Available online at www.sciencedirect.com
SCIENCE DIRECT®
Information and Computation 200 (2005) 107–147
Information and Computation
www.elsevier.com/locate/ic

Notions of bisimulation and congruence formats for SOS with data☆

Mohammad Reza Mousavi *, Michel A. Reniers, Jan Friso Groote

Department of Computer Science, Eindhoven University of Technology, P.O. Box 513, NL-5600 MB Eindhoven, The Netherlands

Received 06 August 2004; revised 23 November 2004
Available online 17 May 2005

Abstract

While studying the specification of the operational semantics of different programming languages and formalisms, one can observe the following three facts. First, Plotkin's style of Structural Operational Semantics 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. Third, 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 article, 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.

Keywords: Formal semantics; Structural operational sematics; Bisimulation; Congruence; SOS formats

☆ A preliminary and summarized version of this article appeared as [28].

* Corresponding author.
E-mail address: m.r.mousavi@tue.nl (M.R. Mousavi).

0890-5401/$ - see front matter © 2005 Elsevier Inc. All rights reserved.
doi:10.1016/j.ic.2005.03.002

1. Introduction

Structural Operational Semantics (called SOS for short) [30–32] has been very popular in defining operational semantics for different formalisms and programming languages. Congruenceproperties of notions of (bi-)simulation have been investigated for 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 are usually standard but very tedious and lengthy [18,36]. This has resulted in a line of research for defining standard syntactical formats for different types of SOS 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 [30]). 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 [16,11], real-time languages [4,10,21,24] and hybrid languages [14,6] 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 [10,21,14,27].

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 [27] or are just neglected by making a reference to a standard format that does not cover the data state [10]. The proposal that comes closest [9] is unfinished and encodes rules for state-bearing processes into rules without a state, for which a format is given.

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

The rest of this article is structured as follows. In Section 2, we review the related work in the area of congruence formats for SOS. 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 article 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 transition system specifications from the literature in the domains of coordination languages, real-time process algebra, and hybrid process algebra. Finally, Section 6 concludes the article and presents possible extensions of the proposed approach.

2. Related work

The first standard format for congruence of a transition system specification was proposed by De Simone in [17]. The De Simone format allows for deduction rules of the following form:

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

where x_i and y_i are distinct 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 a process term that does not have repeated occurrences of any variable (so-called *architectural* term, disallowing copying of variables), 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 rest of this article.)

Bloom et al. [8], in their study of the relationship between bisimilarity and completed-trace congruence, define an extension of the De Simone format, called *GSOS* (for Structural Operational Semantics with Guarded recursion), 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:

$$\frac{\{x_i \xrightarrow{l_{ij}} y_{ij} \mid i \in I, j \leqslant m_i\} \cup \{x_j \xrightarrow{l_{jk}} \not\rightarrow \mid j \in J, k \leqslant n_j\}}{f(x_0,\ldots,x_{n-1}) \xrightarrow{l} t}.$$

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_j's and n_k's are natural numbers (to set an upper bound on the number of premises) and t is a term with variables only among x_k's ($0 \leqslant k < n$) and y_{ij}'s ($i \in I$ and $j \leqslant m_i$).

Another orthogonal extension of the De Simone format, called *tyft/tyxt*, which allows for look-ahead in the premises of a rule, is proposed in [23] (*tyft/tyxt* is the code summarizing the structure of SOS rule in this format). This format allows for arbitrary terms in the left-hand sides (called *sources*) of the premises and in the right-hand side (called *target*) of the conclusion but requires the source of the conclusion to be a variable or a function applied to variables only, the targets of premises to be variables, and all these variables to be distinct. In other words, it allows for deduction rules of the following forms:

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

Again, all notations 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 [23], is the acyclicity of the *variable dependency graph* for each deduction rule. 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 source and the latter in the target of a premise. Later, in [19] 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 [20] 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 $t_i \xrightarrow{l_i} \not\rightarrow$ can appear as well, provided that the transition system specifi-

cation 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 [5] (for Predicates And Tyft/tyxt Hybrid format) and the PANTH format [39] (for Predicates And Negative Tyft/tyxt Hybrid format) 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 [25], 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 [2].

In an unpublished note [9], the issue of state-bearing and parameterized processes is treated and a congruence format is proposed (called Super-SOS). The approach of [9] 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 [9] (for processes parameterized on data-state) seems to be what we call stateless bisimilarity in this article. However, congruence proofs for these formats are not provided. As we do not consider parameterized processes in our setting, their formats may not be directly comparable to ours. For example, some of the dataflow conditions proposed in [9] are not necessary in our setting for stateless bisimilarity. Furthermore, the formats in [9] do not induce congruence with respect to the other notions of bisimilarity that we discuss in this article.

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 article, 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 [25,33] or artificially by transforming a multi-sorted state to a single-sorted one, e.g., in [9]). 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 one to define 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 terms, but also data terms. This allows both for more expressiveness 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 [7], 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 [7] is not applicable.

In [15], a preliminary version of [14], 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 [26,29] 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} (typical members: $x, y, x_i, y_i, \ldots$) and data variables V_{d} (typical member: v). A process signature Σ_{p} is a set of function symbols (typical members: f) with their fixed arity. Functions with zero arity are called constants (typical members: a, b, c). A process term $t \in T(\Sigma_{\mathrm{p}})$ is defined inductively as follows: a variable $x \in V_{\mathrm{p}}$ is a process term, if $t_0, \ldots, t_{n-1}$ are process terms then for all $f \in \Sigma_{\mathrm{p}}$ 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 $u \in T(\Sigma_{\mathrm{d}})$ 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 $C(\Sigma_{\mathrm{p}})$ and $C(\Sigma_{\mathrm{d}})$ in each of these contexts are defined as expected (closed process terms are typically denoted by $p, q, p', q', p_i, q_i, p'_i, q'_i \ldots$).

A process substitution (σ or σ') replaces a process variable in an open process term with another (possibly open) process term. A data substitution (ξ) replaces a data variable in an open data term with another (possibly open) data 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 $(\Sigma_{\mathrm{p}}, \Sigma_{\mathrm{d}}, L, D(Rel))$ 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 $(s, s') \in T(\Sigma_{\mathrm{p}}) \times T(\Sigma_{\mathrm{d}})$ 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 formulae and c is a formula. The formula c is called the conclusion and the formulae from H are called premises.

Notions of open and closed and the concept of substitution are lifted to formulae 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 $\dfrac{H}{c}$.

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

- the root node is labelled by ϕ, and
- if ψ is the label of a node and $\{\psi_i \mid i \in I\}$ is the set of labels of the nodes directly above this node, then there are a deduction rule $\dfrac{\{\chi_i \mid i \in I\}}{\chi}$, 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.

The transition relation associated with a transition system specification is the smallest relation containing all provable closed formulae.

Note that in this paper, we only consider transition relations and notions of bisimulation on closed terms. Techniques developed in [34] can be used to define these concepts on open terms.

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 [26,29] to the extended state. In such a case processes are compared regardless of the data they have. Our survey of the literature has revealed that such a notion of strong bisimilarity is not used at all. In this article, we restrict ourselves to comparing processes with respect to the same data state. In this way, we get to what we call a *state-based bisimilarity*, depicted in Fig. 1.

Definition 2 (*state-based bisimilarity*)**.** A relation $R_{\mathrm{sb}} \subseteq (C(\Sigma_{\mathrm{p}}) \times C(\Sigma_{\mathrm{d}})) \times (C(\Sigma_{\mathrm{p}}) \times C(\Sigma_{\mathrm{d}}))$ is a *state-based bisimulation* relation if and only if $\forall_{p,q,d_0,d_1,r,l}\ ((p,d_0),(q,d_1)) \in R_{\mathrm{sb}} \Rightarrow d_0 = d_1 \wedge$

(1) $\forall_{p',d'}(p,d_0) \xrightarrow{l}_r (p',d') \Rightarrow \exists_{q'}(q,d_0) \xrightarrow{l}_r (q',d') \wedge ((p',d'),(q',d')) \in R_{\mathrm{sb}}$;
(2) $\forall_{q',d'}(q,d_1) \xrightarrow{l}_r (q',d') \Rightarrow \exists_{p'}(p,d_1) \xrightarrow{l}_r (p',d') \wedge ((p',d'),(q',d')) \in R_{\mathrm{sb}}$.

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

Definition 3 (*process-congruence*)**.** For $\sim \subseteq (C(\Sigma_{\mathrm{p}}) \times C(\Sigma_{\mathrm{d}})) \times (C(\Sigma_{\mathrm{p}}) \times C(\Sigma_{\mathrm{d}}))$, $\sim$ is called a *process-congruence* w.r.t. an n-ary process function $f \in \Sigma_{\mathrm{p}}$ if and only if for all $p_i, q_i \in C(\Sigma_{\mathrm{p}})$ $(0 \leqslant i < n)$, for all $d \in C(\Sigma_{\mathrm{d}})$, if $(p_i,d) \sim (q_i,d)$ (for all $0 \leqslant i < n$), 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.

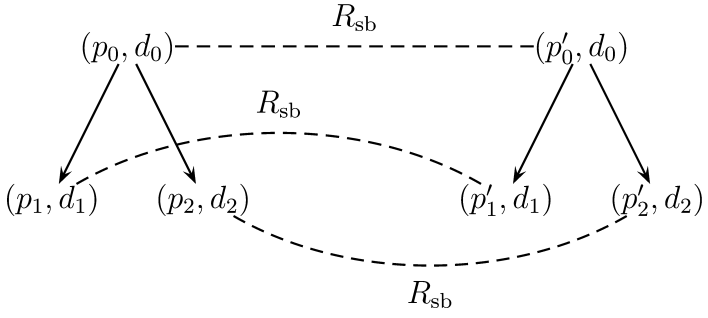


Fig. 1. State-based bisimilarity.

