\Rightarrow$) Suppose that for a rule

$$r = \frac{\{t_k \xrightarrow{a_k} t_k' \mid k \in K\} \cup \{t_l \stackrel{a_l}{\nrightarrow} \mid l \in L\}}{t \xrightarrow{a} t'} \in R$$

and a closed substitution σ all premises hold in $\longrightarrow_{P,S}$. Define $\beta = S(\sigma(t \xrightarrow{a} t'))$. For a negative premise $t_l \stackrel{a_l}{\nrightarrow}$ it trivially holds that for every $t'' \in T(\Sigma)$ $t_l \xrightarrow{a_l} t'' \notin \bigcup_{0 \leqslant i < \beta} \longrightarrow^P_i$. For a positive premise $t_k \xrightarrow{a_k} t_k'$ it holds that either $\sigma(t_k \xrightarrow{a_k} t_k') \in \bigcup_{0 \leqslant i < \beta} \longrightarrow^P_i$ or $\sigma(t_k \xrightarrow{a_k} t_k') \in \longrightarrow^P_\beta$. Consider the set $T = \{j \mid j < degree(P)$ and for some $k \in K$ j is the smallest ordinal such that $\sigma(t_k \xrightarrow{a_k} t_k') \in \longrightarrow^P_{\beta j}\}$. $|T| \leqslant |K| < degree(P)$. 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': $\sigma(t \xrightarrow{a} t') \in \longrightarrow^P_{\beta j'}$ by definition. Hence, $\sigma(t \xrightarrow{a} t') \in \longrightarrow_{P,S}$.

$\Leftarrow$) Suppose $\psi \in \longrightarrow_{P,S}$. Then for some $0 \leqslant i < \alpha$, $0 \leqslant j < degree(P)$ $\psi \in \longrightarrow^P_{ij}$. According to the definition of $\longrightarrow_{P,S}$ this means that there is a closed 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 $\sigma(\chi_k) \in \bigcup_{0 \leqslant j' < j} \longrightarrow^P_{ij'} \cup \bigcup_{0 \leqslant i' < i} \longrightarrow^P_{i'}$. But then $\sigma(\chi_k) \in \longrightarrow_{P,S}$. If $\chi_k \equiv t \stackrel{a}{\nrightarrow}$ then for every $t' \in T(\Sigma)$: $\sigma(t \xrightarrow{a} t') \notin \bigcup_{0 \leqslant i' < i} \longrightarrow^P_{i'}$. Due to the stratification $S(\sigma(t \xrightarrow{a} t')) < i$. Hence, $\sigma(t \xrightarrow{a} t') \notin \longrightarrow^P_{i'}$ for $i' \geqslant i$ and therefore $\sigma(t \xrightarrow{a} t') \notin \longrightarrow_{P,S}$. So all premises of $\sigma(r)$ hold in $\longrightarrow_{P,S}$. □

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

2.5.4. LEMMA. *Let P be a TSS which is stratified by stratifications 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 $\longrightarrow_{P,S} \neq \longrightarrow_{P,S'}$. This means that there is some ϕ such that either $\phi \in \longrightarrow_{P,S} - \longrightarrow_{P,S'}$ or $\phi \in \longrightarrow_{P,S'} - \longrightarrow_{P,S}$. Assume that ϕ is minimal with respect to S, i.e. $S(\phi) \leqslant S(\psi)$ for all $\psi \in (\longrightarrow_{P,S} - \longrightarrow_{P,S'}) \cup (\longrightarrow_{P,S'} - \longrightarrow_{P,S})$. Define $i = S(\phi)$.

1. Suppose $\phi \in \longrightarrow_{P,S} - \longrightarrow_{P,S'}$. Then $\phi \in \longrightarrow^P_{ij}$ for some $0 \leqslant j < degree(P)$ (see definition 2.2). Assume that ϕ is minimal with respect to $\longrightarrow^P_{ij}$, i.e. for all ψ with $S(\psi) = i$ and $\psi \in \longrightarrow_{P,S} - \longrightarrow_{P,S'}$: $\psi \notin \longrightarrow^P_{ij'}$ with $j' < j$.
   As $\longrightarrow_{P,S}$ agrees with P there is a closed instantiated rule $\sigma(r)$ with conclusion ϕ and premises χ_k $(k \in K)$ such that $\longrightarrow_{P,S} \models \chi_k$. As $\phi \notin \longrightarrow_{P,S'}$ it cannot be that all premises χ_k $(k \in K)$ hold in $\longrightarrow_{P,S'}$. Hence, $\longrightarrow_{P,S'} \not\models \chi_{k'}$ for some $k' \in K$. If $\chi_{k'}$ is a positive literal then $\chi_{k'} \in \bigcup_{0 \leqslant j'' < j} \longrightarrow^P_{ij''} \cup \bigcup_{0 \leqslant i'' < i} \longrightarrow^P_{i''}$ and $\chi_{k'} \notin \longrightarrow_{P,S'}$. But this contradicts one of the assumptions

that ϕ is minimal.

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

2. Considering $\phi \in \longrightarrow_{P,S'} - \longrightarrow_{P,S}$ 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 $\longrightarrow_{P,S}$ associated to 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 $\longrightarrow_P$ associated with P.

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

2.6.1. Definition. 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

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

and every substitution σ, $\sigma(t \xrightarrow{a} t')$ is in a strictly higher stratum than $\sigma(t_k \xrightarrow{a_k} t_k')$ $(k \in K)$ and $\sigma(t_l \xrightarrow{a_l} t'')$ 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 chain ending in some literal ϕ.

2.6.2. Theorem. *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 $\longrightarrow_1$ is a transition relation that agrees with P. P has a strict stratification $S: T(\Sigma) \rightarrow \alpha$ for some ordinal α. Let $\longrightarrow_{P,S}$ be the transition relation that is associated with P. Assume, in order to generate a contradiction, that $\longrightarrow_{P,S} \neq \longrightarrow_1$. This implies that there is some literal ϕ such that $\phi \in \longrightarrow_{P,S} - \longrightarrow_1$ or $\phi \in \longrightarrow_1 - \longrightarrow_{P,S}$. Assume furthermore that ϕ is minimal, i.e. for all $\psi \in (\longrightarrow_{P,S} - \longrightarrow_1) \cup (\longrightarrow_1 - \longrightarrow_{P,S})$: $S(\phi) \leqslant S(\psi)$. For reasons of symmetry it is enough to consider only one case: $\phi \in \longrightarrow_{P,S} - \longrightarrow_1$. As $\longrightarrow_{P,S}$ agrees with P there is a rule

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

and a substitution $\sigma: V \rightarrow T(\Sigma)$ such that $\phi = \sigma(\chi)$, $\longrightarrow_{P,S} \models \sigma(\chi_k)$ for all $k \in K$. Then for some $k' \in K$ $\longrightarrow_1 \not\models \sigma(\chi_{k'})$ because otherwise, as $\longrightarrow_1$ agrees with P, $\phi \in \longrightarrow_1$ contradicting the assumption.

