Congruence for SOS with data

Congruence for SOS with Data

MohammadReza Mousavi, Michel Reniers, Jan Friso Groote

Department of Computer Science,
Eindhoven University of Technology (TU/e),
Postbox 513, NL-5600 MB, Eindhoven, The Netherlands
Email: {m.r.mousavi,m.a.reniers,j.f.groote}@tue.nl

Abstract

While studying the specification of the operational semantics of different programming languages and formalisms, one can observe the following three facts. Firstly, Plotkin's style of Structured Operational Semantics (SOS) has become a standard in defining operational semantics. Secondly, congruence with respect to some notion of bisimilarity is an interesting property for such languages and it is essential in reasoning about them. Thirdly, there are numerous languages that contain an explicit data part in the state of the operational semantics.

The first two facts, have resulted in a line of research exploring syntactic formats of operational rules to derive the desired congruence property for free. However, the third point (in combination with the first two) is not sufficiently addressed and there is no standard congruence format for operational semantics with an explicit data state. In this paper, we address this problem by studying the implications of the presence of a data state on the notion of bisimilarity. Furthermore, we propose a number of formats for congruence.

1 Introduction

Structured Operational Semantics (called SOS for short) [23] has been very popular in defining operational semantics for different formalisms and programming languages. Moreover, congruence properties of notions of (bi-)simulation have been investigated in many of these languages as a key property for compositional reasoning and refinement. Congruence simply means that if one replaces a component in an arbitrary system with a (bi-)similar counterpart, the resulting system is (bi-)similar to the original one. Proofs of congruence for SOS semantics are usually standard but very tedious involving several pages. This has resulted in a line of research for defining standard syntactical formats for different types of SOS semantics in order to obtain the congruence property for a given notion of bisimilarity automatically.

From the early beginning, SOS has been used for languages with data as an integrated part of their operational state (e.g., the original report on SOS contains several examples of state-bearing transition system specifications [23]). As systems get more complex, the integration of a data state in their semantics becomes more vital. Besides the systems that have an explicit notion of data such as [4] and [9], real-time languages [15, 2, 18, 8] and hybrid languages [11] are other typical examples of systems in which a data state shows itself in the operational semantics in one way or another. However, the introduction of data turns out not to be as trivial as it seems and leads to new semantical issues such as adapted notions of bisimilarity [15, 11, 8, 21].

To the best of our knowledge, no standard congruence format for these different notions of bisimilarity with a data state has been proposed so far. Henceforth, most of the congruence proofs are done manually [21] or are just neglected by making a reference to a standard format that does not cover the data state [8]. The proposal that comes closest ([7]) is unfinished and encodes rules for state-bearing processes into rules without a state, for which a format is given.

In this paper, we address the implications of the presence of a data state on notions of bisimilarity and propose standard formats that induce congruence with respect to these notions of bisimilarity.

The rest of this paper is structured as follows. In Section 2, we review the related work in the area of congruence formats for SOS semantics. Then, in Section 3, we set the scene by defining transition system specifications with data and several notions of bisimilarity. In this section, we also sketch the relationship between these notions of bisimilarity and point out their application areas. The main contribution of this paper is introduced in Section 4, where we define standard syntactic formats for proving congruence with respect to the defined notions of bisimilarity. Furthermore, we give a full comparison between congruence results for the notions of bisimilarity with data. Subsequently, Section 5 presents applications of the proposed theory on a number of transition system specifications from the literature. Finally, Section 6 concludes the paper and presents possible extensions of the proposed approach.

2 Related Work

The first standard format for congruence was proposed by De Simone in [12]. The De Simone format allows for deduction rules of the following form:

$$\frac{\{x_i \xrightarrow{l_i} y_i | i \in I\}}{f(x_0, \ldots, x_{n-1}) \xrightarrow{l} t} \quad P(\overrightarrow{l_i}, l).$$

where x_i and y_i are variables ranging over process terms, f is an n-ary function from the signature (e.g., sequential composition, parallel composition, etc.), I is a subset of the set $\{0, \ldots, n-1\}$ (indices of arguments of f), t is an arbitrary process term and P is a predicate stating the relationship between the labels of the premises and the label of the conclusion. (It turns out that side conditions of this kind do not play any role on the congruence result and thus we do not mention them in the remainder.)

Bloom, Istrail and Meyer, in their study of the relationship between bisimilarity and trace congruence [6], define an extension of the De Simone format, called *GSOS*, to capture *reasonable* language definitions. GSOS extends the De Simone format to cover negative premises. In other words, it allows for deduction rules of the following form:

$$\frac{\{x_i \xrightarrow{l_{ij}} y_{ij} | i \in I, j \leq m\} \cup \{x_j \overset{l_{jk}}{\nrightarrow} | j \in J, k \leq n\}}{f(x_0, \ldots, x_{n-1}) \xrightarrow{l} t}.$$

All common notations in the rule above have the same intuition as those of the De Simone format, J is a subset of indices of f (similar to I) and m and n are two natural numbers (to set an upper bound on the number of premises).

Another orthogonal extension of the De Simone format, called *tyft/tyxt*, is proposed in [17]. This format allows for arbitrary terms in the left-hand-sides of the premises and in the right-hand-side of the conclusion but requires the left-hand-side of the conclusion to be a variable or a function applied to variables only, the right-hand-sides of premises to be variables and all these variables to be distinct. In other words, it allows for the following forms of deduction rules:

$$\frac{\{t_i \xrightarrow{l_i} y_i | i \in I\}}{f(x_0, \ldots, x_{n-1}) \xrightarrow{l} t}, \qquad \frac{\{t_i \xrightarrow{l_i} y_i | i \in I\}}{x \xrightarrow{l} t}.$$

All notations again share the same intuition with those of the De Simone format, apart from I, which is now an arbitrary (possibly infinite) set. The only further restriction on the set of premises, imposed for proving the congruence theorem in [17], is the acyclicity of a *variable dependency graph.* A variable dependency graph has variables as its nodes and there exists an edge from one variable to another if the former appears in the left-hand-side and the latter in the right-hand-side of a premise. Later, in [13] it is shown that the acyclicity constraint can be relaxed and that for every *tyft/tyxt* rule with a non-well-founded set of premises (with respect to the variable dependency ordering) there exists a rule that induces the same transition relation and is indeed well-founded.

The merits of the two extensions were merged in [14] where negative premises were added to the *tyft/tyxt* format, resulting in the *ntyft/ntyxt* format. In the premises of *ntyft/ntyxt* rules negative transition relations of the form $t_i \overset{l_i}{\nrightarrow}$ can appear as well, provided that the transition system specification can be *stratified.* Stratification is concerned with defining a measure that decreases from the conclusion to negative premises and that does not increase from the conclusion to positive premises.

Finally, the PATH format [3] and the PANTH format [25] extend *tyft/tyxt* and *ntyft/ntyxt* with predicates, respectively. A deduction rule in PANTH format may have predicates, negative predicates, transitions and negative transitions in its premises and a predicate or a transition in its conclusion.

In [19], the PANTH format is extended for multi-sorted variable binding. This covers the problem of operators such as recursion or choice over a time domain. The issue of binding operators for multi-sorted process terms is also briefly introduced in [1]. In an unpublished note [7], the issue of state-bearing processes and multi-sorted transition system specifications is treated and three congruence formats are proposed (Super-SOS, Data-SOS, and Special-SOS). The approach of [7] relies on transforming state-bearing processes to multi-sorted ones and thus, state-bearing transition system specifications cannot be dealt with in their original representation (whilst this is possible in our approach). The notion of bisimilarity in [7] seems to be what we call stateless bisimilarity in this paper. However, the formats they propose for this notion of bisimilarity are not proved to induce congruence and moreover they impose some unneeded restrictions that are not present in our format for stateless bisimilarity. Furthermore, the formats in [7] do not induce congruence with respect to the other notions of bisimilarity that we discuss in this paper.

We recognize the problem of multi-sorted process terms and admit that it is an interesting problem in itself. However, we address a different issue in this paper, that is, the issue of states having a particular structure (possibly from different sorts). In the above works, the data state is coded into process terms (either naturally due to the definition of operators like time-choice operators, e.g., in [19, 24] or artificially by transforming a multi-sorted state to a single-sorted one, e.g., in [7]). Thus, the transition system specification as well as the notion of equivalence are confined to look at the behavior of process terms (since standard formats allow defining only one function symbol at a time in the conclusion of a deduction rule). However, if the data state is made explicit as a part of the state in the transition system specification, then the transition system specification may not only address process composition operators, but also data composition operators. This allows both for more expressivity in the specification of SOS and for the possibility of introducing new notions of equivalence (w.r.t. the relationship between data and process terms).

In [5], an extension of the *tyft/tyxt* format is proposed to cover the semantics of higher order languages. The extended format, called *promoted tyft/tyxt*, allows for putting (open) terms on the labels as well as on the two sides of the transition relation specification. Since labels are assumed to be of the same sort as process terms, their results do not apply to our problem domain directly (in which we have at least two different signatures for processes and data). However, by assuming two disjoint parts of the same signature as data and process signatures we may get a weaker result for the case of stateless bisimilarity in Section 4 (i.e., the resulting format would be more restrictive than ours). For the more involved notions of bisimilarity, however, we have to move to a multi-sorted state paradigm in order to define our criteria for the standard format and thus the format of [5] is not applicable.

In [11], to prove congruence for a specification language with a data state, the transition system specification is partially transformed to another transition system specification that is isomorphic to the original one and does not contain a data state. Furthermore, it is shown that the original notion of bisimilarity corresponds to strong bisimilarity [20, 22] in the new specification. Since the resulting transition system specification is in PATH format, it is deduced that the original notion of bisimilarity is a congruence. Although, the formal proof for these steps is not given there in detail, the proof sketch seems to be convincing for this particular notion of bisimilarity (i.e., stateless bisimilarity). We combine all these steps here in one theorem and prove it so that such transformations and proofs are not necessary anymore. Furthermore, we give standard formats for other notions of bisimilarity for which such a straightforward transformation does not exist.

3 Preliminaries

3.1 Basic Definitions

We assume infinite and disjoint sets of process variables V_p (with typical members x, y, x', y', x_0, $y_0 \ldots$) and data variables V_d (with typical members v and v'). A process signature Σ_p is a set of function symbols with their fixed arity. Functions with zero arity are called constants. A process term $t \in T(\Sigma_p)$ is defined inductively as follows: a variable $x \in V_p$ is a process term, if $t_0, \ldots, t_{n-1}$ are process terms then for all $f \in \Sigma_p$ with arity n, $f(t_0, \ldots, t_{n-1})$ is a process term, as well (i.e., constants are indeed process terms). Process terms are typically denoted by $t, t', t_i, t'_i, \ldots$. Similarly, data terms $u \in T(\Sigma_d)$ are defined based on a data signature Σ_d and the set of variables V_d and typically denoted by $u, u', u_i, u'_i, \ldots$. Closed terms $C(\Sigma_x)$ in each of these contexts are defined as expected (closed process terms are typically denoted by $p, q, p', q', p_0, q_0, p'_0, q'_0 \ldots$). A substitution σ replaces a variable in an open term with another (possibly open) term. The set of variables appearing in term t is denoted by $vars(t)$.

Definition 3.1 (Transition System Specification) A *transition system specification with data* is a tuple $(\Sigma_p, \Sigma_d, L, D(Rel))$ where Σ_p is a process signature, Σ_d is a data signature, L is a set of labels (with typical members $l, l', l_0, \ldots$), and $D(Rel)$ is a set of deduction rules, where Rel is a set of (ternary) relation symbols. For all $r \in Rel$, $l \in L$ and $s, s' \in T(\Sigma_p) \times T(\Sigma_d)$ we define that $(s, l, s') \in r$ is a formula. A deduction rule $dr \in D$, is defined as a tuple (H, c) where H is a set of formulas and c is a formula. The formula c is called the conclusion and the formulas from H are called premises.

Notions of open and closed and the concept of substitution are lifted to formulas in the natural way. A formula $(s, l, s') \in r$ is denoted by the more intuitive notation $s \xrightarrow{l}_r s'$, as well. A deduction rule is mostly denoted by $\frac{H}{c}$.

A *proof* of a formula ϕ is a well-founded upwardly branching tree of which the nodes are labelled by formulas such that

- the root node is labelled by ϕ, and
- if ψ is the label of a node q and $\{\psi_i \mid i \in I\}$ is the set of labels of the nodes directly above q, then there is a deduction rule $\frac{\{\chi_i \mid i \in I\}}{\chi}$, a process substitution σ and a data substitution υ such that application of these substitutions to χ gives the formula ψ, and for all $i \in I$, application of the substitutions to χ_i gives the formula ψ_i.

3.2 Notions of Bisimilarity

The introduction of data to the state adds a new dimension to the notion of bisimilarity. One might think that we can easily deal with data states by imposing the original notion of strong bisimilarity [20, 22] to the extended state. Our survey of the literature has revealed that such a notion of strong bisimilarity is not used at all. It is clear that a format that respects strong bisimilarity as a congruence must necessarily be very restricted. Therefore, in this paper, we restrict ourselves to comparing processes with respect to the same data state. In this way, we get to what we call a *statebased bisimilarity*, depicted in Figure 1.

$$\begin{tikzcd}
& (p_0, d_0) \arrow[dl] \arrow[d] \arrow[rr, dash, dashed, "R_{sb}"] & & (p_0', d_0) \arrow[dl] \arrow[d] \\
(p_1, d_1) \arrow[rr, dash, dashed, bend left, "R_{sb}"] & (p_2, d_2) \arrow[rr, dash, dashed, bend right, "R_{sb}"'] & (p_1', d_1) & (p_2', d_2)
\end{tikzcd}$$

Figure 1: Statebased Bisimilarity

Definition 3.2 (Statebased Bisimilarity) A relation $R_{sb} \subseteq (C(\Sigma_p) \times C(\Sigma_d)) \times (C(\Sigma_p) \times C(\Sigma_d))$ is a *statebased bisimulation* relation if and only if $\forall_{p_0, p_1, d}\ ((p_0, d), (p_1, d)) \in R_{sb} \Rightarrow \forall_r$

1. $\forall_{l_0, p_0', d'}\ (p_0, d) \xrightarrow{l_0}_r (p_0', d') \Rightarrow \exists_{p_1'}\ (p_1, d) \xrightarrow{l_0}_r (p_1', d')\ \wedge ((p_0', d'), (p_1', d')) \in R_{sb}$;

2. $\forall_{l_1, p_1', d'}\ (p_1, d) \xrightarrow{l_1}_r (p_1', d') \Rightarrow \exists_{p_0'}\ (p_0, d) \xrightarrow{l_1}_r (p_0', d')\ \wedge ((p_0', d'), (p_1', d')) \in R_{sb}$.

Two closed state terms (p, d) and (q, d) are *statebased bisimilar*, denoted by $(p, d) \underline{\leftrightarrow}_{sb} (q, d)$, if and only if there exists a statebased bisimulation relation R_{sb} such that $((p, d), (q, d)) \in R_{sb}$.

Definition 3.3 (Process-congruence) For $\sim \subseteq (C(\Sigma_p) \times C(\Sigma_d)) \times (C(\Sigma_p) \times C(\Sigma_d))$, $\sim$ is called a *process-congruence* w.r.t. an n-ary process function $f \in \Sigma_p$ if and only if for all $p_i, q_i \in C(\Sigma_p)$ $(0 \leq i < n)$, for all $d \in C(\Sigma_d)$, if $(p_i, d) \sim (q_i, d)$ then $(f(p_0, \ldots, p_{n-1}), d) \sim (f(q_0, \ldots, q_{n-1}), d)$. Furthermore, $\sim$ is called a *process-congruence* for a transition system specification if and only if it is a process-congruence w.r.t. all process functions of the process signature.