Example 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 state-based bisimilarities hold:

$$(a,d) \underline{\leftrightarrow}_{\mathrm{sb}} (b,d), \qquad (b,d) \underline{\leftrightarrow}_{\mathrm{sb}} (c,d).$$

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

State-based 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 state-based bisimilarity is not useful.

This problem leads to the introduction of a new notion of bisimilarity which takes all possible initial states into account [22,21]. We call this notion *initially stateless bisimilarity* (see Fig. 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.

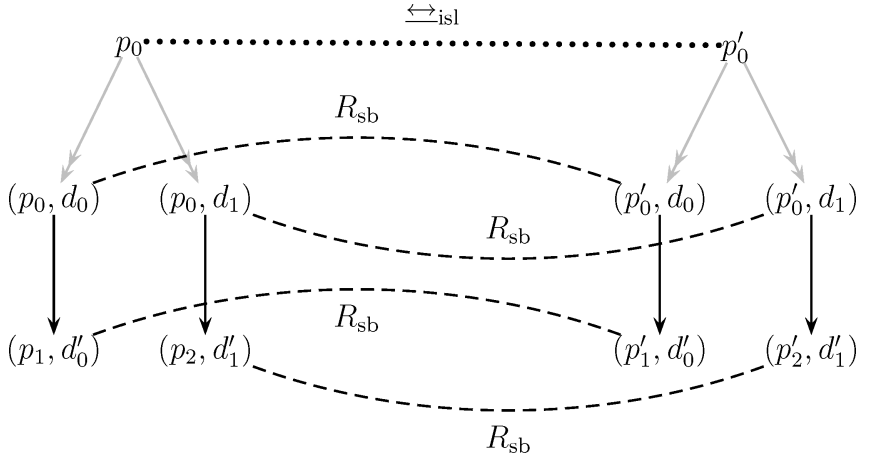


Fig. 2. Initially stateless bisimilarity.

Definition 5 (*initially stateless bisimilarity*)**.** Two closed process terms p and q are *initially stateless bisimilar*, denoted by $p \underline{\leftrightarrow}_{\mathrm{isl}} q$, if and only if there exists a state-based bisimulation relation R_{sb} such that $((p,d),(q,d)) \in R_{\mathrm{sb}}$ for all $d \in C(\Sigma_{\mathrm{d}})$.

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

Definition 6 (*congruence*)**.** For arbitrary $\sim \subseteq C(\Sigma_{\mathrm{p}}) \times C(\Sigma_{\mathrm{p}})$, $\sim$ is called a *congruence* w.r.t. an n-ary process function $f \in \Sigma_{\mathrm{p}}$ if and only if for all $p_i, q_i \in C(\Sigma_{\mathrm{p}})$ $(0 \leqslant i < n)$, if $p_i \sim q_i$ (for all $0 \leqslant i < n$), 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 7. Consider the transition system specification of Example 4. The following initially stateless bisimilarities hold $b \underline{\leftrightarrow}_{\mathrm{isl}} c$ and $f(b,c) \underline{\leftrightarrow}_{\mathrm{isl}} f(c,c)$ but the following initially stateless bisimilarities do not hold $a \underline{\leftrightarrow}_{\mathrm{isl}} b$ and $f(c,a) \underline{\leftrightarrow}_{\mathrm{isl}} f(c,b)$. We observe that the previous problem of congruence does not exist anymore for initially stateless bisimilarity. Later on, in Example 37, 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 [10,14,22,27], shown in Fig. 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

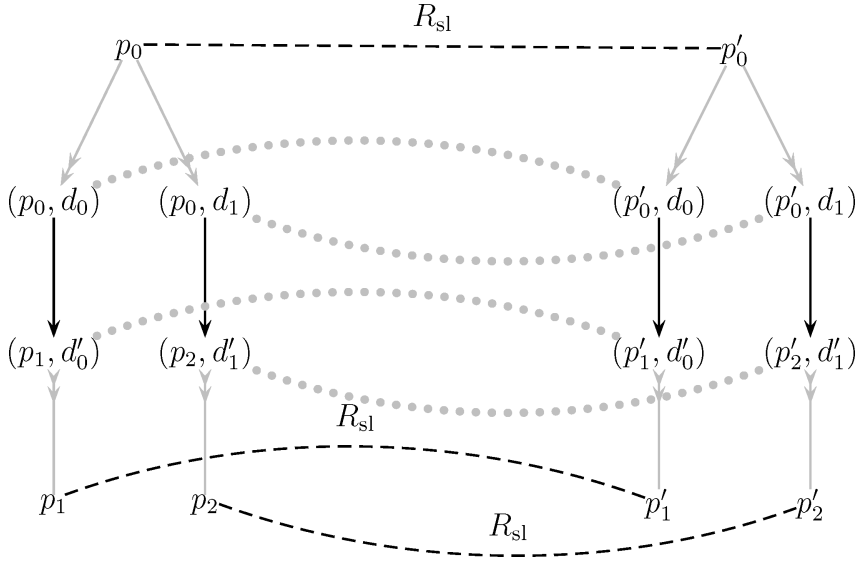


Fig. 3. Stateless bisimilarity.

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 8 (*stateless bisimilarity*)**.** A relation $R_{\mathrm{sl}} \subseteq C(\Sigma_{\mathrm{p}}) \times C(\Sigma_{\mathrm{p}})$ is a *stateless bisimulation* relation if and only if $\forall_{p,q,d,r,l}(p,q) \in R_{\mathrm{sl}} \Rightarrow$

(1) $\forall_{p',d'}(p,d) \xrightarrow{l}_r (p',d') \Rightarrow \exists_{q'}(q,d) \xrightarrow{l}_r (q',d') \wedge (p',q') \in R_{\mathrm{sl}}$;
(2) $\forall_{q',d'}(q,d) \xrightarrow{l}_r (q',d') \Rightarrow \exists_{p'}(p,d) \xrightarrow{l}_r (p',d') \wedge (p',q') \in R_{\mathrm{sl}}$.

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

Example 9. Consider the transition system specification of Example 4. None of the non-trivial examples of bisimilarity hold anymore for stateless bisimilarity. Namely, it does not hold that $a \underline{\leftrightarrow}_{\mathrm{sl}} b$, $a \underline{\leftrightarrow}_{\mathrm{sl}} c$ or $b \underline{\leftrightarrow}_{\mathrm{sl}} c$. From these one may conclude that stateless bisimilarity is a congruence for the above transition system specification. We prove this formally in Example 17.

None of the three notions of bisimilarity is the perfect notion. State-based bisimilarity is the easiest one to check and establish but is not very robust in applications. It is most suitable for systems that are not subject to any further composition and interference. Initially stateless bisimilarity is a bit more difficult to check and establish but is more robust and suitable for systems that are amenable to further sequential compositions. 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 [22]. 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 [22,12,14]. Our congruence format for state-based bisimilarity can be adapted to the parameterized notion of bisimilarity.

Next, we compare the above three notions of bisimilarity.

3.3. Comparing the notions of bisimilarity

In Examples 4 and 9, we have shown that two processes b and c are state-based 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 state-based bisimilarity (w.r.t. a particular data state). The following corollary states that if a state-based bisimulation relation is closed under the change of data state then it induces a stateless bisimilarity.

Corollary 10. *Let R be a state-based bisimulation relation. If* $\forall_{p,q}(\exists_d((p,d),\ (q,d)) \in R \Rightarrow \forall_{d'}((p,d'),(q,d')) \in R)$ *then* $\forall_{p,q}(\exists_d((p,d),(q,d)) \in R \Rightarrow p \underline{\leftrightarrow}_{\mathrm{sl}} q)$.

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

Corollary 11. *For two arbitrary closed process terms p and q, we have*

(1) *if* $p \leftrightarrow_{\mathrm{sl}} q$, *then* $p \leftrightarrow_{\mathrm{isl}} q$;
(2) $p \leftrightarrow_{\mathrm{isl}} q$ *if and only if*, $(p,d) \leftrightarrow_{\mathrm{sb}} (q,d)$ *for all closed data terms d.*

Again, in Examples 4 and 7, we have shown that a and b are state-based bisimilar with respect to d but they are not initially stateless bisimilar. Thus, state-based bisimilarity (with respect to a particular data state) is strictly weaker than initially stateless bisimilarity.

The following corollary states that stateless bisimilarity implies state-based bisimilarity with respect to all data states.

Corollary 12. *For two arbitrary closed process terms p and q; if* $p \leftrightarrow_{\mathrm{sl}} q$, *then* $(p,d) \leftrightarrow_{\mathrm{sb}} (q,d)$ *for all* $d \in C(\Sigma_{\mathrm{d}})$.

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 [21] with data in three steps for stateless, state-based, 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 [39] with data).

4.1. Congruence format for stateless bisimilarity

In this article, 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 13 (*process-tyft*)**.** Let $(\Sigma_{\mathrm{p}}, \Sigma_{\mathrm{d}}, L, D(Rel))$ be a transition system specification. A deduction rule $dr \in D(Rel)$ is in *process-tyft format* if it is of the form

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

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

We name the set of process variables appearing in the source of the conclusion X_{p} and in the target 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. For simplicity in proofs, we require the acyclicity of the variable

dependency graph (see Section 2), as well. However, this requirement can be removed using the result of [19].

Theorem 14. *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 14.

Definition 15 (*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) $(f(p_0,\ldots,p_{n-1}), f(q_0,\ldots,q_{n-1})) \in \tilde{R}$ for all n-ary $f \in \Sigma_p$, and all $p_0,\ldots,p_{n-1}, q_0,\ldots,q_{n-1} \in C(\Sigma_p)$ such that $(p_i,q_i) \in \tilde{R}$ for all $0 \leqslant i < n$.

