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

E-mail: {`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 and lengthy. 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 integral 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 [2, 8, 14, 17] 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 [8, 11, 14, 20].

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. Hence, most of the congruence proofs are done manually [20] 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 an application of the proposed theory on a transition system specification from the domain of coordination languages. Finally, Section 6 concludes the paper and presents possible extensions of the proposed approach. Due to space restrictions we dispense with presenting the proofs of the theorems. Also, for the same reason, we give one application only. Proofs of the theorems and applications from different domains can be found in [21].

2 Related Work

The first standard format for congruence of a transition system specification was proposed by De Simone in [25]. 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* (for Structured Operational Semantics with Guarded recursion). GSOS extends the De Simone format to cover negative premises. Another orthogonal extension of the De Simone format, called *tyft/tyxt*, is proposed in [16] (*tyft/tyxt* is the code summarizing the structure of SOS rule in this format). This format allows for, among others, look-ahead in the premises of a rule. The merits of the two extensions were merged in [13] where negative premises were added to the *tyft/tyxt* format, resulting in the *ntyft/ntyxt* format. Finally, the PATH format [3] (for Predicates And Tyft/tyxt Hybrid format) the PANTH format [26] (for Predicates And Negative Tyft/tyxt Hybrid format) extend *tyft/tyxt* and *ntyft/ntyxt* with predicates, respectively. A deduction rule in PANTH format may thus have predicates, negative predicates, transitions and negative transitions in its premises and a predicate or a transition in its conclusion.

In [18], 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 [18, 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 [19, 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 in the literature.

3 Preliminaries

3.1 Basic Definitions

We assume infinite and disjoint sets of process variables V_p (with typical members $x, y, x_0, y_0 \ldots$) and data variables V_d (with typical member 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 $t, t', t_i, \ldots$. 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 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 ξ 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 [19, 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.

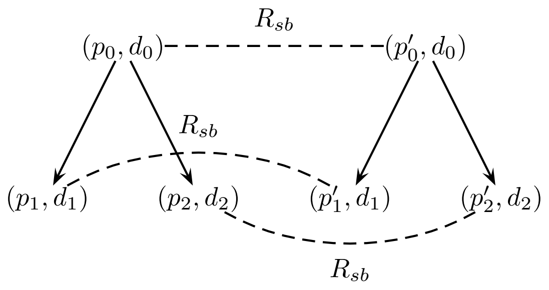

Figure 1. Statebased Bisimilarity

Definition 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 $\forall_{p_0, p_1, d, r}\ ((p_0, d), (p_1, d')) \in R_{sb} \Rightarrow d = d' \wedge$

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 (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.

Statebased bisimilarity is a rather weak notion of bisimilarity for most practical examples. The problem lies in the

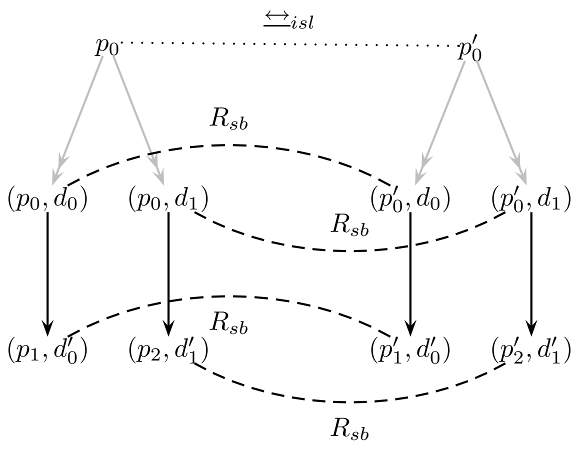

Figure 2. Initially Stateless Bisimilarity

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 [15, 14]. 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.

Definition 4 (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 5 (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.

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

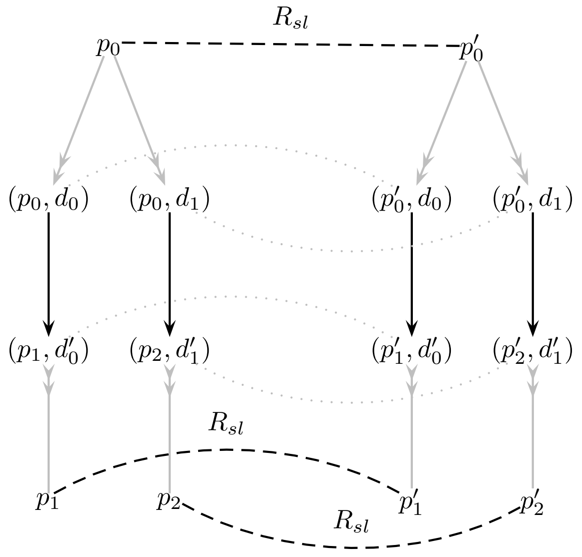

Figure 3. Stateless Bisimilarity

stateless bisimilarity is not preserved in such an environment. This, for instance, happens in open concurrent systems.