Example 3.4 Consider a transition system specification, where the signature consists of three (distinct) process constants a, b and c, one binary process function f, and three (distinct) data constants d, d' and d'', and the deduction rules are the following:

$$(1)\ \frac{}{(a, d) \xrightarrow{l} (a, d'')} \qquad (2)\ \frac{}{(a, d') \xrightarrow{l} (a, d')} \qquad (3)\ \frac{}{(b, d) \xrightarrow{l} (b, d'')}$$

$$(4)\ \frac{}{(c, d) \xrightarrow{l} (a, d'')} \qquad (5)\ \frac{(x_0, v) \xrightarrow{l} (y, v')}{(f(x_0, x_1), v) \xrightarrow{l} (x_1, d')}$$

Then, the following statebased bisimilarities hold: $(a, d) \underline{\leftrightarrow}_{sb} (b, d)$ and $(b, d) \underline{\leftrightarrow}_{sb} (c, d)$. However, the following statebased bisimilarities do not hold: $(a, d') \underline{\leftrightarrow}_{sb} (b, d')$ and $(f(c, a), d) \underline{\leftrightarrow}_{sb} (f(c, b), d)$. From the latter case, we can observe that statebased bisimilarity is not a process-congruence for the above transition system specification.

Statebased bisimilarity is a rather weak notion of bisimilarity for most practical examples. The problem lies in the fact that in this notion of bisimilarity the process parts are only related with respect to a particular data state. Thus, if the common initial data state is not known (e.g., if the components have to start their execution on the result of an unknown or non-deterministic process), then statebased bisimilarity is not useful.

This problem leads to the introduction of a new notion of bisimilarity which takes all possible initial states into account [16, 15]. We call this notion *initially stateless bisimilarity* (see Figure 2). This notion of bisimilarity is very useful for the case where components are composed sequentially. In such cases, when we prove that two components are bisimilar, we do not rely on the initial starting state and thus, we allow for sequential composition with any other component.

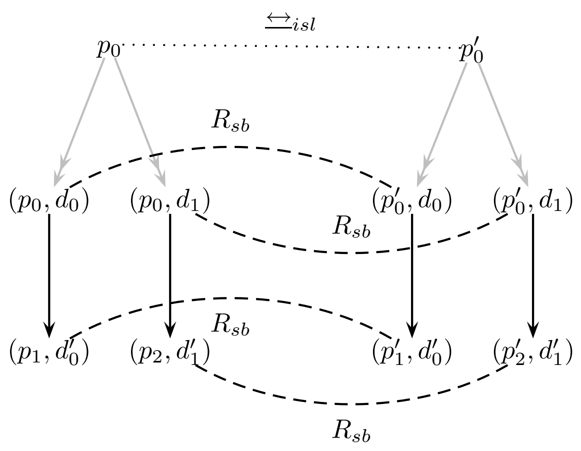

Figure 2: Initially Stateless Bisimilarity

Definition 3.5 (Initially Stateless Bisimilarity) Two closed process terms p and q are *initially stateless bisimilar*, denoted by $p \underline{\leftrightarrow}_{isl} q$, if and only if there exists a statebased bisimulation relation R_{sb} such that $((p, d), (q, d)) \in R_{sb}$ for all $d \in C(\Sigma_d)$.

For initially stateless bisimilarity (and also for stateless bisimilarity), congruence is defined as expected in the following definition.

Definition 3.6 (Congruence) For arbitrary $\sim \subseteq C(\Sigma_p) \times C(\Sigma_p)$, $\sim$ is called a *congruence* w.r.t. an n-ary process function $f \in \Sigma_p$ if and only if for all $p_i, q_i \in C(\Sigma_p)$ $(0 \leq i < n)$, if $p_i \sim q_i$ then $f(p_0, \ldots, p_{n-1}) \sim f(q_0, \ldots, q_{n-1})$. Furthermore, $\sim$ is called a *congruence* for a transition system specification if and only if it is a congruence w.r.t. all process functions of the process signature.

Example 3.7 Consider the transition system specification of Example 3.4. The following initially stateless bisimilarities hold $b \underline{\leftrightarrow}_{isl} c$ and $f(b, c) \underline{\leftrightarrow}_{isl} f(c, c)$ but the following initially stateless bisimilarities do not hold $a \underline{\leftrightarrow}_{isl} b$ and $f(c, a) \underline{\leftrightarrow}_{isl} f(c, b)$. We observe that the previous problem of congruence does not exist anymore for initially stateless bisimilarity. Later on, in Example 4.25, we show that for this transition system specification, initially stateless bisimilarity is indeed a congruence.

However, initially stateless bisimilarity does not solve all problems, either. If there is a possibility of change in the intermediate data states (by an outside process), then initially stateless bisimilarity is not preserved in such an environment. This, for instance, happens in open concurrent systems.

Stateless bisimilarity [16, 8, 21, 11], shown in Figure 3, is the solution to this problem and the finest notion of bisimilarity for state-bearing processes that that one can find in the literature. Two process terms are stateless bisimilar if, for all identical data states, they satisfy the same predicates and they can mimic transitions of each other and the resulting process terms are again stateless bisimilar. In other words, we compare process terms for all identical data states and allow all sorts of change (interference) in the data part after each transition.

Definition 3.8 (Stateless Bisimilarity) A relation $R_{sl} \subseteq C(\Sigma_p) \times C(\Sigma_p)$ is a *stateless bisimulation* relation if and only if $\forall_{p_0,p_1}$ $(p_0, p_1) \in R_{sl} \Rightarrow \forall_r$

1. $\forall_{d_0,l_0,p'_0,d'_0}$ $(p_0, d_0) \xrightarrow{l_0}_r (p'_0, d'_0) \Rightarrow \exists_{p'_1}$ $(p_1, d_0) \xrightarrow{l_0}_r (p'_1, d'_0)$ $\wedge (p'_0, p'_1) \in R_{sl}$;

2. $\forall_{d_1,l_1,p'_1,d'_1}$ $(p_1, d_1) \xrightarrow{l_1}_r (p'_1, d'_1) \Rightarrow \exists_{p'_0}$ $(p_0, d_1) \xrightarrow{l_1}_r (p'_0, d'_1)$ $\wedge (p'_0, p'_1) \in R_{sl}$.

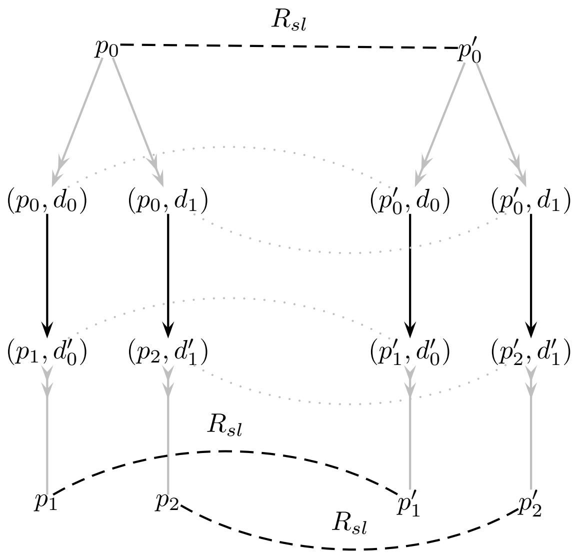

Figure 3: Stateless Bisimilarity

Two closed process terms p and q are *stateless bisimilar*, denoted by $p \underline{\leftrightarrow}_{sl}\ q$, if and only if there exists a stateless bisimulation relation R_{sl} such that $(p, q) \in R_{sl}$.

Example 3.9 Consider the transition system specification of Example 3.4. None of the non-trivial examples of bisimilarity hold anymore for stateless bisimilarity. Namely, it does not hold that $a \underline{\leftrightarrow}_{sl}\ b$, $a \underline{\leftrightarrow}_{sl}\ c$ or $b \underline{\leftrightarrow}_{sl}\ c$. From these one may conclude that stateless bisimilarity is a congruence for the above transition system specification. We prove this formally in Example 4.5.

one of the three notions of bisimilarity is the perfect notion. Statebased bisimilarity is the easiest one to check and establish but is not very robust in application. It is most suitable for closed deterministic sequential systems. Initially stateless bisimilarity is a bit more difficult to check and establish but is more robust and suitable for closed nondeterministic sequential systems. Finally, stateless bisimilarity is the hardest one to establish but it is considered the most robust one for open concurrent systems. In general, a compromise has to be made in order to find the right level of robustness and strength and as a result the most suitable notion of bisimilarity has to be determined for each language / application separately.

A common practice in establishing bisimulation relations for concurrent systems is to transform them to nondeterministic sequential systems preserving stateless bisimilarity and then using initially stateless bisimilarity in that setting [16]. Another option for open systems with a limited possibility of intervention from the environment is to parameterize the notion of bisimilarity with an interference relation [16, 10, 11]. Our congruence format for statebased bisimilarity can easily be adapted to the parameterized notion of bisimilarity.

Next, we compare the above three notions of bisimilarity.

3.3 Comparing the Notions of Bisimilarity

The following corollary states that stateless bisimilarity implies statebased bisimilarity with respect to all data states.

Corollary 3.10 If $p \underline{\leftrightarrow}_{sl}\ q$, then $(p, d) \underline{\leftrightarrow}_{sb}\ (q, d)$ for all $d \in C(\Sigma_d)$.

In Examples 3.4 and 3.9, we have shown that two processes b and c are statebased bisimilar (w.r.t. data state d) but stateless bisimilarity fails to hold between them. Thus, we may infer that stateless bisimilarity is finer than statebased bisimilarity (w.r.t. a particular data state). The following corollary states that if a statebased bisimilarity relation is closed under the change of data state then it induces a stateless bisimulation relation.

Corollary 3.11 Consider a statebased bisimulation relation R. If for $\forall_{p,q,d}$ $((p,d),\ (q,d)) \in R \Rightarrow \forall_{d'}$ $((p,d'),(q,d')) \in R$ then $\forall_{p,q}$ $(\exists_d$ $((p,d),(q,d)) \in R) \Rightarrow p \underline{\leftrightarrow}_{sl}$ q.

Finally, the following lemma positions initially stateless bisimilarity in between the two other notions of bisimilarity we have discussed so far.

Corollary 3.12 For two arbitrary closed process terms p and q and an arbitrary closed data term d, we have

1. if $p \underline{\leftrightarrow}_{sl}$ q, then $p \underline{\leftrightarrow}_{isl}$ q;
2. $p \underline{\leftrightarrow}_{isl}$ q if and only if, $(p,d) \underline{\leftrightarrow}_{sb}$ (q,d) for all d.

Again, in Examples 3.4 and 3.7, we have shown that a and b are statebased bisimilar with respect to d but they are not initially stateless bisimilar. Thus, statebased bisimilarity (with respect to a particular data state) is strictly weaker than initially stateless bisimilarity.

4 Standard Formats for Congruence

In this section we present standard formats and prove congruence results with respect to aforementioned notions of bisimilarity. To do this, we extend the *tyft* format of [15] with data in three steps for stateless, statebased, and initially stateless bisimilarity. Finally, we present how our formats can be extended to cover *tyxt* rules and rules containing predicates and negative premises (thus, extending the PANTH format [25] with data).

4.1 Congruence Format for Stateless Bisimilarity

In this paper, we allow for deduction rules that adhere to the tyft-format with respect to the process terms and are not restricted in the data terms. This format is called *process-tyft.*

Definition 4.1 (Process-tyft) Let $(\Sigma_p, \Sigma_d, L, D(Rel))$ be a transition system specification. A deduction rule $dr \in D(Rel)$ is in *process-tyft format* if it is of the form

$$(dr) \quad \frac{\{(t_i, u_i) \xrightarrow{l_i}_{r_i} (y_i, u_i') | i \in I\}}{(f(x_0, \ldots, x_{n-1}), u) \xrightarrow{l}_{r} (t', u')},$$

where I is a set of indices, $\rightarrow_r\ \in Rel$, $l \in L$, $f \in \Sigma_p$ is a process function of arity n, the variables $x_0, \ldots, x_{n-1}$ and y_i $(i \in I)$ are all distinct variables from V_p, and, for all $i \in I$: $\rightarrow_{r_i}\ \in Rel$, $l_i \in L$, $t_i, t' \in T(\Sigma_p)$ and $u, u', u_i, u_i' \in T(\Sigma_d)$.

We name the set of process variables appearing in the left-hand-side of the conclusion X_p and in the right-hand-side of the premises Y_p. The two sets X_p and Y_p are obviously disjoint following the requirements of the format. The above deduction rule is called an *f-defining deduction rule.*

A transition system specification is in *process-tyft format* if all its deduction rules are in process-tyft format.

It turns out that for any transition system specification in process-tyft format, stateless bisimilarity is a congruence.

Theorem 4.2 If a transition system specification is in process-tyft format, then stateless bisimilarity is a congruence for that transition system specification.

Before we prove this theorem, we first define the closure of a relation under stateless congruence and give and prove a lemma that is very useful in the proof of Theorem 4.2.

Definition 4.3 (Closure under stateless congruence) Let $R \subseteq C(\Sigma_p) \times C(\Sigma_p)$. We define the relation $\tilde{R} \subseteq C(\Sigma_p) \times C(\Sigma_p)$ to be the smallest reflexive congruence on $C(\Sigma_p)$ such that the relation R is contained in $\tilde{R}$. Formally, $\tilde{R}$ is defined to be the smallest relation that satisfies:

1. $\tilde{R}$ is reflexive;
2. $R \subseteq \tilde{R}$;
3. $(f(p_0, \dots, p_{n-1}), f(q_0, \dots, q_{n-1})) \in \tilde{R}$ for all n-ary $f \in \Sigma_p$, and all $p_0, \dots, p_{n-1}, q_0, \dots, q_{n-1} \in C(\Sigma_p)$ such that $(p_i, q_i) \in \tilde{R}$ for all $0 \leq i < n$.

Lemma 4.4 Let $R \subseteq C(\Sigma_p) \times C(\Sigma_p)$ and $t \in T(\Sigma_p)$. For any two substitutions σ and σ' such that $(\sigma(x), \sigma'(x)) \in \tilde{R}$ for all $x \in vars(t)$, we have $(\sigma(t), \sigma'(t)) \in \tilde{R}$.

Proof. By induction on the structure of process term t. First observe that as the relation R is on closed process terms only, necessarily the substitutions σ and σ' must be such that application of them to any process term with only variables from $vars(t)$ results in a closed process term. In case t is a variable, say x, we obtain $\sigma(t) = \sigma(x)$ and $\sigma'(t) = \sigma'(x)$. As $x \in vars(t)$, we have $(\sigma(x), \sigma'(x)) \in \tilde{R}$ and hence $(\sigma(t), \sigma'(t)) \in \tilde{R}$.

In case t is a constant, say c, we obtain $\sigma(t) = \sigma(c) = c = \sigma'(c) = \sigma'(t)$. Then $(\sigma(t), \sigma'(t)) \in \tilde{R}$ follows immediately from the fact that $\tilde{R}$ is reflexive.

Finally, consider the case where $t = f(t_0, \dots, t_{n-1})$ for some n-ary function symbol $f \in \Sigma_p$ $(n > 0)$ and $t_i \in T(\Sigma_p)$ for $0 \leq i < n$. Then, as $vars(t_i) \subseteq vars(t)$ for all $0 \leq i < n$, we obtain, by the induction hypothesis, $(\sigma(t_i), \sigma'(t_i)) \in \tilde{R}$ for all $0 \leq i < n$. Since $\sigma(t) = \sigma(f(t_0, \dots, t_{n-1})) = f(\sigma(t_0), \dots, \sigma(t_{n-1}))$ and $\sigma'(t) = \sigma'(f(t_0, \dots, t_{n-1})) = f(\sigma'(t_0), \dots, \sigma'(t_{n-1}))$, and the relation $\tilde{R}$ is closed under congruence, we obtain $(\sigma(t), \sigma'(t)) \in \tilde{R}$. ⊠

Proof. (*Theorem 4.2*) It suffices to prove that stateless bisimilarity is a congruence for each of the process functions of Σ_p. Let $f \in \Sigma_p$ be an n-ary process function. Let p_i and q_i be closed process terms for $0 \leq i < n$. Suppose that $p_i \leftrightarrow_{sl} q_i$ for $0 \leq i < n$. This means that there are stateless bisimulation relations R_i (for $0 \leq i < n$) that witness these stateless bisimilarities. Let R be the union of these relations R_i: $R = \bigcup_{i=0}^{n} R_i$. Obviously R is also a stateless bisimulation relation. We prove that the relation $\tilde{R}$ contains the pair $(f(p_0, \dots, p_{n-1}), f(q_0, \dots, q_{n-1}))$ and that it is a stateless bisimulation relation. The first part is obvious from the definition of $\tilde{R}$.