Lemma 16. *Let $R \subseteq C(\Sigma_p) \times C(\Sigma_p)$ and $t \in T(\Sigma_p)$. For any two process 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. See [23] for a complete proof. □

Proof of Theorem 14. 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 $0 \leqslant i < n$. Suppose that $p_i \leftrightarrows_{sl} q_i$ for $0 \leqslant i < n$. This means that there are stateless bisimulation relations R_i (for $0 \leqslant i < n$) 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 $(f(p_0,\ldots,p_{n-1}), f(q_0,\ldots,q_{n-1}))$ and that it is a stateless bisimulation relation. The first claim 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 $p = f(p_0,\ldots,p_{n-1})$ 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 $(p_i,q_i) \in \tilde{R}$ for all $0 \leqslant i < n$.

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

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

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

We assume acyclicity of the variable dependency graph. A variable dependency graph has variables as its nodes and for each $i \in I$ there exists an edge from any variable $x \in vars(t_i)$ to variable y_i. Hence, 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 target 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 process substitution σ' as follows:

$$\sigma'(x) = \begin{cases} q_i & \text{if } x = x_i, \\ \sigma(x) & \text{if } x \notin X_{\mathrm{p}} \cup Y_{\mathrm{p}}. \end{cases}$$

Note that thus far this process 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) $(\sigma'(t_i), \xi(u_i)) \xrightarrow{l_i}_{r_i} (\sigma'(y_i), \xi(u'_i))$;
(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 $r \geqslant 1$. 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) $x \in X_{\mathrm{p}}$. Then $x = x_i$ for some $0 \leqslant i < n$. 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) $x \notin X_{\mathrm{p}}$ and $x \notin Y_{\mathrm{p}}$. As $\sigma(x) = \sigma'(x)$ and the identity relation is contained in $\tilde{R}$ obviously $(\sigma(x), \sigma'(x)) \in \tilde{R}$.
(3) $x \in Y_{\mathrm{p}}$. 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 16, that $(\sigma(t_i), \sigma'(t_i)) \in \tilde{R}$; which proves property (1).

As we have a proof of smaller depth for $(\sigma(t_i), \xi(u_i)) \xrightarrow{l_i}_{r_i} (\sigma(y_i), \xi(u'_i))$, by the induction hypothesis, we have the existence of a process term q'_i such that $(\sigma'(t_i), \xi(u_i)) \xrightarrow{l_i}_{r_i} (q'_i, \xi(u'_i))$ 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 $(\sigma'(t_i), \xi(u_i)) \xrightarrow{l_i}_{r_i} (\sigma'(y_i), \xi(u'_i))$ 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 process substitution σ' and data substitution ξ. Then, according to the same deduction rule and using σ' instead of σ, we have $(\sigma'(f(x_0,\ldots,x_{n-1})),\xi(u)) \xrightarrow{l}_r (\sigma'(t'),\xi(u'))$. Since $\sigma'(f(x_0,\ldots,x_{n-1})) = f(q_0,\ldots,q_{n-1}) = q, \xi(u) = d$ and $\xi(u') = d'$ 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 16, 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 $0 \leqslant i < n$. 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 $j \in I$. We have $(\sigma(y_j),\sigma'(y_j)) \in \tilde{R}$ by property (3). But, as $x = y_j$, we also have $(\sigma(x),\sigma'(x)) \in \tilde{R}$.

So this concludes the proof of Theorem 14. □

Example 17. Consider the transition system specification of Example 4. Obviously, all deduction rules are in process-tyft format, hence, by Theorem 14, stateless bisimilarity is a congruence for all process functions from the signature of this transition system specification.

4.2. Congruence format for state-based bisimilarity

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

Example 18. 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:

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

Then, we have: $(a,d) \leftrightarrow_{sb} (b,d)$. On the other hand, however, 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 state-based bisimilar as the first one has an l transition due to deduction rule (1), while the second one does not. Hence, state-based bisimilarity is not a process-congruence (for f).

In the conclusions of the deduction rules 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. Namely, we force the links between process variables and data terms to remain consistent in each of the deduction rules as follows.

Definition 19 (*Sfsb*)**.** A deduction rule *dr* of the following form

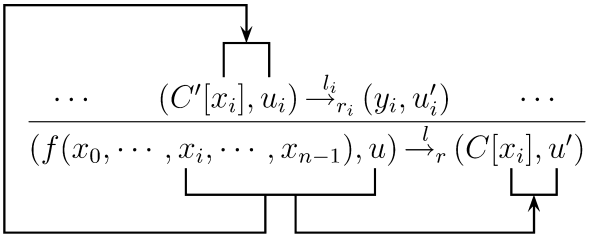
$$(dr)\quad \frac{\{(t_i,u_i)\xrightarrow{l_i}_{r_i}(y_i,u_i')\mid i\in I\}}{(f(x_0,\ldots,x_{n-1}),u)\xrightarrow{l}_{r}(t',u')},$$

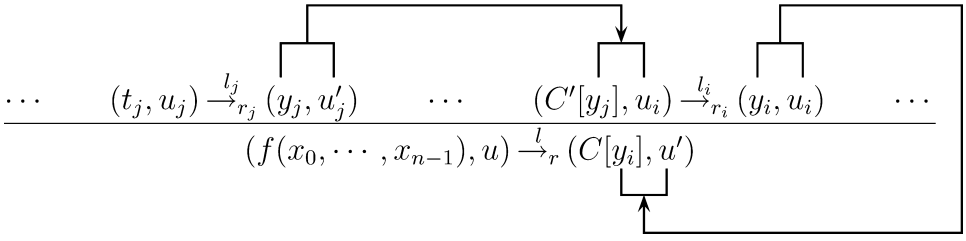
is in *sfsb format* (for Standard Format for state-based bisimulation) if it is in process-tyft format and satisfies the following constraints:

(1) If a variable $x\in X_{\mathrm{p}}$ appears in t', then $u'=u$;
(2) If a variable $y_i\in Y_{\mathrm{p}}$ appears in t', then $u'=u_i$;
(3) If a variable $x\in X_{\mathrm{p}}$ appears in some t_i, then $u_i=u$;
(4) If a variable $y_i\in Y_{\mathrm{p}}$ appears in some t_j, then $u_j=u_i'$.

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, as depicted below, between process variables and data terms from the source of the conclusion to the source of the premises and the target of the conclusion and from the target of the premises to the sources of other premises and finally, to the target of the conclusion.





Theorem 20. *If a transition system specification is in sfsb format, then state-based bisimilarity is a process-congruence for that transition system specification.*

Before proving this theorem, we first define the closure of a relation under state-based congruence and give and prove a lemma that is very useful in the proof of Theorem 20.

Definition 21. 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) $((f(p_0,\ldots,p_{n-1}),d),(f(q_0,\ldots,q_{n-1}),d)) \in \widehat{R}$ 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 $((p_i,d),(q_i,d)) \in \widehat{R}$ for all $0 \leqslant i < n$.

Lemma 22. *Let $R \subseteq (C(\Sigma_p) \times C(\Sigma_d)) \times (C(\Sigma_p) \times C(\Sigma_d))$, $t \in T(\Sigma_p)$, and $d \in C(\Sigma_d)$. For any process 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. The proof is similar to the proof of Lemma 16 which can be consulted in [23]. □

Proof of Theorem 20. It suffices to prove that state-based 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 $0 \leqslant i < n$ and let $d \in C(\Sigma_d)$. Suppose that $(p_i,d) \underline{\leftrightarrow}_{sb} (q_i,d)$ for $0 \leqslant i < n$. This means that there are state-based bisimulation relations R_i (for $0 \leqslant i < n$) that witness these state-based bisimilarities. Let R be the union of these relations R_i: $R = \bigcup_{i=0}^{n} R_i$. Obviously R is also a state-based 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 state-based 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 $0 \leqslant i < n$. The last step of the proof of the transition of p is due to the application of a deduction rule of the following form:

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

This means that there are a process substitution σ and a data substitution ξ such that $\sigma(x_i) = p_i$ for all $0 \leqslant i < n$, $\xi(u) = d$, $\sigma(t') = p'$ and $\xi(u') = d'$. Furthermore, for each $i \in I$, there exists a proof of $(\sigma(t_i),\xi(u_i)) \xrightarrow{l_i}_{r_i} (\sigma(y_i),\xi(u_i'))$ 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 target 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 process substitution σ' as follows:

$$\sigma'(x) = \begin{cases} q_i & \text{if } x = x_i, \\ \sigma(x) & \text{if } x \notin X_{\mathrm{p}} \cup Y_{\mathrm{p}}. \end{cases}$$

Note that thus far this process 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) $((\sigma(t_i), \xi(u_i)), (\sigma'(t_i), \xi(u_i))) \in \widehat{R}$;
(2) $(\sigma'(t_i), \xi(u_i)) \xrightarrow{l_i}_{r_i} (\sigma'(y_i), \xi(u'_i))$;
(3) $((\sigma(y_i), \xi(u'_i)), (\sigma'(y_i), \xi(u'_i))) \in \widehat{R}$.

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 $r \geqslant 1$. 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 show that $((\sigma(x_i), \xi(u_i)), (\sigma'(x_i), \xi(u_i))) \in \widehat{R}$. To do this, the following three cases are distinguished:

(1) $x \in X_{\mathrm{p}}$. Then $x = x_i$ for some $0 \leqslant i < n$. The source of the premise has process variable $x_i \in X_{\mathrm{p}}$; hence, by data-dependency constraint 3, $u_i = u$. 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 $\xi(u) = d$. Thus, we have $((\sigma(x_i), \xi(u_i)), (\sigma'(x_i), \xi(u_i))) \in \widehat{R}$, i.e., $((\sigma(x), \xi(u_i)), (\sigma'(x), \xi(u_i))) \in \widehat{R}$.
(2) $x \notin X_{\mathrm{p}}$ and $x \notin Y_{\mathrm{p}}$. As $\sigma(x) = \sigma'(x)$ and the identity relation is contained in $\widehat{R}$ obviously $((\sigma(x), \xi(u')), (\sigma'(x), \xi(u'))) \in \widehat{R}$.
(3) $x \in Y_{\mathrm{p}}$. Then $x = y_j$ for some $j \in I$. Hence, by data-dependency constraint 4, $u_i = u'_j$. Obviously, also $rank(y_j) < rank(y_i)$. Thus by the induction hypothesis (property (2)) we have, $((\sigma(y_j), \xi(u'_j)), (\sigma'(y_j), \xi(u'_j))) \in \widehat{R}$. But, as $x = y_j$, and $u'_j = u_i$, we also have $((\sigma(x), \xi(u_i)), (\sigma'(x), \xi(u_i))) \in \widehat{R}$.