If $\sigma(\chi_{k'})$ is a positive literal then $\sigma(\chi_{k'}) \in \longrightarrow_{P,S}$, $\sigma(\chi_{k'}) \notin \longrightarrow_1$ and $S(\chi_{k'}) < S(\phi)$. This contradicts the minimality of ϕ. If $\sigma(\chi_{k'}) \equiv t \xrightarrow{a}\!\!\!\!/$ then for some $t' \in T(\Sigma)$ $t \xrightarrow{a} t' \in \longrightarrow_1$, but $t \xrightarrow{a} t' \notin \longrightarrow_{P,S}$ and $S(t \xrightarrow{a} t') < S(\phi)$. This contradicts the minimality of ϕ as well. □

3. Examples showing the use of stratifications

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

3.1. EXAMPLE. Here the GSOS-format is defined. It differs slightly from the GSOS-format as given by BLOOM, ISTRAIL and MEYER [9] 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

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

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

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

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

3.2. EXAMPLE. In [6] a priority operator is defined on process graphs. In [12] 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] will give rise to inconsistencies. Here we will show that simple conditions on either the relabeling operator or recursion can circumvent this problem.

We base this example on the rules for $BPA_{\epsilon\delta}$ as given in [12] (rules 1-6 in table 1). The TSS $P_{prio}=(\Sigma_{prio}, A_{prio}, R_{prio})$ with $\Sigma_{prio}=(F_{prio}, r_{prio})$ 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. Furthermore, the signature contains constant names ϵ for the *empty process*, and δ representing *inaction*, resembling *NIL* in CCS [16].

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 $\Xi \subseteq \Sigma_{prio}$ with *process names*. E is a set of *process declarations* of the form $X \Leftarrow t_X$ for all process names $X \in \Xi$ ($t_X \in T(\Sigma_{prio})$). In $X \Leftarrow t_X$, t_X is the *body* of process name X.

The labels in A_{prio} are given by $Act_{\surd}$ $(=Act \cup \{\surd\})$. $\surd$ 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 $Act_{\surd}$. 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. 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 $X \xrightarrow{b} \epsilon$ 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 $a \in A_{prio}$ by:

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

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

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

TABLE 1

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

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

Another solution is to disallow that the priority and unless operators appear in the body of a process name. In this case a stratification can be given by:

$$S(t \xrightarrow{a} t') = n \quad \text{where } n \text{ is the total number of occurrences of } \theta\text{'s in } t.$$

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 $S: Tr(\Sigma_{prio}, A_{prio}) \rightarrow \omega$ by $S(t \xrightarrow{a} t') = n + m$. One can check that S is a stratification of P_{prio}.

4. STRONG BISIMULATION EQUIVALENCE

The notion of strong bisimulation equivalence as defined below is from PARK [20].

4.1. DEFINITION. 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:

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

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

5. 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 if it guarantees that all operators defined by this format respect bisimulation.

For TSS's without negative premises, the *tyft/tyxt*-format [12] is the most general format for which bisimulation is a congruence. Here 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.

5.1. DEFINITION. 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:

$$\frac{\{t_k \xrightarrow{a_k} y_k \mid k\in K\} \cup \{t_l \stackrel{b_l}{\not\longrightarrow} \mid l\in L\}}{f(x_1,\ldots,x_{r(f)}) \xrightarrow{a} t}$$

with K and L index sets, y_k,x_i $(1\leq i\leq r(f))$ 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:

$$\frac{\{t_k \xrightarrow{a_k} y_k \mid k\in K\} \cup \{t_l \stackrel{b_l}{\not\longrightarrow} \mid l\in L\}}{x \xrightarrow{a} t}$$

with K,L index sets, y_k,x all different variables, $a_k,b_l,a\in A$, t_k,t_l and $t\in\mathbb{T}(\Sigma)$. 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.

From examples given in [12] it follows that the *tyft/tyxt*-format cannot be generalized in any obvious way without endangering the congruence property of bisimulation equivalence. This implies that for the *ntyft/ntyxt*-format, the positive premises cannot be generalized. Since the negative premises are already as general as possible, the *ntyft/ntyxt*-format cannot be generalized in any obvious way without losing the congruence property of strong bisimulation.

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

5.2. DEFINITION (*well-founded*). Let $P=(\Sigma,A,R)$ be a TSS. Let $S=\{t_k \xrightarrow{a_k} t_k' \mid k\in K\}\subseteq \mathbb{T}(\Sigma)\times A\times\mathbb{T}(\Sigma)$ be a set of positive literals over Σ and A. The *Variable Dependency Graph (VDG)* of S is a directed (unlabeled) graph with:

- Nodes: $\bigcup_{k\in K} Var(t_k \xrightarrow{a_k} t_k')$,
- Edges: $\{<x,y> \mid x\in Var(t_k),\ y\in Var(t_k') \text{ for some } k\in K\}$.

S 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.

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

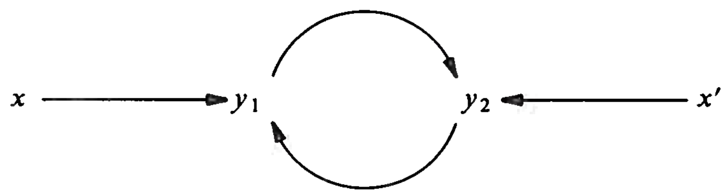

FIGURE 3

5.2.2. EXAMPLE. Consider the variable dependency graph G of $\{x_{n+1} \xrightarrow{a} x_n \mid n \in \mathbb{N}\}$. 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 chain ending in this node. A part of G is depicted in figure 4.

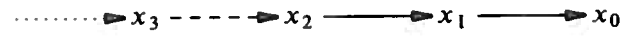

FIGURE 4

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

If S is a set of positive premises of a rule in *ntyft/ntyxt*-format then $n_{VDG}(x) \in \mathbb{N}$ for each $x \in Var(S)$; every variable y_k only occurs once in 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 $U = \{n_{VDG}(x)+1 \mid \langle x,y_k \rangle$ is an edge of $G\}$ is finite. Hence, $n_{VDG}(y_k) = sup(U)$ is a natural number.

5.4. DEFINITION. Two stratifiable TSS's P and P' are *transition equivalent* if $\longrightarrow_P \; = \; \longrightarrow_{P'}$.

Hence, two TSS's are transition equivalent if they have the same signature, the same set of labels and if the sets of rules determine the same associated transition relation. The particular form of the rules is not of importance.

