A congruence theorem for structured operational semantics with predicates and negative premises

Eindhoven University of Technology

Department of Mathematics and Computing Science

A congruence theorem for structured operational semantics with predicates and negative premises

by

C. Verhoef

93/18

Computing Science Note 93/18
Eindhoven, June 1993

Copies can be ordered from:
Mrs. M. Philips
Eindhoven University of Technology
Department of Mathematics and Computing Science
P.O. Box 513
5600 MB EINDHOVEN
The Netherlands
ISSN 0926-4515

A congruence theorem for structured operational semantics with predicates and negative premises

C. VERHOEF

Department of Mathematics and Computing Science
Eindhoven University of Technology
P.O. Box 513, 5600 MB Eindhoven, The Netherlands
e-mail: chrisv@win.tue.nl

ABSTRACT. We proposed a syntactic format, the *panth* format, for structured operational semantics in which besides ordinary transitions also predicates, negated predicates, and negated transitions may occur such that if the rules are stratifiable strong bisimulation equivalence is a congruence for all the operators that can be defined within the *panth* format. To show that this format is useful we took some examples from the literature satisfying the *panth* format but no formats proposed by others. The examples touch upon issues such as priorities, termination, convergence, discrete time, recursion, (infinitary) Hennessy-Milner logic, and universal quantification.
Collation: pp. 22, ill. 2, tab. 7, ref. 24.

Key Words & Phrases: *structured operational semantics, term deduction system, transition system specification, strong bisimulation, congruence theorem, predicate, negative premise, negated predicate, stratifiable, stratification.*

1980 Mathematics Subject Classification (1985 Revision): *68Q05, 68Q55.*
CR Categories: *D.3.1, F.1.1, F.3.2, F.4.3.*

Note: *Full support received from the European Communities under CONCUR 2, BRA 7166.*

1. Introduction

In recent years, it has become a standard method to provide process algebras, process calculi, and programming and specification languages with an operational semantics in the style of Plotkin [22]. As a consequence, the Plotkin style rules themselves became an object of research. A number of so-called *formats* were proposed; a format is a syntactical constraint on the form of the rules. A central issue in the area of structured operational semantics is to define formats ensuring that some important property holds, for instance, that strong bisimulation equivalence is a congruence relation. Of course, we want such a format to be as general as possible.

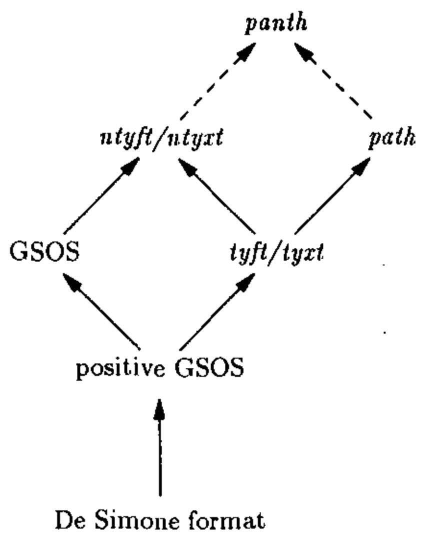

Figure 1. The lattice of formats

In this way a whole lattice of formats came into being. We depict this lattice in figure 1. An arrow from one format to another indicates that all operators definable in the first format can also be defined in the second one. If there are no arrows connecting two formats they are (syntactically) incomparable. The most basic format originates from De Simone [23]. Yet it is already powerful enough to define all the usual operators of, for instance, *CCS* or *ACP*. The GSOS format of Bloom, Istrail and Meyer [8] allows negative premises but no lookahead and the *tyft/tyxt* format of Groote and Vaandrager [15] allows lookahead but no negative premises. They both generalize the format of De Simone. The positive GSOS format is, so the speak, the greatest common divisor of the GSOS and the *tyft/tyxt* format. The *ntyft/ntyxt* format of Groote [14] is, in fact, the least common multiple of the *tyft/tyxt* format and the GSOS format: it allows both lookahead and negative premises. The *path* format of Baeten and Verhoef [5] generalizes the *tyft/tyxt* format with predicates; *path* format stands for "predicates and *tyft/tyxt* hybrid format". In this paper we discuss the *panth* format, which stands for "predicates and *ntyft/ntyxt* hybrid format". The dashed arrows in figure 1 point to it. We will not give the definitions of all the formats in the lattice except the definitions of the four formats in the upper diamond.

The main result of this paper is a congruence theorem stating that if a so-called *term deduction system* satisfies the *panth* format and is *stratifiable* then *strong bisimulation* is a congruence for all the operators that can be defined within the format. First, we will briefly explain the italics. A term deduction system is a generalization of a transition system specification [15]: it allows not only transitions but also (unary) predicates on states. The *panth* format is a syntactical constraint on a term deduction system; still we may simultaneously use transitions and predicates and their negations in the premises and the conclusion may be either a transition or a predicate. A term deduction system is stratifiable if the complexity of the conclusion of each rule is greater than the complexity of its premises. This notion is based on Groote [14]. Strong bisimulation is based on Park [21] but we require also that bisimilar processes satisfy the same predicates; cf. [5]. Now that we have an idea of the significant notions occurring in the main result we briefly discuss its proof. Baeten and Verhoef [5] already conjectured that this result could be proved in the same way as their congruence theorem for the *path* format. Indeed, this turns out to be the case: we code each predicate as a binary relation and we apply the congruence theorem of Groote [14] to the coded system. Consequently, all the operators that can be defined in the *panth* format can also be defined in Groote's *ntyft/ntyxt* format. This observation might give rise to the question if there is need for the *panth* format at all. In the following, we will motivate the need for this new format.

An advantage of the *panth* format is that it provides more syntactic freedom than other formats for defining rules since we can use transitions and predicates and negations of both, whereas in other formats we either have predicates but no negative antecedents or negative antecedents but no predicates. This is not just a theoretical advantage since there are examples of such operational semantics in the literature in which the combination of transitions and predicates with negative transitions and/or negated predicates occurs. We will sketch this in the next paragraph.

In the literature we see more and more that operational rules in the style of Plotkin are decorated with extra predicates on states to express matters like (un)successful termination, convergence, divergence [1], enabledness [7], maximal delay, side conditions [20], etc. Baeten and Verhoef give many examples of this kind of decorated transition rules in their paper on the *path* format [5] thereby showing that there is a need for a general format describing such decorated rules. Another phenomenon that we see in the literature is the use of negative antecedents in rules defining the operational semantics. We mention negative antecedents to operationally describe deadlock detection [18], sequencing [8], priorities [4] probabilistic behaviour [19], urgency [10], and various real [17] and discrete time [2] settings. Now it will not be very surprising that there are also hybrid rules using both decorations and negative antecedents (we will treat some of them in the applications). This is where the *panth* format comes into play, since these hybrid rules quite often turn out to satisfy the *panth* format and are stratifiable. Now the advantage is that we immediately have that strong bisimulation is a congruence for all the operators defined in this way, which is very practical in many cases.

The above advantage is not only of practical value but also of intuitive value since encoding rules to fit one of the known formats in order to get congruenceness in return often contraindicates the intuitive character of the original rules. Another disadvantage of such a coding trick is that there now are two transition systems that have to be shown equivalent. A fast solution to the latter problem is to throw away the original transition system, which is inadvisable in our opinion. In fact, many people prefer to use their own rules rather than encoded rules (that the reader has to decode) and choose to verify the congruence property without a general congruence theorem. We think that our *panth* format is very user-friendly in the sense that people immediately can apply our congruence result to their own rules instead of first having to become coding experts.

There are also theoretical advantages to adding predicates to known formats. For instance, Baeten and Verhoef observe that some negative antecedents can be expressed positively using predicates and pose the question which negative premises can be written positively with predicates. Vaandrager gives a partial answer: for any GSOS system there exists an equivalent positive GSOS system over the same language, extended with positive predicates. Vaandrager and Verhoef proved on a scratch paper that this result extends to the infinite case. However, in this paper we do not dive into these theoretical issues.

Now that we have given some motivation for this paper we discuss the organization of it in the remainder of this section. The paper consists of two parts: a practical and a theoretical part. This is due to the fact that we pursue two communicative targets. The first target is that we want to give rules of thumb accompanied with instructive examples for readers merely interested in applying our congruence theorem. The second target is to formally treat our theory and prove the congruence theorem; this part is for readers more interested in the theoretical side of this paper. We did not choose for a chronological ordering of our paper. In section 2 we start with the end: namely the applications. At first sight this may seem a bit unlogical but there are good reasons for this ordering. An important reason advocating this ordering is that (uninitiated) readers can see that it is not at all necessary to go through all the theory to be able to apply the congruence theorem and that mostly a few simple rules of thumb will do. Another reason justifying this ordering is that the area of application is operational semantics. Operational rules often are easy to read and, moreover, they can be understood without the theoretical part. The last and maybe most important reason for this ordering is that the reader immediately can see if his or her operational semantics has a good change to fit our format. If this is the case the time has come to read on and enter the theoretical part of this paper. An additional advantage is that those readers already have a good impression of the notions that will be made precise in the second part. This part starts in section 3 where the notions stratifiable and term deduction system are made precise. Also in this section we do our very best not to loose the reader by interspersing a running example among the abstract definitions. Following Groote **[14]** we show that stratifiability is a sufficient condition on a term deduction system to guarantee that there exists a transition relation that agrees with it. In section 4, we define the *panth* format and the notion of strong bisimulation in the presence of predicates on states. Then we state and prove our main result: the congruence theorem. The last section contains concluding remarks and discusses future work.

2. Applica'ions

In this section we give some examples that we (mostly) took from the literature. These examples turn out to satisfy the *panth* format and are stratifiable but do not satisfy formats earlier proposed. With the aid of our congruence theorem we then find that strong bisimulation is a congruence. The examples include issues such as priorities, termination, convergence, discrete time, recursion, (infinitary) Hennessy-Milner logic, and universal quantification (in particular, so-called weak predicates).

We use the first example to informally define the significant notions: the *panth* format and stratifiability.

Priorities

The first example is an operational semantics of a basic process algebra with priorities BPA_θ that originates from Baeten and Bergstra [4]; it can also be found in Baeten and Weijland [6]. In this language we have alternative and sequential composition and a priority operator (denoted $+$, $\cdot$, and θ resp.) and a set A of atomic actions. There is also a partial ordering $<$ on the set of atomic actions to express priorities. For instance, if $a < b$ and b and c are not related we have $\theta(a + b) = b$ and $\theta(b + c) = b + c$. We list the operational semantics of BPA_θ in table 1. This operational semantics is a small one; still it contains besides transitions also (postfix) predicates $\cdot \xrightarrow{a} \surd$ and both their negations. So this example is particularly suitable to informally introduce our *panth* format.

$a \xrightarrow{a} \surd$	$\dfrac{x \xrightarrow{a} x'}{x + y \xrightarrow{a} x'}$	$\dfrac{y \xrightarrow{a} y'}{x + y \xrightarrow{a} y'}$
$\dfrac{x \xrightarrow{a} \surd}{x + y \xrightarrow{a} \surd}$	$\dfrac{y \xrightarrow{a} \surd}{x + y \xrightarrow{a} \surd}$	$\dfrac{x \xrightarrow{a} x'}{xy \xrightarrow{a} x'y}$
$\dfrac{x \xrightarrow{a} \surd}{xy \xrightarrow{a} y}$	$\dfrac{x \xrightarrow{a} x',\ \{x \not\xrightarrow{b}, x \not\xrightarrow{b} \surd \mid b > a\}}{\theta(x) \xrightarrow{a} \theta(x')}$	$\dfrac{x \xrightarrow{a} \surd,\ \{x \not\xrightarrow{b}, x \not\xrightarrow{b} \surd \mid b > a\}}{\theta(x) \xrightarrow{a} \surd}$

Table 1. A Transition system for BPA_θ.

There are two conditions that must hold for a transition system before we can apply our congruence theorem. They are that the rules have to be in *panth* format and that the system has to be stratifiable. We first list the conditions for the *panth* format.

Check for each rule the following. All the transitions in the premises must end in distinct variables; denote this set by Y. If the conclusion is a transition $t \xrightarrow{a} t'$ then either $t = x$ for a variable $x \notin Y$ or $t = f(x_1, \ldots, x_n)$ with $x_1, \ldots, x_n$ distinct variables not occurring in Y. If the conclusion is of the form Pt then we treat t as above. Of course, f is an n-ary function symbol.