From the fact that $((\sigma(x), \xi(u_i)), (\sigma'(x), \xi(u_i))) \in \widehat{R}$ for all $x \in vars(t_i)$, by Lemma 22, we have $((\sigma(t_i), \xi(u_i)), (\sigma'(t_i), \xi(u_i))) \in \widehat{R}$; which proves property (1).

As we have a proof of smaller depth for $(\sigma(t_i), \xi(u_i)) \xrightarrow{l_i}_{r_i} (\sigma(y_i), \xi(u'_i))$, by the induction hypothesis, we have the existence of a process term q'_i such that $(\sigma'(t_i), \xi(u_i)) \xrightarrow{l_i}_{r_i} (q'_i, \xi(u'_i))$ and $((\sigma(y_i), \xi(u'_i)), (q'_i, \xi(u'_i))) \in \widehat{R}$. We choose $\sigma'(y_i)$ to be q'_i. Observe that this proves existence of an appropriate process term $\sigma'(y_i)$. Then, obviously, we also have $(\sigma'(t_i), \xi(u_i)) \xrightarrow{l_i}_{r_i} (\sigma'(y_i), \xi(u'_i))$ and $((\sigma(y_i), \xi(u'_i)), (\sigma'(y_i), \xi(u'_i))) \in \widehat{R}$, 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 process substitution σ' and data substitution ξ. Then, according to the same deduction rule and using σ' instead of σ, we have $(\sigma'(f(x_0,\ldots,x_{n-1})),\xi(u)) \xrightarrow{l}_r (\sigma'(t'),\xi(u'))$. Since $\sigma'(f(x_0,\ldots,x_{n-1})) = f(q_0,\ldots,q_{n-1}) = q, \xi(u) = d$ and $\xi(u') = d'$ 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 22, 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) $x \in X_{\mathrm{p}}$. Then $x = x_i$ for some $0 \leqslant i < n$. Hence, $x \in X_{\mathrm{p}}$ and by data-dependency constraint 1, $u = u'$. Hence, $d = \xi(u) = \xi(u') = d'$. 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) $x \notin X_{\mathrm{p}}$ and $x \notin Y_{\mathrm{p}}$. 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) $x \in Y_{\mathrm{p}}$. Then $x = y_j$ for some $j \in I$. Hence, by data-dependency constraint 2, $u'_j = u'$. We obtain $d' = \xi(u') = \xi(u'_j)$. By property (3) we have, $((\sigma(y_j),\xi(u'_j)),(\sigma'(y_j),\xi(u'_j))) \in \widehat{R}$. But, as $x = y_j$, and $\xi(u'_j) = d'$, we also have $((\sigma(x),d'),(\sigma'(x),d')) \in \widehat{R}$.

So this concludes the proof of Theorem 20. □

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 23. 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:

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

These deduction rules are in process-tyft format. Data-dependency constraint 1 is not satisfied by deduction rule (2) as in the target of the conclusion the variable $x \in X_{\mathrm{p}}$ appears but $d \neq d'$. The other data-dependency constraints are indeed satisfied.

The process-congruence result fails on the above specification. As both (a,d) and (b,d) cannot perform any transitions, we have $(a,d) \leftrightarrow_{\mathrm{sb}} (b,d)$. However, it does not hold that $(f(a),d) \leftrightarrow_{\mathrm{sb}} (f(b),d)$ 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 $(a,d') \leftrightarrow_{\mathrm{sb}} (b,d')$ (see deduction rule (1)).

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

Example 24. 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:

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

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 state-based bisimilar. However, $(f(a),d)$ is not state-based 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 state-based 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 25. 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:

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

The 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 (2) since variable $x \in X_{\mathrm{p}}$ appears in the source of the premise but $d' \neq v$.

For this transition system specification, state-based bisimilarity is not a process-congruence. Although we have $(a,d) \leftrightarrow_{\mathrm{sb}} (b,d)$ (because both states deadlock), it is not the case that $(f(a),d)$ and $(f(b),d)$ are state-based bisimilar, 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 26. 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:

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

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 source of its second premise. This also turns out to be harmful for the congruence property, since we have $(a,d) \leftrightarrow_{\mathrm{sb}} (b,d)$ but not $(f(a),d) \leftrightarrow_{\mathrm{sb}} (f(b),d)$ 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 27. 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:

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

This transition system specification does not satisfy the sfsb format and state-based bisimilarity is not a congruence (since $(a,d)\underline{\leftrightarrow}_{\mathrm{sb}}(b,d)$, but it does not hold that $(f(b,a),d)\underline{\leftrightarrow}_{\mathrm{sb}}(f(b,b),d)$). However, it can be checked that initially stateless bisimilarity is indeed a congruence. The reason is that although deduction rule (3) violates data-dependency constraint 1, the violating change in the data 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 $a\underline{\leftrightarrow}_{\mathrm{isl}}b$).

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

Definition 28 (*Sfisl*)**.** A deduction rule dr of the following form

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

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 variable $y_i\in Y_{\mathrm{p}}$ appears in t', then $u'=u_i$;
(2) If a variable $y_i\in Y_{\mathrm{p}}$ appears in some t_j, then $u_j=u_i'$.

Note that the above two constraints are the same as constraints 2 and 4 in Definition 19. However, the two other data-dependency constraints of Definition 19 that were required for variables from the set X_{p}, 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 29. A variable $x\in X_{\mathrm{p}}$ is called *unresolved* if

$$\exists_{i\in I}(x\in vars(t_i)\Rightarrow u\neq u_i)\vee(x\in vars(t')\Rightarrow u\neq u').$$

We define $X_{\mathrm{p}}^{\mathrm{u}}$ to be the set of unresolved variables from the set X_{p}.

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 source 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 target of the conclusion or the source of a premise.

Definition 30. For a given transition system specification in process-tyft format, we define, for all $f \in \Sigma_p$, IV_f as a minimal set that satisfies, 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 source of a premise or the target of the conclusion of dr:

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

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. The following two examples illustrate existence and absence of IV_f.

Example 31. Consider the transition system specification of Example 27, in deduction rule (3), variable x_1 is unresolved, and thus $1 \in IV_f$. For the deduction rules defining the two constants a and b, there are no unresolved variables and $IV_a = IV_b = \emptyset$. The second global condition is trivially satisfied for all IV sets.

Example 32. Consider the following transition system specification with process constants a, b, and c, unary process functions f and g, and data constants d, d', and d''.

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

$$(4)\ \frac{(x_0,d) \xrightarrow{l} (y_0,d')}{(f(x_0),d) \xrightarrow{l} (g(y_0),d')}, \qquad (5)\ \frac{}{(g(x_0),d') \xrightarrow{l} (x_0,d'')}.$$

In deduction rule (5), variable x_0 is unresolved and hence $0 \in IV_g$. However, global constraint 2 requires that in deduction rule (4), $y_0 = x_0$ which is a contradiction (since $0 \in IV_g$, $y_0 \in vars(y_0)$ and f is a unary process function). Hence, we may conclude that no consistent IV_g exists. In fact, one can check that initially stateless is not a congruence for the above transition system specification, as it holds that $a \underline{\leftrightarrow}_{isl} b$ but not $f(a) \underline{\leftrightarrow}_{isl} f(b)$ ($(f(a),d)$ after two steps arrives in (b,d'') which deadlocks but $(f(b),d)$ arrives in (c,d'') which can perform self-transitions).

Definition 33 (*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 arguments 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 34. *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 34.

Definition 35 (*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).

$$\begin{aligned} V(x) &= \emptyset, \\ V(f(t_0, \ldots, t_{n-1})) &= \bigcup_{0 \leqslant i < n, i \in IV_f} vars(t_i) \cup \bigcup_{0 \leqslant i < n, i \notin IV_f} V(t_i). \end{aligned}$$

Lemma 36. *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 process 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 ($n \geqslant 1$) function symbol $f \in \Sigma_p$ and $t_i \in T(\Sigma_p)$ ($0 \leqslant i < n$). 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 35.

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 36, 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 36, 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 of Theorem 34. 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 $0 \leqslant i < n$ and let $d \in C(\Sigma_d)$. Suppose that $p_i \underline{\leftrightarrow}_{isl} q_i$ for $0 \leqslant i < n$. This means that there are state-based bisimulation relations R_i (for $0 \leqslant i < n$) 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 state-based 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 state-based bisimulation relation.

As $((p_i, d), (q_i, d)) \in R_i$ and $R_i \subseteq R \subseteq \overline{R}$, for all $0 \leqslant i < n$ and all $d \in C(\Sigma_d)$, it follows that $((p_i, d), (q_i, d)) \in \overline{R}$, for all $0 \leqslant i < n$ 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 a process substitution σ and a data substitution ξ such that $\sigma(x_i) = p_i$ for all $0 \leqslant i < n$, $\xi(u) = d$, $\sigma(t') = p'$ and $\xi(u') = d'$. Furthermore, for each $i \in I$, there exists a proof of $(\sigma(t_i), \xi(u_i)) \xrightarrow{l_i}_{r_i} (\sigma(y_i), \xi(u_i'))$ 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 target 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 process substitution σ' as follows:

$$\sigma'(x) = \begin{cases} q_i & \text{if } x = x_i, \\ \sigma(x) & \text{if } x \notin X_{\mathrm{p}} \cup Y_{\mathrm{p}}. \end{cases}$$

Note that thus far this process 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) $((\sigma(t_i), \xi(u_i)), (\sigma'(t_i), \xi(u_i))) \in \overline{R}$;
(2) $(\sigma'(t_i), \xi(u_i)) \xrightarrow{l_i}_{r_i} (\sigma'(y_i), \xi(u'_i))$;
(3) $((\sigma(y_i), \xi(u'_i)), (\sigma'(y_i), \xi(u'_i))) \in \overline{R}$.