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

⊆) Suppose $\phi \in \longrightarrow_{ij}^{P}$ for some i and j. According to the definition of $\longrightarrow_{ij}^{P}$ this means that there is a closed instantiated rule $\sigma(r)$ with conclusion ϕ and premises χ_k ($k \in K$) such that $\bigcup_{0 \leqslant j' < j} \longrightarrow_{ij'}^{P} \cup \bigcup_{0 \leqslant i' < i} \longrightarrow_{i'}^{P} \vDash \chi_k$. If χ_k is positive then, inductively, $\bigcup_{0 \leqslant j' < j} \longrightarrow_{ij'}^{P'} \cup \bigcup_{0 \leqslant i' < i} \longrightarrow_{i'}^{P'} \vDash \chi_k$. If $\chi_k \equiv t \overset{a}{\not\longrightarrow}$ then for all $t' \in T(\Sigma)$: $t \xrightarrow{a} t' \notin \bigcup_{0 \leqslant i' < i} \longrightarrow_{i'}^{P}$ and therefore $t \xrightarrow{a} t' \notin \bigcup_{0 \leqslant i' < i} \longrightarrow_{i'}^{P'}$. Hence, in both cases $\bigcup_{0 \leqslant j' < j} \longrightarrow_{ij'}^{P'} \cup \bigcup_{0 \leqslant i' < i} \longrightarrow_{i'}^{P'} \vDash \chi_k$ for all $k \in K$. If r is a *ntyft*-rule, one can apply $\sigma(r)$ again to obtain $\phi \in \longrightarrow_{ij}^{P'}$. If r is in *ntyxt*-format and the lefthandside 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, $\phi \in \longrightarrow_{ij}^{P'}$.

⊇) The reverse implication can be shown in the same way. □

5.6. DEFINITION. 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.

5.7. LEMMA. *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:

$$\sigma(x)=t \in T(\Sigma) \quad \text{if } x \text{ is free in } r,$$
$$\sigma(x)=x \quad \text{otherwise.}$$

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 5.5. □

Next, we state the congruence theorem.

5.8. THEOREM. *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 the same theorem in [12]. Assume $P=(\Sigma,A,R_0)$ with $\Sigma=(F,r)$. According to lemma 5.5 and lemma 5.7 we may assume that P is in pure *ntyft*-format. As P is stratifiable, there is a stratification $S:T(\Sigma,A)\rightarrow\alpha$ for some ordinal α of P. Furthermore, there is a transition relation $\longrightarrow_P$ associated with P. We will show that for all $f \in F$, $u_1,\ldots,u_{r(f)}$, $v_1,\ldots,v_{r(f)} \in T(\Sigma)$:

$$\forall 1 \leq k \leq r(f)\text{: } u_k \leftrightarrows_P v_k \;\Rightarrow\; f(u_1,\ldots,u_{r(f)}) \leftrightarrows_P f(v_1,\ldots,v_{r(f)}).$$

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

For the relation R we have the following useful fact.

FACT. *Let $t \in \mathbb{T}(\Sigma)$ and let $\sigma,\sigma':V\rightarrow T(\Sigma)$ be substitutions such that for all x in $Var(t)$: $\sigma(x)\,R\,\sigma'(x)$. Then $\sigma(t)\,R\,\sigma'(t)$.*
PROOF. Straightforward induction on the structure of t. □

Once 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 $1 \leq k \leq r(f)$. We are ready if we show (by induction on β) that the following holds for all β:

If $\mathfrak{L}(f(u_1,\ldots,u_{r(f)}),a)+\mathfrak{L}(f(v_1,\ldots,v_{r(f)}),a)=\beta$ then
- *$f(u_1,\ldots,u_{r(f)}) \xrightarrow{a} u' \in \longrightarrow_P$ and $u_k R\, v_k$ for $1 \leq k \leq r(f)$ implies $\exists v'$ $f(v_1,\ldots,v_{r(f)}) \xrightarrow{a} v' \in \longrightarrow_P$ and $u'\,R\,v'$*
- *vice versa.*

Here $\mathcal{L}(t,a)=sup(\{S(t \xrightarrow{a} t')+1 \mid t' \in T(\Sigma)\})$ for $t \in T(\Sigma)$ and $a \in A$. As the induction hypothesis is symmetric we need only check one halve of it. Suppose the induction hypothesis holds for all $\beta' < \beta$. The validity of the induction hypothesis for β follows immediately if the following statement holds for all $1 \leqslant i < \alpha$ and $1 \leqslant j \leqslant degree(P)$:

If $\mathcal{L}(f(u_1,\ldots,u_{r(f)}),a)+\mathcal{L}(f(v_1,\ldots,v_{r(f)}),a)=\beta,\ f(u_1,\ldots,u_{r(f)}) \xrightarrow{a} u' \in \longrightarrow_{ij}^{P}$ *and* $u_k R v_k$ *for* $1 \leqslant k \leqslant r(f)$ *then* $\exists v'\ f(v_1,\ldots,v_{r(f)}) \xrightarrow{a} v' \in \longrightarrow_P$ *and* $u' R v'$

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

$$r = \frac{\{t_k \xrightarrow{a_k} y_k \mid k \in K\} \cup \{t_l \xrightarrow{a_l}\!\!\!\!\!\not\ \ \mid l \in L\}}{f(x_1,\ldots,x_{r(f)}) \xrightarrow{a} t} \in R_0$$

and a substitution σ such that:

1. $\sigma(f(x_1,\ldots,x_{r(f)}))=u$,
2. $\sigma(x_i)=u_i$ for $1 \leqslant i \leqslant r(f)$,
3. $\sigma(t)=u'$,
4. $\longrightarrow_P \vDash \sigma(t_k \xrightarrow{a_k} y_k)$ and $\longrightarrow_P \vDash \sigma(t_l) \xrightarrow{a_l}\!\!\!\!\!\not\ \ $.

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

CLAIM. *There is a closed substitution* σ' *such that for any* $x \in nodes(G)$ *with* $n_{VDG}(x) < n$ $\sigma(x) R \sigma'(x)$ *and if* $x = y_k$ *for some* $k \in K$ *then* $\sigma'(t_k \xrightarrow{a} y_k) \in \longrightarrow_P$.

For the moment we defer proving this claim and, assuming that it holds for all n, the proof of the theorem is finished first.

For all positive premises ϕ of r it follows that we can prove that $\sigma'(\phi) \in \longrightarrow_P$ for some closed substitution σ'. We will show that for each negative premise $t_l \xrightarrow{a_l}\!\!\!\!\!\not\ \ $ in r $\sigma'(t_l) \xrightarrow{a_l}\!\!\!\!\!\not\ \ $ also holds in $\longrightarrow_P$. By the claim $\sigma(x) R \sigma'(x)$ for all variables x in r. Hence, we know using the previously proved fact that $\sigma(t_l) R \sigma'(t_l)$. By definition of R there are two possibilities.