Now it is easy to verify that the rules of table 1 are in ***panth*** format but it will be even more easy if we also list the things that we do not have to worry about.

There is no restriction on the number of premises. There is also no restriction on terms occurring in predicates, negated predicates, and negated transitions in the premises. There is no restriction on a term occurring in the left-hand side of a transition in a premise or in the right-hand side of a conclusion.

As an example we treat the last but one rule of table 1. There is just one positive transition ending in a variable x', for the negated predicates and negative transitions there is nothing to check, since there are no restrictions on their terms. The conclusion begins with a term of the form $f(x)$ and $x \not\equiv x'$. So this rule is in ***panth*** format. The other rules are treated likewise.

Now we give the rules of thumb for the stratifiability. This condition is a bit more involved: we have to define a map, called a stratification, for which two conditions must hold for each rule instantiated with closed terms. If a stratification exists for a set of rules we call this set stratifiable. Roughly, a rule is stratifiable if the complexity of the conclusion is greater than the complexity of its premises. This complexity is measured with a stratification. The arguments that a stratification takes are positive transitions and predicates; we call them positive formulas. A stratification measures the complexity of its arguments in terms of numbers, so it ranges over numbers. To express that the complexity of the conclusion may not exceed the complexity of the premises we also have the following two conditions on a stratification S for every rule instantiated with closed terms. Let c be the conclusion of a closed instantiation of a rule and let h be a positive premise of it. Then we want that $S(h) \leq S(c)$. Now we treat the negative antecedents. Since S is only defined on positive formulas we have to turn the negative formulas into positive ones. There are two cases: first let $t \overset{a}{\not\longrightarrow}$ be a closed instantiation of a negative transition. Then we want that $S(t \xrightarrow{a} s) < S(c)$ for all closed terms s. Secondly, let $\neg Pt$ be a closed instantiation of a negated predicate P then we want that $S(Pt) < S(c)$.

Next, we will give a recipe for finding a stratification. In most cases we can find a stratification (for which the two conditions hold) by measuring the complexity of a positive formula in terms of counting a particular symbol occurring in the conclusion of a rule with negative antecedents.

As an example we give a stratification for the rules in table 1. The rules containing negative antecedents have in their conclusion a θ. We define a map that counts the number of θ's as follows: let t be a closed term with n occurrences of θ's then $S(t \xrightarrow{a} s) = S(t \xrightarrow{a} \surd) = n$. Now we check the two conditions for the last but one rule. Replace each x and x' by closed terms t and t'. Since the number of θ's occurring in $\theta(t)$ is one greater than the number of θ's occurring in t we are done. The other rules are equally simple.

Termination and convergence

The next example is an operational semantics originating from Aceto and Hennessy [1]. It is an operational semantics of a *CCS* like process algebra extended with a successful termination predicate and a convergence predicate. Their approach is to first inductively define both predicates and then define the transition relation using one of the predicates. In this semantics they use a negative premise to express unsuccessful termination. Baeten and Verhoef [5] showed that this operational semantics can be written positively by explicitly defining a third unsuccessful termination predicate. This approach is sometimes* less work than our approach, which is finding a stratification. In table 2 we list the rules for the (postfix denoted) termination predicate $\surd$.

With the aid of this predicate Aceto and Hennessy inductively define their convergence predicate $\downarrow$; we list their rules in table 3. Note the negative premise in the last rule.

Finally, Aceto and Hennessy give the rules for the non-deterministic choice $+$, the sequential composition $;$, the parallel composition $|$, the binding constructor $recx.\_$, and the encapsulation

* Especially when only the negated predicate is important. See, for instance, [3] or [7].

$nil\surd$	$\dfrac{x\surd, y\surd}{(x+y)\surd}$	$\dfrac{x\surd, y\surd}{(x;y)\surd}$
$\dfrac{x\surd, y\surd}{(x \mid y)\surd}$	$\dfrac{x\surd}{\partial_H(x)\surd}$	$\dfrac{t[recx.t/x]\surd}{recx.t\surd}$

Table 2. The rules of Aceto and Hennessy for the $\surd$.

$\delta \downarrow$	$nil \downarrow$	$\mu \downarrow, \mu \in Act_\tau$
$\dfrac{x \downarrow}{\partial_H(x) \downarrow}$	$\dfrac{t[recx.t/x] \downarrow}{recx.t \downarrow}$	$\dfrac{x \downarrow, y \downarrow}{(x+y) \downarrow}$
$\dfrac{x \downarrow, y \downarrow}{(x \mid y) \downarrow}$	$\dfrac{x\surd, y \downarrow}{(x;y) \downarrow}$	$\dfrac{\neg(x\surd), x \downarrow}{(x;y) \downarrow}$

Table 3. The rules for $\downarrow$.

operator $\partial_H(\cdot)$. We treat recursion in the same way as Groote and Vaandrager [15] by adding process names $recx.t$ to the signature for each $t \in O(\Sigma)$ to obtain that the recursion rules fit our format (we will do this in more detail in an example later on; see table 6). However, it would be a better idea to incorporate recursion within our format as is done for the GSOS format [8] and De Simone's format [23].

$\dfrac{x \xrightarrow{\mu} x'}{x + y \xrightarrow{\mu} x'}$	$\dfrac{x \xrightarrow{\mu} x'}{y + x \xrightarrow{\mu} x'}$	$\dfrac{x \xrightarrow{\mu} x'}{x; y \xrightarrow{\mu} x'; y}$
$\dfrac{x\surd, y \xrightarrow{\mu} y'}{x; y \xrightarrow{\mu} y'}$	$\dfrac{x \xrightarrow{\mu} x'}{x \mid y \xrightarrow{\mu} x' \mid y}$	$\dfrac{x \xrightarrow{\mu} x'}{y \mid x \xrightarrow{\mu} y \mid x'}$
$\dfrac{x \xrightarrow{a} x', y \xrightarrow{\bar{a}} y'}{x \mid y \xrightarrow{\tau} x' \mid y'}$	$\dfrac{x \xrightarrow{\mu} x'}{\partial_H(x) \xrightarrow{\mu} \partial_H(x')}, \mu \notin H$	$\dfrac{t[recx.t/x] \xrightarrow{\mu} x'}{recx.t \xrightarrow{\mu} x'}$

Table 4. The action relations for each $\mu \in Act_\tau$.

It is easy to see that the operational semantics consisting of the rules in tables 2–4 satisfy the ***panth*** format. We will give a stratification. We already explained that the first thing to do is to look at the rules with negative antecedents. In this case their is just one such rule. In the conclusion we see the symbol $\downarrow$. Define a map S that counts the number of $\downarrow$'s occurring in a positive formula. It is easy to see that this map is a stratification. We check the two conditions for the negative rule. Replace each x and y by closed terms t and s respectively. Since $S(t\surd) = 0 < 1 = S((t;s) \downarrow)$ the negative condition holds. For the positive condition we have $S(s \downarrow) = 1 \leq 1 = S((t;s) \downarrow)$. The other rules are also very simple.

Aceto and Hennessy are interested in rooted weak bisimulation instead of strong bisimulation, so our theorem will not directly apply to their situation. However, Baeten and Verhoef [5] show for an operational semantics of Van Glabbeek [13] for *ACP* with abstraction that rooted *weak* bisimulation is a congruence with the aid of their congruence theorem for *strong* bisimulation. We leave as an open problem whether a similar trick can also be applied for the *CCS* like process algebra of Aceto and Hennessy.

Discrete time

The next example is an operational semantics of Baeten and Bergstra [2] describing a basic process algebra with relative discrete time. In table 5 we list their rules. The $\underline{\underline{a}}$ stands for the action a in the current time slice and σ_d stands for the discrete time unit delay. It is easy to see that the rules of table 5 satisfy the *panth* format. Baeten and Bergstra apply a coding trick to obtain that their rules

$\underline{\underline{a}}\xrightarrow{a}\surd$	$\dfrac{x\xrightarrow{a}x'}{xy\xrightarrow{a}x'y}$	$\dfrac{x\xrightarrow{\sigma}x'}{xy\xrightarrow{\sigma}x'y}$
$\dfrac{x\xrightarrow{a}\surd}{xy\xrightarrow{a}y}$	$\sigma_d(x)\xrightarrow{\sigma}x$	$\dfrac{x\xrightarrow{a}x'}{x+y\xrightarrow{a}x'\xleftarrow{a}y+x}$
$\dfrac{x\xrightarrow{a}\surd}{x+y\xrightarrow{a}\surd\xleftarrow{a}y+x}$	$\dfrac{x\xrightarrow{\sigma}x',\, y\xrightarrow{\sigma}y'}{x+y\xrightarrow{\sigma}x'+y'}$	$\dfrac{x\xrightarrow{\sigma}x',\, y\overset{\sigma}{\not\to}}{x+y\xrightarrow{\sigma}x'\xleftarrow{\sigma}y+x}$

Table 5. *BPA* with discrete time.

satisfy the *ntyft/ntyxt* format of Groote and give a stratification S with $S(t\xrightarrow{a}t') = n$ if the number of function symbols of t equals n. If we just add $S(t\xrightarrow{a}\surd) = n$ with the number of function symbols of t equals n (and not encode the rules) we are done. Baeten and Bergstra still have to show that their transition system is equivalent to the encoded system.

$\dfrac{\langle t_X\|E\rangle\xrightarrow{a}y}{\langle X\|E\rangle\xrightarrow{a}y}$	$\dfrac{\langle t_X\|E\rangle\xrightarrow{\sigma}y}{\langle X\|E\rangle\xrightarrow{\sigma}y}$	$\dfrac{\langle t_X\|E\rangle\xrightarrow{a}\surd}{\langle X\|E\rangle\xrightarrow{a}\surd}$	$\dfrac{\langle t_X\|E\rangle\xrightarrow{\sigma}\surd}{\langle X\|E\rangle\xrightarrow{\sigma}\surd}$

Table 6. Recursion rules for *BPA* with discrete time.

Discrete time and recursion

We extend the above operational semantics with rules concerning recursion. The resulting example will be particularly interesting, since it deepens our insight in the notion of a stratification. Before we continue, we briefly explain some recursion terminology. We extend the signature with a set of process names with typical elements $X, Y, Z, \ldots$. A recursive specification E over a set V of process names is a set of equations of the form $X = t$ such that t is a closed term that may only contain guarded occurrences of process names in V and $X \in V$. An occurrence of a process name in a term is guarded if it occurs in a subterm of the form $a \cdot s$ or $\sigma_d(s)$. The intention of a recursive specification is to (uniquely) specify the behaviour of infinite processes. The guardedness demand is to exclude specifications that specify more than one process like $X = X$.

Now $\langle X|E\rangle$ denotes the X component of a solution of the recursive specification E. The expression $\langle t_X|E\rangle$ is short-hand for the right-hand side of the equation belonging to X with every process name Y replaced by $\langle Y|E\rangle$. So, for example, if $E = \{X = aX + \sigma_d(X)\}$ the expression $\langle aX + \sigma_d(X)|E\rangle$ is short-hand for $a\langle X|E\rangle + \sigma_d(\langle X|E\rangle)$. It is easy to see with the rules of tables 5 and 6 that $\langle X|E\rangle\xrightarrow{a}\langle X|E\rangle$ and $\langle X|E\rangle\xrightarrow{\sigma}\langle X|E\rangle$.

Our recipe for finding a stratification was to count a particular symbol occurring in the conclusion of a rule with a negative premise. In this case it is the $+$ symbol. Since there can be any finite number of $+$ symbols in the premise of a recursion rule whereas in its conclusion there is not a single $+$ symbol our approach no longer works; so a fortiori the stratification of Baeten and Bergstra will not work. We solve this by assigning to these dangerous conclusions the ordinal number $\omega = \omega_0$. Define a stratification S as follows. Let n be the number of unguarded occurrences of process names in t and let m be the number of $+$ symbols that occur in t. Let $S(t\xrightarrow{\alpha}t') = S(t\xrightarrow{\alpha}\surd) = \omega \cdot n + m$ with α either a or σ. Now it is not hard to check that the two conditions hold for S. As an example, we check a recursion rule: $S(\langle t_X|E\rangle\xrightarrow{a}y) = \omega \cdot 0 + m$, since we have forbidden unguarded occurrences in right-hand sides. Now $S(\langle X|E\rangle\xrightarrow{a}y) = \omega \cdot 1 + 0$. The other rules are also simple.