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 $r \geqslant 1$. 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 36. Hence we prove

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

and

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

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:
- $x \notin X_{\mathrm{p}}$ and $x \notin Y_{\mathrm{p}}$. Then, we have $\sigma(t_i) = \sigma'(t_i)$. Since $\overline{R}$ is reflexive we obtain $((\sigma(x), \xi(u_i)), (\sigma'(x), \xi(u_i))) \in \overline{R}$.
- $x \in Y_{\mathrm{p}}$. Then $x = y_j$ for some $j \in I$. Hence, by local data-dependency constraint 2, $u_i = u'_j$. Observe that $rank(y_j) < r$. By the induction hypothesis (property (3)), we have $((\sigma(y_j), \xi(u'_j)), (\sigma'(y_j), \xi(u'_j))) \in \overline{R}$. Hence, as $y_j = x$ and $\xi(u_j) = \xi(u_i)$, we have $((\sigma(x), \xi(u_i)), (\sigma'(x), \xi(u_i))) \in \overline{R}$.
- $x \in X_{\mathrm{p}}$. Then, $x = x_j$ for some $0 \leqslant j < n$. We distinguish two cases:
  - $j \in IV_f$. Then, we use assumption (4) to obtain the desired $((\sigma(x), \xi(u_j)), (\sigma'(x), \xi(u_j))) \in \overline{R}$;
  - $j \notin IV_f$. 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 $u_i = u$. Hence $d = \xi(u) = \xi(u_i)$. Thus, we have $((\sigma(x), \xi(u_i)), (\sigma'(x), \xi(u_i))) \in \overline{R}$.

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 $0 \leqslant j < n'$. 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 $((\sigma(x), \xi(u_i)), (\sigma'(x), \xi(u_i))) \in \overline{R}$.
For the second property observe that $x \in V(t_i)$ implies (1) $x \in vars(t'_j)$ for some $0 \leqslant j < n'$ 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 $0 \leqslant k < n$ 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 $((\sigma(t_i), \xi(u_i)), (\sigma'(t_i), \xi(u_i))) \in \overline{R}$. We also have a proof of smaller depth for $(\sigma(t_i), \xi(u_i)) \xrightarrow{l_i}_{r_i} (\sigma(y_i), \xi(u'_i))$. Then, by the induction hypothesis, we have the existence of a process term q'_i such that $(\sigma'(t_i), \xi(u_i)) \xrightarrow{l_i}_{r_i} (q'_i, \xi(u'_i))$ and $((\sigma(y_i), \xi(u'_i)), (q'_i, \xi(u'_i))) \in \overline{R}$. We choose $\sigma'(y_i)$ to be q'_i. Observe that this proves existence of an appropriate process term $\sigma'(y_i)$. This concludes the proof of 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 process substitution σ' and data substitution ξ. Then, according to the same deduction rule and using σ' instead of σ, we have $(\sigma'(f(x_0, \ldots, x_{n-1})), \xi(u)) \xrightarrow{l}_r (\sigma'(t'), \xi(u'))$. Since $\sigma'(f(x_0, \ldots, x_{n-1})) = f(q_0, \ldots, q_{n-1}) = q, \xi(u) = d$ and $\xi(u') = d'$ 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 36. 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 $j \in I$. Hence, by local data-dependency constraint 1, $u' = u'_j$. By property (3), we have $((\sigma(y_j), \xi(u'_j)), (\sigma'(y_j), \xi(u'_j))) \in \overline{R}$. Hence, as $y_j = x$ and $\xi(u_j) = \xi(u') = d'$, we have $((\sigma(x), d'), (\sigma'(x), d')) \in \overline{R}$.

- $x \in X_{\mathrm{p}}$. Then, $x = x_j$ for some $0 \leqslant j < n$. We distinguish two cases:
  - $j \in IV_f$. Then, we use assumption (4) to obtain the desired $((\sigma(x), d')), (\sigma'(x), d')) \in \overline{R}$;
  - $j \notin IV_f$, 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 $u' = u$. Hence $d = \xi(u) = \xi(u') = d'$. 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 $g \in \Sigma_{\mathrm{p}}$ and $t'_j \in T(\Sigma_{\mathrm{p}})$ for $0 \leqslant j < n'$. 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 $0 \leqslant j < n'$ 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 $0 \leqslant k < n$ 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 $d'' \in C(\Sigma_{\mathrm{d}})$. In the second case, by the induction hypothesis (second property), we have $((\sigma(x), d''), (\sigma'(x), d'')) \in \overline{R}$ for all $d'' \in C(\Sigma_{\mathrm{d}})$.

So this concludes the proof of Theorem 34. □

Example 37. Consider the transition system specification of Example 4. Obviously the deduction rules are in process-tyft format. They also satisfy the sfisl format as no variables introduced in the target of any premise are used in the source of a premise or in the target 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 38. 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:

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

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

The deduction rules (1)–(3) are in sfisl format, trivially. Deduction rule (4) does not satisfy local data-dependency constraint 1, since $y \in Y_{\mathrm{p}}$ but $d \neq d'$. 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 $a \underline{\leftrightarrow}_{\mathrm{isl}} b$, but not that $f(a) \underline{\leftrightarrow}_{\mathrm{isl}} f(b)$ 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 $(c, d') \underline{\leftrightarrow}_{\mathrm{sb}} (c', d')$.

Example 39. Consider the transition system specification from Example 38 with deduction rule (4) replaced by

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

The deduction rules (1)–(3) are in sfisl format, trivially. Deduction rule (4) satisfies local data-dependency constraint 1 of sfisl, but local data-dependency constraint 2 is not satisfied as $y \in Y_{\mathrm{p}}$ but $d' \neq v'$. 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: $a \underline{\leftrightarrow}_{\mathrm{isl}} b$ holds, but it does not hold that $f(a) \underline{\leftrightarrow}_{\mathrm{isl}} f(b)$ 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 state-based 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 state-based 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 state-based 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 state-based 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 state-based 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 40. 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:

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

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 $(a, d) \underline{\leftrightarrow}_{\mathrm{sb}} (b, d)$, but not $(f(a), d) \underline{\leftrightarrow}_{\mathrm{sb}} (f(b), d)$. Hence, congruence of state-based bisimilarity does not hold.

Example 41. 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) $\dfrac{}{(c,d') \xrightarrow{l'} (c,d')}$, (2) $\dfrac{}{(f(a),d) \xrightarrow{l} (b,d)}$,

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

State-based 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), whereas $(f(b),d)$ is forced to make a transition to (c,d). Clearly, b and c are not stateless bisimilar (due to their difference w.r.t. data state d').

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

Lemma 42. *For a transition system specification, if state-based bisimilarity is a congruence, then initially stateless bisimilarity is a congruence, as well.*

Proof. Suppose that $p_i \underline{\leftrightarrow}_{\mathrm{isl}} q_i$ for $0 \leqslant i < n$. By definition this means that there exist state-based bisimulation relations R_i such that $((p_i,d),(q_i,d)) \in R_i$ for all d. Since state-based bisimilarity is a congruence (by assumption), we have, for each d, the existence of a state-based 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 state-based 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 means that $f(p_0,\ldots,p_{n-1}) \ \underline{\leftrightarrow}_{\mathrm{isl}} \ f(q_0,\ldots,q_{n-1})$. □

Corollary 43. *If a transition system specification is in sfsb format, then initially stateless bisimilarity is a congruence for it.*

Lemma 42 shows that congruence for initially stateless bisimilarity is either stronger than or incomparable to congruence for stateless bisimilarity (since in Example 41, we have already shown that there exists a case were state-based bisimilarity, thus initially stateless bisimilarity, is a congruence but stateless bisimilarity is not). To prove the incomparability result, we need a counterexample where stateless bisimilarity is a congruence but initially stateless bisimilarity is not (the counterexample of Example 40 does not work in this case). The following example establishes this fact.

Example 44. 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) $\dfrac{}{(a,d') \xrightarrow{l} (a,d)}$, (2) $\dfrac{}{(b,d') \xrightarrow{l} (c,d')}$, (3) $\dfrac{}{(c,d) \xrightarrow{l} (c,d)}$,

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

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 $a \underline{\leftrightarrow}_{\mathrm{isl}} b$ but not $f(a) \underline{\leftrightarrow}_{\mathrm{isl}} f(b)$.

So, to conclude, we have proved in this section, that congruence for state-based 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:

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

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 formats in presence of such constructs.

4.5.1. Deduction rules in tyxt format

A common type of deduction rules used in transition system specifications is the *tyxt* form which has the following structure:

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

Rules of the above form fit within the *tyft* form if we replace it with a copy the above rule for each function symbol $f \in \Sigma_p$ with (arbitrary) arity n where all occurrences of x are substituted with $f(x_0, \ldots, x_{n-1})$ with $x_i \notin vars(dr)$ for $0 \leqslant i < n$:

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

Observe that the resulting deduction rule is in process-tyft format indeed. In [23], it is shown that the original transition system specification and the unfolded one are transition equivalent (meaning that the same transitions can be derived).

For our congruence results there is no problem in also allowing deduction rules in tyxt format. It is not necessary to explicitly transform the transition system specification as described above to apply our congruence results for stateless and state-based bisimilarity since deduction rules in tyxt format transform into deduction rules in process-tyft format and since any data-dependency constraint involving x in the original deduction rule is replaced by data-dependency constraints involving the x_i variables in the unfolded deduction rule and vice versa.

