A congruence theorem for structured operational semantics with predicates

J.C.M. Baeten and 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, josb@win.tue.nl

Abstract. We proposed a syntactical format, the *path* format, for structured operational semantics in which predicates may occur. We proved that strong bisimulation is a congruence for all the operators that can be defined within the *path* format. To show that this format is useful we provided many examples that we took from the literature about *CCS*, *CSP*, and *ACP*; they do satisfy the *path* format but no formats proposed by others. The examples include concepts like termination, convergence, divergence, weak bisimulation, a zero object, side conditions, functions, real time, discrete time, sequencing, negative premises, negative conclusions, and priorities (or a combination of these notions).

Key Words & Phrases: *structured operational semantics, term deduction system, transition system specification, structured state system, labelled transition system, strong bisimulation, congruence theorem, predicate.*
1980 Mathematics Subject Classification (1985 Revision): *68Q05, 68Q55.*
CR Categories: *D.3.1, F.1.1, F.3.2, F.4.3.*
Note: *Partial support received from the European Communities under CONCUR 2, BRA 7166.*

1. Introduction

Since the paper of Plotkin [**23**] about structural operational semantics it has become quite popular to provide a semantics for process calculi, programming, and specification languages by using a labelled transition system. Since then a number of papers about the subject of structured operational semantics appeared and it gradually became a subject of research of its own. Various so-called *formats* were proposed for labelled transition systems in Plotkin's style. Such a format is a syntactical constraint on the form of the rules such that some nice property holds, for instance, that bisimulation equivalence is a congruence for all the operators that can be defined within the format.

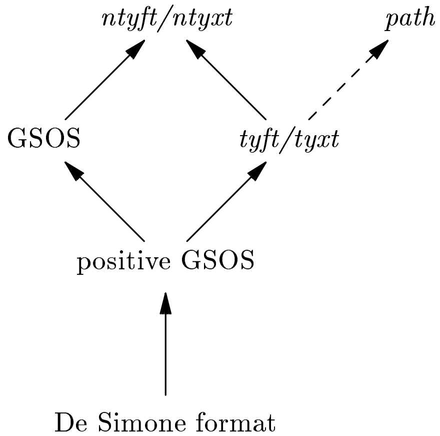

Figure 1. The lattice of formats

In figure 1 we depict the various formats together with a new one; the dashed arrow points to it. An arrow from one format to another indicates that all operators that can be defined in the first format can also be defined in the second one. The most basic format is a modest reformulation of a format proposed by De Simone [**25**]. 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 [**11**] allows negative premises but no lookahead and the *tyft/tyxt* format of Groote and Vaandrager [**16**] allows lookahead but no negative premises. They both generalize the format of De Simone. The positive GSOS format in the figure is the greatest common divisor (so to speak) of the GSOS and the *tyft/tyxt* format. The *ntyft/ntyxt* format of Groote [**15**] is the least common multiple (using the same informal phrasing) of the *tyft/tyxt* format and the GSOS format. In this paper we will extend the *tyft/tyxt* format with predicates. We propose the *path* format, which stands for *predicates and tyft/tyxt hybrid format.* We will not give the exact definitions of all these formats except the definitions of the *path* format and the *tyft/tyxt* format.

The main result of this paper is a congruence theorem stating that if a so-called term deduction system satisfies the *path* format (and is well-founded) then strong bisimulation is a congruence. We prove this using the congruence theorem of Groote and Vaandrager [**16**] (which is the case without predicates) by coding each predicate as a binary relation. So all the operators that can be defined in the *path* format can also be defined in the *tyft/tyxt* format. Now the question arises why we need the *path* format anyway. Next, we will motivate the need for this new format.

A format is, in general, a purely syntactical constraint on the form of the deduction rules, so the *path* format syntactically generalizes the *tyft/tyxt* format since we may mix predicates and ordinary transitions (in fact, our format uses relations instead of transitions), whereas we cannot use predicates in the *tyft/tyxt* format.

Another argument that advocates the *path* format is that usually an operational semantics in Plotkin's style that contains predicates can be easily understood, but if we have to code each predicate as a binary relation to obtain an operational semantics in *tyft/tyxt* format we immediately lose the simplicity of the operational semantics.

Even if we do not care about syntactical freedom or simplicity at all we still have to be careful with coding of predicates by relations since this can lead to non-intuitive situations. Next, we will give some examples of operational semantics that are better understood with than without predicates.

Two examples can be found in a paper of Baeten and Vaandrager [**7**] in which several operational semantics are given for the simple language Basic Process Algebra (*BPA*); this consists of the first five laws of the theory *PA* of Bergstra and Klop [**10**]. The language *BPA* has only atomic actions and alternative and sequential composition.

First, Baeten and Vaandrager extend the signature with a constant δ. Then they give an operational semantics with negative premises. For completeness they need that $\delta x = x$, which is different from the usual law in the *ACP* framework, namely $\delta x = \delta$; the constant δ stands for inaction or deadlock. In section 2 we will return to this operational semantics. An operational semantics using predicates that to us is more intuitively appealing is given by Van Glabbeek [**14**]. Further on in this section we will use his semantics to explain our format.

When discussing another operational semantics of *BPA*, Baeten and Vaandrager moreover add the constant ε (see Koymans and Vrancken [**19**]) to the algebraic description language featuring the rule

$$\varepsilon \xrightarrow{\surd} \delta;$$

here $\surd$ is just an extra label. Our intuition of the empty process is that it is only capable of immediately successfully terminating, but this rule suggests that it can, in fact, perform an action $\surd$ and then terminate unsuccessfully. Baeten and Van Glabbeek [**6**] give an operational semantics with a termination predicate that is more in accordance with our intuition about the empty process. We will return to this issue in another context where we treat an operational semantics of Aceto and Hennessy [**1**] in which a termination predicate occurs; see section 2.