So, we only have to prove the following for any $(p, q) \in \tilde{R}$: if for arbitrary $\rightarrow_r$, l, p', d and d', $(p, d) \xrightarrow{l}_r (p', d')$, then there exists a q' such that $(q, d) \xrightarrow{l}_r (q', d')$ and $(p', q') \in \tilde{R}$ and vice versa for transitions of q. Due to symmetry, it suffices to provide the proofs for the transitions of p only.

We prove this by induction on the depth of the proof of a transition. We do not show the proof for the induction base as it is an instance of the proof of the induction step where there are no premises.

For the induction step, we distinguish three cases based on the structure of the definition of $\tilde{R}$. In case the pair (p, q) is contained in $\tilde{R}$ due to reflexivity of $\tilde{R}$ or due to the requirement that $\tilde{R}$ contains R, the proof is obvious (and requires no induction at all). For the remaining case, we find $p = f(p_0, \dots, p_{n-1})$ and $q = f(q_0, \dots, q_{n-1})$ for some $p_0, \dots, p_{n-1}, q_0, \dots, q_{n-1}$ such that $(p_i, q_i) \in \tilde{R}$ for all $0 \leq i < n$. The last step of the proof of the transition of p is due to the application of a deduction rule of the following form:

$$\frac{\{(t_i, u_i) \xrightarrow{l_i}_{r_i} (y_i, u_i') | i \in I\}}{(f(x_0, \dots, x_{n-1}), u) \xrightarrow{l}_r (t', u')}.$$

This means that there are substitutions σ and υ such that $\sigma(x_i) = p_i$ for all $0 \leq i < n$, $\upsilon(u) = d$, $\sigma(t') = p'$ and $\upsilon(u') = d'$. Furthermore, for each $i \in I$, there exist a proof of $(\sigma(t_i), \upsilon(u_i)) \xrightarrow{l_i}_{r_i} (\sigma(y_i), \upsilon(u'_i))$ with smaller depth.

Since we have assumed acyclicity of the variable dependency graph, we can define a rank, $rank(x)$, for each variable x, as the maximum length of a backward chain starting from x in the variable dependency graph. The rank of a premise is the rank of its right-hand-side variable. Then, for each $x \in vars(t_i)$ of each premise $(t_i, u_i) \xrightarrow{l_i}_{r_i} (y_i, u'_i)$ of the deduction rule, it holds that $rank(x) < rank(y_i)$.

We define the substitution σ' as follows:

$$\sigma'(x) = \begin{cases} q_i & \text{if } x = x_i, \\ \sigma(x) & \text{if } x \notin X_p \cup Y_p. \end{cases}$$

Note that thus far this substitution is not defined for variables from Y_p. We extend the definition while proving, by induction on the rank of a premise r, three essential properties: for all r, for all $i \in I$ such that $rank(y_i) = r$,

1. $(\sigma(t_i), \sigma'(t_i)) \in \tilde{R}$;
2. $(\sigma'(t_i), \upsilon(u_i)) \xrightarrow{l_i}_{r_i} (\sigma'(y_i), \upsilon(u'_i))$;
3. $(\sigma(y_i), \sigma'(y_i)) \in \tilde{R}$.

Again, we do not show the proof of the induction base ($r = 0$) as it is an instance of the proof of the induction step.

For the induction step, suppose $r \geq 1$. Let $(t_i, u_i) \xrightarrow{l_i}_{r_i} (y_i, u'_i)$ for some $i \in I$ be a premise of rank r. First, we prove property (1). Let $x \in vars(t_i)$. We distinguish three cases:

1. $x \in X_p$. Then $x = x_i$ for some $0 \leq i < n$. From the definition of σ' we have that $\sigma(x) = \sigma(x_i) = p_i$ and $\sigma'(x_i) = q_i$ and we already know that $(p_i, q_i) \in \tilde{R}$. Thus, we have $(\sigma(x), \sigma'(x)) \in \tilde{R}$.
2. $x \notin X_p$ and $x \notin Y_p$. As $\sigma(x) = \sigma'(x)$ and the identity relation is contained in $\tilde{R}$ obviously $(\sigma(x), \sigma'(x)) \in \tilde{R}$.
3. $x \in Y_p$. Then $x = y_j$ for some $j \in I$. Obviously, also $rank(y_j) < rank(y_i)$. Thus by the induction hypothesis (property (3)) we have, $(\sigma(y_j), \sigma'(y_j)) \in \tilde{R}$. But, as $x = y_j$, we also have $(\sigma(x), \sigma'(x)) \in \tilde{R}$.

From the fact that $(\sigma(x), \sigma'(x)) \in \tilde{R}$ for all $x \in vars(t_i)$, we have, by Lemma 4.4, that $(\sigma(t_i), \sigma'(t_i)) \in \tilde{R}$; which proves property (1).

As we have a proof of smaller depth for $(\sigma(t_i), \upsilon(u_i)) \xrightarrow{l_i}_{r_i} (\sigma(y_i), \upsilon(u'_i))$, by the induction hypothesis, we have the existence of a process term q'_i such that $(\sigma'(t_i), \upsilon(u_i)) \xrightarrow{l_i}_{r_i} (q'_i, \upsilon(u'_i))$ and $(\sigma(y_i), q'_i) \in \tilde{R}$. We choose $\sigma'(y_i)$ to be q'_i. Observe that this proves existence of an appropriate process term $\sigma'(y_i)$. Then, we also have $(\sigma'(t_i), \upsilon(u_i)) \xrightarrow{l_i}_{r_i} (\sigma'(y_i), \upsilon(u'_i))$ and $(\sigma(y_i), \sigma'(y_i)) \in \tilde{R}$, which prove properties (2) and (3).

Now, we finish our reasoning using process substitution σ' and the same data substitution and deduction rule. Observe that indeed $\sigma'(f(x_0, \ldots, x_{n-1})) = f(q_0, \ldots, q_{n-1}) = q$. By property (2) we have proven that there are proofs for all premises using the substitutions σ' and υ. Then, according to the same deduction rule and using σ' instead of σ, we have $(\sigma'(f(x_0, \ldots, x_{n-1})), \upsilon(u)) \xrightarrow{l}_{r} (\sigma'(t'), \upsilon(u'))$. Since $\sigma'(f(x_0, \ldots, x_{n-1})) = f(q_0, \ldots, q_{n-1}) = q$, $\upsilon(u) = d$ and $\upsilon(u') = d'$ we obtain $(q, d) \xrightarrow{l}_{r} (\sigma'(t'), d')$.

We only have to show that $(\sigma(t'), \sigma'(t')) \in \tilde{R}$. By Lemma 4.4, it suffices to show that $(\sigma(x), \sigma'(x)) \in \tilde{R}$ for all $x \in vars(t')$. Three cases can be distinguished:

1. $x \in X_p$. Then $x = x_i$ for some $0 \leq i < n$. We have that $\sigma(x_i) = p_i$ and $\sigma'(x_i) = q_i$ and we already know that $(p_i, q_i) \in \tilde{R}$ and $x_i = x$. Thus, we have $(\sigma(x), \sigma'(x)) \in \tilde{R}$.

2. $x \notin X_p$ and $x \notin Y_p$. As $\sigma(x) = \sigma'(x)$ and the identity relation is contained in $\tilde{R}$ obviously $(\sigma(x), \sigma'(x)) \in \tilde{R}$.

3. $x \in Y_p$. Then $x = y_j$ for some $j \in I$. By property (3) we have, $(\sigma(y_j), \sigma'(y_j)) \in \tilde{R}$. But, as $x = y_j$, we also have $(\sigma(x), \sigma'(x)) \in \tilde{R}$.

So this concludes the proof of Theorem 4.2. ⊠

Example 4.5 Consider the transition system specification of Example 3.4. Obviously, all deduction rules are in process-tyft format, hence, by Theorem 4.2, stateless bisimilarity is a congruence for all process functions from the signature of this transition system specification.

4.2 Congruence Format for Statebased Bisimilarity

In this section, we introduce a format for establishing congruence of statebased bisimilarity. First, we show that we cannot simply use the previously introduced process-tyft format.

Example 4.6 Consider a transition system specification in process-tyft format, where the signature consists of three process constants a, b, and c, one unary process function f, and two data constants d and d' and the deduction rules are the following:

$$(1)\ \frac{}{(a,v) \xrightarrow{l} (c,d')} \qquad (2)\ \frac{}{(b,d) \xrightarrow{l} (c,d')} \qquad (3)\ \frac{}{(f(x),v) \xrightarrow{l'} (x,d')}$$

Then, we have $(a, d) \leftrightarrow_{sb} (b, d)$. On the other hand, it does not hold that $(f(a), d) \leftrightarrow_{sb} (f(b), d)$, since $(f(a), d)$ has an l' transition to (a, d'), while $(f(b), d)$ only has an l' transition to (b, d') and these two states are not statebased bisimilar as the first one has an l transition due to deduction rule (1), while the second one does not. Hence, statebased bisimilarity is not a process-congruence (for f).

In deduction rules (1) and (3) of the above example, we have transitions that (potentially) change the data state while keeping the process variable. That is the reason why we fail to have that state-based bisimilarity is a process-congruence.

We remedy this shortcoming by adding more constraints to the format. We define the binding between process variables and data terms and force it to remain consistent in each of the deduction rules.

Definition 4.7 A state (t, u) satisfies the data dependency $x \Rrightarrow u'$, denoted by $(t, u) \models x \Rrightarrow u'$, if and only if $x \in \mathit{vars}(t)$ and $u' = u$.

Example 4.8 Consider once more the transition system specification from Example 4.6. The left-hand-side of the conclusion of deduction rule (3) has a data dependency $(f(x), v) \models x \Rrightarrow v$ and the right-hand-side of the conclusion has a data dependency $(x, d') \models x \Rrightarrow d'$.

Definition 4.9 (Sfsb) A deduction rule (dr) is in *sfsb format* if it is in process-tyft format and satisfies the following data-dependency constraints:

1. If a data dependency on a variable $x \in X_p$ is satisfied in the right-hand-side of the conclusion, the dependency is satisfied in the left-hand-side of the conclusion, that is,

$$\forall_{x \in X_p}\ (t', u') \models x \Rrightarrow u' \Rightarrow (f(x_0, \ldots, x_{n-1}), u) \models x \Rrightarrow u'.$$

2. If a data dependency on a variable $y \in Y_p$ is satisfied in the right-hand-side of the conclusion, the dependency is satisfied in the right-hand-side of a premise, that is,

$$\forall_{y \in Y_p}\ (t', u') \models y \Rrightarrow u' \Rightarrow \exists_{i \in I}\ (y_i, u'_i) \models y \Rrightarrow u'.$$

3. If a data dependency on a variable $x \in X_p$ is satisfied in the left-hand-side of a premise, the dependency is satisfied in the left-hand-side of the conclusion:

$$\forall_{i \in I, x \in X_p}\ (t_i, u_i) \models x \Rrightarrow u_i \Rightarrow (f(x_0, \dots, x_{n-1}), u) \models x \Rrightarrow u_i.$$

4. If a data dependency on a variable $y \in Y_p$ is satisfied in the left-hand-side of a premise, the dependency is satisfied in the right-hand-side of a premise:

$$\forall_{i \in I, y \in Y_p}\ (t_i, u_i) \models y \Rrightarrow u_i \Rightarrow \exists_{j \in I}\ (y_j, u'_j) \models y \Rrightarrow u_i.$$

A transition system specification is in *sfsb format* if and only if all its deduction rules are.

Informally speaking, we foresee a flow of binding between process variables and data terms from the left-hand-side of the conclusion to the left-hand-side of the premises and the right-hand-side of the conclusion and from the right-hand-side of the premises to the left-hand-sides of other premises and finally, to the right-hand-side of the conclusion. For simplicity in proofs, we require the acyclicity of the variable dependency graph, as well. However, this requirement can be removed using the result of [13].

Theorem 4.10 If a transition system specification is in sfsb format, then statebased bisimilarity is a process-congruence for that transition system specification.

Before we prove this theorem, we first define the closure of a relation under statebased congruence and give and prove a lemma that is very useful in the proof of Theorem 4.10.

Definition 4.11 Let $R \subseteq (C(\Sigma_p) \times C(\Sigma_d)) \times (C(\Sigma_p) \times C(\Sigma_d))$. We define the relation $\widehat{R} \subseteq (C(\Sigma_p) \times C(\Sigma_d)) \times (C(\Sigma_p) \times C(\Sigma_d))$ to be the smallest reflexive process-congruence that contains R. Formally, $\widehat{R}$ is defined to be the smallest relation that satisfies:

1. $\widehat{R}$ is reflexive;
2. $R \subseteq \widehat{R}$;
3. $((f(p_0, \dots, p_{n-1}), d), (f(q_0, \dots, q_{n-1}), d)) \in \widehat{R}$ for all n-ary $f \in \Sigma_p$, $d \in C(\Sigma_d)$, and all $p_0, \dots, p_{n-1}, q_0, \dots, q_{n-1} \in C(\Sigma_p)$ such that $((p_i, d), (q_i, d)) \in \widehat{R}$ for all $0 \leq i < n$.

Lemma 4.12 Let $R \subseteq (C(\Sigma_p) \times C(\Sigma_d)) \times (C(\Sigma_p) \times C(\Sigma_d))$, $t \in T(\Sigma_p)$, $d \in C(\Sigma_d)$. For any two substitutions σ and σ' such that $((\sigma(x), d), (\sigma'(x), d)) \in \widehat{R}$ for all $x \in vars(t)$, we have $((\sigma(t), d), (\sigma'(t), d)) \in \widehat{R}$.

Proof. By induction on the structure of process term t. First observe that, as the relation R is on closed state terms only, necessarily the substitutions σ and σ' must be such that application of them to any process term with only variables from $vars(t)$ results in a closed process term. In case t is a variable, say x, we obtain $\sigma(t) = \sigma(x)$ and $\sigma'(t) = \sigma'(x)$. As $x \in vars(t)$, we have $((\sigma(x), d), (\sigma'(x), d)) \in \widehat{R}$ and hence $((\sigma(t), d), (\sigma'(t), d)) \in \widehat{R}$.

In case t is a constant, say c, we obtain $\sigma(t) = \sigma(c) = c = \sigma'(c) = \sigma'(t)$. Then, from the fact that $\widehat{R}$ is reflexive, it follows immediately that $((\sigma(t), d), (\sigma'(t), d)) \in \widehat{R}$.

Finally, consider the case where $t = f(t_0, \dots, t_{n-1})$ for some n-ary function symbol $f \in \Sigma_p$ ($n > 0$) and $t_i \in T(\Sigma_p)$. Then, as $vars(t_i) \subseteq vars(t)$ for all $0 \leq i < n$, we obtain, by the induction hypothesis, $((\sigma(t_i), d), (\sigma'(t_i), d)) \in \widehat{R}$ for all $0 \leq i < n$. Since $\sigma(t) = \sigma(f(t_0, \dots, t_{n-1})) =$

$f(\sigma(t_0), \ldots, \sigma(t_{n-1}))$ and $\sigma'(t) = \sigma'(f(t_0, \ldots, t_{n-1})) = f(\sigma'(t_0), \ldots, \sigma'(t_{n-1}))$, and the relation $\widehat{R}$ is closed under process-congruence, we obtain $((\sigma(t), d), (\sigma'(t), d)) \in \widehat{R}$. ⊠

Proof. (*Theorem 4.10*) It suffices to prove that statebased bisimilarity is a process-congruence for each of the process functions of Σ_p. Let $f \in \Sigma_p$ be an n-ary process function. Let p_i and q_i be closed process terms for $0 \leq i < n$ and let $d \in C(\Sigma_d)$. Suppose that $(p_i, d) \leftrightarrow_{sb} (q_i, d)$ for $0 \leq i < n$. This means that there are statebased bisimulation relations R_i (for $0 \leq i < n$) that witness these statebased bisimilarities. Let R be the union of these relations R_i: $R = \bigcup_{i=0}^{n} R_i$. Obviously R is also a statebased bisimulation relation. We prove that the relation $\widehat{R}$ contains the pair $((f(p_0, \ldots, p_{n-1}), d), (f(q_0, \ldots, q_{n-1}), d))$ and that it is a statebased bisimulation relation. The first part is obvious from the definition of $\widehat{R}$.