1. $\sigma(t_l) \leftrightarrows_P \sigma'(t_l)$. In this case $\sigma'(t_l) \xrightarrow{a_l}\!\!\!\!\!\not\ \ $ clearly holds in $\longrightarrow_P$.
2. $\sigma(t_l)=g(w_1,\ldots,w_{r(g)})$ 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 $w \in T(\Sigma)$ $\sigma'(t_l) \xrightarrow{a_l} w$. Suppose $v \xrightarrow{a} \sigma'(t)$ is in a stratum S_γ $(\gamma < \mathcal{L}(v,a))$. Hence, $\mathcal{L}(\sigma'(t_l),a_l) \leqslant \gamma < \mathcal{L}(v,a)$. In the same way we can show that $\mathcal{L}(\sigma(t_l),a_l) < \mathcal{L}(u,a)$. Clearly, $\mathcal{L}(\sigma(t_l),a_l)+\mathcal{L}(\sigma'(t_l),a_l) < \mathcal{L}(u,a)+\mathcal{L}(v,a)$. So by applying the first induction hypothesis we know that $\exists w'$ $\sigma(t_l) \xrightarrow{a_l} w'$. But this contradicts that $\sigma(t_l) \xrightarrow{a_l}\!\!\!\!\!\not\ \ $ holds in $\longrightarrow_P$. So for every negative premise $t_l \xrightarrow{a_l}\!\!\!\!\!\not\ \ $ of r: $\longrightarrow_P \vDash \sigma'(t_l) \xrightarrow{a_l}\!\!\!\!\!\not\ \ $.

Now as all premises of $\sigma'(r)$ hold, we may conclude that $\sigma'(f(x_1,\ldots,x_{r(f)}) \xrightarrow{a} t) \in \longrightarrow_P$. Define $v' = \sigma'(t)$. For all $x \in Var(t)$: $\sigma(x) R \sigma'(x)$. By an application of the previously proved fact it follows that $\sigma(t) R \sigma'(t)$ or equivalently, $u' R v'$. This completes the induction step for the second induction hypothesis. So we are almost finished. We only have to give the proof of the claim.

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

1. $x = x_i$ $(1 \leqslant i \leqslant r(f))$. In this case the claim holds for n as $\sigma(x)=u_i R v_i = \sigma'(x)$.
2. $x = y_k$ $(k \in K)$ and $t_k \xrightarrow{a} y_k$ is a premise of r. By induction it holds that there is a closed substitution σ' such that for all $y \in Var(t_k)$: $\sigma(y) R \sigma'(y)$. By the previously proved fact $\sigma(t_k) R \sigma'(t_k)$. Now distinguish between two cases:

1. $\sigma(t_k) \leftrightarrows_P \sigma'(t_k)$. In this case there is a $w \in T(\Sigma)$ such that $\sigma'(t_k) \xrightarrow{a_k} w \in \longrightarrow_P$ 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:

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

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

   $w_j R w_j'$ for $1 \leqslant j \leqslant r(g)$.

   Furthermore, we know that $\mathfrak{L}(\sigma(t_k), a_k) + \mathfrak{L}(\sigma'(t_k), a_k) \leqslant \mathfrak{L}(u,a) + \mathfrak{L}(v,a)$ because $S(\sigma(t_k) \xrightarrow{a_k} t'') \leqslant S(u \xrightarrow{a} u')$ for arbitrary $t'' \in T(\Sigma)$ and $S(\sigma'(t_k) \xrightarrow{a_k} t'') \leqslant S(v \xrightarrow{a} v')$ for arbitrary $t'', v' \in T(\Sigma)$. Also $\sigma(t_k \xrightarrow{a_k} y_k) \in \bigcup_{i'<i} \longrightarrow_{i'}^{P} \cup \bigcup_{j'<j} \longrightarrow_{ij'}^{P}$. Now we can apply the first or second induction hypothesis which gives that there is a w such that $g(w_1', \ldots, w_{r(g)}') \xrightarrow{a_k} w \in \longrightarrow_P$ and $\sigma(y_k) R w$.

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

6. Modular properties of TSS's

Often one wants to extend a TSS with new functions and constants. Therefore the *sum* of two TSS's is introduced [12]. The combination of two TSS's P_0 and P_1 is denoted by $P_0 \oplus P_1$. 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 it would be nice if all literals with source $t \in T(\Sigma_0)$ in $\longrightarrow_{P_0 \oplus P_1}$ are exactly the literals in $\longrightarrow_{P_0}$. In this case we say that $P_0 \oplus P_1$ is a *conservative extension* of P_0.

6.1. Definition. 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:

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

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

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

6.3. Definition. Let $P_i = (\Sigma_i, A_i, R_i)$ $(i = 0, 1)$ be two TSS's 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 $t' \in T(\Sigma)$:

$$t \xrightarrow{a} t' \in \longrightarrow_P \quad \Leftrightarrow \quad t \xrightarrow{a} t' \in \longrightarrow_{P_0}.$$

6.4. 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 $t, u \in T(\Sigma_0)$: $t \leftrightarrows_P u \Leftrightarrow t \leftrightarrows_{P_0 \oplus P_1} u$.

The following theorem gives conditions under which a TSS P_1 can be added conservatively to P_0. The theorem is the same as the one that holds for TSS's without negative premises [12], 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 [12].

6.5. THEOREM. *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 $S:Tr(\Sigma,A)\to\alpha$ for some ordinal α for P. Define $S^0:Tr(\Sigma_0,A_0)\to\alpha$ by $S^0(\phi)=S(\phi)$. It is not hard to check that S^0 is a stratification of P_0.

We will prove that:

$$t\in T(\Sigma_0),\ a\in A_0,\ t\xrightarrow{a}t'\in\longrightarrow_P \quad\Leftrightarrow\quad t\xrightarrow{a}t'\in\longrightarrow_{P_0},\ t'\in T(\Sigma_0) \qquad \text{(I)}$$

by induction on the ordinal β $(0\leq\beta<\alpha)$ 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 $t\xrightarrow{a}t'\in\longrightarrow^P_{\beta j}$ for some j. Here $\longrightarrow^P_{\beta j}$ is the relation defined in definition 2.5.2 to construct $\longrightarrow_P$. By induction on j it is shown that:

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

As $\longrightarrow_P$ agrees with P there is a rule $r\in R$ with conclusion $u\xrightarrow{a}u'$ and a substitution $\sigma:V\to T(\Sigma)$ 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 in 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 $(u=f(x_1,\ldots,x_{r(f)}))$:

$$\frac{\{s_k\xrightarrow{a_k}y_k\mid k\in K\}\cup\{u_l\stackrel{b_l}{\not\longrightarrow}\mid l\in L\}}{f(x_1,\ldots,x_{r(f)})\xrightarrow{a}u'}$$

Now we use induction on $n_{VDG}(x)$ 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 $\sigma(s_k\xrightarrow{a_k}y_k)\in\longrightarrow_{P_0}$. Suppose $n_{VDG}(x)=n\in\mathbb{N}$. As P_0 is pure, we distinguish two cases:

1. $x=x_i$ $(1\leq i\leq r(f))$. As $t\in T(\Sigma_0)$, $\sigma(x)\in T(\Sigma_0)$.
2. $x=y_k$ $(k\in K)$ 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 $\sigma(s_k\xrightarrow{a_k}y_k)\in\longrightarrow_{P_0\oplus P_1}$, we can derive $\sigma(s_k\xrightarrow{a_k}y_k)\in\longrightarrow_{P_0}$ 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 $\sigma(\phi)\in\longrightarrow_{P_0}$. For a negative premise $u_l\stackrel{b_l}{\not\longrightarrow}$ we assume, in order to generate a contradiction that $\exists\ u_l'\in T(\Sigma_0)$ $\sigma(u_l\xrightarrow{b_l}u_l')\in\longrightarrow_{P_0}$. 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 $\sigma(u_l\xrightarrow{b_l}u_l')\in\longrightarrow_P$. This contradicts $\sigma(u_l)\stackrel{b_l}{\not\longrightarrow}$.