The reader can easily see that S could be chosen more minimal: it suffices to define S only on σ-transitions and let $S(t\xrightarrow{a}t') = 0$. However, the above stratification also works for all the extensions that Baeten and Bergstra discuss in their paper whereas the minimal version only works for this example. So for all their extensions we have that bisimulation equivalence is a congruence.

The above recursion problem also arises if we add recursion to our first example concerning priorities. Fortunately, it can be solved in the same way; for more information on the combination of priorities and recursion we refer to Groote [14]. In fact, we did not have this problem with the example of Aceto and Hennessy since we there counted the symbol $\downarrow$. This illustrates that it is wise not to count too much. For, if we had counted all function symbols we immediately ran into the above recursion problem.

Hennessy-Milner logic

The next example concerning Hennessy-Milner logic [16] is due to Frits Vaandrager [24]. This example shows the expressive power of the *panth* format. The set of Hennessy-Milner logic formulas over a given alphabet $A = \{a, b, \ldots\}$ is given by the following grammar: $\varphi ::= T | \varphi \wedge \varphi | \langle a \rangle \varphi | \neg \varphi$. Suppose that we have a positive transition system specification, in say *tyft/tyxt* format, defining transition relations $\xrightarrow{a}$ for each $a \in A$. With the four simple rules in table 7 we can define the satisfaction relation $\models$ within the *panth* format by defining postfix predicates $\_ \models \varphi$ for all formulas φ.

With the aid of the fundamental result of Hennessy and Milner saying that two processes are bisimilar if and only if they satisfy the same Hennessy-Milner logic formulas we also have that this extension does not change the definition of bisimulation in the presence of the satisfaction predicates.

$x \models T$	$\dfrac{x \xrightarrow{a} x',\ x' \models \varphi}{x \models \langle a \rangle \varphi}$	$\dfrac{x \models \varphi,\ x \models \psi}{x \models \varphi \wedge \psi}$	$\dfrac{\neg(x \models \varphi)}{x \models \neg \varphi}$

Table 7. The satisfaction relation $\models$ as postfix predicates $\_ \models \varphi$.

Let S be a stratification given by $S(t \xrightarrow{a} t') = 0$ and $S(t \models \varphi) = n$ $(t, t' \in C(\Sigma))$ with n the number of $\neg$ symbols occurring in φ. It is easy to see that the rules of table 7 together with our positive operational semantics defining $\cdot \xrightarrow{a} \cdot$ satisfy the *panth* format and that they are stratifiable.

We can do the same with infinitary Hennessy-Milner logic formulas. Only the third rule of table 7 changes to the following rule (I is an index set of arbitrary cardinality)

$$\frac{\{x \models \varphi_i : i \in I\}}{x \models \bigwedge_{i \in I} \varphi_i}.$$

It is easy to see that this rule satisfies the *panth* format. We have to be careful with our choice of a stratification. The above one will no longer work, since the addition of ordinals is not commutative. The stratification that we now need measures maximal nesting of $\neg$ symbols within a formula. For instance $S(x \models \neg T \wedge \neg\neg T) = 2$. We inductively define S on the postfix predicates $\_ \models \varphi$:

$$S(x \models T) = 0, \quad S(x \models \langle a \rangle \varphi) = S(x \models \varphi),$$
$$S(x \models \bigwedge_{i \in I} \varphi_i) = \sup_{i \in I} S(x \models \varphi_i), \quad S(x \models \neg \varphi) = S(x \models \varphi) + 1.$$

Note that this stratification also works for the finite case.

Universal quantification

Some predicates that we find in the literature are defined with a universal quantifier in their hypotheses. The purpose of this last example is to show the expressive power of the *panth* format by showing that it is often possible to define such predicates within our format. We illustrate this with the weak termination predicate $\surd\!\!\!/$ of Aceto and Hennessy [1]. A process p is weakly terminating ($p\surd\!\!\!/$) if for all q that cannot perform any silent moves but are reachable from p with only (zero or more) silent steps we have that $q\surd$; see table 2 for the termination predicate $\surd$. Or in a Plotkin style rule:

$$\frac{\forall q : p \xrightarrow{\epsilon} q, q \xrightarrow{\tau}\!\!\!\!\!/ \implies q\surd}{p\surd\!\!\!/}.$$

Clearly, this rule does not fit our format. This is due to the fact that our format is of an existentional nature. However, the combination of an existentional quantifier and negation leads to a universal quantifier. With this we can define the weak termination predicate $\surd\!\!\!/$ of Aceto and Hennessy within our format. We mention that $p \xrightarrow{\epsilon} q$ means that p evolves into q by performing zero or more silent actions, which can be easily defined within our format. We need an auxiliary predicate to define the weak termination predicate. The first rule below defines this auxiliary predicate which holds if the negation of the hypothesis of the above rule holds. The second rule defines the weak termination predicate by simply negating the auxiliary predicate.

$$\frac{p \xrightarrow{\epsilon} q,\ q \xrightarrow{\tau}\!\!\!\!\!/\ ,\ \neg(q\surd)}{p\overline{\surd\!\!\!/}}, \quad \frac{\neg p\overline{\surd\!\!\!/}}{p\surd\!\!\!/}.$$

We can find a stratification with a cumulative application of our recipe: count the number of $\overline{\surd\!\!\!/}$ symbols plus two times the number of $\surd\!\!\!/$ symbols in a positive formula.

In this way we can also define Aceto and Hennessy's weak convergence predicate and its parameterized version (and the resulting operational semantics is stratifiable). We recall that Aceto and Hennessy are interested in rooted weak bisimulation instead of strong bisimulation. Moreover, we think it would be a better idea to study a format that allows universal quantification.

3. Term deduction systems

The examples that we discussed in section 2 are term deduction systems. In this section we will define this notion, which generalizes the concept of a transition system specification [15]. We will also define the notion of stratifiability, which is due to Groote [14]. Following Groote, we prove that being stratifiable is a sufficient condition to define a transition relation in the presence of predicates and negative antecedents. We intersperse a running example among the abstract definitions so that the reader immediately has a concrete idea about them.

Before we continue with the definitions we will list some preliminaries for completeness sake.

We assume that we have an infinite set V of variables with typical elements $x, y, z, \ldots$. A (single sorted) signature Σ is a set of function symbols together with their arity. If the arity of a function symbol $f \in \Sigma$ is zero we say that f is a constant symbol. We restrict ourselves to signatures that contain at least one constant symbol. The notion of a term (over Σ) is defined as expected: $x \in V$ is a term; if $t_1, \ldots, t_n$ are terms and if $f \in \Sigma$ is n-ary then $f(t_1, \ldots, t_n)$ is a term. A term is also called an open term; if it contains no variables we call it closed. We denote the set of closed terms by $C(\Sigma)$ and the set of open terms by $O(\Sigma)$ (note that a closed term is also open). We also want to speak about the variables occurring in terms: let $t \in O(\Sigma)$ then $var(t) \subseteq V$ is the set of variables occurring in t.

A substitution σ is a map from the set of variables into the set of terms over a given signature. This map can easily be extended to the set of all terms by substituting for each variable occurring in an open term its σ-image.

Definition (3.1)

A term deduction system is a structure (Σ, D) with Σ a signature and D a set of deduction rules. The set $D = D(T_p, T_r)$ is parameterized with two sets, which are called respectively the set of predicate symbols and the set of relation symbols. Let s, t, and $u \in O(\Sigma)$, $P \in T_p$, and $R \in T_r$. We call expressions $Ps, \neg Ps, tRu$, and $t\neg R$ formulas. We call the formulas Ps and tRu positive and $\neg Ps$ and $t\neg R$ negative. If S is a set of formulas we write $PF(S)$ for the subset of positive formulas of S and $NF(S)$ for the subset of negative formulas of S.

A deduction rule $d \in D$ has the form

$$\frac{H}{C}$$

with H a set of formulas and C a positive formula; to save space we will also use the notation H/C. We call the elements of H the hypotheses of d and we call the formula C the conclusion of d. If the set of hypotheses of a deduction rule is empty we call such a rule an axiom. We denote an axiom simply by its conclusion provided that no confusion can arise. The notions "substitution", "*var*", and "closed" extend to formulas and deduction rules as expected.

Definition (3.2)

Let T be a term deduction system. Let $F(T)$ be the set of all closed formulas over T. We denote the set of all positive formulas over $F(T)$ by $PF(T)$ and the negative formulas by $NF(T)$. Let $X \subseteq PF(T)$. We define when a formula $\varphi \in F(T)$ holds in X; notation $X \vdash \varphi$.

$X \vdash sRt$ if $sRt \in X$,
$X \vdash Ps$ if $Ps \in X$,
$X \vdash s\neg R$ if $\forall t \in C(\Sigma) : sRt \notin X$,
$X \vdash \neg Ps$ if $Ps \notin X$.

The purpose of a term deduction system is to define a set of positive formulas that can be deduced using the deduction rules. For instance, if the term deduction system is a transition system specification then a transition relation is such a set. For term deduction systems without negative formulas this set comprises all the formulas that can be proved by a well-founded proof tree. If we allow negative formulas in the premises of a deduction rule it is no longer obvious which set of positive formulas can be deduced using the deduction rules. Bloom, Istrail, and Meyer [8] formulate that a transition relation must agree with a transition system specification. We will use their notion; it is only adapted to the framework of this paper.

Definition (3.3)

Let $T = (\Sigma, D)$ be a term deduction system and let $X \subseteq PF(T)$ be a set of positive closed formulas. We say that X agrees with T if a formula φ holds in X if and only if there is a deduction rule instantiated with a closed substitution such that the instantiated conclusion equals φ and all the instantiated hypotheses hold in X. More formally: X agrees with T if

$$\varphi \in X \iff \exists\, H/C \in D \text{ and } \sigma : V \longrightarrow C(\Sigma) \text{ such that } \sigma(C) = \varphi \text{ and } \forall h \in H : X \vdash \sigma(h).$$

Not every term deduction system defines a set of positive formulas that agrees with it. A term deduction system can define more than one set of positive formulas that agrees with it. We show this in the following two examples. Groote [14] gives similar examples with relations instead of predicates.

Example (3.4)

Let T_1 be the term deduction system that consists of one constant symbol c and one deduction rule $\neg Pc/Pc$. For all $X \subseteq PF(T_1)$ that agree with T_1 we have $Pc \in X \iff Pc \notin X$. Clearly, such an X does not exist.

Let T_2 be the term deduction system that consists of one constant symbol c and one deduction rule Pc/Pc. Then $\emptyset$ and $\{Pc\}$ both agree with T_2.

Groote [14] formulates a sufficient condition for the existence of a transition relation that agrees with a given transition system specification. We essentially follow Groote by formulating a similar condition: we incorporate predicates in his notion. Indeed, this condition is sufficient for the existence of a set of positive formulas for a given term deduction system. We obtain this result in a similar way as Groote by extending his notions with predicates and by proving his results for these extended notions.

Definition (3.5)

Let $T = (\Sigma, D)$ be a term deduction system. The formula dependency graph G of T is a labelled directed graph with as nodes positive formulas. For all deduction rules $H/C \in D$ and for all closed substitutions σ we have the following edges in G: for all $h \in PF(H)$ there is an edge $\sigma(h) \xrightarrow{p} \sigma(C)$; for all $s\neg R \in NF(H)$ there is for all $t \in C(\Sigma)$ an edge $\sigma(sRt) \xrightarrow{n} \sigma(C)$; for all $\neg Ps \in NF(H)$ there is an edge $\sigma(Ps) \xrightarrow{n} \sigma(C)$. If e is an edge of G we denote this by $e \in G$. An edge labelled with a p is called positive and if it is labelled with an n it is called a negative edge. A set of edges is called positive if all its elements are positive and negative if all they are all negative.