So, we only have to prove the following for any $((p, d), (q, d)) \in \widehat{R}$: if for an arbitrary $\rightarrow_r$, l, p' and d', $(p, d) \xrightarrow{l}_r (p', d')$, then there exists a q' such that $(q, d) \xrightarrow{l}_r (q', d')$ and $((p', d'), (q', d'))$ $\in \widehat{R}$ and vice versa for transitions of q. Due to symmetry, it suffices to provide the proofs for the transitions of p only.

We prove this by induction on the depth of the proof of a transition. We do not show the proof for the induction base as it is an instance of the proof of the induction step where there are no premises.

For the induction step, we distinguish three cases based on the structure of the definition of $\widehat{R}$. In case the pair $((p, d), (q, d))$ is contained in the identity relation or the relation R, the proof is obvious (and requires no induction at all). For the remaining case, we find $p = f(p_0, \ldots, p_{n-1})$ and $q = f(q_0, \ldots, q_{n-1})$ for some $p_0, \ldots, p_{n-1}, q_0, \ldots, q_{n-1}$ such that $((p_i, d), (q_i, d)) \in \widehat{R}$ for all $0 \leq i < n$. The last step of the proof of the transition of p is due to the application of a deduction rule of the following form:

$$\frac{\{(t_i, u_i) \xrightarrow{l_i}_{r_i} (y_i, u_i') | i \in I\}}{(f(x_0, \ldots, x_{n-1}), u) \xrightarrow{l}_r (t', u')}.$$

This means that there are substitutions σ and υ such that $\sigma(x_i) = p_i$ for all $0 \leq i < n$, $\upsilon(u) = d$, $\sigma(t') = p'$ and $\upsilon(u') = d'$. Furthermore, for each $i \in I$, there exist a proof of $(\sigma(t_i), \upsilon(u_i)) \xrightarrow{l_i}_{r_i} (\sigma(y_i), \upsilon(u_i'))$ with smaller depth.

Since we have assumed acyclicity of the variable dependency graph, we can define a rank, $rank(x)$, for each variable x, as the maximum length of a backward chain starting from x in the variable dependency graph. The rank of a premise is the rank of its right-hand-side variable. Then, for each $x \in vars(t_i)$ of each premise $(t_i, u_i) \xrightarrow{l_i}_{r_i} (y_i, u_i')$ of the deduction rule, it holds that $rank(x) < rank(y_i)$.

We define the substitution σ' as follows:

$$\sigma'(x) = \begin{cases} q_i & \text{if } x = x_i, \\ \sigma(x) & \text{if } x \notin X_p \cup Y_p. \end{cases}$$

Note that thus far this substitution is not defined for variables from Y_p. We extend the definition while proving, by induction on the rank of a premise r, three essential properties: for all r, for all $i \in I$ such that $rank(y_i) = r$,

1. $((\sigma(t_i), \upsilon(u_i)), (\sigma'(t_i), \upsilon(u_i))) \in \widehat{R}$;
2. $(\sigma'(t_i), \upsilon(u_i)) \xrightarrow{l_i}_{r_i} (\sigma'(y_i), \upsilon(u_i'))$;
3. $((\sigma(y_i), \upsilon(u_i')), (\sigma'(y_i), \upsilon(u_i'))) \in \widehat{R}$.

Again, we do not show the proof of the induction base ($r = 0$) as it is an instance of the proof of the induction step.

For the induction step, suppose $r \geq 1$. Let $(t_i, u_i) \xrightarrow{l_i}_{r_i} (y_i, u_i')$ for some $i \in I$ be a premise of rank r. First, we prove property (1). Let $x \in vars(t_i)$. We distinguish three cases:

1. $x \in X_p$. Then $x = x_i$ for some $0 \leq i < n$. The left-hand-side of the premise has a data dependency $x_i \Rrightarrow u_i$. Hence, by data-dependency constraint 3, this dependency also has to be satisfied in the left-hand-side of the conclusion. Hence, $u_i = u$. We also have that $\sigma(x_i) = p_i$ and $\sigma'(x_i) = q_i$ and we already know that $((p_i, d), (q_i, d)) \in \widehat{R}$ and $\upsilon(u) = d$. Thus, we have $((\sigma(x_i), \upsilon(u_i)), (\sigma'(x_i), \upsilon(u_i))) \in \widehat{R}$, i.e., $((\sigma(x), \upsilon(u_i)), (\sigma'(x), \upsilon(u_i))) \in \widehat{R}$.

2. $x \notin X_p$ and $x \notin Y_p$. As $\sigma(x) = \sigma'(x)$ and the identity relation is contained in $\widehat{R}$ obviously $((\sigma(x), \upsilon(u')), (\sigma'(x), \upsilon(u'))) \in \widehat{R}$.

3. $x \in Y_p$. Then $x = y_j$ for some $j \in I$. The left-hand-side of the premise has a data dependency $y_j \Rrightarrow u_i$. Hence, by data-dependency constraint 4, this dependency also has to be satisfied in the right-hand-side of a premise. Only the premise with index j is a candidate. Hence, $(y_j, u'_j) \models x \Rrightarrow u_i$. Hence $u'_j = u_i$. Obviously, also $rank(y_j) < rank(y_i)$. Thus by the induction hypothesis(property (2)) we have, $((\sigma(y_j), \upsilon(u'_j)), (\sigma'(y_j), \upsilon(u'_j))) \in \widehat{R}$. But, as $x = y_j$, and $u'_j = u_i$, we also have $((\sigma(x), \upsilon(u_i)), (\sigma'(x), \upsilon(u_i))) \in \widehat{R}$.

From the fact that $((\sigma(x), \upsilon(u_i)), (\sigma'(x), \upsilon(u_i))) \in \widehat{R}$ for all $x \in vars(t_i)$, by Lemma 4.12, we have $((\sigma(t_i), \upsilon(u_i)), (\sigma'(t_i), \upsilon(u_i))) \in \widehat{R}$; which proves property (1).

As we have a proof of smaller depth for $(\sigma(t_i), \upsilon(u_i)) \xrightarrow{l_i}_{r_i} (\sigma(y_i), \upsilon(u'_i))$, by the induction hypothesis, we have the existence of a process term q'_i such that $(\sigma'(t_i), \upsilon(u_i)) \xrightarrow{l_i}_{r_i} (q'_i, \upsilon(u'_i))$ and $((\sigma(y_i), \upsilon(u'_i)), (q'_i, \upsilon(u'_i))) \in \widehat{R}$. We choose $\sigma'(y_i)$ to be q'_i. Observe that this proves existence of an appropriate process term $\sigma'(y_i)$. Then, we also have $(\sigma'(t_i), \upsilon(u_i)) \xrightarrow{l_i}_{r_i} (\sigma'(y_i), \upsilon(u'_i))$ $((\sigma(y_i), \upsilon(u'_i)), (\sigma'(y_i), \upsilon(u'_i))) \in \widehat{R}$, which prove properties (2) and (3).

Now, we finish our reasoning using process substitution σ' and the same data substitution and deduction rule. Observe that indeed $\sigma'(f(x_0, \ldots, x_{n-1})) = f(q_0, \ldots, q_{n-1}) = q$. By property (2) we have proven that there are proofs for all premises using the substitutions σ' and υ. Then, according to the same deduction rule and using σ' instead of σ, we have $(\sigma'(f(x_0, \ldots, x_{n-1})), \upsilon(u)) \xrightarrow{l}_{r}$ $(\sigma'(t'), \upsilon(u'))$. Since $\sigma'(f(x_0, \ldots, x_{n-1})) = f(q_0, \ldots, q_{n-1}) = q$, $\upsilon(u) = d$ and $\upsilon(u') = d'$ we obtain $(q, d) \xrightarrow{l}_{r} (\sigma'(t'), d')$.

We only have to show that $((\sigma(t'), d'), (\sigma'(t'), d')) \in \widehat{R}$. By Lemma 4.12, it suffices to show that $((\sigma(x), d'), (\sigma'(x), d')) \in \widehat{R}$ for all $x \in vars(t')$. Three cases can be distinguished:

1. $x \in X_p$. Then $x = x_i$ for some $0 \leq i < n$. The right-hand-side of the conclusion of the deduction rule has a data dependency $(t', u') \models x \Rrightarrow u'$. Hence, by data-dependency constraint 1, this data dependency has to be satisfied in the left-hand-side of the conclusion. Therefore, necessarily $(f(x_0, \ldots, x_{n-1}), u) \models x \Rrightarrow u'$, and thus $u = u'$. Hence $d = \upsilon(u) = \upsilon(u') = d'$. We also have that $\sigma(x_i) = p_i$ and $\sigma'(x_i) = q_i$ and we already know that $((p_i, d), (q_i, d)) \in \widehat{R}$, $x_i = x$, and $d = d'$. Thus, we have $((\sigma(x), d'), (\sigma'(x), d')) \in \widehat{R}$.

2. $x \notin X_p$ and $x \notin Y_p$. As $\sigma(x) = \sigma'(x)$ and the identity relation is contained in $\widehat{R}$ obviously $((\sigma(x), d'), (\sigma'(x), d')) \in \widehat{R}$.

3. $x \in Y_p$. Then $x = y_j$ for some $j \in I$. The right-hand-side of the conclusion has a data dependency $y_j \Rrightarrow u'$. Hence, by data-dependency constraint 2, this dependency also has to be satisfied in the right-hand-side of a premise. Only the premise with index j is a candidate. Hence, $(y_j, u'_j) \models x \Rrightarrow u'$. This can only be the case if $u'_j = u'$. We obtain $d' = \upsilon(u') = \upsilon(u'_j)$. By property (3) we have, $((\sigma(y_j), \upsilon(u'_j)), (\sigma'(y_j), \upsilon(u'_j))) \in \widehat{R}$. But, as $x = y_j$, and $\upsilon(u'_j) = d'$, we also have $((\sigma(x), d'), (\sigma'(x), d')) \in \widehat{R}$.

So this concludes the proof of Theorem 4.10. ⊠

Next, we show that if the proposed format is relaxed in any conceivable way, the congruence result is lost. The first example shows that we cannot remove data-dependency constraint 1.

Example 4.13 Consider a transition system specification, where the process signature consists of process constants a and b, and a unary function symbol f; the data signature consists of data constants d and d'; and the following deduction rules:

$$(1)\quad \frac{}{(a, d') \xrightarrow{l} (a, d')}, \qquad (2)\quad \frac{}{(f(x), d) \xrightarrow{l} (x, d')}.$$

These deduction rules are in process-tyft format. Data-dependency constraint 1 is not satisfied by deduction rule (2) as the data dependency $x \Rrightarrow d'$ that is satisfied in the right-hand-side of the conclusion (i.e., state (x, d')) is not satisfied in the left-hand-side of the conclusion. The other data-dependency constraints are satisfied.

The process-congruence result fails on the above specification. We have $(a, d) \leftrightarroweq_{sb} (b, d)$ (both cannot perform any transitions). However, it does not hold that $(f(a), d) \leftrightarroweq_{sb} (f(b), d)$ since the former state can perform a transition due to deduction rule (2) to (a, d'), while the latter is forced to make the same transition to (b, d') and it clearly does not hold that $(a, d') \leftrightarroweq_{sb} (b, d')$ (see deduction rule (1)).

The next example shows that we cannot remove data-dependency constraint 2.

Example 4.14 Consider a process signature consisting of process constants a and b and a unary process function f; a data signature consisting of data constants d and d'; and a transition system specification with the following deduction rules:

$$(1)\quad \frac{}{(a, v) \xrightarrow{l} (a, v)}, \qquad (2)\quad \frac{}{(b, d) \xrightarrow{l} (b, d)}, \qquad (3)\quad \frac{(x, d) \xrightarrow{l} (y, d)}{(f(x), d) \xrightarrow{l} (y, d')}.$$

These deduction rules are in process-tyft format and all data-dependency constraints, except for constraint 2, which is violated by deduction rule (3). This violation results in breaking the process-congruence result. Two states (a, d) and (b, d) are statebased bisimilar. However, $(f(a), d)$ is not statebased bisimilar to $(f(b), d)$ since the former can perform a transition using deduction rule (3) to (a, d'), while the latter performs a similar transition to (b, d'). These two states are not statebased bisimilar as the former performs an l-transition and the latter deadlocks.

The next example shows that we cannot remove data-dependency constraint 3.

Example 4.15 Consider a transition system specification, where the process signature consists of process constants a and b, and a unary function symbol f; the data signature consists of data constants d and d'; and the following deduction rules:

$$(1)\quad \frac{}{(a, d') \xrightarrow{l} (a, d')}, \qquad (2)\quad \frac{(x, d') \xrightarrow{l} (y, v')}{(f(x), v) \xrightarrow{l} (x, v)}.$$

The above deduction rules are in process-tyft format and satisfy data-dependency constraints 1, 2, and 4. Data-dependency constraint 3 is violated in deduction rule (3) since the data dependency $x \Rrightarrow d'$ that is satisfied in the left-hand-side of the premise is not satisfied in the left-hand-side of the conclusion.

For this transition system specification, statebased bisimilarity is not a process-congruence. We have $(a, d) \leftrightarroweq_{sb} (b, d)$ (both states deadlock). However, $(f(a), d) \leftrightarroweq_{sb} (f(b), d)$ does not hold, since the former state can make a transition due to deduction rule (2) while the latter cannot make any transition.

The next example shows that we cannot remove data-dependency constraint 4.

Example 4.16 Consider a process signature consisting of process constants a and b and a unary process function f; a data signature consisting of data constants d and d'; and a transition system specification with the following deduction rules:

$$(1)\quad \frac{}{(a,v)\xrightarrow{l}(a,v)},\qquad (2)\quad \frac{}{(b,d)\xrightarrow{l}(b,d)},\qquad (3)\quad \frac{(x,d)\xrightarrow{l}(y,d)\quad (y,d')\xrightarrow{l}(y',d')}{(f(x),d)\xrightarrow{l}(y',d')}.$$

The above deduction rules are in process-tyft format and satisfy all data-dependency constraints apart from constraint 4. Deduction rule (3) breaks this constraint in the left-hand-side of its second premise. This also turns out to be harmful for the congruence property, since we have $(a,d) \leftrightarrow_{sb} (b,d)$ but not $(f(a),d) \leftrightarrow_{sb} (f(b),d)$ because deduction rule (3) allows for a transition of the former but not the latter.

4.3 Congruence Format for Initially Stateless Bisimilarity

Later, when comparing congruence conditions for the different notions of bisimilarity, we show that the sfsb format works perfectly well for initially stateless bisimilarity. However, it may turn out to be too restrictive in application. The following example shows a common problem in this regard.

Example 4.17 Consider the following transition system specification (with process constants a and b, unary process function f, and data constants d and d') and the following deduction rules:

$$(1)\quad \frac{}{(a,v)\xrightarrow{l}(a,v)},\qquad (2)\quad \frac{}{(b,d)\xrightarrow{l}(b,d)},\qquad (3)\quad \frac{(x_0,v)\xrightarrow{l}(y,v)}{(f(x_0,x_1),v)\xrightarrow{l}(x_1,d')}.$$

This transition system specification does not satisfy the sfsb format and statebased bisimilarity is not a congruence (since $(a,d) \leftrightarrow_{sb} (b,d)$, but it does not hold that $(f(b,a),d) \leftrightarrow_{sb} (f(b,b),d)$). However, it can be checked that initially stateless bisimilarity is indeed a congruence. The reason is that the change in the data state in deduction rule (3) is harmless since x_1's are now related using all data states including d' (e.g., the above counterexample does not work anymore since it does not hold that $a \leftrightarrow_{isl} b$).

This gives us some clue that for initially stateless bisimilarity, we may weaken the data-dependency constraints.