As $\longrightarrow_{P_0}$ agrees with P_0 and all premises of $\sigma(r)$ hold in $\longrightarrow_{P_0}$ it follows that $\sigma(u\xrightarrow{a}u')$ also holds in $\longrightarrow_{P_0}$. 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 intermediate induction hypothesis:

$$t\xrightarrow{a}t'\in\longrightarrow_{P_0} \quad\Rightarrow\quad t\xrightarrow{a}t'\in\longrightarrow_P.$$

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

$$t\xrightarrow{a}t'\in\longrightarrow_{P_0} \quad\Rightarrow\quad t\xrightarrow{a}t'\in\longrightarrow_P,\ t\in T(\Sigma_0),\ a\in A_0.$$

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 TSS's 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 TSS's is not stratifiable.

6.6. EXAMPLE. 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:

$$\frac{f(x) \stackrel{a}{\not\longrightarrow}}{f(x) \stackrel{a}{\longrightarrow} f(x)}$$

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 $\longrightarrow$ is a relation that agrees with $P_0 \oplus P_1$ then $\longrightarrow \models f(c) \stackrel{a}{\longrightarrow} f(c)$ iff $\longrightarrow \models f(c) \stackrel{a}{\not\longrightarrow}$.

6.7. EXAMPLE. This is a less trivial example that shows a problem that can occur when stratifying the sum of stratifiable TSS's. Let P_0 consist of a unary function name g, a constant δ, labels a,b and a rule:

$$\frac{x \stackrel{b}{\not\longrightarrow}}{g(x) \stackrel{a}{\longrightarrow} \delta}$$

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

$$\frac{g(f(x)) \stackrel{a}{\longrightarrow} y}{f(x) \stackrel{b}{\longrightarrow} \delta}$$

P_0 and P_1 both have an associated transition relation. $P_0 \oplus P_1$, however, makes it possible to show that $f(\delta) \stackrel{b}{\longrightarrow} \delta$ iff $f(\delta) \stackrel{b}{\not\longrightarrow}$ for any transition relation $\longrightarrow$ agreeing with $P_0 \oplus P_1$. In figure 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.

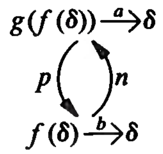

FIGURE 5

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

6.8. THEOREM. *Let $\Sigma_0=(F_0,\overline{r_0})$ and $\Sigma_1=(F_1,\overline{r_1})$ 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 stratified TSS's. Let $\Sigma_0 \oplus \Sigma_1$ be defined. If for all closed substitutions σ_0 and σ_1 and rules $r_0 \in R_0$ and $r_1 \in R_1$ where ϕ is the conclusion of r_1 and ψ is a positive premise of r_0 or ($\psi=t \stackrel{a}{\longrightarrow} t'$ and $t \stackrel{a}{\not\longrightarrow}$ is a negative premise of r_0): $\sigma_0(\psi) \neq \sigma_1(\phi)$ then $P_0 \oplus P_1$ is a stratifiable TSS.*

PROOF. Assume that P_0 has stratification $S^0: Tr(\Sigma_0,A_0) \rightarrow \alpha_0$ and that P_1 has stratification $S^1: Tr(\Sigma_1,A_1) \rightarrow \alpha_1$. 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 $\overline{\phi}$ by replacing all subterms $f(\overline{u})$ for $f \in F_1$ in ϕ by a_0. As the label of ϕ is in A_0, $\overline{\phi} \in Tr(\Sigma_0,A_0)$ and thus $\overline{\phi}$ 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 $\overline{\phi}$

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 $r \in R_0 \cup R_1$. Suppose σ is a closed substitution and ψ is the conclusion, ϕ a positive premise (if present in $\sigma(r)$) and $t \xrightarrow{a}\!\!\!\!/\;$ a negative premise (also if present) of $\sigma(r)$. We proceed by case analysis.

1. $\psi \in U$. By the condition in this theorem ψ is not an instance of a conclusion in a rule from 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 the same way as $\bar{\phi}, \bar{\psi}$ and $\overline{t \xrightarrow{a} t'}$ are related in S^0. As $\bar{\phi}, \bar{\psi}$ and $\overline{t \xrightarrow{a} t'}$ 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.
2. $\psi \notin U$.
   1. 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.
   2. 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 $S(\phi) = \alpha_0 + \gamma$. If $t \xrightarrow{a} t' \notin U$ and $\overline{t \xrightarrow{a} t'} \in S^1_{\gamma_{t'}}$ then $S(t \xrightarrow{a} t') = \alpha_0 + \gamma_{t'}$. Now as $\bar{\psi}$, $\bar{\phi}$ and $\overline{t \xrightarrow{a} t'}$ 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. □

7. 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. Then, in figure 6, we show how we will prove our result. The arrows denote set inclusion and 'IF' indicates that we need image finiteness.

7.1. Definition. Let $P = (\Sigma, A, R)$ be a stratifiable TSS and let $\longrightarrow_P$ be the transition relation associated with P. Let $s \in T(\Sigma)$. A sequence $a_1 \star \cdots \star a_n \in A^*$ is a *(P-)trace* from s if there are terms $s_1, \ldots, s_n \in T(\Sigma)$ for some $n \in \mathbb{N}$ such that $s \xrightarrow{a_1}_P s_1 \xrightarrow{a_2}_P \cdots \xrightarrow{a_n}_P s_n$. $Tr(s)$ is the set of all P-traces from s. Two process terms $s, s' \in T(\Sigma)$ are *trace equivalent with respect to P* if $Tr(s) = Tr(s')$. This is also denoted as $s \equiv^P_T s'$.

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

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

7.4. Definition. Let $P = (\Sigma, A, R)$ be a stratifiable TSS with associated transition relation $\longrightarrow_P$. A relation $R^n \subseteq T(\Sigma) \times T(\Sigma)$, for $n \in \mathbb{N}$, is called an *n-bounded bisimulation* relation if:

1. $R^0 = T(\Sigma) \times T(\Sigma)$,
2. $\forall s, t \in T(\Sigma)\ s R^{n+1} t$ and $s \xrightarrow{a}_P s' \Rightarrow \exists t'\ t \xrightarrow{a}_P t'$ and $s' R^n t'$,
3. $\forall s, t \in T(\Sigma)\ s R^{n+1} t$ and $t \xrightarrow{a}_P t' \Rightarrow \exists s'\ s \xrightarrow{a}_P s'$ and $s' R^n t'$.