Stateless bisimilarity [8, 11, 15, 20], shown in Figure 3, is the solution to this problem and the finest notion of bisimilarity for state-bearing processes 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 6 (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. $\forall_{d_0,l_0,p'_0,d'_0}\quad (p_0, d_0) \xrightarrow{l_0}_r (p'_0, d'_0) \quad \Rightarrow$
   $\exists_{p'_1}\ (p_1, d_0) \xrightarrow{l_0}_r (p'_1, d'_0) \wedge (p'_0, p'_1) \in R_{sl}$;

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

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 $(p, q) \in R_{sl}$.

None 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 non-deterministic 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 non-deterministic sequential systems preserving stateless bisimilarity and then using initially stateless bisimilarity in that setting [15]. 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 [15, 10, 11]. Our congruence format for statebased bisimilarity can easily be adapted to the parameterized notion of 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 [14] 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 [26] 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 7 (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 1 If a transition system specification is in process-tyft format, then stateless bisimilarity is a congruence for that 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 1 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:

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

$$(3) \quad \frac{}{(f(x), v) \xrightarrow{l} (x, d')}$$

Then, we have: $(a, d) \leftrightarrows_{sb} (b, d)$. On the other hand, however, it does not hold that $(f(a), d) \leftrightarrows_{sb} (f(b), d)$, 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 8 A state (t, u) satisfies the data dependency $x \Rrightarrow u'$, denoted by $(t, u) \models x \Rrightarrow u'$ (pronounced as x is bound to u in state (t, u)), if and only if $x \in \mathit{vars}(t)$ and $u' = u$.

Example 2 Consider once more the transition system specification from Example 1. 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 $(x, d') \models x \Rrightarrow d'$. This means that we rely on initially stateless bisimilarity of arguments appearing at this index.

Definition 9 (Sfsb) A deduction rule (dr) is in *sfsb format* (for Standard Format for StateBased bisimulation) 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,

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

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,

$$\begin{array}{ll}\forall_{y \in Y_p} & (t', u') \models y \Rrightarrow u' \Rightarrow \\ & \exists_{i \in I}\ (y_i, u_i') \models y \Rrightarrow u'.\end{array}$$

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:

$$\begin{array}{ll}\forall_{i \in I, x \in X_p} & (t_i, u_i) \models x \Rrightarrow u_i \Rightarrow \\ & (f(x_0, \ldots, x_{n-1}), u) \models x \Rrightarrow u_i.\end{array}$$

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:

$$\begin{array}{ll}\forall_{i \in I, y \in Y_p} & (t_i, u_i) \models y \Rrightarrow u_i \Rightarrow \\ & \exists_{j \in I}\ (y_j, u_j') \models y \Rrightarrow u_i.\end{array}$$

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 [12].

Theorem 2 If a transition system specification is in sfsb format, then statebased bisimilarity is a process-congruence for that transition system specification.

In [21], we have shown that if the proposed format is relaxed in any conceivable way, the congruence result is lost.

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 3 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:

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

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

This transition system specification does not satisfy the sfsb format and statebased bisimilarity is not a congruence (since $(a, d) \underline{\leftrightarrow}_{sb} (b, d)$, but it does not hold that $(f(b, a), d) \underline{\leftrightarrow}_{sb} (f(b, b), d)$). 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 $a \underline{\leftrightarrow}_{isl}\ b$).

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

Definition 10 (Sfisl) A deduction rule (dr) is in *sfisl format* (for Standard Format for Initially StateLess bisimulation) 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,

$$\begin{array}{ll}\forall_{y \in Y_p} & (t', u') \models y \Rrightarrow u' \Rightarrow \\ & \exists_{i \in I}\ (y_i, u_i') \models y \Rrightarrow u'.\end{array}$$

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:

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

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

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

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 12 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.