Example (3.6)

We depict in figure 2 the formula dependency graphs of the term deduction systems T_1, T_2 of example (3.4), and the formula dependency graph of a new one: T_3. The last term deduction system consists of a constant symbol c and for all $n \geq 0$ a deduction rule $\neg P_n c / P_{n+2} c$ and for all odd n a deduction rule $\neg P_n c / P_0 c$. The term deduction system T_3 is based on an example of Groote [14].

$$\begin{tikzcd} Pc \arrow[loop right, "n"] \end{tikzcd} \qquad \begin{tikzcd} Pc \arrow[loop right, "p"] \end{tikzcd} \qquad \begin{tikzcd} P_1c \arrow[r, "n"] \arrow[d, "n"'] & P_3c \arrow[r, "n"] \arrow[dl, "n"'] & P_5c \arrow[r, "n"] \arrow[dll, "n"'] & P_7c \arrow[dlll, "n"'] \arrow[r, dashed, no head] & {} \\ P_0c \arrow[r, "n"'] & P_2c \arrow[r, "n"'] & P_4c \arrow[r, "n"'] & P_6c \arrow[r, dashed, no head] & {} \end{tikzcd}$$

Figure 2. Three formula dependency graphs.

Definition (3.7)

A term deduction system is stratifiable if there is no node in its formula dependency graph that is the start of a backward chain of edges containing an infinite negative subset.

A term deduction system is called strictly stratifiable if there is no node in its formula dependency graph that is the start of an infinite backward chain of edges.

Definition (3.8)

Let $T = (\Sigma, D)$ be a stratifiable term deduction system and let G be its formula dependency graph. We inductively define a mapping $|\cdot|$ from the set of positive formulas of T to an ordinal α that calculates the number of negative edges in G that can be reached with a backward chain of edges beginning in φ. Note that if G contains a cycle with a negative edge we cannot define this mapping. However, we can define this mapping if G only contains positive cycles. Two formulas φ and ψ are equivalent if they occur in a cycle of the formula dependency graph G or if they are identical; notation $\varphi \sim \psi$. We write $[\varphi]$ for the equivalence class containing φ. Note that since T is stratifiable there are only positive cycles. Define $|\cdot| : PF(T)/\!\sim\ \longrightarrow \alpha$ as follows

$$|[\varphi]| = \sup(\{|[\psi]| + 1 : \psi \xrightarrow{n} \chi \in G,\ [\chi] = [\varphi]\} \cup \{|[\psi]| : \psi \xrightarrow{p} \chi \in G,\ [\psi] \neq [\varphi] = [\chi]\}).$$

We assume that $\sup(\emptyset) = 0$. Now define $|\varphi| = |[\varphi]|$.

Example (3.9)

With the aid of figure 2 we see that T_1 is not stratifiable and that T_2 and T_3 are stratifiable. It is not hard to see that for the term deduction system T_3 we have $|P_{2n+1}c| = n$ and $|P_{2n}c| = \omega + n$ for all $n \geq 0$.

Definition (3.10)

Let $T = (\Sigma, D)$ be a term deduction system. A mapping $S : PF(T) \longrightarrow \alpha$ for an ordinal α is called a stratification for T if for all deduction rules $H/C \in D$ and closed substitutions σ the following conditions hold. For all $h \in PF(H)$ we have $S(\sigma(h)) \leq S(\sigma(C))$; for all $s \neg R \in NF(H)$ we have for all $t \in C(\Sigma) : S(\sigma(sRt)) < S(\sigma(C))$; for all $\neg Ps \in NF(H)$ we have $S(\sigma(Ps)) < S(\sigma(C))$. A stratification is called strict if we in addition have that $S(\sigma(h)) < S(\sigma(C))$ for all $h \in PF(H)$.

Lemma (3.11)

A term deduction system is (strictly) stratifiable if and only if there exists a (strict) stratification for it.

Proof. First, we prove the lemma for the non-strict case. Assume that T is stratifiable. Then we can define the mapping $|\cdot| : PF(T) \longrightarrow \alpha$ for an ordinal α. It is not hard to see that $|\cdot|$ is a stratification for T.

Assume that there exists a stratification S for a term deduction system T. Let G be the formula dependency graph of T. We show that for all nodes φ in G with $S(\varphi) = \alpha$ there is no backward chain of edges with an infinite negative subset. Suppose that this holds for all $\alpha < \beta$. We prove it for β. Let φ be a node in G with $S(\varphi) = \beta$ and suppose that there is a backward chain of edges containing an infinite negative subset. Then there is a path $\varphi \xleftarrow{p} \varphi_1 \xleftarrow{p} \varphi_2 \cdots \xleftarrow{p} \varphi_n \xleftarrow{n} \psi \cdots$ and for ψ there is a backward chain with an infinite negative subset. It is easy to see that $S(\psi) < \beta$ so with induction we find that ψ has no backward chain containing an infinite subset, which contradicts our assumption on φ.

Now we prove the lemma for the strict case. Suppose that T is strictly stratifiable. Let G be its formula dependency graph. Since there is no node with an infinite backward chain of edges we can define a map S that counts the number of incoming edges as follows

$$S(\varphi) = sup\{S(\psi) + 1 : \psi \xrightarrow{n \text{ or } p} \varphi \in G\}.$$

It is not hard to see that S is a strict stratification.

The other direction is easy. It can be proved analogously to the non-strict case.

We need the next definition and lemma to construct a set of positive formulas that agrees with a given term deduction system. We define a mapping, called the degree, that assigns to a term deduction system an ordinal number with a property (called regularity) that is proved in the lemma. We assume that the axiom of choice holds for this definition and for the lemma, since we we assume that the only cardinal numbers that exist are the natural numbers or $\aleph_\alpha$ for all ordinal numbers $\alpha \geq 0$. We recall that an ordinal number is a transitive set of transitive sets and that for all ordinal numbers $\alpha \geq 0$ the cardinality of the (initial) ordinal number ω_α equals $\aleph_\alpha$ (by definition).

Definition (3.12)

Let V be a set. If $0 \leq |V| < \aleph_0$ we define $d(V) = \omega_0$. If $|V| = \aleph_\alpha$ for an ordinal $\alpha \geq 0$ we define $d(V) = \omega_{\alpha+1}$.

Let $T = (\Sigma, D)$ be a term deduction system. The degree $d(H/C)$ of a deduction rule $H/C \in D$ is the degree of its set of positive premises: $d(H/C) = d(PF(H))$. Let $\omega_\alpha = \sup\{d(H/C) : H/C \in D\}$. The degree $d(T)$ of a term deduction system T is ω_0 if $\alpha = 0$ and $\omega_{\alpha+1}$ otherwise.

Example (3.13)

Let T_3 be as in example (3.6). It is easy to see that $d(T_3) = \omega_0$.

The following lemma is inserted for the readers that are not familiar with the notion of regularity of a cardinal number and thus with the fact that $\aleph_0$ and $\aleph_{\alpha+1}$ are regular for all $\alpha \geq 0$ (if we assume the axiom of choice).

Lemma (3.14)

Let $|I| < \aleph_\alpha$ with α an ordinal in $\{0\} \cup \{\gamma + 1 : \gamma \geq 0\}$. Suppose that for all $i \in I$ we have an ordinal $\alpha_i < \omega_\alpha$. Then there is an ordinal number β such that for all $i \in I : \alpha_i < \beta < \omega_\alpha$.

Proof. We assume the axiom of choice. Let $\beta = \sup\{\alpha_i : i \in I\} + 1$. Then we clearly have for all $i \in I$ that $\alpha_i < \beta$. So we only need to show that $\beta < \omega_\alpha$. The case $\alpha = 0$ is trivial, so let $\alpha = \gamma + 1$ for an ordinal $\gamma \geq 0$ and suppose that $\beta \geq \omega_\alpha$. Then $|\beta| \geq \aleph_{\gamma+1}$. It is easy to see that $|\beta| = |\sup\{\alpha_i : i \in I\}| \leq |I| \cdot \aleph_\gamma \leq \aleph_\gamma^2 = \aleph_\gamma < \aleph_{\gamma+1}$. This contradicts our assumption so $\beta < \omega_\alpha$.

Next, we will define a set of positive formulas from which we will show that it agrees with a given term deduction system.

Definition (3.15)

Let $T = (\Sigma, D)$ be a term deduction system and let $S : PF(T) \longrightarrow \alpha$ be a stratification for an ordinal number α. We define a set $T_S \subseteq PF(T)$ as follows.

$$T_S = \bigcup_{i<\alpha} T_i^S, \quad T_i^S = \bigcup_{j<d(T)} T_{i,j}^S.$$

We will frequently use unions over T_i^S and $T_{i,j}^S$; therefore, we define the following notations

$$U_i^S = \bigcup_{i'<i} T_{i'}^S \ (i \leq \alpha), \quad U_{i,j}^S = \bigcup_{j'<j} T_{i,j'}^S \ (j \leq d(T)).$$

We drop the superscripts S, provided that no confusion arises. Now we define for all $i < \alpha$ and for all $j < d(T)$ the set $T_{i,j} = T_{i,j}^S$:

$$T_{ij} = \Big\{\varphi \mid S(\varphi) = i,\ \exists H/C \in D \text{ and } \sigma : V \longrightarrow C(\Sigma) \text{ with } \sigma(C) = \varphi,$$
$$\forall h \in PF(H) : U_{i,j} \cup U_i \vdash \sigma(h) \text{ and } \forall h \in NF(H) : U_i \vdash \sigma(h)\Big\}.$$

Example (3.16)

Let $T = T_3$ be the term deduction system of example (3.6). Let $S : PF(T) \longrightarrow \omega_0 \cdot 2$ be the strict stratification that we defined in lemma (3.11). Note that $S = |\cdot|$ in this example; see definition (3.8). We will calculate T_S so it suffices to calculate $T_{i,j}$ for all $i < \alpha$ and $j < d(T)$. Since there are no positive premises we have that $T_{i,0} = T_{i,j}$ for all $j < d(T)$. So $T_i = T_{i,0}$. It is not hard to verify that for all $n \geq 0$ we have $T_{2n} = T_{\omega_0+2n+1} = \emptyset$, $T_{2n+1} = \{P_{4n+3}c\}$, and $T_{\omega_0+2n} = \{P_{4n}c\}$. So we find that $T_S = \{P_0c, P_3c, P_4c, P_7c, P_8c, P_{11}c, \ldots\}$.

Theorem (3.17)

Let $T = (\Sigma, D)$ be a term deduction system and let $S : PF(T) \longrightarrow \alpha$ be a stratification for an ordinal number α. Then T_S agrees with T.

Proof. First, assume that $T_S \vdash \varphi$. Then there are $i < \alpha$ and $j < d(T)$ with $\varphi \in T_{i,j}$. So there is a deduction rule H/C and a closed substitution σ with $\sigma(C) = \varphi$. It follows immediately that $T_S \vdash \sigma(h)$ for all $h \in PF(H)$. Now let $h \in NF(H)$ then $U_i \vdash \sigma(h)$. Since $S(\sigma(h)) < S(\sigma(C))$ we also have that $\sigma(h) \notin T_{i'}$ for all $i \leq i' < \alpha$. So $T_S \vdash \sigma(h)$.

Now assume that there is a deduction rule $H/C \in D$ and a closed substitution with $T_S \vdash \sigma(h)$ for all $h \in H$. We show that $T_S \vdash \sigma(C)$. Let $i = S(\sigma(C))$. It suffices to show that there is a $j < d(T)$ with $T_{i,j} \vdash \sigma(C)$. First, let $h \in NF(H)$. Since $T_S \vdash \sigma(h)$ and $U_i \subseteq T_S$ we also have $U_i \vdash \sigma(h)$. Second, let $h \in PF(H)$. Since $S(\sigma(h)) \leq i$ we have $\sigma(h) \in U_i \cup T_i$. For all $h \in PF(H)$ with $\sigma(h) \in T_i$ there is a $j_h < d(T)$ such that $\sigma(h) \in T_{i,j_h}$. Since $|PF(H)| < |d(H/C)| \leq |d(T)|$ we find using lemma (3.14) that there exists a $j < d(T)$ with for all $h \in PF(H) : j_h < j < d(T)$. So for all $h \in PF(H)$ we have $U_i \cup U_{i,j} \vdash \sigma(h)$. By definition of $T_{i,j}$ we find that $T_{i,j} \vdash \sigma(C)$.

Theorem (3.18)

There exists a term deduction system with an agreeing set of positive formulas that is not stratifiable.

Proof. See Bol and Groote [9]. They show that this is already the case for a term deduction system without predicates.