Two process expressions $t, t' \in T(\Sigma)$ are *n-bounded bisimilar (for P)*, notation $t \leftrightarrows^n_P t'$ if there is a *n*-

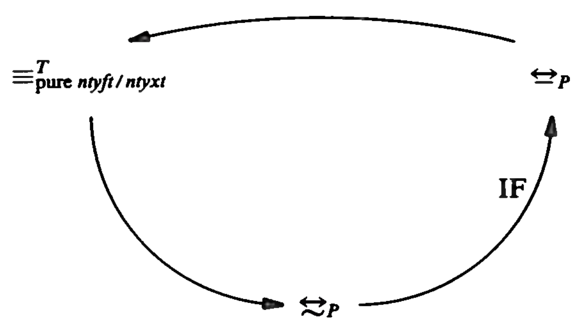

FIGURE 6

bounded bisimulation relation R^n such that $t\,R^n\,t'$. Two terms $t,t' \in T(\Sigma)$ are *bounded bisimilar for P*, notation $t \underset{\sim}{\leftrightarrows}_P t'$, if for all $n \in \mathbb{N}$ $t \overset{n}{\underset{\sim}{\leftrightarrows}}_P t'$.

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

7.4.1. LEMMA. *Let $P=(\Sigma,A,R)$ be a stratifiable TSS such that $\longrightarrow_P$ is image finite. Let $s,t \in T(\Sigma)$. Then:*

$$s \underset{\sim}{\leftrightarrows}_P t \quad \Leftrightarrow \quad s \leftrightarrows_P t.$$

PROOF. "$\Leftarrow$" is trivial. See for "$\Rightarrow$" [11]. □

7.5. We will now give the basic definitions and lemmas to prove that $\equiv^T_{\text{pure } ntyft/ntyxt} \subseteq \underset{\sim}{\leftrightarrows}_P$. The main component is the following test system. We will show that this test system is stratifiable and that it can test equality between n-bounded bisimilar processes.

7.5.1. DEFINITION. Let $P=(\Sigma,A,R)$ be a TSS. The *bisimulation tester of P* $P_T=(\Sigma_T,A_T,R_T)$ is a TSS with signature $\Sigma_T=(F_T,r_T)$ containing binary function names B^n and Q^n_a for all $n \in \mathbb{N}$, $a \in A$ and with constant δ. The labels of P_T are $A_T=A \cup \{ok,yes,no\}$. The rules in R_T are given in table 3.

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

TABLE 3

The rules in table 3 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.

The rules in table 3 just encode n-bounded bisimilarity. The negative premises model the universal quantifiers in definition 7.4.

7.5.2. REMARK. 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 rule 2 and 3 of table 3 $a \neq ok,yes,no$. If a would be allowed to range over $A \cup \{ok,yes,no\}$, then it is impossible to give a stratification as is done in this paper.

7.5.3. LEMMA. *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 $S:Tr(\Sigma_T,A_T) \rightarrow \omega$ as follows:

1. for all $a \in A$ and $t,t' \in T(\Sigma_T)$ $S(t \xrightarrow{a} t')=1$,
2. for $n \in \mathbb{N}$ and $t,u,v \in T(\Sigma_T)$ $S(B^n(t,u) \xrightarrow{yes} v)=2n+1$,
3. for $n \in \mathbb{N}-\{0\}$, $a \in A_T$ and $t,u,v \in T(\Sigma_T)$ $S(Q_a^n(t,u) \xrightarrow{ok} v)=2n-1$,
4. 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. □

7.5.4. LEMMA. *Let $P=(\Sigma,A,R)$ be a stratifiable TSS in pure ntyft/ntyxt-format containing at least one constant in its signature. Furthermore, A must not contain the labels ok,no,yes and Σ must 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*

$$B^n(t,u) \xrightarrow{yes} \delta \in \longrightarrow_{P \oplus P_T} \Leftrightarrow t \overset{n}{\leftrightarrows}_P u.$$

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 6.8, $P \oplus P_T$ is stratifiable. So $P \oplus P_T$ has an associated transition relation $\longrightarrow_{P \oplus P_T}$. As a consequence of theorem 6.5 $P \oplus P_T$ is a conservative extension of P.
"$\Rightarrow$" Use induction on n. *Basis*. For $n=0$ $t \overset{n}{\leftrightarrows}_P u$ for any $t,u \in T(\Sigma)$. Hence, the theorem holds in this case.
Induction. We have to show that (1): If $B^{n+1}(t,u) \xrightarrow{yes} \delta \in \longrightarrow_{P \oplus P_T}$ and $t \xrightarrow{a} t' \in \longrightarrow_P$ then $\exists u'$ s.t. $u \xrightarrow{a} u' \in \longrightarrow_P$ and $t' \overset{n}{\leftrightarrows}_P u'$ and vice versa (2): if $u \xrightarrow{a} u' \in \longrightarrow_P$ then $\exists t'$ s.t. $t \xrightarrow{a} t' \in \longrightarrow_P$ and $t' \overset{n}{\leftrightarrows}_P u'$. As $B^{n+1}(t,u) \xrightarrow{yes} \delta \in \longrightarrow_{P \oplus P_T}$ and $\longrightarrow_{P \oplus P_T}$ agrees with $P \oplus P_T$, it must be the case that using rule 4 $B^{n+1}(t,u) \overset{no}{\not\longrightarrow}$ and $B^{n+1}(u,t) \overset{no}{\not\longrightarrow}$ hold in $\longrightarrow_{P \oplus P_T}$. 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 $t \xrightarrow{a} t' \in \longrightarrow_P$ and by conservativity also $t \xrightarrow{a} t' \in \longrightarrow_{P \oplus P_T}$. Hence for some v $Q_a^{n+1}(t',u) \xrightarrow{ok} v \in \longrightarrow_{P \oplus P_T}$. But then the premises of rule 2 must be true with $\sigma(y)=u$ and $\sigma(x')=t'$. Hence for some u' $u \xrightarrow{a} u' \in \longrightarrow_{P \oplus P_T}$ and $B^n(t',u') \xrightarrow{yes} \delta \in \longrightarrow_{P \oplus P_T}$. By conservativity $u \xrightarrow{a} u' \in \longrightarrow_P$. With the induction hypothesis $t' \overset{n}{\leftrightarrows}_P u'$.
We can show (2) in the same way. Hence if $B^{n+1}(t,u) \xrightarrow{yes} \delta \in \longrightarrow_{P \oplus P_T}$ then $t \overset{n+1}{\leftrightarrows}_P u$.
"$\Leftarrow$" Again, we use induction on n. *Basis*. If $n=0$, the theorem is trivial as $B^0(t,u) \xrightarrow{yes} \delta \in \longrightarrow_{P \oplus P_T}$ for all $t,u \in T(\Sigma)$.
Induction. Suppose $t \overset{n+1}{\leftrightarrows}_P u$. We will show that $B^{n+1}(t,u) \xrightarrow{yes} \in \longrightarrow_{P \oplus P_T}$. By rule 4 it is sufficient to show that $B^{n+1}(t,u) \overset{no}{\not\longrightarrow}$ and $B^{n+1}(u,t) \overset{no}{\not\longrightarrow}$ hold in $\longrightarrow_{P \oplus P_T}$. This means that we have to show that rule 3 can never be applied, i.e. neither (3): $t \xrightarrow{a} t'$ or $Q_a^{n+1}(t',u) \overset{ok}{\not\longrightarrow}$ nor (4): $u \xrightarrow{a} u'$ or $Q_a^{n+1}(u',t) \overset{ok}{\not\longrightarrow}$ for any $a \in A$ holds in $\longrightarrow_{P \oplus P_T}$. Suppose for some $a \in A$ $t \overset{a}{\not\longrightarrow}$ holds in $\longrightarrow_{P \oplus P_T}$. Then (3) trivially does not hold. Now suppose $t \xrightarrow{a} t' \in \longrightarrow_{P \oplus P_T}$ for some t'. As P_T conservatively