Definition 4.18 (Sfisl) A deduction rule (dr) is in *sfisl format* if it is in process-tyft format and satisfies the following local (relaxed) data-dependency constraints:

1. If a data dependency on a variable $y \in Y_p$ is satisfied in the right-hand-side of the conclusion, the dependency is satisfied in the right-hand-side of a premise, that is,

$$\forall_{y\in Y_p}\ (t',u') \models y \Rrightarrow u' \Rightarrow \exists_{i\in I}\ (y_i,u'_i) \models y \Rrightarrow u'.$$

2. If a data dependency on a variable $y \in Y_p$ is satisfied in the left-hand-side of a premise, the dependency is satisfied in the right-hand-side of a premise:

$$\forall_{i\in I, y\in Y_p}\ (t_i,u_i) \models y \Rrightarrow u_i \Rightarrow \exists_{j\in I}\ (y_j,u'_j) \models y \Rrightarrow u_i.$$

The data-dependency constraints that were required for variables from the set X_p for congruence of statebased bisimilarity, need not be satisfied for this format anymore. The reason of violating these constraints is that we rely on the fact that certain positions are instantiated by process terms that are related for all possible data. To formalize this concept, first we define positions for which the two constraints are violated and then we check the global consequences of this violation.

Definition 4.19 A variable $x \in X_p$ is called *unresolved* if

$$\begin{array}{l}\exists_{i \in I}\ x \in \mathit{vars}(t_i) \Rightarrow (t_i, u_i) \not\models x \Rrightarrow u \\ \vee \\ x \in \mathit{vars}(t') \Rightarrow (t', u') \not\models x \Rrightarrow u.\end{array}$$

We define X_p^u to be the set of unresolved variables.

For each process function f, we define a set IV_f that contains indices of f for which we need initially stateless bisimilarity because a data-dependency is violated with respect to the variable that occurs in that position in the left-hand-side of the conclusion. The set IV_f contains at least the indices of the unresolved variables of the f-defining deduction rules, but it may contain more indices due to the use of f in other deduction rules in the right-hand-side of the conclusion or the left-hand-side of a premise.

Definition 4.20 For a given transition system specification in process-tyft format, we define, for all $f \in \Sigma_p$, the sets IV_f as the smallest sets that satisfy, for all f-defining deduction rules dr:

1. the indices of unresolved variables (i.e., variables from X_p^u) of dr are in IV_f;
2. for all n-ary process functions $g \in \Sigma_p$: for each occurrence of a process term $g(t_0, \ldots, t_{n-1})$ in the left-hand-side of a premise or the right-hand-side of the conclusion of dr:

$$\forall_{i \in IV_g}\ \forall_{x \in \mathit{vars}(t_i)}\ \exists_{j \in IV_f}\ x = x_j.$$

Note that with the above definition, it is possible that such a set does not exist. In such cases, the global data-dependency constraint given below cannot be established.

Definition 4.21 (Sfisl) A transition system specification is in *sfisl format* if all its deduction rules are in sfisl format and furthermore for each process function f the set IV_f exists.

Informally, this means that a deduction rule may change the data state associated with a process term (arbitrarily) if according to the other rules, the process term is guaranteed to be among the initial argument of the topmost process function (thus, benefitting from the initially stateless bisimilarity assumption). The positions of a process function f benefitting from the initially stateless bisimilarity assumption are thus denoted by IV_f.

Theorem 4.22 If a transition system specification is in sfisl format, then initially stateless bisimilarity is a congruence for that transition system specification.

Before we prove this theorem, we first define the closure of a relation under initially stateless congruence and give and prove a lemma that is very useful in the proof of Theorem 4.22.

Definition 4.23 (Closure With Initially Stateless Congruence) Let $R \subseteq (C(\Sigma_p) \times C(\Sigma_d)) \times (C(\Sigma_p) \times C(\Sigma_d))$. We define the relation $\overline{R} \subseteq (C(\Sigma_p) \times C(\Sigma_d)) \times (C(\Sigma_p) \times C(\Sigma_d))$ to be the smallest relation that satisfies:

1. $\overline{R}$ is reflexive;
2. $R \subseteq \overline{R}$;
3. $((f(p_0, \ldots, p_{n-1}), d), (f(q_0, \ldots, q_{n-1}), d)) \in \overline{R}$ for all n-ary $f \in \Sigma_p$, $d \in C(\Sigma_d)$, and all $p_0, \ldots, p_{n-1}, q_0, \ldots, q_{n-1} \in C(\Sigma_p)$ such that
   (a) $\forall_{i \notin IV_f}\ ((p_i, d), (q_i, d)) \in \overline{R}$;
   (b) $\forall_{i \in IV_f, d' \in C(\Sigma_d)}\ ((p_i, d'), (q_i, d')) \in \overline{R}$.

For a process term t, we define the set $V(t)$ to be the set of variables that appear in the places indicated by the sets IV_f (for all f).

$$\begin{array}{lcl} V(x) & = & \emptyset, \\ V(f(t_0,\ldots,t_{n-1})) & = & \bigcup\limits_{0\le i<n, i\in IV_f} vars(t_i) \cup \bigcup\limits_{0\le i<n, i\notin IV_f} V(t_i). \end{array}$$

Lemma 4.24 Let $R \subseteq (C(\Sigma_p)\times C(\Sigma_d))\times(C(\Sigma_p)\times C(\Sigma_d))$, $t\in T(\Sigma_p)$, $d\in C(\Sigma_d)$. For any two substitutions σ and σ' such that

1. $((\sigma(x),d'),(\sigma'(x),d'))\in \overline{R}$ for all $x\in V(t)$, $d'\in C(\Sigma_d)$, and
2. $((\sigma(x),d),(\sigma'(x),d))\in \overline{R}$ for all $x\in vars(t)\setminus V(t)$;

we have $((\sigma(t),d),(\sigma'(t),d))\in\overline{R}$.

Proof. By induction on the structure of process term t. In case t is a variable, say x, we obtain $\sigma(t)=\sigma(x)$ and $\sigma'(t)=\sigma'(x)$ and $V(t)=V(x)=\emptyset$. As $x\in vars(t)\setminus V(t)$, we have $((\sigma(x),d),(\sigma'(x),d))\in\overline{R}$ and therefore $((\sigma(t),d),(\sigma'(t),d))\in\overline{R}$ as well.

In case t is a constant, say c, we obtain $\sigma(t)=\sigma(c)=c=\sigma'(c)=\sigma'(t)$. Then, from reflexivity of $\overline{R}$, it follows immediately that $((\sigma(t),d),(\sigma'(t),d))\in\overline{R}$.

Finally, consider the case where $t=f(t_0,\ldots,t_{n-1})$ for some n-ary ($n\ge 1$) function symbol $f\in\Sigma_p$ and $t_i\in T(\Sigma_p)$ ($0\le i<n$). If we prove

$$((\sigma(t_i),d),(\sigma'(t_i),d))\in\overline{R} \tag{1}$$

for all $i\notin IV_f$, and

$$((\sigma(t_i),d'),(\sigma'(t_i),d'))\in\overline{R} \tag{2}$$

for all $i\in IV_f$ and $d'\in C(\Sigma_d)$, then $((\sigma(t),d),(\sigma'(t),d))\in\overline{R}$ according to Definition 4.23.

For the first part, assume that $i\notin IV_f$. Then, by definition of V, we have $V(t_i)\subseteq V(t)$. Therefore, by the first assumption on σ and σ' of Lemma 4.24, we have $((\sigma(x),d'),(\sigma'(x),d'))\in\overline{R}$ for all $x\in V(t_i)$ and $d'\in C(\Sigma_d)$. By the first and second assumption and the fact that $vars(t_i)\setminus V(t_i)\subseteq vars(t)$, we have $((\sigma(x),d),(\sigma'(x),d))\in\overline{R}$ for all $x\in vars(t_i)\setminus V(t_i)$. Thus, by the induction hypothesis, we have $((\sigma(t_i),d),(\sigma'(t_i),d))\in\overline{R}$.

For the second part, assume that $i\in IV_f$ and that $d'\in C(\Sigma_d)$. From the definition of V we obtain $vars(t_i)\subseteq V(t)$. Hence, by the first assumption on σ and σ' of Lemma 4.24, we have $((\sigma(x),d'),(\sigma'(x),d'))\in\overline{R}$ for all $x\in vars(t_i)$. Thus, by the induction hypothesis, we have $((\sigma(t_i),d'),(\sigma'(t_i),d'))\in\overline{R}$. ⊠

Proof. (*Theorem 4.22*) It suffices to prove that initially stateless bisimilarity is a congruence for each of the process functions of Σ_p. Let $f\in\Sigma_p$ be an n-ary process function. Let p_i and q_i be closed process terms for $0\le i<n$ and let $d\in C(\Sigma_d)$. Suppose that $p_i \leftrightarroweq_{isl} q_i$ for $0\le i<n$. This means that there are statebased bisimulation relations R_i (for $0\le i<n$) such that $((p_i,d),(q_i,d))\in R_i$ for all $d\in C(\Sigma_d)$. Let R be the union of these relations R_i: $R=\bigcup_{i=0}^{n} R_i$. Obviously R is also a statebased bisimulation relation. We prove that the relation $\overline{R}$ contains the pair $((f(p_0,\ldots,p_{n-1}),d),(f(q_0,\ldots,q_{n-1}),d))$, for all $d\in C(\Sigma_d)$, and that it is a statebased bisimulation relation.

As $((p_i,d),(q_i,d))\in R_i$ and $R_i\subseteq R\subseteq\overline{R}$, for all $0\le i<n$ and all $d\in C(\Sigma_d)$, it follows that $((p_i,d),(q_i,d))\in\overline{R}$, for all $0\le i<n$ and all $d\in C(\Sigma_d)$. Hence, by the definition of $\overline{R}$ obviously also $((f(p_0,\ldots,p_{n-1}),d),(f(q_0,\ldots,q_{n-1}),d))\in\overline{R}$, for $d\in C(\Sigma_d)$.

So, we only have to prove the following for any $((p,d),(q,d))\in\overline{R}$: if for arbitrary $\rightarrow_r$, l, p' and d', $(p,d)\xrightarrow{l}_r(p',d')$, then there exists a q' such that $(q,d)\xrightarrow{l}_r(q',d')$ and $((p',d'),(q',d'))\in\overline{R}$ and vice versa for transitions of q. Due to symmetry, it suffices to provide the proofs for the transitions of p only.

We prove this by induction on the depth of the proof of a transition. We do not show the proof for the induction base as it is an instance of the proof of the induction step where there are no premises.

For the induction step, we distinguish three cases based on the structure of the definition of $\overline{R}$. In case the pair $((p,d),(q,d))$ is contained in $\overline{R}$ due to reflexivity of $\overline{R}$ or due to the requirement that $\overline{R}$ contains R, the proof is obvious (and requires no induction at all). For the remaining case, we find $p = f(p_0,\ldots,p_{n-1})$ and $q = f(q_0,\ldots,q_{n-1})$ for some $p_0,\ldots,p_{n-1},q_0,\ldots,q_{n-1}$ such that

$$\forall_{i \notin IV_f}\ ((p_i,d),(q_i,d)) \in \overline{R}, \tag{3}$$

and

$$\forall_{i \in IV_f, d' \in C(\Sigma_d)}\ ((p_i,d'),(q_i,d')) \in \overline{R}. \tag{4}$$

The last step of the proof of the transition of p is due to the application of a deduction rule of the following form:

$$\frac{\{(t_i,u_i) \xrightarrow{l_i}_{r_i} (y_i,u_i') \mid i \in I\}}{(f(x_0,\ldots,x_{n-1}),u) \xrightarrow{l}_{r} (t',u')}.$$

This means that there are substitutions σ and υ such that $\sigma(x_i) = p_i$ for all $0 \leq i < n$, $\upsilon(u) = d$, $\sigma(t') = p'$ and $\upsilon(u') = d'$. Furthermore, for each $i \in I$, there exist a proof of $(\sigma(t_i),\upsilon(u_i)) \xrightarrow{l_i}_{r_i} (\sigma(y_i),\upsilon(u_i'))$ with smaller depth.

Since we have assumed acyclicity of the variable dependency graph, we can define a rank, $rank(x)$, for each variable x, as the maximum length of a backward chain starting from x in the variable dependency graph. The rank of a premise is the rank of its right-hand-side variable. Then, for each $x \in vars(t_i)$ of each premise $(t_i,u_i) \xrightarrow{l_i}_{r_i} (y_i,u_i')$ of the deduction rule, it holds that $rank(x) < rank(y_i)$.

We define the substitution σ' as follows:

$$\sigma'(x) = \begin{cases} q_i & \text{if } x = x_i, \\ \sigma(x) & \text{if } x \notin X_p \cup Y_p. \end{cases}$$

Note that thus far this substitution is not defined for variables from Y_p. We extend this definition while proving, by induction on the rank of a premise r, three essential properties: for all r, for all $i \in I$ such that $rank(y_i) = r$,

1. $((\sigma(t_i),\upsilon(u_i)),(\sigma'(t_i),\upsilon(u_i))) \in \overline{R}$;
2. $(\sigma'(t_i),\upsilon(u_i)) \xrightarrow{l_i}_{r_i} (\sigma'(y_i),\upsilon(u_i'))$;
3. $((\sigma(y_i),\upsilon(u_i')),(\sigma'(y_i),\upsilon(u_i'))) \in \overline{R}$.

Again, we do not show the proof of the induction base ($r = 0$) as it is an instance of the proof of the induction step.

For the induction step, suppose $r \geq 1$. Let $(t_i,u_i) \xrightarrow{l_i}_{r_i} (y_i,u_i')$ for some $i \in I$ be a premise of rank r. First, we prove property (1). We aim at using Lemma 4.24. Hence we prove

$$\forall_{x \in vars(t_i) \setminus V(t_i)}\ ((\sigma(x),\upsilon(u_i)),(\sigma'(x),\upsilon(u_i))) \in \overline{R} \tag{5}$$

and

$$\forall_{x \in V(t_i), d'' \in C(\Sigma_d)}\ ((\sigma(x),d''),(\sigma'(x),d'')) \in \overline{R} \tag{6}$$

by induction on the structure of term t_i.

1. Suppose that t_i is a variable, say x. Then $vars(t_i) \setminus V(t_i) = \{x\} \setminus \emptyset = \{x\}$. For the first property, we distinguish three cases:

- $x \notin X_p$ and $x \notin Y_p$. Then, we have $\sigma(t_i) = \sigma'(t_i)$. Since $\overline{R}$ is reflexive we obtain $((\sigma(x), \upsilon(u_i)), (\sigma'(x), \upsilon(u_i))) \in \overline{R}$.
- $x \in Y_p$. Then $x = y_j$ for some $j \in I$. The left-hand-side of the premise has a data dependency $y_j \Rrightarrow u_i$. Hence, by local data-dependency constraint 2, this dependency also has to be satisfied in the right-hand-side of a premise. Only the premise with index j is a candidate. Hence, $(y_j, u'_j) \models y_j \Rrightarrow u_j$. So, $u_i = u'_j$. Observe that $rank(y_j) < r$. By the induction hypothesis (property (3)), we then have $((\sigma(y_j), \upsilon(u'_j)), (\sigma'(y_j), \upsilon(u'_j))) \in \overline{R}$. Hence, as $y_j = x$ and $\upsilon(u_j) = \upsilon(u_i)$, we have $((\sigma(x), \upsilon(u_i)), (\sigma'(x), \upsilon(u_i))) \in \overline{R}$.
- $x \in X_p$. Then, $x = x_j$ for some $0 \leq j < n$. We distinguish two cases: (1) If $j \in IV_f$, then we use assumption (4) to obtain $((\sigma(x), \upsilon(u_j)), (\sigma'(x), \upsilon(u_j))) \in \overline{R}$; (2) If $j \notin IV_f$, then by assumption (3) we have $((p_j, d), (q_j, d)) \in \overline{R}$. By definition of IV we obtain that x_j is not an unresolved variable. Hence, by definition of unresolved variables, we have the data-dependency $(t_i, u_i) \models x_j \Rrightarrow u$; and thus $u_i = u$. Hence $d = \upsilon(u) = \upsilon(u_i)$. Thus, we have $((\sigma(x), \upsilon(u_i)), (\sigma'(x), \upsilon(u_i))) \in \overline{R}$.

The second property holds trivially, as $V(t_i) = \emptyset$.

2. Suppose that t_i is a process constant, say c. Then both properties hold trivially, as $vars(t_i) = \emptyset$ and $V(t_i) = \emptyset$.