As a last example of semantics that are better understood with than without predicates we will mention the clear operational semantics using predicates that Klusener [**18**] gives for various real time process algebras. He also gives non-intuitive semantics with negative premises for the same languages. Klusener introduces these negative premises to eliminate the predicates (in order to obtain a congruence result). We claim that this is not necessary; see section 2 for more details.

Next we will informally introduce the *path* format by giving the operational semantics of Van Glabbeek [**14**] of the language *BPA* to demonstrate the congruence theorem. In fact, we take a fragment of the structured operational semantics of Van Glabbeek [**14**]. In *BPA* we have alternative and sequential composition (denoted by $+$ and $\cdot$ resp.) and a set A of atomic actions. Often we will write xy instead of $x \cdot y$. In table 1 we have that x, x', y, and y' are distinct variables and a ranges over the set A. We have ordinary transitions in this table and predicates. For each $a \in A$ there is a postfix predicate $\cdot \xrightarrow{a} \surd$. The intuition of $x \xrightarrow{a} \surd$ is that x can perform an a action and then terminate successfully. The intended meaning of $x \xrightarrow{a} x'$ is as expected: x evolves into x' by performing an a action. We have to check two properties of the transition system in order to be able to use our congruence theorem. We will informally introduce them and show that they hold for the system in table 1.

The first demand is that every rule must be well-founded. Well-foundedness roughly says that there are no transitions in the premises with cyclic occurrences of variables; for instance, the rules

$$\frac{x \xrightarrow{a} x}{C}, \quad \frac{y \xrightarrow{a} z, z \xrightarrow{b} y}{C} \tag{1}$$

are not well-founded. It is easy to check that the rules in table 1 are well-founded.

$a \xrightarrow{a} \surd$	$\dfrac{x \xrightarrow{a} x'}{xy \xrightarrow{a} x'y}$	$\dfrac{x \xrightarrow{a} \surd}{xy \xrightarrow{a} y}$	
$\dfrac{x \xrightarrow{a} x'}{x + y \xrightarrow{a} x'}$	$\dfrac{y \xrightarrow{a} y'}{x + y \xrightarrow{a} y'}$	$\dfrac{x \xrightarrow{a} \surd}{x + y \xrightarrow{a} \surd}$	$\dfrac{y \xrightarrow{a} \surd}{x + y \xrightarrow{a} \surd}$

Table 1. A Transition system for *BPA*.

The last demand is a little bit more involved: the rules have to be in *path* format. We list the conditions.

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 then it must begin with either a variable $x \notin Y$ or a term $f(x_1, \ldots, x_n)$ with $x_1, \ldots, x_n$ distinct variables that are not in Y. If the conclusion contains a predicate we treat the term that occurs in it as if it were the beginning of an ordinary transition.

Now it is easy to verify that the rules of table 1 are in *path* 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 whatsoever on terms occurring in predicates in the premises. There is no restriction on a term that occurs in the left-hand side of a transition in a premise or in the right-hand side of a transition in a conclusion.

As an example we treat the last rule of table 1. For the premise of this rule there is nothing to check. The beginning of the transition in the conclusion is of the form $f(x, y)$. So this rule is in *path* format and bisimulation is a congruence.

In the remainder of this section we will discuss the organization 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 several reasons that advocate this ordering, in fact, we already discussed an application in the introduction. The first reason for this choice is that the area of application is operational semantics and they are often easy to read. The second reason is that the informal definitions that we just gave for the well-foundedness and the *path* format will be enough to understand what is going on. The third and maybe most important reason for this choice is that the reader immediately can see if her or his operational semantics has a good chance to fit in our format. If this is the case then the time has come to read on and enter section 3. In this section we introduce the basics and the notion of a term deduction system, which is a generalization of a transition system specification. In the next section we define a structured state system, which generalizes the concept of a labelled transition system and we define a notion of bisimulation equivalence in the presence of predicates. The connection between these two sections is that a term deduction system induces in a natural way a structured state system. Now we have all the prerequisites to continue with section 5. In this section we will give the exact definitions of well-foundedness and the *path* and *tyft/tyxt* formats. Thereafter, we will state and prove the congruence theorem. The last section contains concluding remarks and discusses future work.

2. Applications

In this section we will discuss a number of known operational semantics in which predicates occur so none of the examples satisfy formats proposed by others. It will turn out that they are well-founded and satisfy the *path* format or that they can be easily modified to satisfying the *path* format. With the aid of our congruence theorem we then find that bisimulation is a congruence for all the operators that are defined within the operational semantics. We recall that the informal definitions of well-founded and the *path* format can be found in the introduction.

The examples are taken from *CCS*, *CSP*, and *ACP*. They touch upon concepts like termination, convergence, divergence, weak bisimulation, a zero object, side conditions, functions, real time, discrete time, sequencing, negative premises, negative conclusions, and priorities (or a combination of these notions).

The first example concerns an operational semantics that originates 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 the termination predicate $\surd$, which is a postfix denoted predicate. In table 2 we enumerate the rules as they essentially appear in [**1**]. In fact, Baeten and Van Glabbeek [**6**] have the same approach to successful termination. We treat recursion in the same way as Groote and Vaandrager [**16**] by adding process names $rec x.\, t$ to the signature for each $t \in O(\Sigma)$ to obtain that the recursion rules fit our format. However, it would be a better idea to incorporate recursion whitin our format as is done for the GSOS format [**11**] and De Simone's format [**25**].

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

Table 2. The rules for $\surd$.

Next, they define the unsuccessful termination predicate by simply taking the complement of the successful termination predicate; we will denote this predicate by $\neg\surd$. In our approach we will have to explicitly define this predicate with deduction rules since we define the transition system in one step. In table 3 we give our rules for $\neg\surd$. It might be argued that this is not so intuitive but the rules for $\neg\surd$ are quite straightforward. Moreover, if only the negative version of a predicate is necessary it can be an advantage; see the $\_ \neq 0$ predicate later on.

With the aid of both predicates Aceto and Hennessy inductively define their convergence predicate $\downarrow$; we list their rules in table 4.