Example 4 Consider the transition system specification of Example 3, in deduction rule (3), variable x_1 is unresolved, and thus $2 \in IV_f$.

Definition 13 (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 3 If a transition system specification is in sfisl format, then initially stateless bisimilarity is a congruence for that transition system specification.

In [21], we have shown that none of the two constraints of sfisl can be relaxed in any conceivable way.

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 5 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 $(a, d) \leftrightarroweq_{sb} (b, d)$, but not $(f(a), d) \leftrightarroweq_{sb} (f(b), d)$.

Example 6 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 1 For a transition system specification, if statebased bisimilarity is a congruence, then initially stateless bisimilarity is a congruence, as well.

Corollary 1 If a transition system specification is in sfsb format, then initially stateless bisimilarity is a congruence for it.

Lemma 1 shows that congruence for initially stateless bisimilarity is either stronger than or incomparable to congruence for stateless bisimilarity (since in Example 6, 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 5 do not work in this case). The following example establishes this fact.

Example 7 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:

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

$$(3)\quad \frac{}{(c,d) \xrightarrow{l} (c,d)},$$

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

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 $a \leftrightarrow_{isl} b$ but not $f(a) \leftrightarrow_{isl} f(b)$.

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:

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

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 [13, 26]).

5 Case study: Operational semantics of Linda

In [9], the operational semantics of Linda is given using a combination of SOS rules and a structural congruence. As this kind of transition system specification is not purely in the format used in this paper, we have transformed it in such a way that it fits the format (by extending the language with a process constant ϵ). A formulation of this semantics without the constant ϵ is also possible, but the resulting transition system specification is much larger. In this section, we apply the proposed formats on the extended language. Process constants (atomic process terms) in this language are ϵ (for terminating process), $ask(t)$ and $nask(t)$ (for checking existence and absence of tuple t in the shared data space, respectively), $tell(t)$ (for adding tuple t to the space) and $get(t)$ (for taking tuple t from the space). Process composition operators in this language include nondeterministic choice ($+$), sequential composition (;) and parallel composition ($||$). Operational state of a Linda program is denoted by (p, ς) where p is a process term in the above syntax and ς is a set modeling the shared data space.

$$\frac{}{(\epsilon, \varsigma)\downarrow} \qquad \frac{}{(ask(t), \varsigma \cup \{t\}) \rightarrow (\epsilon, \varsigma \cup \{t\})}$$

$$\frac{}{(tell(t), \varsigma) \rightarrow (\epsilon, \varsigma \cup \{t\})} \qquad \frac{}{(get(t), \varsigma \cup \{t\}) \rightarrow (\epsilon, \varsigma)}$$

$$\frac{}{(nask(t), \varsigma) \rightarrow (\epsilon, \varsigma)}[t \notin \varsigma]$$

$$\frac{(x_0, \varsigma)\downarrow}{(x_0 + x_1, \varsigma)\downarrow} \qquad \frac{(x_1, \varsigma)\downarrow}{(x_0 + x_1, \varsigma)\downarrow}$$

$$\frac{(x_0, \varsigma) \rightarrow (y, \varsigma')}{(x_0 + x_1, \varsigma) \rightarrow (y, \varsigma')} \qquad \frac{(x_1, \varsigma) \rightarrow (y, \varsigma')}{(x_0 + x_1, \varsigma) \rightarrow (y, \varsigma')}$$

$$\frac{(x_0, \varsigma) \rightarrow (y, \varsigma')}{(x_0 \ ; x_1, \varsigma) \rightarrow (y \ ; x_1, \varsigma')} \qquad \frac{(x_0, \varsigma)\downarrow \ \ (x_1, \varsigma) \rightarrow (y, \varsigma')}{(x_0 \ ; x_1, \varsigma) \rightarrow (y, \varsigma')}$$

$$\frac{(x_0, \varsigma) \rightarrow (y, \varsigma')}{(x_0||x_1, \varsigma) \rightarrow (y||x_1, \varsigma')} \qquad \frac{(x_1, \varsigma) \rightarrow (y, \varsigma')}{(x_0||x_1, \varsigma) \rightarrow (x_0||y, \varsigma')}$$

$$\frac{(x_0, \varsigma)\downarrow \ \ (x_1, \varsigma)\downarrow}{(x_0 \ ; x_1, \varsigma)\downarrow} \qquad \frac{(x_0, \varsigma)\downarrow \ \ (x_1, \varsigma)\downarrow}{(x_0||x_1, \varsigma)\downarrow}$$

Obviously these deduction rules are all in process-tyft format (with appropriate seasoning for termination predicate $\downarrow$). As a consequence, stateless bisimilarity is a congruence. Initially stateless bisimilarity is a congruence for all operators except parallel composition. Note that $IV_+ = \emptyset$ and $IV_; = \{1\}$. Thus, initially stateless bisimilarity is a congruence for the sequential part of Linda.