Theorem (3.19)

Let T be a term deduction system and let S and S' be stratifications for T. Then $T_S = T_{S'}$.

Proof. Let $T = (\Sigma, D)$ and let S and S' be stratifications for T and suppose that $T_S \neq T_{S'}$. Let, for instance $\varphi \in T_S \setminus T_{S'}$; the other case is treated analogously. Choose φ minimal in the following sense: for all $\psi \in T_S \setminus T_{S'} \cup T_{S'} \setminus T_S$ we have $S(\varphi) \leq S(\psi)$. Let $i = S(\varphi)$. Then there is a $j < d(T)$ with $\varphi \in T^S_{i,j}$. Choose this j minimal in the following sense: for all $\psi \in T_S \setminus T_{S'}$ with $S(\psi) = i$ we have that $\psi \in T^S_{i,j'}$ with $j' \geq j$. Since T_S agrees with T there is a rule $H/C \in D$ and a closed substitution σ with $\sigma(C) = \varphi$ and $T_S \vdash \sigma(h)$ for all $h \in H$. If also $T_{S'} \vdash \sigma(h)$ for all $h \in H$ we would have that $\varphi \in T_{S'}$, so there is an $h \in H$ with $T_{S'} \not\vdash \sigma(h)$. If $h \in PF(H)$ then $\sigma(h) \notin T_{S'}$ but also $\sigma(h) \in U^S_i \cup U^S_{i,j}$, which contradicts the minimality of φ with respect to i or j. If $h = s\neg R$ there is an $s' \in C(\Sigma)$ with $\sigma(sRs') \in T_{S'}$. Since $S(\sigma(sRs')) < i$ and $\sigma(sRs') \notin T_S$ this contradicts the minimality of φ with respect to i. If $h = \neg Ps$, we immediately find a contradiction with the minimality of φ with respect to i since $s(\sigma(Ps)) < i$ and $Ps \notin T_S$.

Theorem (3.20)

Let T be a strictly stratifiable term deduction system then there is at most one set of closed positive formulas that agrees with T.

Proof. Let $T = (\Sigma, D)$ be a term deduction system and let $S : PF(T) \longrightarrow \alpha$ be a strict stratification for T. Suppose that X and Y agree with T and that $X \neq Y$, say $\varphi \in X \setminus Y$; the other case is treated analogously. Choose φ minimal in the following sense: for all $\psi \in X \setminus Y \cup Y \setminus X$ we have $i = S(\varphi) \leq S(\psi)$. Since X agrees with T and $X \vdash \varphi$ there is a deduction rule $H/C \in D$ and a closed substitution σ with $\sigma(C) = \varphi$ and $X \vdash \sigma(h)$ for all $h \in H$. Note that $S(\sigma(h)) < i$ for all $h \in H$ since S is strict. If $Y \vdash \sigma(h)$ for all $h \in H$ we would have $\varphi \in Y$ so there is an $h \in H$ with $Y \not\vdash \sigma(h)$. If $h \in PF(H)$ we find that $\sigma(h) \in X \setminus Y$, which contradicts the minimal choice of φ. If $h = s\neg R \in NF(H)$ then there is a $s' \in C(\Sigma)$ with $\sigma(sRs') \in Y \setminus X$, which also contradicts the minimality of φ. If $h = \neg Ps \in NF(H)$ we have $\sigma(Ps) \in Y \setminus X$, which is also impossible.

Corollary (3.21)

If S is a strict stratification for a term deduction system T then T_S is the unique set of positive formulas that agrees with T.

Definition (3.22)

Let T and T' be stratifiable term deduction systems. Let S and S' be stratifications for T and T'. If T and T' have the same signature and if $T_S = T'_{S'}$ we say that T and T' are equivalent.

4. The congruence theorem

In this section we will formally define the *panth* format and other notions necessary to state the congruence theorem and then we will prove it. We expect that this result can be proved by adapting the congruence theorem of Groote to our situation. However, we prove a stronger result since we moreover show that every term deduction system in *panth* format can be reduced to a term deduction system in *ntyft/ntyxt* format.

Definition (4.1)

Let $T = (\Sigma, D)$ be a term deduction system with $D = D(T_p, T_r)$. Let in the following K, L, M, and N be index sets of arbitrary cardinality, let $s_k, t_l, u_m, v_n, t \in O(\Sigma)$ for all $k \in K$, $l \in L$, $m \in M$, and $n \in N$, let $P_k, P_m, P \in T_p$ be predicate symbols for all $k \in K$ and $m \in M$, and let $R_l, R_n, R \in T_r$ be relation symbols for all $l \in L$ and $n \in N$.

A deduction rule $d \in D$ is in *pntyft* format if it has the form

$$\frac{\{P_k s_k : k \in K\} \cup \{t_l R_l y_l : l \in L\} \cup \{\neg P_m u_m : m \in M\} \cup \{v_n \neg R_n : n \in N\}}{f(x_1, \ldots, x_n) R t}$$

with $f \in \Sigma$ an n-ary function symbol and $X \cup Y = \{x_1, \ldots, x_n\} \cup \{y_i : i \in I\} \subseteq V$ a set of distinct variables. If $var(d) = X \cup Y$ we call d pure. A variable in $var(d)$ that does not occur in $X \cup Y$ is called free.

A deduction rule $d \in D$ is in *pntyxt* format if it has the form

$$\frac{\{P_k s_k : k \in K\} \cup \{t_l R_l y_l : l \in L\} \cup \{\neg P_m u_m : m \in M\} \cup \{v_n \neg R_n : n \in N\}}{xRt}$$

with $X \cup Y = \{x\} \cup \{y_i : i \in I\} \subseteq V$ a set of distinct variables. If $var(d) = X \cup Y$ we call d pure. A variable in $var(d)$ that does not occur in $X \cup Y$ is called free.

A deduction rule is in *pntyf* format if it has the form

$$\frac{\{P_k s_k : k \in K\} \cup \{t_l R_l y_l : l \in L\} \cup \{\neg P_m u_m : m \in M\} \cup \{v_n \neg R_n : n \in N\}}{Pf(x_1, \ldots, x_n)}$$

with $f \in \Sigma$ an n-ary function symbol and $\{x_1, \ldots, x_n\} \cup \{y_i : i \in I\} \subseteq V$ a set of distinct variables. The notions pure and free are defined as expected.

A deduction rule is in *pntyx* format if it has the form

$$\frac{\{P_k s_k : k \in K\} \cup \{t_l R_l y_l : l \in L\} \cup \{\neg P_m u_m : m \in M\} \cup \{v_n \neg R_n : n \in N\}}{Px}$$

with $\{x\} \cup \{y_i : i \in I\} \subseteq V$ a set of distinct variables. The notions pure and free are defined as expected.

We explain the names of the deduction rules. The p in the phrases *pntyft*, *pntyxt*, *pntyf*, and *pntyx* refers to the predicates occurring in the rules, the n refers to the presence of negative formulas in the premises, the *ty* refers to the positive relation part in the set of hypotheses, and the *ft*, *xt*, *f*, and *x* refer to the various conclusions. The names *ntyft* and *ntyxt* are taken from Groote [14].

If a deduction rule $d \in D$ has one of the above forms we say that this rule is in ***panth*** format, which stands for "predicates and *ntyft/ntyxt* hybrid format". A term deduction system is in ***panth*** format if all its rules are. A term deduction system is called pure if all its rules are pure.

A term deduction system is in *ntyft/ntyxt* format if it is in ***panth*** format and its set of predicate symbols is empty. The *ntyft/ntyxt* format originates from Groote [14]. A term deduction system is in ***path*** format if it is in *panth* format and there are no negative formulas in the rules. The ***path*** format originates from Baeten and Verhoef [5]. A term deduction system is in *tyft/tyxt* format if it is in ***path*** format and its set of predicate symbols is empty. The *tyft/tyxt* format originates from Groote and Vaandrager [15].

We need the technical notion of well-foundedness of a term deduction system, which will be used in the proof of the congruence theorem. The notion of well-foundedness is taken from Groote and Vaandrager [15], where it is also used in the proof of their congruence theorem. The same phenomenon occurs in Groote's paper [14] and in Baeten and Verhoef [5]. It is an open question whether the requirement of well-foundedness is really necessary. Recent work of Fokkink [12] seems to show that the well-foundedness is not necessary for the congruence theorem of Groote and Vaandrager [15] and that this seems to generalize to our format. Therefore, we omitted the well-foundedness demand in the examples we discussed in section 2. But all these examples are well-founded.

Definition (4.2)

Let $T = (\Sigma, D)$ be a term deduction system and let F be a set of formulas. The variable dependency graph of F is a directed graph with variables occurring in F as its nodes. The edge $x \longrightarrow y$ is an edge of the variable dependency graph if and only if there is a positive relation $tRs \in F$ with $x \in var(t)$ and $y \in var(s)$.

The set F is called well-founded if any backward chain of edges in its variable dependency graph is finite. A deduction rule is called well-founded if its set of hypotheses is so. A term deduction system is called well-founded if all its deduction rules are well-founded.

The following lemma and its proof are essentially due to Groote [14].

Lemma (4.3)

For every well-founded stratifiable term deduction system in ***panth*** format there is an equivalent pure well-founded term deduction system in ***panth*** format.

Proof. Let $T = (\Sigma, D)$ be a well-founded stratifiable term deduction system in ***panth*** format. Let $T' = (\Sigma, D')$ be defined as follows. The deduction rules of D' are the rules of D without free variables plus for each rule with free variables a set of new rules. Let $d \in D$ be a deduction rule with free variables then such a set of new rules is the set containing a new deduction rule for every possible substitution of closed terms for the free variables in d. Clearly, T' is pure, well-founded and in ***panth*** format. Let $S : PF(T) \longrightarrow \alpha$ be a stratification for T, then S is also a stratification for T', so T' is stratifiable. It remains to check that T and T' are equivalent. So we have to verify that $T_S = T'_S$.

It suffices to prove that

$$\forall i < \alpha : T_i = T'_i. \tag{1}$$

We prove (1) with transfinite induction on i. So suppose that (1) holds for all $i' < i$; we prove it for i. Note that $d(T) = d(T')$. To prove (1) for i it suffices to show that

$$\forall j < d(T) : T_{i,j} = T'_{i,j}. \tag{2}$$

We prove (2) with transfinite induction on j. So suppose that (2) is valid for all $j' < j$; we prove it for j.

First, let $\varphi \in T_{i,j}$. By definition of $T_{i,j}$ there is a rule $H/C \in D$ and a closed substitution σ with $\sigma(C) = \varphi$. If $h \in PF(H)$ then $U_{i,j} \cup U_i \vdash \sigma(h)$. With the induction hypotheses for i or j we find that $U'_{i,j} \cup U'_i \vdash \sigma(h)$. If $h \in NF(H)$ then $U_i \vdash \sigma(h)$. With the induction hypothesis on i we also have that $U'_i \vdash \sigma(h)$. Now we only need to find an appropriate deduction rule $d' \in D'$. Let ρ be the substitution with $\rho(x) = \sigma(x)$ for all free variables in d and let $\rho(x) = x$ otherwise. It is easy to see that $\sigma \circ \rho(x) = \sigma(x)$ for all $x \in V$ and that $\rho(H/C) \in D'$. So by definition of $T'_{i,j}$ we find that $\varphi \in T'_{i,j}$.

Second, suppose that $\varphi \in T'_{i,j}$. By definition of $T'_{i,j}$ there is a deduction rule $\rho(H/C) \in D'$ and a closed substitution σ' with $\sigma' \circ \rho(C) = \varphi$. Let $H/C \in D$ be the original rule. With the induction hypotheses for i or j we find for all $h \in PF(H)$ that $U_{i,j} \cup U_i \vdash \sigma' \circ \rho(h)$ and using induction on i we find for all $h \in NF(H)$ that $U_i \vdash \sigma' \circ \rho(h)$. So by definition of $T_{i,j}$ we find that $\sigma' \circ \rho(C) = \varphi \in T_{i,j}$.

Next, we will define the notion of strong bisimulation, which is based on Park [21]. See also Baeten and Verhoef [5].