3. Suppose that $t_i = g(t'_0, \ldots, t'_{n'-1})$ for some n'-ary process function $g \in \Sigma_p$ and $t'_j \in T(\Sigma_p)$ for $0 \leq j < n'$. For the first property observe that $x \in vars(t_i) \setminus V(t_i)$ implies that $x \in vars(t'_j) \setminus V(t'_j)$ for some $j \notin IV_g$. By the induction hypothesis (first property), we then have $((\sigma(x), \upsilon(u_i)), (\sigma'(x), \upsilon(u_i))) \in \overline{R}$.

   For the second property observe that $x \in V(t_i)$ implies (1) $x \in vars(t'_j)$ for some $0 \leq j < n'$ such that $j \in IV_g$; or (2) $x \in V(t'_j)$ for some $j \notin IV_g$. In the first case, the global data-dependency constraint requires that $x = x_k$ for some $0 \leq k < n$ such that $k \in IV_f$. We have $\sigma(x) = p_k$ and $\sigma'(x) = q_k$. Using assumption (4) we then obtain $((\sigma(x), d''), (\sigma'(x), d'')) \in \overline{R}$ for all $d'' \in C(\Sigma_d)$. In the second case, by the induction hypothesis (second property), we have $((\sigma(x), d''), (\sigma'(x), d'')) \in \overline{R}$ for all $d'' \in C(\Sigma_d)$.

From property (1), we have that $((\sigma(t_i), \upsilon(u_i)), (\sigma'(t_i), \upsilon(u_i))) \in \overline{R}$. We also have a proof of smaller depth for $(\sigma(t_i), \upsilon(u_i)) \xrightarrow{l_i}_{r_i} (\sigma(y_i), \upsilon(u'_i))$. Then, by the induction hypothesis, we have the existence of a process term q'_i such that $(\sigma'(t_i), \upsilon(u_i)) \xrightarrow{l_i}_{r_i} (q'_i, \upsilon(u'_i))$ and $((\sigma(y_i), \upsilon(u'_i)), (q'_i, \upsilon(u'_i)))$ $\in \overline{R}$. We choose $\sigma'(y_i)$ to be q'_i. Observe that this proves existence of an appropriate process term $\sigma'(y_i)$. Then, we also have $(\sigma'(t_i), \upsilon(u_i)) \xrightarrow{l_i}_{r_i} (\sigma'(y_i), \upsilon(u'_i))$ $((\sigma(y_i), \upsilon(u'_i)), (\sigma'(y_i), \upsilon(u'_i))) \in \overline{R}$, which prove properties (2) and (3).

Now, we finish our reasoning using process substitution σ' and the same data substitution and deduction rule. Observe that indeed $\sigma'(f(x_0, \ldots, x_{n-1})) = f(q_0, \ldots, q_{n-1}) = q$. By property (2) we have proven that there exist proofs for all premises using the substitutions σ' and υ. Then, according to the same deduction rule and using σ' instead of σ, we have $(\sigma'(f(x_0, \ldots, x_{n-1})), \upsilon(u)) \xrightarrow{l}_r (\sigma'(t'), \upsilon(u'))$. Since $\sigma'(f(x_0, \ldots, x_{n-1})) = f(q_0, \ldots, q_{n-1}) = q$, $\upsilon(u) = d$ and $\upsilon(u') = d'$ we obtain $(q, d) \xrightarrow{l}_r (\sigma'(t'), d')$.

We only have to show that $((\sigma(t'), d'), (\sigma'(t'), d')) \in \overline{R}$. We aim at using Lemma 4.24. Hence we prove

$$\forall_{x \in vars(t') \setminus V(t')} \; ((\sigma(x), d'), (\sigma'(x), d')) \in \overline{R} \tag{7}$$

and

$$\forall_{x \in V(t'), d'' \in C(\Sigma_d)} \; ((\sigma(x), d''), (\sigma'(x), d'')) \in \overline{R} \tag{8}$$

by induction on the structure of term t'.

1. Suppose that t' is a variable, say x. Then $vars(t') \setminus V(t') = \{x\} \setminus \emptyset = \{x\}$. For the first property, we distinguish three cases:

- $x \notin X_p$ and $x \notin Y_p$. Then, we have $\sigma(t_i) = \sigma'(t_i)$. Since $\overline{R}$ is reflexive we obtain $((\sigma(x), d'), (\sigma'(x), d')) \in \overline{R}$.
- $x \in Y_p$. Then $x = y_j$ for some $j \in I$. The right-hand-side of the conclusion has a data dependency $y_j \Rrightarrow u'$. Hence, by local data-dependency constraint 1, this dependency also has to be satisfied in the right-hand-side of a premise. Only the premise with index j is a candidate. Hence, $(y_j, u'_j) \models y_j \Rrightarrow u_j$. So, $u' = u'_j$. By property (3), we have $((\sigma(y_j), v(u'_j)), (\sigma'(y_j), v(u'_j))) \in \overline{R}$. Hence, as $y_j = x$ and $v(u_j) = v(u') = d'$, we have $((\sigma(x), d'), (\sigma'(x), d')) \in \overline{R}$.
- $x \in X_p$. Then, $x = x_j$ for some $0 \leq j < n$. We distinguish two cases: (1) If $j \in IV_f$, then we use assumption (4) to obtain $((\sigma(x), d')), (\sigma'(x), d')) \in \overline{R}$; (2) If $j \notin IV_f$, then by assumption (3) we have $((p_j, d), (q_j, d)) \in \overline{R}$. By definition of IV we obtain that x_j is not an unresolved variable. Hence, by definition of unresolved variables, we have the data-dependency $(t', u') \models x_j \Rrightarrow u$; and thus $u' = u$. Hence $d = v(u) = v(u') = d'$. Thus, we have $((\sigma(x), d'), (\sigma'(x), d')) \in \overline{R}$.

The second property holds trivially, as $V(t') = \emptyset$.

2. Suppose that t' is a process constant, say c. Then both properties hold trivially, as $vars(t') = \emptyset$ and $V(t') = \emptyset$.

3. $t' = g(t'_0, \ldots, t'_{n'-1})$ for some n'-ary process function $g \in \Sigma_p$ and $t'_j \in T(\Sigma_p)$ for $0 \leq j < n'$. For the first property observe that $x \in vars(t') \setminus V(t')$ implies that $x \in vars(t'_j) \setminus V(t'_j)$ for some $j \notin IV_g$. By the induction hypothesis (first property), we then have $((\sigma(x), d'), (\sigma'(x), d')) \in \overline{R}$.

   For the second property observe that $x \in V(t')$ implies (1) $x \in vars(t'_j)$ for some $0 \leq j < n'$ such that $j \in IV_g$; or (2) $x \in V(t'_j)$ for some $j \notin IV_g$. In the first case, the global data-dependency constraint requires that $x = x_k$ for some $0 \leq k < n$ such that $k \in IV_f$. We have $\sigma(x) = p_k$ and $\sigma'(x) = q_k$. Using assumption (4) we then obtain $((\sigma(x), d''), (\sigma'(x), d'')) \in \overline{R}$ for all $d'' \in C(\Sigma_d)$. In the second case, by the induction hypothesis (second property), we have $((\sigma(x), d''), (\sigma'(x), d'')) \in \overline{R}$ for all $d'' \in C(\Sigma_d)$.

So this concludes the proof of Theorem 4.22. ⊠

Example 4.25 Consider the transition system specification of Example 3.4. Obviously the deduction rules are in process-tyft format. They also satisfy the sfisl format as no variables introduced in the right-hand-side of any premise are used in the left-hand-side of a premise or in the right-hand-side of the conclusion. Variable x_1 in deduction rule (5) is unresolved. Hence, we obtain $IV_f \supseteq \{1\}$. As the process function f is not used in any other deduction rule we find $IV_f = \{1\}$. Obviously, for all process constants we find that the set IV is empty: $IV_a = IV_b = IV_c = \emptyset$. Hence, the transition system specification is also in sfisl format. From this we conclude that initially stateless bisimilarity is a congruence.

In the next two examples, we show that none of the two constraints of sfisl can be relaxed in any conceivable way.

Example 4.26 Consider the following transition system specification (with process constants a, b, c, and c', unary process function f, and data constants d and d') and the following deduction rules:

$$(1)\quad \frac{}{(a,d) \xrightarrow{l} (c,d)}, \qquad (2)\quad \frac{}{(b,d) \xrightarrow{l} (c',d)},$$

$$(3)\quad \frac{}{(c,d') \xrightarrow{l} (c,d')}, \qquad (4)\quad \frac{(x,v) \xrightarrow{l} (y,d)}{(f(x),v) \xrightarrow{l} (y,d')}.$$

The deduction rules (1)-(3) are in sfisl format, trivially. Deduction rule (4) does not satisfy local data-dependency constraint 1, since $y \Rrightarrow d'$ is satisfied in the right-hand-side of the conclusion but not in the right-hand-side of the premise. Local data-dependency constraint 2 and the global data-dependency constraint are satisfied (with $IV_f = \emptyset$).

That initially stateless bisimilarity is not a congruence w.r.t. f can be seen as follows: we have that $a \underline{\leftrightarrow}_{isl} b$, but not that $f(a) \underline{\leftrightarrow}_{isl} f(b)$ since $(f(a), d)$ can perform a transition to (c, d') while $(f(b), d)$ is forced to perform the same transition to (c', d') and it does not hold that $(c, d') \underline{\leftrightarrow}_{sb} (c', d')$.

Example 4.27 Consider the transition system specification from Example 4.26 with deduction rule (4) replaced by

$$(4) \quad \frac{(x,v) \xrightarrow{l} (y,v') \quad (y,d') \xrightarrow{l} (y',v'')}{(f(x),v) \xrightarrow{l} (y',v'')}.$$

The deduction rules (1)-(3) are in sfisl format, trivially. Deduction rule (4) satisfies local data-dependency constraint 1 of sfisl, but not local data-dependency constraint 2 as $y \Rrightarrow d'$ is satisfied in the left-hand-side of a premise but not in the right-hand-side of a premise. Also, the global data-dependency constraint is satisfied by this transition system specification.

That initially stateless bisimilarity is not a congruence w.r.t. f can be seen as follows: $a \underline{\leftrightarrow}_{isl} b$ holds, but it does not hold that $f(a) \underline{\leftrightarrow}_{isl} f(b)$ since $(f(a), d)$ is able to perform an l transition (due to rules (4), (3) and (1)) while $(f(b), d)$ deadlocks.

4.4 Comparing Congruence Results

When motivating different notions of bisimilarity, we stated that statebased bisimilarity is considered the weakest (least distinguishing) and least robust notion of bisimilarity with respect to data change. This statement, especially the least robust part, may suggest that if for a transition system specification statebased bisimilairty is a congruence, stateless and initially stateless bisimilarity are trivially congruences, as well. This conjecture can be supported by the standard formats that we gave in this section where the statebased format is the most restrictive and stateless is the most relaxed one. Surprisingly, this conclusion is not entirely true. It turns out that congruence for statabased bisimilarity is indeed stronger than congruence for initially stateless bisimilarity but incomparable to congruence for stateless bisimilarity. A similar incomparability result holds for congruence for initially stateless bisimilarity versus stateless bisimilarity, as well.

The following two examples show that congruence results for statebased bisimilarity and stateless bisimilarity are incomparable. In other words, there are both cases in which one of the two notions is a congruence and the other is not.

Example 4.28 Consider the following transition system specification (with process constants a and b, unary process function f, and data constants d and d') and the following deduction rules:

$$(1) \quad \frac{}{(a,d') \xrightarrow{l} (a,d')}, \qquad (2) \quad \frac{}{(f(a),d) \xrightarrow{l} (a,d')}.$$

In the above transition system specification, the process constants a and b are not stateless bisimilar and hence, congruence of stateless bisimilarity follows trivially. However, we have $(a, d) \underline{\leftrightarrow}_{sb} (b, d)$, but not $(f(a), d) \underline{\leftrightarrow}_{sb} (f(b), d)$.

Example 4.29 Consider the following transition system specification (with process constants a, b, and c, unary process function f, and data constants d and d') and the following deduction rules:

$$(1) \quad \frac{}{(c,d') \xrightarrow{l'} (c,d')}, \qquad (2) \quad \frac{}{(f(a),d) \xrightarrow{l} (b,d)},$$

$$(3) \quad \frac{}{(f(b),d) \xrightarrow{l} (c,d)}, \qquad (4) \quad \frac{}{(f(c),d) \xrightarrow{l} (a,d)}.$$

Statebased bisimilarity is obviously a congruence though the transition system specification does not satisfy the proposed format. Now, consider the processes a and b. These two processes are stateless bisimilar, however, $f(a)$ and $f(b)$ are not stateless bisimilar, since $(f(a), d)$ can make a transition to (b, d), then $(f(b), d)$ is forced to make a transition to (c, d) while b and c are clearly not stateless bisimilar (due to their difference w.r.t. data d').

The following lemma states that if statebased bisimilarity is a congruence, then initially stateless bisimilarity is a congruence as well.

Lemma 4.30 For a transition system specification, if statebased bisimilarity is a congruence, then initially stateless bisimilarity is a congruence, as well.

Proof. Suppose that $p_i \leftrightarroweq_{isl} q_i$ for $0 \leq i < n$. By definition this means that there exist statebased bisimulation relations R_i such that $((p_i, d), (q_i, d)) \in R_i$ for all d. Since statebased bisimilarity is a congruence (by assumption), we have, for each d, the existence of a statebased bisimulation relation S_d such that $((f(p_0, \ldots, p_{n-1}), d), (f(q_0, \ldots, q_{n-1}), d)) \in S_d$. Let $S = \bigcup_d S_d$, and observe that S is a statebased bisimulation relation such that, for all d, $((f(p_0, \ldots, p_{n-1}), d), (f(q_0, \ldots, q_{n-1}), d)) \in S$. This, in turn, means that $f(p_0, \ldots, p_{n-1}) \leftrightarroweq_{isl} f(q_0, \ldots, q_{n-1})$. ⊠

Corollary 4.31 If a transition system specification is in sfsb format, then initially stateless bisimilarity is a congruence for it.

Lemma 4.30 shows that congruence for initially stateless bisimilarity is either stronger than or incomparable to congruence for stateless bisimilarity (since in Example 4.29, we have already shown that there exists a case were statebased bisimilarity, thus initially stateless bisimilarity, is a congruence but stateless bisimilarity is not). To prove the incomparability result, we need a counter example where stateless bisimilarity is a congruence but initially stateless bisimilarity is not (the counter-examples of Example 4.28 do not work in this case). The following example establishes this fact.

Example 4.32 Consider the following transition system specification (with process constants a, b, and c, unary process function f, and data constants d and d') and the following deduction rules:

$$(1)\ \frac{}{(a, d') \xrightarrow{l} (a, d)}, \qquad (2)\ \frac{}{(b, d') \xrightarrow{l} (c, d')}, \qquad (3)\ \frac{}{(c, d) \xrightarrow{l} (c, d)},$$

$$(4)\ \frac{}{(f(a), d) \xrightarrow{l} (c, d)}, \qquad (5)\ \frac{}{(f(b), d') \xrightarrow{l} (c, d')}.$$

According to the above transition system specification, none of the three constants a, b and c are stateless bisimilar, thus congruence of stateless bisimilarity is obvious. However, we have $a \leftrightarroweq_{isl} b$ but not $f(a) \leftrightarroweq_{isl} f(b)$.

So, to conclude, we have proved in this section, that congruence for statebased bisimilarity implies congruence for initially stateless bisimilarity (and not vice versa). However, proving congruence for stateless bisimilarity does not necessarily mean anything for congruence for the two other notions.

4.5 Seasoning the Process-tyft Format

The deduction rules in all three proposed formats are of the following form:

$$\frac{\{(t_i, u_i) \xrightarrow{l_i}_{r_i} (y_i, u_i') | i \in I\}}{(f(x_0, \ldots, x_{n-1}), u) \xrightarrow{l}_{r} (t, u')}.$$

Using this form we cannot go far with proving congruence properties of existing theories since there are many other constructs and patterns that are not present in the above format. In this section, we show how to exploit the format in presence of such constructs. A common type of deduction rules used in transition system specifications is the *tyxt* form which has the following structure:

$$(dr) \quad \frac{\{(t_i, u_i) \xrightarrow{l_i}_{r_i} (y_i, u_i') | i \in I\}}{(x, u) \xrightarrow{l}_{r} (t, u')}.$$

Rules of the above form fit within the *tyft* form if we copy the above rule for all function symbols $f \in \Sigma_p$ with (arbitrary) arity n and substitute all occurrences of x with $f(x_0, \ldots, x_{n-1})$.