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

Table 3. The rules for $\neg\surd$.

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

Table 4. The rules for $\downarrow$.

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 operator $\partial_H(\cdot)$. Aceto and Hennessy also have a divergence predicate, which is the complement of the convergence predicate. We omitted this predicate since they do not need it to define their operational semantics. However it is easy to define this predicate; we refer to [**17**] for an explicit operational semantics of the divergence predicate in *path* format.

For more details on the process algebra of Aceto and Hennessy we refer to [**1**].

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

Table 5. The action relations for each $\mu \in Act_\tau$.

The matter that we are interested in at the moment is that the rules in tables 2–5 are well-founded and in *path* format so with our main result we immediately find that strong bisimulation is a congruence.

At this point one may find that we are not entirely fair since Aceto and Hennessy are, in fact, interested in rooted weak bisimulation and they have a lot of work to do to obtain that this is a congruence for their system. Therefore, we will prove in the next example that rooted *weak* bisimulation is a congruence with our 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.

We have not succeeded in coding the weak termination and convergence predicates of Aceto and Hennessy [**1**] in the *path* format. This seems to involve essential use of negative premises.

This next example is an operational semantics of Van Glabbeek [**14**] for BPA_τ, which is a fragment of the theory ACP_τ that originates from Bergstra and Klop [**9**]. The subscript τ stands for the silent move of Milner [**20**]. The operational semantics is given by the rules of tables 1 and 6. Van Glabbeek [**14**] observes that his model is isomorphic to the graph model of Baeten, Bergstra and Klop [**4**]. This means that two terms are strongly bisimilar [**22**] in his model if and only if they are rooted weakly bisimilar [**20**] in the graph model, so since all the rules of tables 1 and 6 are well-founded and in *path* format we know that strong bisimulation is a congruence and thus, rooted weak bisimulation on the graph model is also a congruence. We can also apply this result to the entire operational semantics that Van Glabbeek gives for ACP_τ.

$$a \xrightarrow{a} \tau$$

$$\frac{x \xrightarrow{\tau} y,\ y \xrightarrow{a} x'}{x \xrightarrow{a} x'} \qquad \frac{x \xrightarrow{\tau} y,\ y \xrightarrow{a} \surd}{x \xrightarrow{a} \surd}$$

$$\frac{x \xrightarrow{a} y,\ y \xrightarrow{\tau} x'}{x \xrightarrow{a} x'} \qquad \frac{x \xrightarrow{a} y,\ y \xrightarrow{\tau} \surd}{x \xrightarrow{a} \surd}$$

Table 6. Operational rules for the silent step.

The next example is an operational semantics of the language BPA_0 that originates from Baeten and Bergstra [**3**]. The theory BPA_0 is the language BPA (that we already saw above) with a zero object denoted by 0. They have for this constant the following laws

$$x + 0 = x, \quad x0 = 0, \quad 0x = 0.$$

Baeten and Bergstra explicitly state in their paper: "We cannot define action rules in the format of Groote and Vaandrager [**16**] or Groote [**15**]" (see [**3**], loc. cit. p. 87). The problems arise with the rules for the sequential composition: if $x \xrightarrow{a} x'$ then we will only have $xy \xrightarrow{a} x'y$ if $y \neq 0$. In table 7 we give the rules that define the (postfix) predicate $\cdot \neq 0$ and we give the rules for the sequential composition. Together with the first five rules of table 1 they form an operational semantics for BPA_0. It is not hard to check that all the rules are well-founded and in *path* format, so we obtain that bisimulation is a congruence. Our approach also works for the extensions that Baeten and Bergstra discuss.

$$a \neq 0 \qquad \frac{x \neq 0}{x + y \neq 0} \qquad \frac{x \neq 0}{y + x \neq 0}$$

$$\frac{x \neq 0,\ y \neq 0}{xy \neq 0} \qquad \frac{x \xrightarrow{a} x',\ y \neq 0}{xy \xrightarrow{a} x'y} \qquad \frac{x \xrightarrow{a} \surd,\ y \neq 0}{xy \xrightarrow{a} y}$$

Table 7. The $\neq 0$ predicate and rules for sequential composition.

Often we see that the rules of an operational semantics have so-called side conditions. For instance, the following rule from Groote and Vaandrager [**16**] has a side condition:

$$a \xrightarrow{a} \varepsilon,\ a \neq \surd.$$

This side condition is an abbreviation for the list containing for all $a \in A$ a rule $a \xrightarrow{a} \varepsilon$. Thus, it is in fact a harmless side condition, since the rules can be replaced by an enumaration. Often, though, side conditions are not so harmless. This is the case if a side condition contains a process term. An example of such a rule can be found in a paper of Moller and Tofts [**21**] on temporal *CCS*:

$$\frac{P \xrightarrow{t} P'}{P \oplus Q \xrightarrow{t} P'},\ |Q|_\mathcal{T} < t. \tag{1}$$

Here $|\cdot|_\mathcal{T}$ is called the maximal delay, which is a *function* and not a predicate. In the operational semantics of Moller and Tofts there are no "free" occurrences of the maximal delay function, but only side conditions as in display (1). Since this is the case, we can see the side condition $|Q|_{\mathcal{T}} < t$ as a mixfix (or distfix) predicate: $|\cdot|_{\mathcal{T}} < t$ (see [**13**] for the fix terminology). The only problem left is to define the maximal delay function in these terms. Moller and Tofts define their maximal delay function in terms of $=$ instead of $<$, so we will also define the corresponding predicates in terms of $=$ (we will do this in a minute). But now we have to rephrase display (1), since we no longer have a predicate in terms of $<$. This is easy:

$$\frac{P \xrightarrow{t} P',\ |Q|_{\mathcal{T}} = s}{P \oplus Q \xrightarrow{t} P'},\ s < t.$$

Note that the new side condition does not contain Q anymore. It has become an ordinary enumeration: for all s, t with $s < t$ we have such a rule.