Because congruence of initially stateless bisimilarity w.r.t. parallel composition cannot be concluded using our format, we may wonder whether this result must have been expected. In the following example, we show that this is indeed the case and the indications given by our format are true (i.e., initially stateless bisimilarity is not a congruence for the language with parallel composition operator).

Example 8 Consider the processes $p = ask(1) \ ; (nask(1) \ ; ask(2))$ and $q = ask(1) \ ; nask(1)$. According to the above transition system specification, it holds that $p \underline{\leftrightarrow}_{isl} q$ (in both processes, using an arbitrary common initial state, either $ask(1)$ executes followed by deadlock or both deadlock immediately). However, if we compose each of the two processes in parallel with the process $r = get(1)$, then the two processes may behave differently for some data states. For example, consider the data state $\{1, 2\}$. For this data state, one execution path of $(p \mid\mid r, \{1, 2\})$ is: first executing $ask(1)$ from p successfully, then $get(1)$ from r (thus, resulting in data state $\{2\}$), and executing $nask(1)$ followed by $ask(2)$ successfully. However, all possible executions of $(q \mid\mid r, \{1, 2\})$ can never make four consecutive transitions before termination. Thus, we conclude that initially stateless bisimilarity is not a congruence with respect to parallel composition.

In [21], the formats developed in this paper have also been applied successfully to transition system specifications from the domains of real-time and hybrid systems.

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. Investigating the possibility of applying the same techniques for congrence with respect to weaker notions of bisimulation (e.g., branching bisimulation) is another interesting direction for our future research.

Acknowledgements The authors would like to express their appreciation to Pieter Cuijpers and Ana Sokolova for their comments on early versions of this documents. Valuable comments of the anonymous referees are also acknowledged.

References

[1] L. Aceto, W. J. Fokkink, and C. Verhoef. Structural operational semantics. In J. A. Bergstra, A. Ponse, and S. A. Smolka, editors, *Handbook of Process Algebra, Chapter 3*. Elsevier Science, Dordrecht, The Netherlands, 2001.

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

[3] J. C. M. Baeten and C. Verhoef. A congruence theorem for structured operational semantics with predicates. In E. 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] J. W. d. Bakker and E. P. d. Vink. *Control Flow Semantics*. Foundations of Computing Series. The MIT Press, 1996.

[5] K. 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] B. Bloom, S. Istrail, and A. R. Meyer. Bisimulation can't be traced. *Journal of the ACM (JACM)*, 42:232–268, Jan. 1995.

[7] B. Bloom and F. 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] V. Bos and J. J. Kleijn. Redesign of a systems engineering language — formalisation of χ. *Formal Aspects of Computing*, 15(4), Dec. 2003.

[9] A. Brogi and J.-M. Jacquet. On the expressiveness of linda-like concurrent languages. In I. Castellani and C. 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] M. 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] P. J. Cuijpers and M. A. Reniers. Hybrid process algebra. Technical Report 03-07, Department of Computer Science, Eindhoven University of Technology, Eindhoven, The Netherlands, 2003.

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

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

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

[15] J. F. Groote and A. Ponse. Process algebra with guards combining hoare logic with process algebra. Technical Report CS-R9069, Amsterdam, The Netherlands, 1990.

[16] J. F. Groote and F. Vaandrager. Structured operational semantics and bisimulation as a congruence. *Information and Computation*, 100, Oct. 1992.

[17] J.-M. Jacquet, K. De Bosschere, and A. Brogi. On timed coordination languages. In A. Porto and G.-C. 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.

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

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

[20] M. Mousavi, T. Basten, M. Reniers, M. Chaudron, and G. 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.

[21] M. Mousavi, M. Reniers, and J. F. Groote. Congruence for SOS with data. Technical Report 04-05, Department of Computer Science, Eindhoven University of Technology, 2004.

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

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

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

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

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