Checking whether initially stateless bisimilarity is a congruence is not as straightforward due to the global data-dependency constraint. There are two simple solutions. First, check whether state-based bisimilarity is a congruence; if so, so is initially stateless bisimilarity. Or second, check congruence of initially stateless bisimilarity on the unfolded transition system specification.

4.5.2. Deduction rules with predicates

Another common phenomenon is the presence of predicates. Predicates of the form $P(t,u)$ may be present in the premises or the conclusion of a deduction rule. Thus, we allow deduction rules of the following forms:

$$(dr_1)\ \frac{\{(t_i,u_i)\xrightarrow{l_i}_{r_i}(y_i,u_i')|i\in I\}\cup\{P_j(t_j',v_j)\mid j\in J\}}{(f(x_0,\ldots,x_{n-1}),u)\xrightarrow{l}_{r}(t',u')}$$

and

$$(dr_2)\ \frac{\{(t_i,u_i)\xrightarrow{l_i}_{r_i}(y_i,u_i')|i\in I\}\cup\{P_j(t_j',v_j)\mid j\in J\}}{P(f(x_0,\ldots,x_{n-1}),u)}.$$

Predicates can be dealt with in the above formats, as if they are source of a transition relation. This can be formally proved by introducing a fresh dummy transition relation for each predicate and replacing occurrences of that predicate in premises by this transition relation with a target consisting of a fresh dummy process variable and a fresh dummy data variable and occurrences in conclusions by this transition relation with a state consisting of a fresh process constant and a fresh data constant in the target. This transformation is similar to the transformation from [5] for the path format.

Hence, a deduction rule of the form (dr_1) is replaced by a deduction rule of the form

$$(dr_1')\ \frac{\{(t_i,u_i)\xrightarrow{l_i}_{r_i}(y_i,u_i')|i\in I\}\cup\{(t_j',v_j)R_{P_j}(z_j,w_j)\mid j\in J\}}{(f(x_0,\ldots,x_{n-1}),u)\xrightarrow{l}_{r}(t',u')},$$

where the z_j are all different process variables that did not occur in (dr_1) and the w_j are all different data variables that did not occur in (dr_1).

A deduction rule of the form (dr_2) is replaced by a deduction rule of the form

$$(dr_2')\ \frac{\{(t_i,u_i)\xrightarrow{l_i}_{r_i}(y_i,u_i')|i\in I\}\cup\{(t_j',v_j)R_{P_j}(z_j,w_j)\mid j\in J\}}{(f(x_0,\ldots,x_{n-1}),u)R_P(a,d)}.$$

where additionally a is a process constant such that $a\notin\Sigma_{\mathrm{p}}$ and d is a data constant such that $d\notin\Sigma_{\mathrm{d}}$.

As before, this transformation does not have to be carried out explicitly. Observe that the original transition system specification is in process-tyft format (by considering the arguments of the predicates as sources of premises and conclusions) iff the transformed transition system specification is in process-tyft format. Thus, stateless bisimilarity is a congruence for the original transition system specification iff it is a congruence for the transformed transition system specification.

With respect to the sfsb format, observe that the sources of data-dependencies are the same for the original rules and the transformed rules. This is due to the decision to treat the argument of

the predicates as sources of conclusions and premises. Also observe that there are new targets of premises introduced by the transformation, but those only contain fresh variables and can therefore never be used to satisfy a data-dependency constraint. Hence, the transformed transition system specification satisfies the sfsb format iff the original transition system specification does.

For the local data-dependencies of the sfisl format a similar observation holds. For, the new process constant a, we find $IV_a = \emptyset$ and for all other process constants and functions we find that the sets IV are the same for the original transition system specification and the transformed one. Therefore, the transformed transition system specification satisfies the sfisl format iff the original transition system specification does.

4.5.3. Deduction rules with negative premises

As argued by Groote [20], it is often convenient to describe that certain activity can be performed based on the absence of certain actions. Thus, we allow for deduction rules of the following form:

$$(dr)\quad \frac{\{(t_i,u_i) \xrightarrow{l_i}_{r_i} (y_i,u_i') \mid i \in I\} \cup \{(t_j,u_j) \overset{l_j}{\nrightarrow}_{r_j} \mid j \in J\}}{(f(x_0,\ldots,x_{n-1}),u) \xrightarrow{l}_{r} (t',u')}.$$

For such transition system specifications another definition is required of what a proof of a transition is (see [20,39]). Not every transition system specification with negative premises defines a transition relation. Different interpretations of negative premises can be considered (see [40]), but here we adopt the interpretation put forward by [20]. A sufficient condition for the existence of a transition relation is that the transition system specification is *stratifiable*.

A stratification is a metric on formulae 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). For stratifiable transition system specifications, our congruence results can be used safely.

5. Applications of the formats

In this section, some process languages from the 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 notions of bisimilarity introduced in this article are used (possibly with a different formulation) and whether the deduction rules are in the corresponding format. We focus on stateless bisimilarity and initially stateless bisimilarity as these seem to be of most interest in these applications.

5.1. The coordination language Linda

In [11], 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 article, 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 non-deterministic choice ($+$), sequential composition ($;$), and parallel composition ($||$). The data signature of this language consists of a constant $\{\}$ for the empty multiset and a class of unary function symbols $\cup\{t\}$, for all tuples t, denoting the union of a multiset with a singleton multiset containing tuple t. The 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 multiset modelling the shared data space.

The transition system specification defines one relation $\rightarrow$ and one predicate $\downarrow$. Negative premises and deduction rules in tyxt format are not used.

The deduction rules of our reformulation of the SOS for Linda from [11] are the following:

$$(1)\ \frac{}{(\epsilon,\varsigma)\downarrow},\quad (2)\ \frac{}{(ask(t),\varsigma\cup\{t\})\rightarrow(\epsilon,\varsigma\cup\{t\})},$$

$$(3)\ \frac{}{(tell(t),\varsigma)\rightarrow(\epsilon,\varsigma\cup\{t\})},\quad (4)\ \frac{}{(get(t),\varsigma\cup\{t\})\rightarrow(\epsilon,\varsigma)},$$

$$(5)\ \frac{[t\notin\varsigma]}{(nask(t),\varsigma)\rightarrow(\epsilon,\varsigma)},\quad (6)\ \frac{(x_0,\varsigma)\downarrow}{(x_0+x_1,\varsigma)\downarrow},\quad (7)\ \frac{(x_1,\varsigma)\downarrow}{(x_0+x_1,\varsigma)\downarrow},$$

$$(8)\ \frac{(x_0,\varsigma)\rightarrow(y,\varsigma')}{(x_0+x_1,\varsigma)\rightarrow(y,\varsigma')},\quad (9)\ \frac{(x_1,\varsigma)\rightarrow(y,\varsigma')}{(x_0+x_1,\varsigma)\rightarrow(y,\varsigma')},$$

$$(10)\ \frac{(x_0,\varsigma)\rightarrow(y,\varsigma')}{(x_0\ ;\ x_1,\varsigma)\rightarrow(y\ ;\ x_1,\varsigma')},\quad (11)\ \frac{(x_0,\varsigma)\downarrow\ (x_1,\varsigma)\rightarrow(y,\varsigma')}{(x_0\ ;\ x_1,\varsigma)\rightarrow(y,\varsigma')},$$

$$(12)\ \frac{(x_0,\varsigma)\rightarrow(y,\varsigma')}{(x_0||x_1,\varsigma)\rightarrow(y||x_1,\varsigma')},\quad (13)\ \frac{(x_1,\varsigma)\rightarrow(y,\varsigma')}{(x_0||x_1,\varsigma)\rightarrow(x_0||y,\varsigma')},$$

$$(14)\ \frac{(x_0,\varsigma)\downarrow\ (x_1,\varsigma)\downarrow}{(x_0\ ;\ x_1,\varsigma)\downarrow},\quad (15)\ \frac{(x_0,\varsigma)\downarrow\ (x_1,\varsigma)\downarrow}{(x_0||x_1,\varsigma)\downarrow}.$$

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 45. 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 $p \leftrightarrow_{\mathrm{isl}} q$ (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 $(p \parallel r, \{1,2\})$ 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 $(q \parallel r, \{1,2\})$ can never make four consecutive transitions before termination. Thus, we conclude that initially stateless bisimilarity is not a congruence with respect to the parallel composition.

5.2. The timed process algebra timed μCRL

In [21], a timed extension of the language μCRL, called timed 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: $\left(\sum_x \_\right)_{x\in V}$, $(\_ \cdot t)_{t\in T}$;
- binary process functions: $\_ + \_$, $\_ \cdot \_$, $(\_ \lhd b \rhd \_)_{b\in B}$, $\_ \parallel \_$.

The meaning of the sets A, V, T, and B, and the meaning of the process constants and functions are irrelevant. The process functions that we do not consider here are either only introduced for axiomatization purposes ($\between$, $|$, $\ll$) 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 $\left(\_ \xrightarrow{a} \surd\right)_{a\in A}$;
- a family of 'action-transition' relations $\left(\_ \xrightarrow{a} \_\right)_{a\in A}$;
- a 'time-transition' relation $\_ \xrightarrow{\iota} \_$.

The data state consists of an element of the set T (reflecting time). In [21], $U(p,t)$ is written as $U(t,p)$ and $(p,t) \xrightarrow{a} \surd$ is written as $(p,t) \xrightarrow{a} (\surd, t)$. In this section, we use the notations $U(p,t)$ and $(p,t) \xrightarrow{a} \surd$. The deduction rules are given below.