$\lvert 0 \rvert_{\mathcal{T}} = 0$	$\lvert X \rvert_{\mathcal{T}} = 0$	$\lvert a.P \rvert_{\mathcal{T}} = 0$	$\lvert \delta.P \rvert_{\mathcal{T}} = \omega$
$\dfrac{\lvert P \rvert_{\mathcal{T}} = t}{\lvert (s)P \rvert_{\mathcal{T}} = s + t}$	$\dfrac{\lvert P \rvert_{\mathcal{T}} = t}{\lvert P \backslash a \rvert_{\mathcal{T}} = t}$	$\dfrac{\lvert P \rvert_{\mathcal{T}} = t}{\lvert P[S] \rvert_{\mathcal{T}} = t}$	$\dfrac{\lvert P_i\{\mu\tilde{x}.\tilde{P}/\tilde{x}\} \rvert_{\mathcal{T}} = t}{\lvert \mu_i \tilde{x}.\tilde{P} \rvert_{\mathcal{T}} = t}$
$\dfrac{\lvert P \rvert_{\mathcal{T}} = t,\ \lvert Q \rvert_{\mathcal{T}} = s}{\lvert P \oplus Q \rvert_{\mathcal{T}} = \max(s,t)}$	$\dfrac{\lvert P \rvert_{\mathcal{T}} = t,\ \lvert Q \rvert_{\mathcal{T}} = s}{\lvert P + Q \rvert_{\mathcal{T}} = \min(s,t)}$	$\dfrac{\lvert P \rvert_{\mathcal{T}} = t,\ \lvert Q \rvert_{\mathcal{T}} = s}{\lvert P \mid Q \rvert_{\mathcal{T}} = \min(s,t)}$	

Table 8. The maximal delay function as a mixfix predicate.

In table 8 we listed the maximal delay function as a mixfix predicate. Note that we are a bit sloppy with the notation: in fact, we cannot speak about $|(s)P|_{\mathcal{T}} = s + t$. The ultra correct way to state this rule is to put the sum in a side condition:

$$\frac{|P|_{\mathcal{T}} = t}{|(s)P|_{\mathcal{T}} = r},\ \forall r, s, t : r = s + t.$$

The same holds for the rules that use max and min.

The rephrased operational semantics of Moller and Tofts is well-founded and in *path* format, so strong bisimulation is a congruence.

We will briefly mention some related examples in the area of real time process algebra.

Klusener [**18**] gives on the one hand some clear operational semantics for several real time process algebras but on the other hand he introduces non-intuitive operational semantics for the same real time process algebras using negative premises. This is a lot of work and is merely done to obtain that the transition system specifications are in *ntyft* format such that he gets the congruenceness for "free" with a theorem of Groote [**15**]. We claim that it is not necessary: the operational semantics that Klusener actually wants to discuss are well-founded and in *path* format. This result can be obtained in the same way as we sketched this for the above example of Moller and Tofts.

The last temporal example is an operational semantics for timed *CSP* of Schneider [**24**]. His operational semantics also consists of a function and ordinary transitions. We claim that this function is, in fact, a mixfix predicate and that the resulting operational semantics is well-founded and in *path* format.

We promised in the introduction to return to the operational semantics with negative premises that Baeten and Vaandrager [**7**] give for *BPA*. We recall that they extend the signature of *BPA* with an auxiliary constant δ and that they have $\delta x = x$. They have this property because they model the sequential composition as a sequencing operator. If x is sequenced with y, notation xy, the process xy starts with the execution of x, and if it cannot do more actions, then the execution of y starts. In table 9 we list their semantics with extra rules (from

$a \xrightarrow{a} \delta$	$\delta \not\longrightarrow$
$\dfrac{x \xrightarrow{a} x'}{x + y \xrightarrow{a} x' \xleftarrow{a} y + x}$	$\dfrac{x \xrightarrow{a} x'}{xy \xrightarrow{a} x'y}$
$\dfrac{x \not\longrightarrow, y \xrightarrow{a} y'}{xy \xrightarrow{a} y'}$	$\dfrac{x \not\longrightarrow, y \not\longrightarrow}{xy \not\longrightarrow, x + y \not\longrightarrow}$

Table 9. *BPA* with sequencing instead of sequential composition.

us) containing negative conclusions (note that we compressed eight rules to six). The intended interpretation of $x \not\longrightarrow$ is that x has no outgoing transitions. Baeten and Vaandrager define this in terms of the transition relation, whereas we define this in terms of itself, namely inductively as a predicate.

It is easy to see that this transition system is well-founded and in *path* format, so bisimulation is a congruence.

This example deviates from the previous ones in the sense that we introduce a predicate $\not\longrightarrow$, whereas in the other examples a predicate already existed. So we additionally have to show that $x \not\longrightarrow$ if and only if there is no a and x' such that $x \xrightarrow{a} x'$ in order to show that strong bisimulation is the same as our bisimulation with the extra predicate $\not\longrightarrow$.

The next example is also an operational semantics with negative premises. It is a semantics of *BPA* with discrete time from Baeten and Bergstra [**2**]. In table 10 '$\xrightarrow{\sigma}$' is just an extra relation: σ is not in their language. The extra rules are the ones with negative conclusions. Again it is easy to see that the rules are well-founded and in *path* format, so bisimulation is a congruence. As in the previous example we additionally have to show that $x \stackrel{\sigma}{\not\longrightarrow}$ if and only if there is no x' with $x \xrightarrow{\sigma} x'$.

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

Table 10. *BPA* with discrete time.