Definition (4.4)

Let $T = (\Sigma, D)$ be a stratifiable term deduction system with stratification S and let $D = D(T_p, T_r)$. A binary relation $B \subseteq C(\Sigma) \times C(\Sigma)$ is called a (strong) bisimulation if for all $s, t \in C(\Sigma)$ with sBt the following conditions hold. For all $R \in T_r$

$$\forall s' \in C(\Sigma)\ (T_S \vdash sRs' \Rightarrow \exists t' \in C(\Sigma) : T_S \vdash tRt' \wedge s'Bt'),$$
$$\forall t' \in C(\Sigma)\ (T_S \vdash tRt' \Rightarrow \exists s' \in C(\Sigma) : T_S \vdash sRs' \wedge s'Bt'),$$

and for all $P \in S_p$

$$T_S \vdash Ps \Leftrightarrow T_S \vdash Pt.$$

The first two conditions are known as the transfer property. Two states $s, t \in C(\Sigma)$ are bisimilar if there exists a bisimulation relation containing the pair (s, t). If s and t are bisimilar we write $s \sim t$. Note that bisimilarity is an equivalence relation.

Theorem (4.5)

Let $T = (\Sigma, D)$ be a well-founded stratifiable term deduction system in *panth* format then strong bisimulation is a congruence for all function symbols occurring in Σ.

Proof. The structure of this proof resembles the proof of the congruence theorem of Baeten and Verhoef [5].

Groote [14] proved this theorem in the case that the set of predicate symbols is empty, that is, if the term deduction system is in *ntyft/ntyxt* format. Our strategy to prove the non-empty case is to construct from a term deduction system a new one without predicates with the property that two terms are bisimilar in the old term deduction system if and only if they are bisimilar in the new one. We make the new term deduction system from the old one by coding each predicate symbol in the old system as a special relation symbol in the modified one.

To begin with, we construct from a term deduction system a new one by extending the original signature with a xenoconstant and moving the predicates of the original system to xenorelations.

Let $T = (\Sigma, D)$ be a well-founded stratifiable term deduction system in *panth* format and suppose that $D = D(T_p, T_r)$ with $T_p \neq \emptyset$. In accordance with lemma (4.3) we may assume that T is pure. We define a new term deduction system $T' = (\Sigma', D')$. Let ξ be a constant function symbol that is strange to Σ and define $\Sigma' = \Sigma \cup \{\xi\}$. Let $D' = D'(\emptyset, T_r')$ with $T_r' = T_r \cup \{R_P \mid P \in T_p\}$ (disjoint union). A relation symbol R_P for $P \in T_p$ is defined as follows. For two terms s and t over Σ' we will have sR_Pt if and only if Ps and $t = \xi$. The set of deduction rules is $D' = \{d' \mid d \in D\}$ and a deduction rule d' is constructed from an old rule $d \in D$ as follows. Let $d = H/C$. The set of hypotheses of d' is the set H but with the positive predicates $\{P_k s_k : k \in K\}$ replaced by $\{s_k R_{P_k} z_k : k \in K\}$ with $\{z_k : k \in K\}$ a set of distinct variables disjoint with $var(d)$ and the negative predicates $\{\neg P_m u_m : m \in M\}$ replaced by $\{u_m \neg R_{P_m} : m \in M\}$. If the conclusion of the old rule is of the form Pt then the conclusion of the new rule is $tR_P\xi$. Otherwise C remains the same. Note that T' is pure, well-founded and in *ntyft/ntyxt* format.

Next, we stratify the xenosystem by assigning to each positive xenoformula the ordinal that a stratification for the original system assigns to this positive formula with every ξ replaced by a fixed closed original term.

We verify that T' is stratifiable. Let $S : PF(T) \longrightarrow \alpha$ be a stratification for T and let c be a closed term over Σ. For all $s \in O(\Sigma')$ we inductively define a term $s[\xi/c] \in O(\Sigma)$ as follows. Let $\xi[\xi/c] = c$. If $x \in V$ is variable then $x[\xi/c] = x$. Let $f \in \Sigma$ be n-ary, and let $t_1, \ldots, t_n \in O(\Sigma')$ then $f(t_1, \ldots, t_n)[\xi/c] = f(t_1[\xi/c], \ldots, t_n[\xi/c])$. Now we can define $S : PF(T') \longrightarrow \alpha$ as follows. Let $s, t \in C(\Sigma')$ and let $R \in T_r$ and $P \in T_p$ then

$$S'(sRt) = S(s[\xi/c]Rt[\xi/c]), \quad S'(sR_Pt) = S(Ps[\xi/c]).$$

For all $\sigma' : V \longrightarrow C(\Sigma')$ we define a substitution $\sigma'[\xi/c] : V \longrightarrow C(\Sigma)$ as follows. Define $\sigma'[\xi/c](x) = \sigma'(x)[\xi/c]$ for all $x \in V$. Note that the following holds for all $s \in O(\Sigma)$ and $t \in O(\Sigma')$

$$s[\xi/c] = s, \quad \sigma'(t)[\xi/c] = \sigma'[\xi/c](t[\xi/c]). \tag{3}$$

To show that S' is a stratification for T' we have to check the conditions of definition (3.10). Let $d' = H'/C' \in D'$ be a rule and let $d = H/C \in D$ be its corresponding rule in the original system T. The deduction rule d' is of the form

$$\frac{\{s_k R_{P_k} z_k : k \in K\} \cup \{t_l R_l y_l : l \in L\} \cup \{u_m \neg R_{P_m} : m \in M\} \cup \{v_n \neg R_n : n \in N\}}{C'}. \tag{4}$$

Let $\sigma' : V \longrightarrow C(\Sigma')$ be a closed substitution and let $\sigma = \sigma'[\xi/c]$. Note that with (3) we have for all $s, t \in O(\Sigma)$ and $u \in O(\Sigma')$

$$S'(\sigma'(sRt)) = S(\sigma(sRt)), \quad S'(\sigma'(sR_Pu)) = S(\sigma(Ps)).$$

So for each type of conclusion we have $S'(\sigma'(C')) = S(\sigma(C))$. We have four cases. First, we treat the case $s_k R_{P_k} z_k$.

$$S'(\sigma'(s_k R_{P_k} z_k)) = S(\sigma(P_k s_k)) \leq S(\sigma(C)) = S'(\sigma'(C'))$$

since S is a stratification. The case $t_l R_l y_l$ is treated analogously. We treat the negative premises. Let $u'_m \in C(\Sigma')$

$$S'(\sigma'(u_m R_{P_m} u'_m)) = S(\sigma(P_m u_m)) < S(\sigma(C)) = S'(\sigma'(C'))$$

since S is a stratification. We treat the last case. Let $v'_n \in C(\Sigma')$.

$$S'(\sigma'(v_n R_n v'_n)) = S(\sigma(v_n R_n v'_n[\xi/c])) < S(\sigma(C)) = S'(\sigma'(C')).$$

Again, we used that S is a stratification. So S' is a stratification and we find that T' is stratifiable.

Now we will prove that two closed Σ-terms u and v are bisimilar in the xenosystem T' if and only if they are bisimilar in the original system T. In order to do this we use a number of properties that are listed below.

Let $P \in T_p$, $R \in T_r$, and $u, v \in C(\Sigma') \supseteq C(\Sigma)$ (unless otherwise specified) then the following properties hold for all $i < \alpha$ and $j < d(T') = d(T)$. (We denote the bisimulation relation in T by $\sim$ and in T' by $\sim'$.)

(i)	$T'_{i,j} \vdash uR_P v \Longrightarrow v = \xi$
(ii)	$u \in C(\Sigma), T'_{i,j} \vdash uRv \Longrightarrow v \in C(\Sigma)$
(iii)	$u, v \in C(\Sigma), T'_i \vdash uRv \Longrightarrow T_i \vdash uRv$
(iv)	$u \in C(\Sigma), T'_i \vdash uR_P\xi \Longrightarrow T_i \vdash Pu$
(v)	$u \in C(\Sigma), T_i \vdash Pu \Longrightarrow T'_i \vdash uR_P\xi$
(vi)	$u, v \in C(\Sigma), T_i \vdash uRv \Longrightarrow T'_i \vdash uRv$
(vii)	$u \in C(\Sigma), T_i \vdash \neg Pu \Longrightarrow T'_i \vdash u\neg R_P$
$(viii)$	$u \in C(\Sigma), T'_i \vdash u\neg R_P \Longrightarrow T_i \vdash \neg Pu$
(ix)	$u \in C(\Sigma), T_i \vdash u\neg R \Longrightarrow T'_i \vdash u\neg R$
(x)	$u \in C(\Sigma), T'_i \vdash u\neg R \Longrightarrow T_i \vdash u\neg R$
(xi)	$u, v \in C(\Sigma) \Longrightarrow (u \sim v \iff u \sim' v)$

The proof of (i) follows directly from the definitions of $T'_{i,j}$ and the deduction rules in D'.

We prove (ii) with transfinite induction on i. So suppose that (ii) holds for all $i' < i$. We show the induction step for i with transfinite induction on j. So suppose that it is valid for all $j' < j$; we prove it for j. By definition of $T'_{i,j}$ there is a rule $d' = H'/C' \in D'$ as in (4) with $C' = sRt$ and a closed substitution σ with $\sigma(s) = u$ and $\sigma(t) = v$. In particular, for all $l \in L$ we have $U'_i \cup U'_{i,j} \vdash \sigma(t_l R_l y_l)$. Since the deduction rules are pure we have that $var(t) \subseteq var(s) \cup \{y_l : l \in L\}$. So it suffices to show that $\sigma(y_l) \in C(\Sigma)$ for all $l \in L$ since $\sigma(s) \in C(\Sigma)$. We denote the set of all y_l by Y. Suppose that there is an $y_{l_0} \in Y$ with $\sigma(y_{l_0}) \in C(\Sigma') \setminus C(\Sigma)$. This contradicts the well-foundedness of the rule d', for $T'_{i'} \vdash \sigma(t_{l_0}) R_{l_0} \sigma(y_{l_0})$ for an $i' < i$ or $T'_{i,j'} \vdash \sigma(t_{l_0}) R_{l_0} \sigma(y_{l_0})$ for a $j' < j$ so with the induction hypotheses for i or j we find that $\sigma(t_{l_0}) \in C(\Sigma') \setminus C(\Sigma)$. Since t_{l_0} is a Σ-term, this must be the result of a substitution. This can only be due to a variable $y_{l_1} \in Y$. With induction on the subsubscript we find an infinite backward chain of edges $y_{l_0} \longleftarrow y_{l_1} \longleftarrow \dots$ in the variable dependency graph of d'. This concludes the induction step for j so by definition of T'_i we find that (ii) holds for i, which concludes the induction step for i.

A simple example due to Fokkink [11] shows that for this property the well-foundedness cannot be missed. Suppose that we have a signature that consists of a single constant a. We have two rules: the axiom xRx and the rule xRx/aSx. Clearly, this system is not well-founded. The xenosystem has the same rules; only the signature is extended with the xenoconstant ξ. It is easy to see that we can derive in this new system that $aS\xi$.

We simultaneously verify (iii)–(x) with transfinite induction on i. Suppose that they hold for all $i' < i$; we prove them for i by verifying the following four properties for all $j < d(T) = d(T')$.

$$u, v \in C(\Sigma), T'_{i,j} \vdash uRv \Longrightarrow T_{i,j} \vdash uRv \tag{5}$$

$$u \in C(\Sigma), T'_{i,j} \vdash uR_P\xi \Longrightarrow T_{i,j} \vdash Pu \tag{6}$$
$$u \in C(\Sigma), T_{i,j} \vdash Pu \Longrightarrow T'_{i,j} \vdash uR_P\xi \tag{7}$$
$$u, v \in C(\Sigma), T_{i,j} \vdash uRv \Longrightarrow T'_{i,j} \vdash uRv \tag{8}$$

We prove (5)–(8) with transfinite induction on j, so assume that they are valid for all $j' < j$ then we check them for j.