Another common phenomenon is the presence of predicates. Predicates of the form $Pred(t, u)$ may be present in the premises or the conclusion of a deduction rule. Predicates can be dealt with in the above formats, as if they are left-hand-side of a transition relation (this can be formally proved by introducing fresh dummy transition relations for each predicate that always have a fresh dummy variable in their right-hand-side [3]).

Regarding negative premises, if we can define a measure on formulas over the signature Σ_p that, for each deduction rule of the transition system specification, does not increase from conclusion to all positive premises and strictly decreases from conclusion to negative premises (i.e., if a stratification for all rules exists) then the congruence results can be used safely. Note that an extension to negative premises requires another definition of what a proof of a transition is (see [14, 25]).

5 Case-Studies and Comparison

In this section, some process languages from literature for which an operational semantics is provided by means of a transition system specification with a data state are considered.

For each of these languages, we establish which of the bisimilarities introduced in this paper, are used (possibly with a different formulation) and whether the deduction rules are in the corresponding format. Also, the other notions of bisimilarity defined in this paper are discussed for these process languages.

5.1 HyPA

In [11], a process algebra is presented for the description of hybrid systems, i.e., systems with both discrete events and continuous change of variables. The process signature of HyPA consists of the following process constants and functions:

- process constants: δ, ϵ, $(a)_{a \in A}$, $(c)_{c \in C}$;
- unary process functions: $(d \gg \_)_{d \in D}$, $(\partial_H(\_))_{H \subseteq A}$;
- binary process functions: $\oplus$, $\odot$, $\blacktriangleright$, $\triangleright$, $\parallel$, $\lfloor\!\lfloor$, and $|$.

The data state consists of mappings from model variables to values, denoted by Val. The data signature is not made explicit.

The transition system specification defines the following predicate and relations:

- a 'termination'-predicate $\checkmark$;
- a family of 'action-transition' relations $\left(\_ \xrightarrow{l} \_\right)_{l \in A \times Val}$;
- a family of 'flow-transition' relations $\left(\_ \overset{\sigma}{\rightsquigarrow} \_\right)_{\sigma \in T \to Val}$.

The deduction rules are given in Table 1 and Table 2.

On HyPA process terms, a notion of bisimilarity is defined that coincides with our definition of stateless bisimilarity.

Table 1: Operational semantics of HyPA

$$\frac{}{\langle \epsilon, \nu \rangle \checkmark}(1) \qquad \frac{}{\langle a, \nu \rangle \xrightarrow{a,\nu} \langle \epsilon, \nu \rangle}(2) \qquad \frac{(\nu, \sigma) \models_{\mathrm{f}} c,\ \ dom(\sigma) = [0, t]}{\langle c, \nu \rangle \overset{\sigma}{\rightsquigarrow} \langle c, \sigma(t) \rangle}(3)$$

$$\frac{(\nu, \nu') \models_{\mathrm{r}} d,\ \langle x, \nu' \rangle \checkmark}{\langle d \gg x, \nu \rangle \checkmark}(4) \qquad \frac{(\nu, \nu') \models_{\mathrm{r}} d,\ \langle x, \nu' \rangle \xrightarrow{l} \langle y, \nu'' \rangle}{\langle d \gg x, \nu \rangle \xrightarrow{l} \langle y, \nu'' \rangle}(5)$$

$$\frac{\langle x_0, \nu \rangle \checkmark}{\begin{array}{l}\langle x_0 \oplus x_1, \nu \rangle \checkmark \\ \langle x_1 \oplus x_0, \nu \rangle \checkmark\end{array}}(6) \qquad \frac{\langle x_0, \nu \rangle \xrightarrow{l} \langle y, \nu' \rangle}{\begin{array}{l}\langle x_0 \oplus x_1, \nu \rangle \xrightarrow{l} \langle y, \nu' \rangle \\ \langle x_1 \oplus x_0, \nu \rangle \xrightarrow{l} \langle y, \nu' \rangle\end{array}}(7) \qquad \frac{\langle x_0, \nu \rangle \checkmark,\ \langle y_0, \nu \rangle \checkmark}{\langle x_0 \odot y_0, \nu \rangle \checkmark}(8)$$

$$\frac{\langle x_0, \nu \rangle \xrightarrow{l} \langle y, \nu' \rangle}{\langle x_0 \odot x_1, \nu \rangle \xrightarrow{l} \langle y \odot x_1, \nu' \rangle}(9) \qquad \frac{\langle x_0, \nu \rangle \checkmark,\ \langle x_1, \nu \rangle \xrightarrow{l} \langle y, \nu' \rangle}{\langle x_0 \odot x_1, \nu \rangle \xrightarrow{l} \langle y, \nu' \rangle}(10)$$

$$\frac{\langle x_0, \nu \rangle \checkmark}{\begin{array}{l}\langle x_0 \blacktriangleright x_1, \nu \rangle \checkmark \\ \langle x_0 \triangleright x_1, \nu \rangle \checkmark\end{array}}(11) \qquad \frac{\langle x_0, \nu \rangle \xrightarrow{l} \langle y, \nu' \rangle}{\begin{array}{l}\langle x_0 \blacktriangleright x_1, \nu \rangle \xrightarrow{l} \langle y \blacktriangleright x_1, \nu' \rangle \\ \langle x_0 \triangleright x_1, \nu \rangle \xrightarrow{l} \langle y \blacktriangleright x_1, \nu' \rangle\end{array}}(12)$$

$$\frac{\langle x_1, \nu \rangle \checkmark}{\langle x_0 \blacktriangleright x_1, \nu \rangle \checkmark}(13) \qquad \frac{\langle x_1, \nu \rangle \xrightarrow{l} \langle y, \nu' \rangle}{\langle x_0 \blacktriangleright x_1, \nu \rangle \xrightarrow{l} \langle y, \nu' \rangle}(14)$$

$$\frac{\langle x, \nu \rangle \xrightarrow{a,\nu'} \langle y, \nu'' \rangle,\ a \notin H}{\langle \partial_H(x), \nu \rangle \xrightarrow{a,\nu'} \langle \partial_H(y), \nu'' \rangle}(20)$$

$$\frac{\langle x, \nu \rangle \overset{\sigma}{\rightsquigarrow} \langle y, \nu' \rangle}{\langle \partial_H(x), \nu \rangle \overset{\sigma}{\rightsquigarrow} \langle \partial_H(y), \nu' \rangle}(21) \qquad \frac{\langle x, \nu \rangle \checkmark}{\langle \partial_H(x), \nu \rangle \checkmark}(22)$$

Table 2: Operational semantics of HyPA, parallel composition

$$\frac{\langle x_0,\nu\rangle\checkmark,\ \langle x_1,\nu\rangle\checkmark}{\begin{array}{c}\langle x_0\parallel x_1,\nu\rangle\checkmark\\ \langle x_0\mid x_1,\nu\rangle\checkmark\end{array}}(15)\qquad \frac{\langle x_0,\nu\rangle\overset{\sigma}{\rightsquigarrow}\langle y_0,\nu'\rangle,\ \langle x_1,\nu\rangle\overset{\sigma}{\rightsquigarrow}\langle y_1,\nu'\rangle}{\begin{array}{c}\langle x_0\parallel x_1,\nu\rangle\overset{\sigma}{\rightsquigarrow}\langle y_0\parallel y_1,\nu'\rangle\\ \langle x_0\mid x_1,\nu\rangle\overset{\sigma}{\rightsquigarrow}\langle y_0\parallel y_1,\nu'\rangle\end{array}}(16)$$

$$\frac{\langle x_0,\nu\rangle\overset{\sigma}{\rightsquigarrow}\langle y,\nu'\rangle,\ \langle x_1,\nu\rangle\checkmark}{\begin{array}{c}\langle x_0\parallel x_1,\nu\rangle\overset{\sigma}{\rightsquigarrow}\langle y,\nu'\rangle\\ \langle x_1\parallel x_0,\nu\rangle\overset{\sigma}{\rightsquigarrow}\langle y,\nu'\rangle\\ \langle x_0\mid x_1,\nu\rangle\overset{\sigma}{\rightsquigarrow}\langle y,\nu'\rangle\\ \langle x_1\mid x_0,\nu\rangle\overset{\sigma}{\rightsquigarrow}\langle y,\nu'\rangle\end{array}}(17)\qquad \frac{\langle x_0,\nu\rangle\xrightarrow{a,\nu'}\langle y,\nu''\rangle}{\begin{array}{c}\langle x_0\parallel x_1,\nu\rangle\xrightarrow{a,\nu'}\langle y\parallel x_1,\nu''\rangle\\ \langle x_1\parallel x_0,\nu\rangle\xrightarrow{a,\nu'}\langle x_1\parallel y,\nu''\rangle\\ \langle x_0\mathbin{\lfloor\!\lfloor} x_1,\nu\rangle\xrightarrow{a,\nu'}\langle y\parallel x_1,\nu''\rangle\end{array}}(18)$$

$$\frac{\langle x_0,\nu\rangle\xrightarrow{a,\nu'}\langle y_0,\nu''\rangle,\ \langle x_1,\nu\rangle\xrightarrow{a',\nu'}\langle y_1,\nu''\rangle,\ a''=a\,\gamma\,a'}{\begin{array}{c}\langle x_0\parallel x_1,\nu\rangle\xrightarrow{a'',\nu'}\langle y_0\parallel y_1,\nu''\rangle\\ \langle x_0\mid x_1,\nu\rangle\xrightarrow{a'',\nu'}\langle y_0\parallel y_1,\nu''\rangle\end{array}}(19)$$

Stateless bisimilarity One can easily observe that all deduction rules of HyPA are in sfsl-format. Hence, stateless bisimilarity is a congruence for all constant and function symbols from the process signature of HyPA.

Statebased bisimilarity With respect to the notion of statebased bisimilarity, as defined in this paper, it can be established that statebased bisimilarity is a process-congruence for the constants of HyPA, the alternative composition operator ($\oplus$), and the encapsulation operator ∂_H (), based on the format of the deduction rules. For the other operators however, this is not the case. Deduction rules (4) (after unseasoning) and (5) violate data-dependency constraint 3. The data dependency $x \Rrightarrow \nu'$ that is satisfied in the left-hand-side of the premise is not satisfied in the left-hand-side of the conclusion.

Deduction rules (9), (12), and (18) for sequential composition ($\odot$), disrupt ($\blacktriangleright$) and left-disrupt ($\triangleright$), and the parallel composition operators ($\parallel$, $\lfloor\!\lfloor$, and $\mid$) all violate data-dependency constraint 1 as the data dependency for variable y in the right-hand-side of the conclusion has no base in the left-hand side of the conclusion.

One might wonder whether this means that our format for statebased bisimilarity is too restrictive in the sense that process-congruence cannot be concluded for many operators. This is not the case, for none of these operators statebased bisimilarity a process-congruence!

Initially stateless bisimilarity In case we consider initially stateless bisimilarity, it turns out that the deduction rules are all in sfisl. Hence, what remains is to check whether the global constraints are satisfied. For this, we need to compute the sets IV_f for each process function f of HyPA. For alternative composition and encapsulation, we obtain $IV_{\oplus} = IV_{\partial_H()} = \emptyset$ as there are no unresolved variables in the deduction rules defining these process functions and there are no process functions used in left-hand-sides of premises or right-hand-sides of conclusions.

For re-initialization, due to the unresolvedness of variable x (at position 0) in deduction rules (4) and (5), and the fact that no process functions are used in left-hand-sides of premises or right-hand-sides of conclusions of re-initialization defining deduction rules, we have $IV_{d\gg} = \{0\}$.

For sequential composition $IV_{\odot} \supseteq \{1\}$ since x_1 is unresolved in deduction rule (9). Also note that in the same deduction rule sequential composition is used in the right-hand-side of the conclusion. The term occurring as argument 1 of this use, x_1, is the index 1 variable from the left-hand-side of the conclusion and hence this occurrence of sequential composition does not add to the set $IV_{\odot}$. As there are no other process functions used in $\odot$-defining deduction rules, we have $IV_{\odot} = \{1\}$. Using a similar reasoning as for sequential composition, we obtain $IV_{\blacktriangleright} = IV_{\triangleright} = \{1\}$.

For the parallel composition operators, based on the unresolvedness of variables in deduction rule (18) we need $IV_{\parallel} \supseteq \{0, 1\}$ and $IV_{⫝̸} \supseteq \{1\}$. All parallel composition operators use parallel composition in the right-hand-side of at least one of their defining deduction rules. This leads to the additional requirement that all variables occurring in the use of parallel composition are from the set X_p. That this is not the case can be seen easily by considering the deduction rules (18) and (19). Hence, it turns out that the sets $IV_{\parallel}$, $IV_{⫝̸}$, and $IV_{|}$ are not defined.

The transition system specification though does not respect the global constraints imposed by sfisl. However, if we restrict to the part of HyPA without parallel composition operators, i.e., sequential HyPA, then we can conclude that initially stateless bisimilarity is a congruence.

The fact that we cannot derive that initially stateless bisimilarity is a congruence w.r.t. the parallel composition operators is not a weakness of our format. Also in this case, initially stateless bisimilarity is not a congruence w.r.t. parallel composition.

5.2 Timed μCRL

In [15], a timed extension of the language μCRL is defined. In this section, we consider a fragment of this language consisting of the following process constants and functions:

- process constants: δ, $(a)_{a \in A}$;
- unary process functions: $\left(\sum_x \_\right)_{x \in V}$, $(\_ {}^{\varsigma} t)_{t \in T}$;
- binary process functions: $+$, $\cdot$, $(\_ \lhd b \rhd \_)_{b \in B}$, $\parallel$.

The process functions that we do not consider here, are either only introduced for axiomatization purposes ($⫝̸$, $|$, $\ll$) or renaming of actions (∂_H, ρ_R, τ_I). The transition system specification defines the following predicates and relations:

- a delay predicate U;
- a family of action-termination predicates $\left(\_ \xrightarrow{a} \checkmark\right)_{a \in A}$;
- a family of action-transition relations $\left(\_ \xrightarrow{a} \_\right)_{a \in A}$;
- a time-transition relation $\_ \xrightarrow{\iota} \_$.

In [15], $U(p, t)$ is written as $U(t, p)$ and $\langle p, t \rangle \xrightarrow{a} \checkmark$ is written as $\langle p, t \rangle \xrightarrow{a} \langle \checkmark, t \rangle$. In this section, we use the notations $U(p, t)$ and $\langle p, t \rangle \xrightarrow{a} \checkmark$. The deduction rules are given in Tables 3 and 4.

The equivalence used in [15] for timed μCRL process terms is timed bisimilarity, which coincides with our notion of initially stateless bisimilarity. The definition of timed bisimilarity in [15] does not require the delay predicate to be transferred between related processes. The notion of initially stateless bisimilarity presented in this paper is based on transferring all predicates and relations used in the transition system specification. This difference is not problematic as it can easily be proved that any two timed bisimilar process terms are also initially stateless bisimilar and vice versa. Congruence of timed bisimilarity is claimed without proof.