Finally, we give an operational semantics of *BPA* with the priority operator of Baeten, Bergstra, and Klop [**5**]. We want to stress that this example is not meant for practical purpose. The reason for giving it is to show that it can be done within our format. For practical use it is better to extend our format with negative premises. Groote and Vaandrager [**16**] gave with a single rule a more intuitive operational definition of the priority operator using negative premises in the presence of deadlock and the empty process. Our operational semantics of *BPA* with the priority operator is built up from the rules in tables 1 and 11. As in the above examples we inductively define the negative premises with a postfix predicate $\_ \stackrel{a}{\not\longrightarrow}$. The meaning of $x \stackrel{a}{\not\longrightarrow}$ is that there is no x' with $x \xrightarrow{a} x'$ and x is not capable of terminating successfully by performing the a action. It is easy to see that the rules are well-founded and in *path* format, so bisimulation is a congruence. Also here we have to show that $x \stackrel{a}{\not\longrightarrow}$ if and only if there is no x' with $x \xrightarrow{a} x'$ and not $x \xrightarrow{a} \surd$.

As a final remark we want to mention that all the positive transition systems appearing in the book of Baeten and Weijland [**8**] are in *path* format. There is only one negative transition

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

Table 11. The additional rules for *BPA* with priority operator.

system in their book, which is not in *path* format but it can easily be made into *path* format since this system treats the priority operator (cf. the above example). We also want to stress that none of the transition systems in their book satisfy the *tyft/tyxt* or the *ntyft/ntyxt* format.

3. Term deduction systems

All the examples that we treated in section 2 are, in fact, term deduction systems. In this section we formalize this notion, which generalizes the notion of a transition system specification. In a transition system specification we have a transition relation $\cdot \xrightarrow{a} \cdot$ for each label a, whereas in a term deduction system we have just a set of relation symbols and a set of predicate symbols. Having general relation symbols is an advantage. For instance, the operational semantics for temporal *CCS* of Moller and Tofts (from which we treated the tricky part in section 2) provides two transition relations; this can be easily modelled with a term deduction system but not with a transition system specification. But we can also think of the combination of a transition relation with a totally different relation, such as Wang's syntactical inequality relation $P \neq Q$ that he uses in an operational semantics for probabilistic *CCS* (q.v. [**26**]).

Before we give the definition of a term deduction system 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. 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 $P \in T_p$ and $R \in T_r$, and $s, t, u \in O(\Sigma)$. We call expressions Ps and tRu formulas. A deduction rule $d \in D$ has the form

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

with H a set of formulas and C a formula. 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 $T = (\Sigma, D)$ be a term deduction system. A proof of a formula ψ from T is a well-founded upwardly branching tree of which the nodes are labelled by formulas such that the root is labelled with ψ and if χ is the label of a node q and $\{\chi_i : i \in I\}$ is the set of labels belonging to the nodes $\{q_i : i \in I\}$ directly above q then there is a deduction rule

$$\frac{\{\phi_i : i \in I\}}{\phi}$$

and a substitution $\sigma : V \longrightarrow O(\Sigma)$ such that $\sigma(\phi) = \chi$ and $\sigma(\phi_i) = \chi_i$ for $i \in I$. The length of q is $|q| = \sup\{|q_i| + 1 : i \in I\}$. The length of a proof is the length of its root.

If a proof of ψ exists, we say that ψ is provable from T, notation $T \vdash \psi$. A proof is closed if it contains only closed formulas.

Lemma (3.3) If a closed formula is provable from a term deduction system then it is provable by a closed proof.

Proof. Easy. Confer lemma 3.3 of Groote and Vaandrager [**16**].

4. Structured state systems and bisimulation

In this section we define the notions of a structured state system and bisimulation. A structured state system is a generalization of the notion of a labelled transition system. In a labelled transition system we have for each label a transition relation, whereas in a structured state system we have just relations and predicates. At first sight, this gives rise to a stronger notion of bisimulation on a structured state system since we have a transfer property for each relation and predicate. But our definition of bisimulation mostly coincides with the ones in the literature. For instance, in the example of Moller and Tofts there are two transition relations so they need to extend the notion of bisimulation to the two relations. In our setting we moreover demand that bisimilar processes must have the same maximal delay. So we have "maximal delay" bisimulation. Moller and Tofts state in their proposition 3.3 that our maximal delay bisimulation coincides with their notion of bisimulation. The same phenomenon occurs in Klusener [**18**] with our UL-bisimulation, Klusener's U-bisimulation, and ordinary bisimulation (U means ultimate delay and L means latest possible action). Also for the examples with negative premises that we gave in section 2 we have to prove that bisimulation is the same as our "negative" bisimulation. For small term deduction systems this will not be a problem but for bigger ones it can become a problem. Therefore, it is better to allow negative premises in our rules. However, this is out of the scope of this paper (but we are currently studying this).

Definition (4.1) A structured state system is a triple (S, S_p, S_r) where S is a set of states, S_p is a subset of the power set of S and S_r is a subset of the power set of $S \times S$. The sets S_p and S_r are called respectively the set of predicates and the set of relations.

Remark (4.2) A structured state system is a generalization of a labelled transition system. To see this take an empty set of predicates and take for the set of relations the labelled transition relation.

Next, we will define the notion of strong bisimulation on a structured state system, which is based on Park [**22**].

Definition (4.3) Let $G = (S, S_p, S_r)$ be a structured state system. A relation $B \subseteq S \times S$ is called a (strong) bisimulation if for all $s, t \in S$ with sBt the following conditions hold. For all $R \in S_r$

$$\forall s' \in S(sRs' \Rightarrow \exists t' \in S : tRt' \wedge s'Bt'),$$
$$\forall t' \in S(tRt' \Rightarrow \exists s' \in S : sRs' \wedge s'Bt'),$$

and for all $P \in S_p$

$$Ps \Leftrightarrow Pt.$$

The first two conditions are known as the transfer property. Two states $s, t \in S$ are bisimilar in the structured state system G if there exists a bisimulation relation containing the pair (s, t). Notation $s \sim_G t$ or $s \sim t$ provided that no confusion can arise. Note that bisimilarity is an equivalence relation.

A term deduction system induces in a natural way a structured state system.