We begin with equation (5). Let $u, v \in C(\Sigma)$ and suppose that $T'_{i,j} \vdash uRv$. By definition of $T'_{i,j}$ there is a deduction rule $d' = H'/C' \in D'$ of the form displayed in (4) with $C' = sRt$, $s, t \in C(\Sigma)$ and a closed substitution σ with $\sigma(s) = u$ and $\sigma(t) = v$. Moreover, we have $U'_i \cup U'_{i,j} \vdash \sigma(s_k R_{P_k} z_k), \sigma(t_l R_l y_l)$ for all $k \in K$ and $l \in L$ and we have $U'_i \vdash \sigma(u_m \neg R_{P_m}), \sigma(v_n \neg R_n)$ for all $m \in M$ and $n \in N$. In order to use induction we have to know that terms like $\sigma(s_k), \sigma(t_l), \ldots \in C(\Sigma)$. It suffices to show that $\sigma(y_l) \in C(\Sigma)$ for all $l \in L$. With (ii) we find for all $l \in L$ that $\sigma(y_l) \in C(\Sigma)$ just like in the proof of (ii). Now we find using (i) and the induction hypotheses on i or j that $U_i \cup U_{i,j} \vdash \sigma(P_k s_k), \sigma(t_l R_l y_l)$ for all $k \in K$ and $l \in L$. We find using the induction hypothesis on i that $U_i \vdash \sigma(\neg P_m u_m), \sigma(v_n \neg R_n)$ for all $m \in M$ and $n \in N$. The deduction rule $d = H/C \in D$ with $C = sRt$ that corresponds with d' takes the form

$$\frac{\{P_k s_k : k \in K\} \cup \{t_l R_l y_l : l \in L\} \cup \{\neg P_m u_m : m \in M\} \cup \{v_n \neg R_n : n \in N\}}{C}. \tag{9}$$

Define $\rho : V \longrightarrow C(\Sigma)$ with $\rho(z_k) = z_k$ for all $k \in K$ and $\rho(x) = \sigma(x)$ otherwise; in particular, we have $\rho(s) = u$ and $\rho(t) = v$. By definition of $T_{i,j}$ we find that $T_{i,j} \vdash uRv$; use (3) to see that $S'(uRv) = S(uRv)$.

Equation (6) is treated analogously.

Now we treat equation (7). Let $u \in C(\Sigma)$ and assume $T_{i,j} \vdash Pu$. By definition of $T_{i,j}$ there is a deduction rule $d = H/C \in D$ of the form (9) with $C = Ps$ for some $s \in O(\Sigma)$ and there is a closed substitution σ with $\sigma(s) = u$ and $U_i \cup U_{i,j} \vdash \sigma(P_k s_k), \sigma(t_l R_l y_l)$ for all $k \in K$ and $l \in L$ and $U_i \vdash \sigma(\neg P_m u_m), \sigma(v_n \neg R_n)$ for all $m \in M$ and $n \in N$. Since $\sigma : V \longrightarrow C(\Sigma)$ we immediately find with the induction hypotheses on i or j that $U'_i \cup U'_{i,j} \vdash \sigma(s_k R_{P_k} \xi), \sigma(t_l R_l y_l)$ for all $k \in K$ and $l \in L$ and $U'_i \vdash \sigma(u_m \neg R_{P_m}), \sigma(v_n \neg R_n)$ for all $m \in M$ and $n \in N$. Let $d' = H'/C' \in D'$ be the deduction rule that corresponds with d. It takes the form displayed in (4) with $C' = sR_P\xi$. Define $\sigma' : V \longrightarrow C(\Sigma')$ by $\sigma'(z_k) = \xi$ for all $k \in K$ and $\sigma'(x) = \sigma(x)$ otherwise; note that $\sigma'(s) = u$. Since $S(Pu) = S'(uR_P\xi)$ (use (3)) we find by definition of $T'_{i,j}$ that $T'_{i,j} \vdash uR_P\xi$.

Equation (8) is verified in the same way. Now (5)–(8) are valid for all $j < d(T)$, which implies that (iii)–(vi) are valid for i. With this we can show that (vii)–(x) also hold for i.

We only treat (vii) since the cases $(viii)$–(x) are treated in the same way. Suppose that $u \in C(\Sigma)$ and $T_i \neg Pu$. Suppose that $T'_i \not\vdash u \neg R_P$ then there is a $u' \in C(\Sigma')$ with $T'_i \vdash uR_P u'$. With (i) we find that $u' = \xi$ so with (iv) for i we find that $T_i \vdash Pu$, which is a contradiction.

This concludes the induction step (on i), so (iii)–(x) are valid for all $i < \alpha$.

Now we verify (xi). Firstly, let $u \sim v$. Then there is a bisimulation relation B with uBv. Define $B' = B \cup \Delta'$, with $\Delta' = \{(t,t) : t \in C(\Sigma')\}$ the diagonal. We show that B' is a bisimulation with $uB'v$ in the new system T'. Clearly, $uB'v$. Now let $sB't$. We distinguish two cases: $s = t$ and $s \neq t$. We verify that the conditions in definition (4.4) hold for the second case since the first case is trivial. Since $s \neq t$ we have $s, t \in C(\Sigma)$ and sBt. Since there are no predicates in T' we only have to verify both transfer properties of definition (4.4). For each transfer property we have two cases: R and R_P. Now let $R \in T_r$ and suppose $T'_{S'} \vdash sRs'$ for some $s' \in C(\Sigma')$. We find with (ii) that $s' \in C(\Sigma)$ and with (iii) that $T_S \vdash sRs'$. Since sBt there is a $t' \in C(\Sigma)$ with $T_S \vdash tRt'$ and $s'Bt'$. So we obtain $s'B't'$ and with (vi) that $T'_{S'} \vdash tRt'$. The second condition in definition (4.4) is verified analogously. We check the first condition for R_P. Let $P \in T_p$ and suppose that $T'_{S'} \vdash sR_Ps'$ for some $s' \in C(\Sigma')$. We find with (i) that $s' = \xi$. So with (iv) we get $T_S \vdash Ps$. Since sBt we find $T_S \vdash Pt$ and with (v) that $T'_{S'} \vdash tR_P\xi$. Clearly $\xi B' \xi$. The second condition in (4.4) is checked analogously.

Secondly, let $u \sim' v$. Then there is a bisimulation relation B' containing the pair (u, v). Let $B = B' \cap \big(C(\Sigma) \times C(\Sigma)\big)$. We show that B is a bisimulation with uBv in the original system T.

Since $u, v \in C(\Sigma)$ we clearly have uBv. Let sBt. We check the conditions of definition (4.4). Let $R \in T_r$ and suppose that $T_S \vdash sRs'$ for some $s' \in C(\Sigma)$. With (vi) we find $T'_{S'} \vdash sRs'$. Since $sB't$ there is a $t' \in C(\Sigma')$ with $T'_{S'} \vdash tRt'$ and $s'B't'$. With (ii) we find $t' \in C(\Sigma)$ so $s'Bt'$. With (iii) we have $T_S \vdash tRt'$. The second condition in definition (4.4) is verified analogously. Next, we show that the last condition of (4.4) from left to right holds. The other direction can be shown analogously. Let $P \in T_p$ and suppose that $T_S \vdash Ps$. With (v) we find $T'_{S'} \vdash sR_P\xi$ so since $sB't$ there is a $t' \in C(\Sigma')$ with $T'_{S'} \vdash tR_Pt'$. With (i) we find $t' = \xi$ so with (iv) we find $T_S \vdash Pt$. This concludes the proof of (xi).

Now we are in a position to prove the congruence theorem. Let $f \in \Sigma$ be an n-ary function symbol. Let $u_i, v_i \in C(\Sigma)$ and $u_i \sim v_i$ for $1 \leq i \leq n$. With (xi) we find $u_i \sim' v_i$ for all $1 \leq i \leq n$. Since the term deduction system T' is well-founded stratifiable and in *ntyft/ntyxt* format we can apply the congruence theorem of Groote [14] so $f(u_1, \ldots, u_n) \sim' f(v_1, \ldots, v_n)$. Since $f(u_1, \ldots, u_n)$ and $f(v_1, \ldots, v_n)$ are Σ-terms we find with (xi) that $f(u_1, \ldots, u_n) \sim f(v_1, \ldots, v_n)$. This concludes the proof of (4.5).

Corollary (4.6)

Every term deduction system in *panth* format can be reduced to a transition system specification in *ntyft/ntyxt* format. If the term deduction system is moreover well-founded and stratifiable then bisimulation equivalence is preserved: two terms are bisimilar in the *panth* system iff they are bisimilar in the *ntyft/ntyxt* system.

5. Conclusions and future work

In this paper we presented a syntactical format, the *panth* format, for structured operational semantics with predicates and negative premises such that if the rules are stratifiable we have that strong bisimulation is a congruence for all the operators that can be defined within this format. With operational semantics mostly taken from the literature we showed that our format is useful: the examples satisfy our format but no formats proposed by others. Moreover, with these examples we informally explained the notions necessary to use our result thereby showing that it can be easily applied without scrutinizing the abstract definitions. The examples include issues such as priorities, termination, convergence, discrete time, recursion, (infinitary) Hennessy-Milner logic, and universal quantification (in particular, so-called weak predicates).

We will briefly discuss future work. Since the stratification technique is not always satisfactory (cf. (3.18)), Bol and Groote [9] proposed the more general reduction technique (for the less general *ntyft/ntyxt* format). A first possibility for future work could be to use their methods to generalize our work. A second possibility is to incorporate recursion within our framework as is done for the GSOS format [8] and De Simone's format [23]. A third generalization could be to allow universal quantification in the hypotheses.

Summarizing, we conclude that the *panth* format is useful, and that our congruence theorem is practical.

Acknowledgements

Thanks to Jos Baeten, Inge Bethke, Frank de Boer, Roland Bol, Jan Friso Groote, Hans Mulder, and Frits Vaandrager for valuable comments and interesting discussions.

6. References

[1] L. Aceto, M. Hennessy, *Termination, deadlock and divergence*, Journal of the ACM, **39**(1):147–187, Januari 1992.

[2] J. C. M. Baeten, J. A. Bergstra, *Discrete Time Process Algebra*, Proceedings CONCUR 92, Stony Brook, LNSC **630**, pp. 401–420, Springer-Verlag, 1992.

[3] J. C. M. Baeten, J. A. Bergstra, *Process algebra with a zero object*, in: J. C. M. Baeten and J. W. Klop, editors, Proceedings CONCUR 90, Amsterdam, volume **458** of Lecture Notes in Computer Science, pp. 83–98, Springer-Verlag, 1990.

[4] J. C. M. Baeten, J. A. Bergstra, *Processen en Procesexpressies* (In Dutch), Informatie, **30**(3), pp. 214–222, 1988.

[5] J. C. M. Baeten and C. Verhoef, *A congruence theorem for structured operational semantics with predicates*, Technical Report CSN 93/05, Eindhoven University of Technology, Eindhoven 1993. Note: to appear in the proceedings of CONCUR'93, Springer LNCS 1993.

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

[7] J. A. Bergstra, A. Ponse, J. J. van Wamel, *Process algebra with backtracking*, Report P9306, Programming Research Group, University of Amsterdam, 1993. Note: to appear in the proceedings of the REX workshop 1993, LNCS, Springer-Verlag.

[8] B. Bloom, S. Istrail, and A. R. Meyer, *Bisimulation can't be traced: preliminary report*, In: Proceedings 15*th* ACM Symposium on Principles of Programming Languages, San Diego, California, pp. 229–239, 1988.

[9] R. N. Bol and J. F. Groote, *The meaning of negative premises in transition system specifications*, Report CS-R9054, CWI, Amsterdam, 1990 An extended abstract appeared in J. Leach Albert, B. Monien, and M. Rodríguez Artalejo, editors, Proceedings 18*th* ICALP, Madrid, LNSC **510**, pp. 481–494, 1991

[10] T. Bolognesi and F. Lucidi, *Timed process algebras with urgent interactions and a unique powerful binary operator*, In J. W. de Bakker, C. Huizing, W. P. de Roever, and G. Rozenberg, editors, Proceedings of the REX Workshop "Real-time: Theory in Practice", LNCS **600**, pp. 124–148, 1992.

[11] W. J. Fokkink, *personal communication*, January 1993.

[12] W. J. Fokkink, *The tyft/tyxt format reduces to tree rules*, In preparation.

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

[14] J. F. Groote, *Transition system specifications with negative premises*, Report CS-R8950, CWI, Amsterdam, 1989. An extended abstract appeared in J. C. M. Baeten and J. W. Klop, editors, Proceedings CONCUR 90, Amsterdam, LNCS **458**, pp. 332–341, Springer-Verlag, 1990.