Before we discuss the three notions of bisimilarity in more detail we emphasize that deduction rule (22) needs to be unseasoned before the formats can be applied. Generally, deduction rule (22) maps to a collection of deduction rules of the form

$$\frac{U(f(x_0, \ldots, x_{n-1}), t') \quad t < t'}{\langle f(x_0, \ldots, x_{n-1}), t \rangle \xrightarrow{\iota} \langle f(x_0, \ldots, x_{n-1}), t' \rangle} \quad (22f)$$

Table 3: Operational semantics of Timed μCRL

$$\frac{}{\langle a, t \rangle \xrightarrow{a} \checkmark}(1) \qquad \frac{}{U(a, t)}(2) \qquad \frac{}{U(\delta, t)}(3)$$

$$\frac{\langle x_0, t \rangle \xrightarrow{l} \checkmark}{\begin{array}{c}\langle x_0 + x_1, t \rangle \xrightarrow{l} \checkmark \\ \langle x_1 + x_0, t \rangle \xrightarrow{l} \checkmark\end{array}}(4) \qquad \frac{\langle x_0, t \rangle \xrightarrow{l} \langle y, t \rangle}{\begin{array}{c}\langle x_0 + x_1, t \rangle \xrightarrow{l} \langle y, t \rangle \\ \langle x_1 + x_0, t \rangle \xrightarrow{l} \langle y, t \rangle\end{array}}(5) \qquad \frac{U(x_0, t)}{\begin{array}{c}U(x_0 + x_1, t) \\ U(x_1 + x_0, t)\end{array}}(6)$$

$$\frac{\langle x_0, t \rangle \xrightarrow{l} \checkmark}{\langle x_0 \cdot x_1, t \rangle \xrightarrow{l} \langle x_1, t \rangle}(7) \qquad \frac{\langle x_0, t \rangle \xrightarrow{l} \langle y, t \rangle}{\langle x_0 \cdot x_1, t \rangle \xrightarrow{l} \langle y \cdot x_1, t \rangle}(8) \qquad \frac{U(x_0, t)}{U(x_0 \cdot x_1, t)}(9)$$

$$\frac{\langle x_0, t \rangle \xrightarrow{l} \checkmark \quad \models b}{\langle x_0 \lhd b \rhd x_1, t \rangle \xrightarrow{l} \checkmark}(10) \qquad \frac{\langle x_0, t \rangle \xrightarrow{l} \langle y, t \rangle \quad \models b}{\langle x_0 \lhd b \rhd x_1, t \rangle \xrightarrow{l} \langle y, t \rangle}(11) \qquad \frac{U(x_0, t) \quad \models b}{U(x_0 \lhd b \rhd x_1, t)}(12)$$

$$\frac{\langle x_1, t \rangle \xrightarrow{l} \checkmark \quad \not\models b}{\langle x_0 \lhd b \rhd x_1, t \rangle \xrightarrow{l} \checkmark}(13) \qquad \frac{\langle x_1, t \rangle \xrightarrow{l} \langle y, t \rangle \quad \not\models b}{\langle x_0 \lhd b \rhd x_1, t \rangle \xrightarrow{l} \langle y, t \rangle}(14) \qquad \frac{U(x_1, t) \quad \not\models b}{U(x_0 \lhd b \rhd x_1, t)}(15)$$

$$\frac{\langle x[e/v], t \rangle \xrightarrow{l} \checkmark}{\langle \sum_v x, t \rangle \xrightarrow{l} \checkmark}(16) \qquad \frac{\langle x[e/v], t \rangle \xrightarrow{l} \langle y, t \rangle}{\langle \sum_v x, t \rangle \xrightarrow{l} \langle y, t \rangle}(17) \qquad \frac{U(x[e/v], t)}{U(\sum_v x, t)}(18)$$

$$\frac{\langle x, t \rangle \xrightarrow{l} \checkmark}{\langle x{}^{\varsigma}t, t \rangle \xrightarrow{l} \checkmark}(19) \qquad \frac{\langle x, t \rangle \xrightarrow{l} \langle y, t \rangle}{\langle x{}^{\varsigma}t, t \rangle \xrightarrow{l} \langle y, t \rangle}(20) \qquad \frac{U(x, t) \quad t \leq t'}{U(x{}^{\varsigma}t', t)}(21)$$

$$\frac{U(x, t') \quad t < t'}{\langle x, t \rangle \xrightarrow{\iota} \langle x, t' \rangle}(22)$$

Table 4: Operational semantics of Timed μCRL, parallelism

$$\frac{\langle x_0, t\rangle \xrightarrow{l} \checkmark \quad \langle x_1, t\rangle \xrightarrow{l'} \checkmark \quad \gamma(l,l') = l''}{\langle x_0 \parallel x_1, t\rangle \xrightarrow{l''} \checkmark} (23)$$

$$\frac{\langle x_0, t\rangle \xrightarrow{l} \checkmark}{\begin{array}{c}\langle x_0 \parallel x_1, t\rangle \xrightarrow{l} \langle x_1, t\rangle \\ \langle x_1 \parallel x_0, t\rangle \xrightarrow{l} \langle x_1, t\rangle\end{array}} (24) \qquad \frac{\langle x_0, t\rangle \xrightarrow{l} \langle y, t\rangle}{\begin{array}{c}\langle x_0 \parallel x_1, t\rangle \xrightarrow{l} \langle y \parallel x_1, t\rangle \\ \langle x_1 \parallel x_0, t\rangle \xrightarrow{l} \langle x_1 \parallel y, t\rangle\end{array}} (25)$$

$$\frac{\langle x_0, t\rangle \xrightarrow{l} \checkmark \quad \langle x_1, t\rangle \xrightarrow{l'} \langle y, t\rangle \quad \gamma(l,l') = l''}{\langle x_0 \parallel x_1, t\rangle \xrightarrow{l''} \langle y, t\rangle \quad \langle x_1 \parallel x_0, t\rangle \xrightarrow{l''} \langle y, t\rangle} (26)$$

$$\frac{\langle x_0, t\rangle \xrightarrow{l} \langle y_0, t\rangle \quad \langle x_1, t\rangle \xrightarrow{l'} \langle y_1, t\rangle \quad \gamma(l,l') = l''}{\langle x_0 \parallel x_1, t\rangle \xrightarrow{l''} \langle y_0 \parallel y_1, t\rangle} (27)$$

$$\frac{U(x_0, t) \quad U(x_1, t)}{U(x_0 \parallel x_1, t)} (28)$$

one for each n-ary process function f in the signature of timed μCRL. Strictly speaking, also all deduction rules involving predicates need to be transformed, though, as this kind of transformation is less involved, it is omitted.

Stateless bisimilarity Note that although stateless bisimilarity is not considered in [15], from the format of the deduction rules, congruence for this equivalence follows easily.

State-based bisimilarity All deduction rules of timed μCRL are in sfsb-format except for those obtained from deduction rule (22) for process functions with arity at least 1 by the previously described unseasoning. For these, the data dependency $x_0 \Rrightarrow t'$ is satisfied in the right-hand-side of the conclusion, but it is not satisfied in the left-hand-side of the conclusion. Hence, such deduction rules do not satisfy local data-dependency constraint 1 of sfsb. Hence, state-based bisimilarity cannot be concluded to be a congruence for any of the non-nullary process functions of timed μCRL.

Nevertheless, using traditional means one can quite easily establish that state-based bisimilarity is a congruence for some of the operators, for example alternative composition.

Initially stateless bisimilarity As mentioned before, all deduction rules of timed μCRL except for deduction rules derived from deduction rule (22) for non-nullary process functions are in sfsb-format. Thus, with respect to the local constraints of sfisl, only those derived deduction rules have to be considered.

Note that the set of variables Y_p is empty for such a deduction rule. Hence, the local data-dependency constraints of sfisl are satisfied trivially.

For an arbitrary function symbol f with arity n, the set of unresolved variables consists of the indices of all arguments. As a consequence, $IV_f \supseteq \{0, \ldots, n-1\}$. For all process functions, except for sequential and parallel composition, the defining deduction rules do not contain any occurrences of process functions in the left-hand-side of a premise or in the right-hand-side of the conclusion. Hence, for all those process functions, we obtain IV_f is the set of all indices of f.

For sequential composition (deduction rule (8)) and parallel composition (deduction rules (25) and (27)) the occurrences of y, y_0 and y_1 in the use of the process functions do not satisfy the requirement that these should be initial variables ($\in X_p$). Hence, for those process functions, the set IV does not exist. Therefore, congruence of initially stateless bisimilarity w.r.t. those process functions cannot be concluded. For the other process functions, as they are independently defined operationally, congruence can be concluded.

We claim that a reformulation of the operational semantics of timed μCRL without the predicate U along the following lines results in an 'equivalent' transition system specification for which the sfisl format can be applied to obtain congruence:

$$\frac{\langle x_0, t\rangle \xrightarrow{\iota} \langle y, t'\rangle}{\langle x_0 \cdot x_1, t\rangle \xrightarrow{\iota} \langle y \cdot x_1, t\rangle}, \qquad \frac{\langle x_0, t\rangle \xrightarrow{\iota} \langle y_0, t'\rangle \quad \langle x_1, t\rangle \xrightarrow{\iota} \langle y_1, t'\rangle}{\langle x_0 \parallel x_1, t\rangle \xrightarrow{\iota} \langle y_0 \parallel y_1, t'\rangle}.$$

The reason is that the first argument of sequential composition and both arguments of parallel composition are no longer forced to be part of the set IV which avoids the problem with y, y_0 and y_1 not being initial variables. Calculation of the sets $IV_{\cdot}$ and $IV_{\parallel}$ gives: $IV_{\cdot} = \emptyset$ and $IV_{\parallel} = \emptyset$.

5.3 Discrete-event χ

In [8], the process language χ_σ is presented. This language is used for the specification, simulation and validation of discrete-event systems.

The signature of χ_σ consists of the following constant and function symbols:

- process constants: δ, ϵ, skip, $(\Delta_t)_{t\in T}$, $(x := e)_{x\in V, e\in E}$, $(c!e)_{c\in C, e\in E}$, $(c?x)_{c\in C, x\in V}$;
- unary process functions: $(b \to \_)_{b\in B}$, $\_^*$, $(|[s \mid \_]|)_{s\in S}$, $(\partial_H)_{H\subseteq A}$, π, $(\tau_I)_{I\subseteq A}$
- binary process functions: $\Box$, ;, $\parallel$

In the transition system specification of this language both predicates and relations are used. For both types of formulas negative occurrences as a premise occur.

The notion of equivalence that is considered in [8] is stateless bisimilarity. They prove that this equivalence is a congruence for the constant and function symbols of χ_σ by using the so-called relaxed PANTH format [19, 2]. We have serious doubts as to the applicability of this format to the given transition system specification due to the presence of a data state in χ_σ. Nevertheless, stateless bisimilarity is a congruence since all deduction rules of the transition system specification are in sfsl format.

6 Conclusion

In this paper, we investigated the impact of the presence of a data state on notions of bisimilarity and standard congruence formats. To do this, we defined three notions of bisimilarity with data and elaborated on their existing and possible uses. Then, we proposed three standard formats that provide congruence results for these three notions. Furthermore, we briefly pointed out the relationships between these notions and between the corresponding congruences. The proposed formats are applied to several examples from the literature successfully. In this paper, we illustrated the use of our format using a data coordination language, called Linda.

Extending the format for a parameterized notion of bisimilarity (with an explicit interference relation or a symbolic / logical representation of interference possibilities) is another interesting extension which should follow the same line as our relaxation of statebased constraints to initially stateless. Furthermore, we may extend the theory to bisimulation relations which allow for different data states but so far we have seen no practical application of such a bisimilarity notion.

Acknowledgements The authors express their best appreciation to Pieter Cuijpers and Ana Sokolova for their valuable comments on early versions of this paper.

References

[1] Luca Aceto, Wan J. Fokkink, and Chris Verhoef. Structural operational semantics. In Jan A. Bergstra, Alban Ponse, and Scott A. Smolka, editors, *Handbook of Process Algebra, Chapter 3*. Elsevier Science, Dordrecht, The Netherlands, 2001.

[2] Jos C. M. Baeten and Cornelis A. Middelburg. *Process Algebra with Timing*. EATCS Monographs. Springer-Verlag, Berlin, Germany, 2002.

[3] Jos C. M. Baeten and Chris Verhoef. A congruence theorem for structured operational semantics with predicates. In Eike Best, editor, *International Conference on Concurrency Theory (CONCUR'93)*, volume 715 of *Lecture Notes in Computer Science*, pages 477–492. Springer-Verlag, Berlin, Germany, 1993.

[4] Jaco W. de Bakker and Erik P. de Vink. *Control Flow Semantics*. Foundations of Computing Series. The MIT Press, 1996.

[5] Karen L. Bernstein. A congruence theorem for structured operational semantics of higher-order languages. In *IEEE Symposium on Logic In Computer Science (LICS'98)*, pages 153–164. IEEE Computer Society, Los Alamitos, CA, USA, 1998.

[6] Bard Bloom, Sorin Istrail, and Albert R. Meyer. Bisimulation can't be traced. *Journal of the ACM (JACM)*, 42(1):232–268, January 1995.

[7] Bard Bloom and Frits Vaandrager. SOS rules formats for parameterized and state-bearing processes (draft). Unpublished note, available through: `http://www.cs.kun.nl/ita/publications/ papers/fvaan/`.

[8] Victor Bos and Jeroen J.T. Kleijn. Redesign of a systems engineering language — formalisation of χ. *Formal Aspects of Computing*, 15(4), December 2003.

[9] Antonio Brogi and Jean-Marie Jacquet. On the expressiveness of Linda-like concurrent languages. In Ilaria Castellani and Catuscia Palamidessi, editors, *Proceedings of Fifth International Workshop on Expressiveness in Concurrency (EXPRESS'98)*, volume 16 of *Electronic Notes in Theoretical Computer Science*. Elsevier Science, Dordrecht, The Netherlands, 1998.

[10] Michel R. V. Chaudron. *Separating Computation and Coordination in the Design of Parallel and Distributed Programs*. PhD thesis, Department of Computer Science, Rijksuniversiteit Leiden, Leiden, The Netherlands, 1998.

[11] Pieter J.L. Cuijpers and Michel A. Reniers. Hybrid process algebra. Technical Report 03-07, Department of Computer Science, Eindhoven University of Technology, Eindhoven, The Netherlands, 2003.

[12] Robert de Simone. Higher-level synchronizing devices in MEIJE-SCCS. *Theoretical Computer Science (TCS)*, 37:245–267, 1985.

[13] Wan J. Fokkink and Rob J. van Glabbeek. Ntyft/ntyxt rules reduce to ntree rules. *Information and Computation*, 126(1):1–10, 1996.

[14] Jan Friso Groote. Transition system specifications with negative premises. *Theoretical Computer Science*, 118(2):263–299, 1993.

[15] Jan Friso Groote. The syntax and semantics of timed *μCRL*. Technical Report SEN-R9709, CWI - Centrum voor Wiskunde en Informatica, Amsterdam, The Netherlands, June 30, 1997.

[16] Jan Friso Groote and Alban Ponse. Process algebra with guards. combining Hoare logic with process algebra. *Formal Aspects of Computing*, 6(2), 1994.

[17] Jan Friso Groote and Frits Vaandrager. Structured operational semantics and bisimulation as a congruence. *Information and Computation*, 100(2), October 1992.

[18] Jean-Marie Jacquet, Koenraad De Bosschere, and Antonio Brogi. On timed coordination languages. In António Porto and Gruia-Catalin Roman, editors, *Proceedings of Coordination Languages and Models, 4th International Conference, Limassol, Cyprus*, volume 1906 of *Lecture Notes in Computer Science*, pages 81–98. Springer-Verlag, Berlin, Germany, 2000.

[19] Cornelis A. Middelburg. Variable binding operators in transition system specifications. *Journal of Logic and Algebraic Programming*, 47(1):15–45, 2001.

[20] Robin A. Milner. *A Calculus of Communicating Systems*, volume 92 of *Lecture Notes in Computer Science*. Springer-Verlag, 1980.

[21] MohammadReza Mousavi, Twan Basten, Michel Reniers, Michel Chaudron, and Giovanni Russello. Separating functionality, behavior and timing in the design of reactive systems: (GAMMA + coordination) + time. Technical Report 02-09, Department of Computer Science, Eindhoven University of Technology, Eindhoven, The Netherlands, 2002.

[22] David M.R. Park. Concurrency and automata on infinite sequences. In *Proceedings of 5th GI Conference*, volume 104 of *Lecture Notes in Computer Science*, pages 167–183. Springer-Verlag, Berlin, Germany, 1981.

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

[24] Michel A. Reniers, Jan Friso Groote, Mark B. van der Zwaag, and Jos van Wamel. Completeness of timed μCRL. *Fundamenta Informaticae*, 50(3-4):361–402, 2002.

[25] Chris Verhoef. A congruence theorem for structured operational semantics with predicates and negative premises. *Nordic Journal of Computing*, 2(2):274–302, 1995.