Definition (4.4) Let $T = (\Sigma, D)$ be a term deduction system and let $D = D(T_p, T_r)$. The structured state system G induced by T has as its set of states $S = C(\Sigma)$; the predicates and relations are the following.

$$S_p = \Big\{ \{ t \in C(\Sigma) \mid T \vdash Pt \} \Bigm| P \in T_p \Big\},$$
$$S_r = \Big\{ \{ (s,t) \in C(\Sigma) \times C(\Sigma) \mid T \vdash sRt \} \Bigm| R \in T_r \Big\}.$$

We will call $s, t \in C(\Sigma)$ bisimilar, notation $s \sim_T t$ if $s \sim_G t$. Note that $\sim_T$ is also an equivalence relation. Again we omit the subscript when it is clear what T is.

Definition (4.5) We say that two term deduction systems are equivalent if they induce the same structured state system.

5. The congruence theorem

This section is devoted to the congruence theorem. We expect that our congruence theorem can be proved by simply adapting the proof of the congruence theorem of Groote and Vaandrager [**16**] to the present situation. However, we prove a stronger result than just a congruence theorem since we moreover show that every term deduction system in *path* format can be reduced to a transition system specification in *tyft/tyxt* format and then apply the theorem of Groote and Vaandrager.

We strongly conjecture that we can prove a similar reduction/congruence theorem for the situation with negative premises by applying the congruence theorem of Groote [**15**] (that treats the situation with negative premises).

First, we will define the notions necessary to state the main result and then we will prove it.

Definition (5.1) Let $T = (\Sigma, D)$ be a term deduction system with $D = D(T_p, T_r)$. Let in the following I and J be index sets of arbitrary cardinality, let $t_i, s_j, t \in O(\Sigma)$ for all $i \in I$ and $j \in J$, let $P_j, P \in T_p$ be predicate symbols for all $j \in J$, and let $R_i, R \in T_r$ be relation symbols for all $i \in I$.

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

$$\frac{\{P_j s_j : j \in J\} \cup \{t_i R_i y_i : i \in I\}}{f(x_1, \ldots, x_n) R t}$$

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 *ptyxt* format if it has the form

$$\frac{\{P_j s_j : j \in J\} \cup \{t_i R_i y_i : i \in I\}}{x R t}$$

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 *ptyf* format if it has the form

$$\frac{\{P_j s_j : j \in J\} \cup \{t_i R_i y_i : i \in I\}}{Pf(x_1, \ldots, x_n)}$$

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 *ptyx* format if it has the form

$$\frac{\{P_j s_j : j \in J\} \cup \{t_i R_i y_i : i \in I\}}{Px}$$

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 explane the names of the deduction rules. The p in the phrases *ptyft*, *ptyxt*, *ptyf*, and *ptyx* refers to the predicates occurring in the rules, the *ty* refers to the relation part in the set of hypotheses, and the *ft, xt, f*, and x refer to the various conclusions. The names *tyft* and *tyxt* are taken from Groote and Vaandrager [**16**].

If a deduction rule $d \in D$ has one of the above forms we say that this rule is in *path* format, which stands for "predicates and *tyft/tyxt* hybrid format". A term deduction system is in *path* 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 *tyft/tyxt* format if it is in *path* format and its set of predicate symbols is empty.

We need the technical notion of well-foundedness of a term deduction system, which will be used in the proof of the congruence theorem. We will use a property in the proof that in general does not hold for a non well-founded term deduction system; we will return to this in the proof of the theorem. It is an open question if the requirement of well-foundedness is really necessary. However, we heard from Fokkink [**12**] that the requirement is probably not necessary for the *tyft/tyxt* format and that his results easily generalize to our setting.

Definition (5.2) Let $T = (\Sigma, D)$ be a term deduction system and let F be a set of formulas. The 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 dependency graph if and only if there is a $tRs \in F$ 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 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.

Lemma (5.3) For every well-founded term deduction system in *path* format there is an equivalent pure well-founded term deduction system in *path* format.

Proof. Essentially the proof of Lemma 5.9 in Groote and Vaandrager [**16**].

Theorem (5.4) Let $T = (\Sigma, D)$ be a well-founded term deduction system in *path* format then strong bisimulation is a congruence for all function symbols occurring in Σ.

Proof. Groote and Vaandrager [**16**] proved this theorem in the case that the set of predicate symbols is empty, that is, if the term deduction system is in *tyft/tyxt* 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 in the old system as a special relation 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 term deduction system and suppose that $D = D(T_p, T_r)$ with $T_p \neq \emptyset$. In accordance with lemma (5.3) 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 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 $d = \frac{H}{C}$. The set of hypotheses of d' is the set H but with the "predicate part" $\{P_j s_j \mid j \in J\}$ replaced by $\{s_j R_{P_j} z_j \mid j \in J\}$ with $\{z_j \mid j \in J\}$ a set of distinct variables disjoint with $var(d)$. If the conclusion of the old rule is of the form Pt then the conclusion of the new rule is $tR_P\xi$. In the other case C remains the same. Note that T' is pure, well-founded and in *tyft/tyxt* format.

Now we will prove that two closed Σ-terms u and v are bisimilar in the original system T if and only if they are bisimilar in the 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. (We denote the bisimulation relation in T by $\sim$ and in T' by $\sim'$.)

(i)	$T' \vdash uR_Pv \Longrightarrow v = \xi$
(ii)	$u \in C(\Sigma), T' \vdash uRv \Longrightarrow v \in C(\Sigma)$
(iii)	$u, v \in C(\Sigma), T' \vdash uRv \Longrightarrow T \vdash uRv$
(iv)	$u \in C(\Sigma), T' \vdash uR_P\xi \Longrightarrow T \vdash Pu$
(v)	$u \in C(\Sigma), T \vdash Pu \Longrightarrow T' \vdash uR_P\xi$
(vi)	$u, v \in C(\Sigma), T \vdash uRv \Longrightarrow T' \vdash uRv$
(vii)	$u, v \in C(\Sigma) \Longrightarrow (u \sim v \iff u \sim' v)$

Before we continue, we discuss properties (ii)–(vi). Property (ii) says that if we can prove in the xenosystem that an old term u is related with a term v by means of an old relation R that v cannot be a xenoterm. A simple example due to Fokkink [**12**] 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:

$$xRx, \quad \frac{xRx}{aSx}.$$

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 $aS\xi$.