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

[16] M. Hennessy, R. Milner, *Algebraic laws for nondeterminism and concurrency*, JACM **32**(1), pp. 137–161.

[17] A. S. Klusener, *Completeness in real time process algebra*, Technical Report CS-R9106, CWI, Amsterdam, 1991. An extended abstract appeared in J. C. M. Baeten and J. F. Groote, editors, Proceedings CONCUR 91, Amsterdam, volume **527** of Lecture Notes in Computer Science, pp. 376–392, 1991.

[18] K. G. Larsen, *Modal Specifications*, Technical Report R89-09, Institute for Electronic Systems, The University of Aalborg, 1989.

[19] K. G. Larsen, A. Skou, *Compositional Verification of Probabilistic Processes*, in: W. R. Cleaveland, editor, Proceedings CONCUR 92, Stony Brook, volume **630** of Lecture Notes in Computer Science, pp. 456–471, Springer-Verlag, 1992.

[20] F. Moller and C. Tofts, *A Temporal Calculus of Communicating Systems*, in: J. C. M. Baeten and J. W. Klop, editors, Proceedings CONCUR 90, Amsterdam, volume **458** of Lecture Notes in Computer Science, pp. 401–415 Springer-Verlag, 1990.

[21] D. M. R. Park, *Concurrency and automata on infinite sequences*, In P. Duessen (ed.) 5*th* GI Conference volume **104** of Lecture Notes in Computer Science, pp. 167–183, Springer-Verlag, 1981.

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

[23] R. de Simone, *Higher-level synchronising devices in* MEIJE-SCCS, Theoretical Computer Science **37**, pp. 245–267, 1985.

[24] F. W. Vaandrager, *personal communication*, April 1993.

In this series appeared:

91/01	D. Alstein	Dynamic Reconfiguration in Distributed Hard Real-Time Systems, p. 14.
91/02	R.P. Nederpelt H.C.M. de Swart	Implication. A survey of the different logical analyses "if...,then...", p. 26.
91/03	J.P. Katoen L.A.M. Schoenmakers	Parallel Programs for the Recognition of *P*-invariant Segments, p. 16.
91/04	E. v.d. Sluis A.F. v.d. Stappen	Performance Analysis of VLSI Programs, p. 31.
91/05	D. de Reus	An Implementation Model for GOOD, p. 18.
91/06	K.M. van Hee	SPECIFICATIEMETHODEN, een overzicht, p. 20.
91/07	E.Poll	CPO-models for second order lambda calculus with recursive types and subtyping, p. 49.
91/08	H. Schepers	Terminology and Paradigms for Fault Tolerance, p. 25.
91/09	W.M.P.v.d.Aalst	Interval Timed Petri Nets and their analysis, p.53.
91/10	R.C.Backhouse P.J. de Bruin P. Hoogendijk G. Malcolm E. Voermans J. v.d. Woude	POLYNOMIAL RELATORS, p. 52.
91/11	R.C. Backhouse P.J. de Bruin G.Malcolm E.Voermans J. van der Woude	Relational Catamorphism, p. 31.
91/12	E. van der Sluis	A parallel local search algorithm for the travelling salesman problem, p. 12.
91/13	F. Rietman	A note on Extensionality, p. 21.
91/14	P. Lemmens	The PDB Hypermedia Package. Why and how it was built, p. 63.
91/15	A.T.M. Aerts K.M. van Hee	Eldorado: Architecture of a Functional Database Management System, p. 19.
91/16	A.J.J.M. Marcelis	An example of proving attribute grammars correct: the representation of arithmetical expressions by DAGs, p. 25.
91/17	A.T.M. Aerts P.M.E. de Bra K.M. van Hee	Transforming Functional Database Schemes to Relational Representations, p. 21.

91/18	Rik van Geldrop	Transformational Query Solving, p. 35.
91/19	Erik Poll	Some categorical properties for a model for second order lambda calculus with subtyping, p. 21.
91/20	A.E. Eiben R.V. Schuwer	Knowledge Base Systems, a Formal Model, p. 21.
91/21	J. Coenen W.-P. de Roever J.Zwiers	Assertional Data Reification Proofs: Survey and Perspective, p. 18.
91/22	G. Wolf	Schedule Management: an Object Oriented Approach, p. 26.
91/23	K.M. van Hee L.J. Somers M. Voorhoeve	Z and high level Petri nets, p. 16.
91/24	A.T.M. Aerts D. de Reus	Formal semantics for BRM with examples, p. 25.
91/25	P. Zhou J. Hooman R. Kuiper	A compositional proof system for real-time systems based on explicit clock temporal logic: soundness and complete ness, p. 52.
91/26	P. de Bra G.J. Houben J. Paredaens	The GOOD based hypertext reference model, p. 12.
91/27	F. de Boer C. Palamidessi	Embedding as a tool for language comparison: On the CSP hierarchy, p. 17.
91/28	F. de Boer	A compositional proof system for dynamic proces creation, p. 24.
91/29	H. Ten Eikelder R. van Geldrop	Correctness of Acceptor Schemes for Regular Languages, p. 31.
91/30	J.C.M. Baeten F.W. Vaandrager	An Algebra for Process Creation, p. 29.
91/31	H. ten Eikelder	Some algorithms to decide the equivalence of recursive types, p. 26.
91/32	P. Struik	Techniques for designing efficient parallel programs, p. 14.
91/33	W. v.d. Aalst	The modelling and analysis of queueing systems with QNM-ExSpect, p. 23.
91/34	J. Coenen	Specifying fault tolerant programs in deontic logic, p. 15.
91/35	F.S. de Boer J.W. Klop C. Palamidessi	Asynchronous communication in process algebra, p. 20.

92/01	J. Coenen; J. Zwiers; W.-P. de Roever	A note on compositional refinement, p. 27.
92/02	J. Coenen; J. Hooman	A compositional semantics for fault tolerant real-time systems, p. 18.
92/03	J.C.M. Baeten; J.A. Bergstra	Real space process algebra, p. 42.
92/04	J.P.H.W.v.d.Eijnde	Program derivation in acyclic graphs and related problems, p. 90.
92/05	J.P.H.W.v.d.Eijnde	Conservative fixpoint functions on a graph, p. 25.
92/06	J.C.M. Baeten; J.A. Bergstra	Discrete time process algebra, p.45.
92/07	R.P. Nederpelt	The fine-structure of lambda calculus, p. 110.
92/08	R.P. Nederpelt; F. Kamareddine	On stepwise explicit substitution, p. 30.
92/09	R.C. Backhouse	Calculating the Warshall/Floyd path algorithm, p. 14.
92/10	P.M.P. Rambags	Composition and decomposition in a CPN model, p. 55.
92/11	R.C. Backhouse; J.S.C.P.v.d.Woude	Demonic operators and monotype factors, p. 29.
92/12	F. Kamareddine	Set theory and nominalisation, Part I, p.26.
92/13	F. Kamareddine	Set theory and nominalisation, Part II, p.22.
92/14	J.C.M. Baeten	The total order assumption, p. 10.
92/15	F. Kamareddine	A system at the cross-roads of functional and logic programming, p.36.
92/16	R.R. Seljée	Integrity checking in deductive databases; an exposition, p.32.
92/17	W.M.P. van der Aalst	Interval timed coloured Petri nets and their analysis, p. 20.
92/18	R.Nederpelt; F. Kamareddine	A unified approach to Type Theory through a refined lambda-calculus, p. 30.
92/19	J.C.M.Baeten; J.A.Bergstra; S.A.Smolka	Axiomatizing Probabilistic Processes: ACP with Generative Probabilities, p. 36.
92/20	F.Kamareddine	Are Types for Natural Language? P. 32.
92/21	F.Kamareddine	Non well-foundedness and type freeness can unify the interpretation of functional application, p. 16.

92/22	R. Nederpelt F.Kamareddine	A useful lambda notation, p. 17.
92/23	F.Kamareddine E.Klein	Nominalization, Predication and Type Containment, p. 40.
92/24	M.Codish D.Dams Eyal Yardeni	Bottum-up Abstract Interpretation of Logic Programs, p. 33.
92/25	E.Poll	A Programming Logic for Fω, p. 15.
92/26	T.H.W.Beelen W.J.J.Stut P.A.C.Verkoulen	A modelling method using MOVIE and SimCon/ExSpect, p. 15.
92/27	B. Watson G. Zwaan	A taxonomy of keyword pattern matching algorithms, p. 50.
93/01	R. van Geldrop	Deriving the Aho-Corasick algorithms: a case study into the synergy of programming methods, p. 36.
93/02	T. Verhoeff	A continuous version of the Prisoner's Dilemma, p. 17
93/03	T. Verhoeff	Quicksort for linked lists, p. 8.
93/04	E.H.L. Aarts J.H.M. Korst P.J. Zwietering	Deterministic and randomized local search, p. 78.
93/05	J.C.M. Baeten C. Verhoef	A congruence theorem for structured operational semantics with predicates, p. 18.
93/06	J.P. Veltkamp	On the unavoidability of metastable behaviour, p. 29
93/07	P.D. Moerland	Exercises in Multiprogramming, p. 97
93/08	J. Verhoosel	A Formal Deterministic Scheduling Model for Hard Real-Time Executions in DEDOS, p. 32.
93/09	K.M. van Hee	Systems Engineering: a Formal Approach Part I: System Concepts, p. 72.
93/10	K.M. van Hee	Systems Engineering: a Formal Approach Part II: Frameworks, p. 44.
93/11	K.M. van Hee	Systems Engineering: a Formal Approach Part III: Modeling Methods, p. 101.
93/12	K.M. van Hee	Systems Engineering: a Formal Approach Part IV: Analysis Methods, p. 63.
93/13	K.M. van Hee	Systems Engineering: a Formal Approach Part V: Specification Language, p. 89.

92/22	R. Nederpelt F.Kamareddine	A useful lambda notation, p. 17.
92/23	F.Kamareddine E.Klein	Nominalization, Predication and Type Containment, p. 40.
92/24	M.Codish D.Dams Eyal Yardeni	Bottum-up Abstract Interpretation of Logic Programs, p. 33.
92/25	E.Poll	A Programming Logic for Fω, p. 15.
92/26	T.H.W.Beelen W.J.J.Stut P.A.C.Verkoulen	A modelling method using MOVIE and SimCon/ExSpect, p. 15.
92/27	B. Watson G. Zwaan	A taxonomy of keyword pattern matching algorithms, p. 50.
93/01	R. van Geldrop	Deriving the Aho-Corasick algorithms: a case study into the synergy of programming methods, p. 36.
93/02	T. Verhoeff	A continuous version of the Prisoner's Dilemma, p. 17
93/03	T. Verhoeff	Quicksort for linked lists, p. 8.
93/04	E.H.L. Aarts J.H.M. Korst P.J. Zwietering	Deterministic and randomized local search, p. 78.
93/05	J.C.M. Baeten C. Verhoef	A congruence theorem for structured operational semantics with predicates, p. 18.
93/06	J.P. Veltkamp	On the unavoidability of metastable behaviour, p. 29
93/07	P.D. Moerland	Exercises in Multiprogramming, p. 97
93/08	J. Verhoosel	A Formal Deterministic Scheduling Model for Hard Real-Time Executions in DEDOS, p. 32.
93/09	K.M. van Hee	Systems Engineering: a Formal Approach Part I: System Concepts, p. 72.
93/10	K.M. van Hee	Systems Engineering: a Formal Approach Part II: Frameworks, p. 44.
93/11	K.M. van Hee	Systems Engineering: a Formal Approach Part III: Modeling Methods, p. 101.
93/12	K.M. van Hee	Systems Engineering: a Formal Approach Part IV: Analysis Methods, p. 63.
93/13	K.M. van Hee	Systems Engineering: a Formal Approach Part V: Specification Language, p. 89.
93/14	J.C.M. Baeten J.A. Bergstra	On Sequential Composition, Action Prefixes and Process Prefix, p. 21.

93/15	J.C.M. Baeten J.A. Bergstra R.N. Bol	A Real-Time Process Logic, p. 31.
93/16	H. Schepers J. Hooman	A Trace-Based Compositional Proof Theory for Fault Tolerant Distributed Systems, p. 27
93/17	D. Alstein P. van der Stok	Hard Real-Time Reliable Multicast in the DEDOS system, p. 19.