extends P, $t \xrightarrow{a} t' \in \longrightarrow_P$. Then using $t \overset{n+1}{\leftrightarrows}_P u$ $\exists u' \in T(\Sigma)$ $u \xrightarrow{a} u' \in \longrightarrow_P$ and $t' \overset{n}{\leftrightarrows}_P u'$. By conservativity $u \xrightarrow{a} u' \in \longrightarrow_{P \oplus P_T}$. Using the induction hypothesis $B^n(t',u') \xrightarrow{yes} \delta \in \longrightarrow_{P \oplus P_T}$. Applying rule 2 yields $Q_a^{n+1}(t',u) \xrightarrow{ok} \delta \in \longrightarrow_{P \oplus P_T}$ and hence $Q_a^{n+1}(t',u) \xrightarrow{ok}\!\!\!\!\!/\;\;$ does not hold in $\longrightarrow_{P \oplus P_T}$. We can prove (4) in the same way. □

The following theorem relates all notions.

7.6. THEOREM. *Let $P=(\Sigma,A,R)$ be a stratifiable TSS in pure ntyft/ntyxt-format such that $\longrightarrow_P$ 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 $a \in A, n \in \mathbb{N}$.*

$$t \equiv^T_{pure\ ntyft/ntyxt} u \;\Leftrightarrow\; t \leftrightarrows_P u \;\Leftrightarrow\; t \leftrightarrow_P u$$

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, $t \equiv^T_{\text{pure}\ ntyft/ntyxt} u$.

Suppose $t \not\leftrightarrows_P u$. This means that for some $n \in \mathbb{N}$ $t \not\leftrightarrows^n_P u$. Construct the context $B^n_T(t,[\,])$. Now by lemma 7.5.4 $B^n(t,u) \xrightarrow{yes}\!\!\!\!\!/\;\;$ holds in $\longrightarrow_{P \oplus P_T}$ while $B^n(t,t) \xrightarrow{yes} \delta \in \longrightarrow_{P \oplus P_T}$. Hence, $t \not\equiv_{\text{pure}\ ntyft/ntyxt} u$ or in other words: $t \equiv^T_{\text{pure}\ ntyft/ntyxt} u \Rightarrow t \leftrightarrows_P u$.

The last case $t \leftrightarrows_P u \Rightarrow t \leftrightarrow_P u$ follows directly from lemma 7.4.1. □

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.

7.7. *A finite test system.* 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 as additional property that if the number of labels in a tested system is finite, then there are only a finite number of rules necessary.

7.7.1. DEFINITION. Let $P=(\Sigma,A,R)$ be a TSS with a countable set of labels A. Assume that there is a function $n{:}A \rightarrow \mathbb{N}$ that gives a unique number for each label, satisfying that if for $a \in A$ $n(a)=m>0$ then $\exists b \in A$ $n(b)=m-1$. 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 4. 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 sub- 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 P_{FT} as for P_T. We only give here the lemmas and we omit the proofs. With these results it can be shown in exactly the same way as in the proof of theorem 7.6 that P_{FT} is also powerful enough to distinguish between non bisimilar processes.

$0 \xrightarrow{0} \delta$		1
$S(x) \xrightarrow{1} x$		2
$1 \xrightarrow{b} \delta$	for $n(b)=0$	3
$\dfrac{x \xrightarrow{b} x'}{S_0(x) \xrightarrow{c} \delta}$	if $n(c)=n(b)+1$	4
$\dfrac{n \xrightarrow{0} n'}{B(n,x,y) \xrightarrow{yes} \delta}$		5
$\dfrac{l \xrightarrow{\bar{a}} l' \quad y \xrightarrow{a} y' \quad B(n,x',y') \xrightarrow{yes} z}{Q(n,l,x',y) \xrightarrow{ok} \delta}$	for $a \in A$	6
$\dfrac{n \xrightarrow{1} n' \quad x \xrightarrow{a} x' \quad Q(n',S_0^{n(a)}(1),x',y) \stackrel{ok}{\not\rightarrow}}{B(n,x,y) \xrightarrow{no} \delta}$	for $n>0$, $a \in A$	7
$\dfrac{B(n,x,y) \stackrel{no}{\not\rightarrow} \quad B(n,y,x) \stackrel{no}{\not\rightarrow}}{B(n,x,y) \xrightarrow{yes} \delta}$	for $n>0$	8

TABLE 4

7.7.2. LEMMA. *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.*

7.7.3. LEMMA. *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. 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:*

$$B(S^n(0),t,u) \xrightarrow{yes}_{P \oplus P_{FT}} \delta \quad \Leftrightarrow \quad t \overset{n}{\leftrightarrow}_P u.$$

8. 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. We find it useful to give an overview of the current state of affairs. Below in table 5 we give an overview of all results that are known up till now. 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 [12] can be obtained from the pure *ntyft/ntyxt*-format by not allowing negative premises in the rules. The GSOS-format [9] has been defined in example 3.1. It is a simplification of the pure *ntyft*-format in the sense that rules in GSOS-format only have conclusions of the form $f(x_1,\ldots,x_{r(f)}) \xrightarrow{a} t$ and premises of the form $x_i \xrightarrow{a} x_i'$ for $1 \leq i \leq r(f)$ and $x_j \stackrel{b}{\not\rightarrow}$ for $1 \leq j \leq r(f)$. In example 3.1 it has been shown that a TSS in GSOS-format has a unique associated transition relation.

The positive GSOS-format [12] 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 positive GSOS format is:

	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

Table 5

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

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 figure 1).
The oldest format is the De Simone-format [26]. It is equal to the positive GSOS-format except that it does not allow copying. Every variable in the left hand side of the conclusion may only occur once in the right hand side of the conclusion or in the left hand side of a premise. Every variable in the right hand side of a premise may appear only once in the right hand side of the conclusion.
The second and third column of table 5 give the trace and completed trace congruences belonging to these formats. The notion of completed trace congruences was not yet defined:

8.1. Definition. Let $P = (\Sigma, A, R)$ be a TSS with associated transition relation $\longrightarrow_P$. Let $s \in T(\Sigma)$. s is a *deadlocked process*, notation $s \not\longrightarrow$, if there are no $t \in T(\Sigma)$ and $a \in A$ with $s \xrightarrow{a}_P t$. A sequence $a_1 \star \cdots \star a_n \in A^*$ is a *completed trace* of s if there are process terms $s_1, \ldots, s_n \in T(\Sigma)$ such that $s \xrightarrow{a_1}_P s_1 \xrightarrow{a_2}_P \cdots \xrightarrow{a_n}_P s_n \not\longrightarrow$. $CT(s)$ is the set of all completed traces of s. Two process terms $s, t \in T(\Sigma)$ are *completed trace equivalent for P* if $CT(s) = CT(t)$. This is denoted as $s \equiv_P^{CT} t$.