The properties (iii)–(vi) actually form two bi-implications saying that every provable formula in the old system is provable in the xenosystem and vice versa. We split these bi-implications into four statements for (proof) technical reasons.

Now we prove the properties.

The proof of (i) follows directly from the definition of the deduction rules in D'.

We prove (ii) with transfinite induction on the length of the proof (see section 3 for the definition of this length). Without loss of generality we may assume that the proof is closed; see lemma (3.3). Suppose that (ii) holds for all lengths $\alpha < \beta$ and that we have a proof of uRv of length β. The last rule d' used in this proof must be of the following form

$$\frac{\{s_j R_{P_j} z_j : j \in J\} \cup \{t_i R_i y_i : i \in I\}}{sRt}. \tag{1}$$

There is a substitution σ with $\sigma(s) = u$ and $\sigma(t) = v$. Note that $var(t) \subseteq var(s) \cup \{y_i : i \in I\}$ so it suffices to show that $\sigma(y_i) \in C(\Sigma)$ for all $i \in I$ since $\sigma(s) \in C(\Sigma)$. We denote the set of all y_i by Y. Suppose that there is an $y_{i_0} \in Y$ with $\sigma(y_{i_0}) \in C(\Sigma') \setminus C(\Sigma)$. This contradicts the well-foundedness of the rule d', for $T' \vdash \sigma(t_{i_0})R_{i_0}\sigma(y_{i_0})$ so with the induction hypothesis we find that $\sigma(t_{i_0}) \in C(\Sigma') \setminus C(\Sigma)$. Since t_{i_0} is a Σ-term, this must be the result of a substitution. This can only be due to a variable $y_{i_1} \in Y$. With induction on the subsubscript we find an infinite backward chain of edges $y_{i_0} \longleftarrow y_{i_1} \longleftarrow \ldots$ in the dependency graph of d'.

We simultaneously verify (iii)–(vi) with transfinite induction on the maximum α of the lengths of the proofs. Suppose that (iii)–(vi) hold for all maxima $\alpha < \beta$ and that we have proofs of (iii)–(vi) with maximum β. We begin with (iii). The last rule in the proof is again of the form that we displayed in (1) together with a substitution σ with $\sigma(s) = u$ and $\sigma(t) = v$. With (ii) we find for all $i \in I$ that $\sigma(y_i) \in C(\Sigma)$ just like in the proof of (ii). So we find using (i) and the induction hypothesis for (iii) and (iv) that $T \vdash \sigma(t_i)R_i\sigma(y_i)$ and $T \vdash P_j\sigma(s_j)$. The original rule in the old system T from which d' is constructed has the form

$$\frac{\{P_j s_j : j \in J\} \cup \{t_i R_i y_i : i \in I\}}{sRt}$$

so we find that $T \vdash \sigma(s)R\sigma(t)$. The case (iv) is treated analogously. Now we treat the case (v). The last rule d in the proof has the form

$$\frac{\{P_j s_j : j \in J\} \cup \{t_i R_i y_i : i \in I\}}{Ps}$$

and there is a substitution σ with $\sigma(s) = u$. With induction we find that $T' \vdash \sigma(t_i)R_i\sigma(y_i)$ and $T' \vdash \sigma(s_j)R_{P_j}\xi$. Let d' be rule that corresponds with d:

$$\frac{\{s_j R_{P_j} z_j : j \in J\} \cup \{t_i R_i y_i : i \in I\}}{sR_P\xi}.$$

Let Z be the set of all z_j. Define a substitution σ' with $\sigma'(z) = \xi$ for all $z \in Z$ and $\sigma'(x) = \sigma(x)$ for all $x \in V \setminus Z$. With d' and σ' we find that $T' \vdash uR_P\xi$. The case (vi) is treated analogously. This concludes the induction step.

Now we verify (vii). Firstly, let $u \sim v$. Then there is a bisimulation relation B with uBv. Define $B' = B \cup \Delta'$, with $\Delta' = \{(x, x) : x \in C(\Sigma')\}$ 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 (4.3) 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 (4.3). For each transfer property we have two cases: R and R_P. Now let $R \in T_r$ and suppose $T' \vdash sRs'$ for some $s' \in C(\Sigma')$. We find with (ii) that $s' \in C(\Sigma)$. Since sBt there is a $t' \in C(\Sigma)$ with $T \vdash tRt'$ and $s'Bt'$. So we obtain $s'B't'$ and with (vi) that $T' \vdash tRt'$. The second condition in definition (4.3) is verified analogously. We check the first condition for R_P. Let $P \in T_p$ and suppose that $T' \vdash sR_Ps'$ for some $s' \in C(\Sigma')$. We find with (i) that $s' = \xi$. So with (iv) we get $T \vdash Ps$. Since sBt we find $T \vdash Pt$ and with (v) that $T' \vdash tR_P\xi$. Clearly $\xi B'\xi$. The second condition in (4.3) 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 (4.3). Let $R \in T_r$ and suppose that $T \vdash sRs'$ for some $s' \in C(\Sigma)$. With (vi) we find $T' \vdash sRs'$. Since $sB't$ there is a $t' \in C(\Sigma')$ with $T' \vdash tRt'$ and $s'B't'$. With (ii) we find $t' \in C(\Sigma)$ so $s'Bt'$. With (iii) we have $T \vdash tRt'$. The second condition in definition (4.3) is verified analogously. Next, we show that the last condition of (4.3) from left to right holds. The other direction can be shown analogously. Let $P \in T_p$ and suppose that $T \vdash Ps$. With (v) we find $T' \vdash sR_P\xi$ so since $sB't$ there is a $t' \in C(\Sigma')$ with $T' \vdash tR_Pt'$. With (i) we find $t' = \xi$ so with (iv) we find $T \vdash Pt$. This concludes the proof of (vii).