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

$$(4)\ \frac{(x_0,t) \xrightarrow{l} \surd}{\begin{array}{l}(x_0+x_1,t) \xrightarrow{l} \surd\\ (x_1+x_0,t) \xrightarrow{l} \surd\end{array}}, \qquad (5)\ \frac{(x_0,t) \xrightarrow{l} (y,t)}{\begin{array}{l}(x_0+x_1,t) \xrightarrow{l} (y,t)\\ (x_1+x_0,t) \xrightarrow{l} (y,t)\end{array}},$$

(6) $\dfrac{U(x_0,t)}{\begin{array}{l}U(x_0+x_1,t)\\ U(x_1+x_0,t)\end{array}}$, (7) $\dfrac{(x_0,t)\xrightarrow{l}\checkmark}{(x_0\cdot x_1,t)\xrightarrow{l}(x_1,t)}$,

(8) $\dfrac{(x_0,t)\xrightarrow{l}(y,t)}{(x_0\cdot x_1,t)\xrightarrow{l}(y\cdot x_1,t)}$, (9) $\dfrac{U(x_0,t)}{U(x_0\cdot x_1,t)}$,

(10) $\dfrac{(x_0,t)\xrightarrow{l}\checkmark\quad[\models b]}{(x_0\lhd b\rhd x_1,t)\xrightarrow{l}\checkmark}$, (11) $\dfrac{(x_1,t)\xrightarrow{l}\checkmark\quad[\not\models b]}{(x_0\lhd b\rhd x_1,t)\xrightarrow{l}\checkmark}$,

(12) $\dfrac{(x_0,t)\xrightarrow{l}(y,t)\quad[\models b]}{(x_0\lhd b\rhd x_1,t)\xrightarrow{l}(y,t)}$, (13) $\dfrac{(x_1,t)\xrightarrow{l}(y,t)\quad[\not\models b]}{(x_0\lhd b\rhd x_1,t)\xrightarrow{l}(y,t)}$,

(14) $\dfrac{U(x_0,t)\quad[\models b]}{U(x_0\lhd b\rhd x_1,t)}$, (15) $\dfrac{U(x_1,t)\quad[\not\models b]}{U(x_0\lhd b\rhd x_1,t)}$,

(16) $\dfrac{(x[e/v],t)\xrightarrow{l}\checkmark}{\left(\sum\limits_v x,t\right)\xrightarrow{l}\checkmark}$, (17) $\dfrac{(x[e/v],t)\xrightarrow{l}(y,t)}{\left(\sum\limits_v x,t\right)\xrightarrow{l}(y,t)}$, (18) $\dfrac{U(x[e/v],t)}{U(\sum\limits_v x,t)}$,

(19) $\dfrac{(x,t)\xrightarrow{l}\checkmark}{(x\lhd t,t)\xrightarrow{l}\checkmark}$, (20) $\dfrac{(x,t)\xrightarrow{l}(y,t)}{(x\lhd t,t)\xrightarrow{l}(y,t)}$, (21) $\dfrac{U(x,t)\quad[t\leqslant t']}{U(x\lhd t',t)}$,

(22) $\dfrac{U(x,t')\quad[t<t']}{(x,t)\xrightarrow{\iota}(x,t')}$,

(23) $\dfrac{(x_0,t)\xrightarrow{l}\checkmark\quad(x_1,t)\xrightarrow{l'}\checkmark\quad[\gamma(l,l')=l'']}{(x_0\parallel x_1,t)\xrightarrow{l''}\checkmark}$,

(24) $\dfrac{(x_0,t)\xrightarrow{l}\checkmark}{\begin{array}{l}(x_0\parallel x_1,t)\xrightarrow{l}(x_1,t)\\ (x_1\parallel x_0,t)\xrightarrow{l}(x_1,t)\end{array}}$, (25) $\dfrac{(x_0,t)\xrightarrow{l}(y,t)}{\begin{array}{l}(x_0\parallel x_1,t)\xrightarrow{l}(y\parallel x_1,t)\\ (x_1\parallel x_0,t)\xrightarrow{l}(x_1\parallel y,t)\end{array}}$,

(26) $\dfrac{(x_0,t)\xrightarrow{l}\checkmark\quad(x_1,t)\xrightarrow{l'}(y,t)\quad[\gamma(l,l')=l'']}{(x_0\parallel x_1,t)\xrightarrow{l''}(y,t)\quad(x_1\parallel x_0,t)\xrightarrow{l''}(y,t)}$,

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

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

Observe that in this transition system specification two relations and two predicates are used. Negative premises do not occur, but there is a deduction rule in tyxt format.

The equivalence used in [21] for timed μCRL process terms is timed bisimilarity, which coincides with our notion of initially stateless bisimilarity. The definition of timed bisimilarity in [21] does not require the delay predicate to be transferred between related processes. The notion of initially stateless bisimilarity presented in this article 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 in [21]. In [33], a reformulation of the semantics of timed μCRL is given in such a way that the data state is encoded into the process terms at the expense of an auxiliary operator. Then, the notion of timed bisimilarity corresponds with the traditional notion of bisimilarity, for which congruence is proven using traditional means.

Stateless bisimilarity. Note that although stateless bisimilarity is not considered in [21], 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 deduction rule (22). For this deduction rule, the data-dependency constraints 1 and 3 of sfsb are violated in the target of the conclusion and the source of the (only) premise as $x \in X_{\mathrm{p}}$ but $t' \neq t$ (note that data-dependency constraints 2 and 4 are respected by this rule). Hence, state-based bisimilarity cannot be concluded to be a congruence for any of the non-nullary[1] 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. Before we discuss initially stateless bisimilarity in more detail we emphasize that deduction rule (22) needs to be unseasoned before the format can be applied. Deduction rule (22) maps to a collection of deduction rules of the form

$$(22_f)\quad \frac{U(f(x_0,\ldots,x_{n-1}),t')\quad [t<t']}{(f(x_0,\ldots,x_{n-1}),t)\xrightarrow{\iota}\left(f(x_0,\ldots,x_{n-1}),t'\right)};$$

one for each n-ary process function f in the signature of timed μCRL.

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.

[1] For nullary process functions there is no data-dependency at all.

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 source of a premise or in the target 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 ($\in X_{\mathrm{p}}$). 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:

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

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. The hybrid process algebra HyPA

In [14], 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: $\oplus$, $\odot$, $\blacktriangleright$, $\rhd$, $\parallel$, $\lfloor\!\lfloor$, and $|$.

Negative premises and rules in tyxt format are not used in this transition system specification.

We refrain from giving further information about the intended meaning of the sets A, C, and D, and the meaning of the process constants and functions as these are irrelevant to the application of our congruence theorems on this language. 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 $\left(\_ \overset{\sigma}{\rightsquigarrow} \_\right)_{\sigma\in T\to Val}$.

Also, the meaning of the set T is irrelevant for our purposes. The deduction rules are given below.

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

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

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

$$(8)\ \frac{(x_0,\nu)\ \checkmark\quad (y_0,\nu)\ \checkmark}{(x_0\odot y_0,\nu)\ \checkmark},\qquad (9)\ \frac{(x_0,\nu)\xrightarrow{l}(y,\nu')}{(x_0\odot x_1,\nu)\xrightarrow{l}(y\odot x_1,\nu')},$$

$$(10)\ \frac{(x_0,\nu)\ \checkmark\quad (x_1,\nu)\xrightarrow{l}(y,\nu')}{(x_0\odot x_1,\nu)\xrightarrow{l}(y,\nu')},$$

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

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

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

$$(17)\ \frac{(x_0,\nu)\overset{\sigma}{\rightsquigarrow}(y,\nu')\quad (x_1,\nu)\ \checkmark}{\begin{array}{l}(x_0\parallel x_1,\nu)\overset{\sigma}{\rightsquigarrow}(y,\nu')\\ (x_1\parallel x_0,\nu)\overset{\sigma}{\rightsquigarrow}(y,\nu')\\ (x_0\mid x_1,\nu)\overset{\sigma}{\rightsquigarrow}(y,\nu')\\ (x_1\mid x_0,\nu)\overset{\sigma}{\rightsquigarrow}(y,\nu')\end{array}},\qquad (18)\ \frac{(x_0,\nu)\xrightarrow{a,\nu'}(y,\nu'')}{\begin{array}{l}(x_0\parallel x_1,\nu)\xrightarrow{a,\nu'}(y\parallel x_1,\nu'')\\ (x_1\parallel x_0,\nu)\xrightarrow{a,\nu'}(x_1\parallel y,\nu'')\\ (x_0\between x_1,\nu)\xrightarrow{a,\nu'}(y\parallel x_1,\nu'')\end{array}},$$

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

$$(20)\quad \frac{(x,\nu)\xrightarrow{a,\nu'}(y,\nu'')\quad [a\notin H]}{(\partial_H(x),\nu)\xrightarrow{a,\nu'}(\partial_H(y),\nu'')},$$

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

On HyPA process terms, in [14], a notion of robust bisimilarity is defined that, for HyPA, coincides with our definition of stateless bisimilarity. Furthermore, in [13], for the purpose of analyzing sequential HyPA processes (i.e., HyPA processes without operators for parallel composition), a notion of bisimilarity is defined that coincides with our notion of initially stateless bisimilarity.

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

State-based bisimilarity. With respect to the notion of state-based bisimilarity, as defined in this article, it can be established that state-based bisimilarity is a process-congruence for the constants of HyPA, the alternative composition operator ($\oplus$), and the encapsulation operator ($\partial_H()$), 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.

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

One might wonder whether this means that our format for state-based 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 state-based bisimilarity is 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 sources of premises or targets 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 sources of premises or targets 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 target of the conclusion. The term

occurring as argument 1 of this use, x_1, is the index 1 variable from the source 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 $IV_{\between} \supseteq \{1\}$. All parallel composition operators use parallel composition in the target 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 $IV_{\parallel}$, $IV_{\between}$, 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. In fact, in [13], our congruence theorem for initially stateless bisimilarity has been used to obtain this result.

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. An example of process terms illustrating this for parallel composition is given in [13].