The notion of *completed trace congruence* can be obtained by replacing 'trace' by 'completed trace', $\equiv_{\mathfrak{S}}^{T}$ by $\equiv_{\mathfrak{S}}^{CT}$ and $\equiv_P^T$ by $\equiv_P^{CT}$ in definition 7.2.

The trace and completed trace congruences for the De Simone-format follow directly from an important result of R. de Simone: 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 [12] where all completed trace congruences, except for the pure *ntyft*/*ntyxt*-format, are given. The notion of 2/3-bisimulation was first mentioned in [15] and simulation equivalence and 2-nested simulation equivalence are defined in [12]. The trace congruences for positive GSOS and GSOS are not published anywhere. However, with the help of the lemmas in [12] one can prove the results. In [12] it is shown that the trace congruence for the pure *tyft*/*tyxt*-format is simulation equivalence.

References

[1] S. Abramsky (1987): *Observation equivalence as a testing equivalence.* Theoretical Computer Science 53, pp. 225-241.

[2] K.R. Apt (1988): *Introduction to logic programming.* Report CS-R8826, Centrum voor Wiskunde en Informatica, Amsterdam, to appear in: Handbook of Theoretical Computer Science, (J. van

Leeuwen, Managing Editor), North-Holland.

[3] E. ASTESIANO, C. BENDIX NIELSEN, N. BOTTA, A. FANTECHI, A. GIOVINI, P. INVERARDI, E. KARLSEN, F. MAZZANTI, G. REGGIO & E. ZUCCA (1986): *The Trial Definition of Ada, Deliverable 7 of the CEC MAP project: The Draft Formal Definition of ANSI/MIL-STD 1815 Ada.*

[4] D. AUSTRY & G. BOUDOL (1984): *Algèbre de processus et synchronisations.* Theoretical Computer Science 30(1), pp. 91-131.

[5] J.C.M. BAETEN & J.A. BERGSTRA (1988): *Global renaming operators in concrete process algebra.* I&C 78(3), pp. 205-245.

[6] J.C.M. BAETEN, J.A. BERGSTRA & J.W. KLOP (1986): *Syntax and defining equations for an interrupt mechanism in process algebra.* Fund. Inf. IX(2), pp. 127-168.

[7] J.A. BERGSTRA (1985): *Put and get, primitives for synchronous unreliable message passing.* Logic Group Preprint Series Nr. 3, CIF, State University of Utrecht.

[8] G. BERRY & G. GONTHIER (1988): *The synchronous programming language ESTEREL: design, semantics, implementation.* Report 842, INRIA, Centre Sophia-Antipolis, Valbonne Cedex, to appear in Science of Computer Programming.

[9] B. BLOOM, S. ISTRAIL & A.R. MEYER (1988): *Bisimulation can't be traced: preliminary report.* In: Conference Record of the 15^{th} ACM Symposium on Principles of Programming Languages (POPL), San Diego, California, pp. 229-239.

[10] R. CLEAVELAND & M. HENNESSY (1988): *Priorities in process algebra.* In: Proceedings 3^{th} Annual Symposium on Logic in Computer Science (LICS), Edinburgh, pp. 193-202.

[11] R.J. VAN GLABBEEK (1987): *Bounded nondeterminism and the approximation induction principle in process algebra.* In: Proceedings STACS 87 (F.J. Brandenburg, G. Vidal-Naquet & M. Wirsing, eds.), LNCS 247, Springer-Verlag, pp. 336-347.

[12] J.F. GROOTE & F.W. VAANDRAGER (1988): *Structured operational semantics and bisimulation as a congruence.* Report CS-R8845, Centrum voor Wiskunde en Informatica, Amsterdam, extended abstract appeared in: Proceedings ICALP 89, Stresa (G. Ausiello, M. Dezani-Ciancaglini & S. Ronchi Della Rocca, eds.), LNCS 372, Springer-Verlag, pp. 423-438.

[13] ISO (1987): *Information processing systems - open systems interconnection - LOTOS - a formal description technique based on the temporal ordering of observational behaviour.* ISO/TC 97/SC 21 N DIS8807.

[14] R. LANGERAK (1989): *A testing theory for LOTOS using deadlock detection.* In: Proceedings 9^{th} IFIP WG6.1 International Symposium on Protocol Specification, Testing, and Verification, June 1989, Enschede, The Netherlands (E. Brinksma, G. Scollo & C.A. Vissers, eds.).

[15] K.G. LARSEN & A. SKOU (1988): *Bisimulation through probabilistic testing.* R 88-29, Institut for Elektroniske Systemer, Afdeling for Matematik og Datalogi, Aalborg Universitetscenter.

[16] R. MILNER (1980): *A Calculus of Communicating Systems*, LNCS 92, Springer-Verlag.

[17] R. MILNER (1983): *Calculi for synchrony and asynchrony.* Theoretical Computer Science 25, pp. 267-310.

[18] R. MILNER (1989): *Communication and concurrency*, Prentice-Hall International.

[19] F. MOLLER (1989): *Axioms for concurrency.* Ph.D. Thesis, Report CST-59-89, Department of Computer Science, University of Edinburgh.

[20] D.M.R. PARK (1981): *Concurrency and automata on infinite sequences.* In: Proceedings 5^{th} GI Conference (P. Deussen, ed.), LNCS 104, Springer-Verlag, pp. 167-183.

[21] I.C.C. PHILLIPS: *CCS with Broadcast Stability.* Unpublished manuscript.

[22] G.D. PLOTKIN (1981): *A structural approach to operational semantics.* Technical Report DAIMI FN-19, Computer Science Department, Aarhus University.

[23] G.D. PLOTKIN (1983): *An operational semantics for CSP.* In: Proceedings IFIP TC2 Working Conference on Formal Description of Programming Concepts - II, Garmisch, 1982 (D. Bjørner, ed.), North-Holland, Amsterdam, pp. 199-225.

[24] A. PNUELI (1985): *Linear and branching structures in the semantics and logics of reactive systems.* In: Proceedings ICALP 85, Nafplion (W. Brauer, ed.), LNCS 194, Springer-Verlag, pp. 15-32.

[25] T.C. PRZYMUSINSKI (1987): *On the declarative semantics of deductive databases and logic programs.* In: Foundations of Deductive Databases and Logic Programming (Jack Minker, ed.), Morgan Kaufmann Publishers, Inc., Los Altos, California, pp. 193-216.

[26] R. DE SIMONE (1985): *Higher-level synchronising devices in* MEIJE-SCCS. Theoretical Computer Science 37, pp. 245-267.