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 (vii) we find $u_i \sim' v_i$ for all $1 \leq i \leq n$. Since the term deduction system T' is well-founded and in *tyft/tyxt* format we can apply the congruence theorem of Groote and Vaandrager [**16**] 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 (vii) that $f(u_1, \ldots, u_n) \sim f(v_1, \ldots, v_n)$. This concludes the proof of (5.4).

Corollary (5.5) Every term deduction system in *path* format can be reduced to a transition system specification in *tyft/tyxt* format. If the term deduction system is moreover well-founded then bisimulation equivalence is preserved: two terms are bisimilar in the *path* system iff they are bisimilar in the *tyft/tyxt* system.

6. Conclusions and future work

In this paper we have proposed a syntactical format (called the *path* format) for structured operational semantics with predicates such that strong bisimulation is a congruence for all the operators that can be defined within this format. For a number of operational semantics that we took from the literature we showed that our format is rather practical: many operational semantics satisfy our format, whereas they do not satisfy formats proposed by others, as for instance the *tyft/tyxt* format proposed by Groote and Vaandrager [**16**] or the *ntyft/ntyxt* format of Groote [**15**]. The examples that we discussed treat concepts like termination, convergence, divergence, a zero object, side conditions, functions, real time, discrete time, sequencing, negative premises, negative conclusions, and priorities (or a combination of these notions).

Future work is mainly concerned with negative rules. For instance, we want to generalize the *path* format with negative antecedents. We think that the structure of our proof for the congruence theorem also works for the negative case. Since we are already able to treat several operational semantics that use negative rules, the question arises how expressive all these formats are. For instance, which negative premises can be written as predicates?

Another possibility to generalizing the *path* format is to incorporate recursion within it as is done for the GSOS format [**11**] and De Simone's format [**25**].

Finally, we conclude that the *path* format is a very practical format that can be used with a lot of ease in a wide range of applications.

Acknowledgements We want to thank Frits Vaandrager for pointing out a redundancy in an earlier version of the proof of theorem (5.4). We also want to thank Wan Fokkink for his example that the well-foundedness cannot be missed in property (5.4)(*ii*). Also thanks to the referees for their valuable comments.

7. References

[**1**] L. Aceto, M. Hennessy, *Termination, deadlock and divergence*, JACM **39**(1):147–187, 1992.

[**2**] J. C. M. Baeten, J. A. Bergstra, *Discrete Time Process Algebra*, Report P9208b, Programming Research Group, University of Amsterdam, 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, LNCS **458**, pp. 83–98, Springer-Verlag, 1990.

[**4**] J. C. M. Baeten, J. A. Bergstra, J. W. Klop, *On the consistency of Koomen's fair abstraction rule*, TCS **51**(1/2), pp. 129–176, 1987.

[**5**] J. C. M. Baeten, J. A. Bergstra, J. W. Klop, *Syntax and defining equations for an interrupt mechanism in process algebra*, Fundamenta Informaticae **IX**, pp. 127–168, 1986.

[**6**] J. C. M. Baeten, R. J. van Glabbeek, *Merge and termination in process algebra*, in: K. V. Nori, editor, Proceedings 7*th* Conference on Foundations of Software Technology and Theoretical Computer Science, Pune, India, LNCS **287**, pp. 153–172, Springer-Verlag, 1987.

[**7**] J. C. M. Baeten and F. W. Vaandrager, *An algebra for process creation*, Acta Informatica **29**, pp. 303–334, 1992.

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

[**9**] J. A. Bergstra, J. W. Klop, *Algebra of communicating processes with abstraction*, TCS **37**, 77–121, 1985.

[**10**] J. A. Bergstra, J. W. Klop, *Fixed point semantics in process algebras*, MC report IW **206**, Mathematical Centre, Amsterdam, 1982. Revised version: J. A. Bergstra, J. W. Klop, *A convergence theorem in process algebra*, in Ten years of concurrency semantics: selected papers of the Amsterdam Concurrency Group, editors J. W. de Bakker, J. J. M. M. Rutten, World Scientific, pp. 164–195, 1992.

[**11**] 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.

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

[**13**] K. Futatsugi, J.A. Goguen, J.-P. Jouannaud, J. Meseguer, *Principles of OBJ2*, in Conference Record of the Twelfth Annual ACM Symposium on Principles of Programming Languages, editor B. Reid, pp. 52–66, ACM, 1985.

[**14**] 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.), LNCS **247**, Springer Verlag, pp. 336–347, 1987.

[**15**] J. F. Groote, *Transition system specifications with negative premises*, Report CS-R9850, CWI, Amsterdam, 1989. An extended abstract appeared in: see [**3**] pp. 332–341.

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

[**17**] A. Ingólfsdóttir, B. Thomsen, *Semantic Models for CCS with Values*, in: Proceedings Chalmers Workshop on Concurrency, 1991, pp. 215–242, Report PMG-R63, Chalmers University of Technology and University of Göteborg, 1992.

[**18**] 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, LNCS **527**, pp. 376–392, 1991.

[**19**] C. P. J. Koymans and J. L. M. Vrancken, *Extending process algebra with the empty process* ε, Logic Group Preprint Series Nr. 1, CIF, Utrecht University, 1985.

[**20**] R. Milner, *A calculus of communicating systems*, LNCS **92**, Springer Verlag, 1980.

[**21**] F. Moller and C. Tofts, *A Temporal Calculus of Communicating Systems*, in: see [**3**], pp. 401–415.

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

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

[**24**] S. Schneider, *An Operational Semantics for Timed CSP*, in: see [**17**] pp. 428–456. To appear in Information and Computation.

[**25**] R. de Simone, *Higher-level synchronising devices in* MEIJE-SCCS, TCS **37**, pp. 245–267, 1985.

[**26**] Wang Yi, *Towards a Theory of Testing for CCS with Probability*, in: see [**17**] pp. 476–492.