5.4. The discrete-event process language χ_σ

In [10], 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 process 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: $[]$, $;$, $\parallel$.

In the transition system specification of this language both predicates and relations are used. For both types of formulae negative occurrences as a premise appear in the semantics.

The notion of equivalence that is considered in [10] is (a different formulation of) stateless bisimilarity. The authors attempt to prove that this equivalence is a congruence for the constant and function symbols of χ_σ by using the so-called relaxed PANTH format [25,4]. For that purpose they consider the begin and end data state of a transition as part of the label of that transition. This way their transition relations and predicates are defined on process terms (without data state). A mistake they make is that in defining which formulae are negative formulae they do not consider the start state as a part of the label. This means that their negative formulae and the ones allowed by [25] are different. Therefore we have serious doubts as to the applicability of the relaxed PANTH format to the given transition system specification of χ_σ. Nevertheless, stateless bisimilarity is a congruence since all deduction rules of the transition system specification are in process-tyft format.

6. Conclusion

In this article, 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 article, we illustrated the use of our format using a data coordination language, called Linda, and several process algebras.

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 state-based 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 congruence with respect to weaker notions of bisimulation (e.g., branching bisimulation) is another interesting direction for our future research.

We are currently investigating a bi-algebraic and categorical interpretation of notions of bisimulation with data, following the approach of [37,38,35]. In this article, we have only proved sufficient conditions for the notions of bisimulation with data to be a congruence. Although we have already shown that no straightforward relaxation of our formats is possible, we could not prove that no relaxation is possible at all. Using the abstract interpretation of semantic rules (as distributive laws), bisimulation, and congruence in a co-algebraic settings, we might be able to investigate whether our imposed formats are indeed necessary for congruence or they can be relaxed in any way.

Generating equational theories from transition systems specifications is another direction of our ongoing research. Deriving algebraic axioms for SOS rules in [1,3] are among notable examples in this direction which try to generate a set of sound and complete axioms for a given operational semantics in a syntactic format. Both [1,3] assume the existence of a number of standard constants and operators in the signature and we believe that these restrictions on the semantics can be relaxed in several ways (even in a setting without data).

Acknowledgments

The authors express their appreciation to Pieter Cuijpers, Murdoch Gabbay, and Ana Sokolova for their comments on early versions of this article. Valuable comments of the anonymous referees of the LICS Symposium and *Information and Computation* journal are also acknowledged.

References

[1] L. Aceto, B. Bloom, F.W. Vaandrager, Turning SOS rules into equations, Information and Computation (I & C) 111 (1) (1994) 1–52.

[2] L. Aceto, W.J. Fokkink, C. Verhoef, Structural operational semantics, in: J.A. Bergstra, A. Ponse, S.A. Smolka (Eds.), Handbook of process algebra, Elsevier Science, Dordrecht, The Netherlands, 2001, pp. 197–292, Chapter 3.
[3] J.C.M. Baeten, E.P. de Vink, Axiomatizing GSOS with termination, Journal of Logic and Algebraic Programming (JLAP) 60–61 (2004) 323–351.
[4] J.C.M. Baeten, C.A. Middelburg, Process algebra with timing, EATCS monographs, Springer-Verlag, Berlin, Germany, 2002.
[5] J.C.M. Baeten, C. Verhoef, A congruence theorem for structured operational semantics with predicates, in: E. Best (Ed.), International conference on concurrency theory (CONCUR'93),Hildesheim, Germany, Lecture notes in computer science, vol. 715, Springer-Verlag, Berlin, Germany, 1993, pp. 477–492.
[6] J.A. Bergstra, C.A. Middelburg, Process algebra for hybrid systems, Theoretical Computer Science (to appear).
[7] K.L. Bernstein, A congruence theorem for structured operational semantics of higher-order languages, in: Symposium on logic in computer science (LICS'98), IEEE Computer Society, Los Alamitos, CA, USA, 1998, pp. 153–164.
[8] B. Bloom, S. Istrail, A.R. Meyer, Bisimulation can't be traced, Journal of the ACM (JACM) 42 (1) (1995) 232–268.
[9] B. Bloom, F.W. Vaandrager, SOS rule formats for parameterized and state-bearing processes (draft), unpublished note. Available from: <http://www.cs.ru.nl/ita/publications/papers/fvaan/>.
[10] V. Bos, J.J. Kleijn, Redesign of a systems engineering language—formalisation of χ, Formal Aspects of Computing 15 (4) (2003) 370–389.
[11] A. Brogi, J.-M. Jacquet, On the expressiveness of Linda-like concurrent languages, in: I. Castellani, C. Palamidessi (Eds.), Proceedings of the fifth international workshop on expressiveness in concurrency (EXPRESS'98), vol. 16, Electronic Notes in Theoretical Computer Science, Elsevier Science, Dordrecht, The Netherlands, 1998.
[12] 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.
[13] P.J.L. Cuijpers, M.A. Reniers, Action and predicate safety of hybrid processes, Tech. Rep. CSR-04-10, Department of Computer Science, Eindhoven University of Technology, Eindhoven, The Netherlands, 2004.
[14] P.J.L. Cuijpers, M.A. Reniers, Hybrid process algebra, Journal of Logic and Algebraic Programming (JLAP) 62 (2) (2005) 191–245.
[15] P.J.L. Cuijpers, M.A. Reniers, Hybrid process algebra, Tech. Rep. CSR-03-07, Department of Computer Science, Eindhoven University of Technology, Eindhoven, The Netherlands, 2003.
[16] J.W. de Bakker, E.P. de Vink, Control flow semantics, in: Foundations of Computing series, The MIT Press, Cambridge, 1996.
[17] R. De Simone, Higher-level synchronizing devices in MEIJE-SCCS, Theoretical Computer Science (TCS) 37 (1985) 245–267.
[18] W. Ferreira, M. Hennessy, A behavioural theory of first-order CML, Theoretical Computer Science (TCS) 216 (1–2) (1999) 55–107.
[19] W.J. Fokkink, R.J. van Glabbeek, Ntyft/ntyxt rules reduce to ntree rules, Information and Computation (I & C) 126 (1) (1996) 1–10.
[20] J.F. Groote, Transition system specifications with negative premises, Theoretical Computer Science (TCS) 118 (2) (1993) 263–299.
[21] 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).
[22] J.F. Groote, A. Ponse, Process algebra with guards: combining hoare logic with process algebra, Formal Aspects of Computing (FAC) 6 (2) (1994) 115–164.
[23] J.F. Groote, F.W. Vaandrager, Structured operational semantics and bisimulation as a congruence, Information and Computation 100 (2) (1992) 202–260.
[24] J.-M. Jacquet, K. de Bosschere, A. Brogi, On timed coordination languages, in: A. Porto, G.-C. Roman (Eds.), Proceedings of coordination languages and models, 4th international conference, limassol, cyprus, Lecture notes in computer science, vol. 1906, Springer-Verlag, Berlin, Germany, 2000, pp. 81–98.
[25] C.A. Middelburg, Variable binding operators in transition system specifications, Journal of Logic and Algebraic Programming (JLAP) 47 (1) (2001) 15–45.
[26] R.A. Milner, A Calculus of Communicating Systems, vol. 92, Lecture Notes in Computer Science, Springer-Verlag, Berlin, 1980.

[27] M.R. Mousavi, T. Basten, M.A. Reniers, M.R.V. Chaudron, G. Russello, Separating functionality, behavior and timing in the design of reactive systems: (GAMMA+coordination)+time, Tech. Rep. CSR-02-09, Department of Computer Science, Eindhoven University of Technology, Eindhoven, The Netherlands (2002).
[28] M.R. Mousavi, M.A. Reniers, J.F. Groote, Congruence for SOS with data, in: Proceedings of the nineteenth annual IEEE symposium on logic in computer science (LICS'04), IEEE Computer Society Press, Turku, Finland, 2004, pp. 302–313.
[29] D.M. Park, Concurrency and automata on infinite sequences, in: Proceedings of 5th GI conference, Lecture notes in computer science, vol. 104, Springer-Verlag, Berlin, Germany, 1981, pp. 167–183.
[30] G.D. Plotkin, A structural approach to operational semantics, Tech. Rep. DAIMI FN-19, Computer Science Department, Aarhus University, Aarhus, Denmark (September 1981).
[31] G.D. Plotkin, The origins of structural operational semantics, Journal of Logic and Algebraic Programming (JLAP) 60–61 (2004) 3–15.
[32] G.D. Plotkin, A structural approach to operational semantics, Journal of Logic and Algebraic Programming (JLAP) 60–61 (2004) 17–139.
[33] M.A. Reniers, J.F. Groote, M.B. van der Zwaag, J. van Wamel, Completeness of timed μCRL, Fundamenta Informaticae 50 (3–4) (2002) 361–402.
[34] A. Rensink, Bisimilarity of open terms, Information and Computation (I&C) 156 (1–2) (2000) 345–385.
[35] J.J.M.M. Rutten, Universal coalgebra: a theory of systems, Theoretical Computer Science (TCS) 249 (1) (2000) 3–80.
[36] B. Thomsen, Plain CHOCS a second generation calculus for higher order processes, Acta Informatica 30 (1) (1993) 1–59.
[37] D. Turi, Functorial operational semanticsand its denotational dual, PhD thesis, Free University of Amsterdam(1996).
[38] D. Turi, G.D. Plotkin, Towards a mathematical operational semantics, in: Proceedings of 12th annual IEEE symposium on logic in computer science (LICS'97), IEEE Computer Society Press, 1997, pp. 280–291.
[39] C. Verhoef, A congruence theorem for structured operational semantics with predicates and negative premises, Nordic Journal of Computing 2 (2) (1995) 274–302.
[40] R. van Glabbeek, The meaning of negative premises in transition system specifications II, Journal of Logic and Algebraic Programming (JLAP) 60–61 (2004) 229–258.