Precongruence Formats for Decorated Trace Preorders

Bard Bloom
IBM T.J. Watson Research Center
Hawthorne, NY 10532, USA
bardb@us.ibm.com

Wan Fokkink*
CWI
PO Box 94079
1090 GB Amsterdam, NL
wan@cwi.nl

Rob van Glabbeek
Computer Science Department
Stanford University
Stanford, CA 94305-9045, USA
rvg@cs.stanford.edu

Abstract

This paper explores the connection between semantic equivalences and preorders for concrete sequential processes, represented by means of labelled transition systems, and formats of transition system specifications using Plotkin's structural approach. For several preorders in the linear time – branching time spectrum a format is given, as general as possible, such that this preorder is a precongruence for all operators specifiable in that format. The formats are derived using the modal characterizations of the corresponding preorders.

1. Introduction

Structural operational semantics [24] provides process algebras and specification languages with an interpretation. It generates a *(labelled) transition system* in which states are the closed terms over a (single-sorted, first-order) signature, and transitions between states may be supplied with labels. The transition relation is obtained from a *transition system specification* (TSS), this being a set of proof rules called *transition rules*.

In case of a TSS with only positive premises, the associated transition relation simply consists of the transitions derivable from the transition rules. In the presence of negative premises it is not always straightforward to associate a transition relation to a TSS. One can for instance express that a transition holds if it does not hold. In VAN GLABBEEK [17] the notion of derivability of transitions from a TSS is extended to negated transitions by incorporating a notion of *negation as failure* (cf. [10]): a *well-supported proof* features a method to derive the negation of a transition by demonstrating that that transition cannot be derived. A TSS is *complete* [17] if for each transition there is a well-supported proof from the TSS either for the transition itself or for its negation. The transition relation associated to a complete TSS consists of the transitions for which there is a well-supported proof. An incomplete TSS arguably does not specify a transition relation in a meaningful way at all. However, it specifies what one could call a *3-valued* transition relation, in which potential transitions are either present, absent or unknown. This approach to the meaning of transition system specifications can be seen as a proof-theoretic characterization of the work of VAN GELDER, ROSS & SCHLIPF [14] in logic programming.

A wide range of semantic equivalences and preorders has been defined on 2-valued transition relations. These preorders are based on the branching structure of processes (ready simulation [7], bisimulation [21]), on execution sequences (partial traces, completed traces, accepting traces), or on decorated versions of execution sequences (ready pairs [22], failure pairs [9, 11], ready traces [2, 25], failure traces [23]). In [15], VAN GLABBEEK classified most equivalences and preorders for concrete, sequential processes[1] that occur in the literature, and motivated them by means of testing scenarios, phrased in terms of 'button pushing experiments' on generative and reactive machines. This gave rise to *modal characterizations* of the preorders, i.e. characterizations in terms of the observations that an experimenter could make during a session with a process. These will here be taken as the definitions of the preorders.

In general a semantic equivalence (or preorder) induced by a TSS is not a congruence (resp. precongruence), i.e. the equivalence class of a term $f(t_1, \ldots, t_n)$ need not be determined by the equivalence classes of its arguments $t_1, \ldots, t_n$. Being a (pre)congruence is an important property, for instance in order to fit the equivalence (or preorder) into an axiomatic framework. Syntactic formats for TSSs have been developed w.r.t. several semantic equivalences and preorders, to ensure that such an equivalence or preorder as induced by a TSS in the corresponding format is a (pre)congruence. These formats have helped

*Supported by a grant from The Nuffield Foundation.

[1] A process is *sequential* if it can do only one action at a time; *concrete* refers to the absence of internal actions or internal choice.

to avoid repetitive (pre)congruence proofs, and to explore the limits of sensible TSS definitions. A first congruence format for bisimulation equivalence was put forward by DE SIMONE [26], which was extended to the GSOS format by BLOOM, ISTRAIL & MEYER [7] and to the tyft/tyxt format by GROOTE & VAANDRAGER [19]. (The latter format was supplied with a well-foundedness criterion, which was subsequently eliminated [13].) The tyft/tyxt format was extended with negative premises [8, 18] to obtain the so-called ntyft/ntyxt format, which works for the class of complete TSSs. The ntyft/ntyxt format generalizes the GSOS format. To mention some formats for other equivalences and preorders, VAANDRAGER [27] observed that de Simone's format is a precongruence format for the partial trace and the failure preorders, BLOOM [5] introduced a more general congruence format for partial trace equivalence, and FOKKINK [12] introduced a precongruence format for the accepting trace preorder. Finally, VAN GLABBEEK [16] introduced a congruence format for ready simulation equivalence, which generalizes the GSOS format.

In this paper precongruence formats are proposed for several semantic preorders based on decorated traces, building on results reported in [4, 16]. We introduce precongruence formats for the ready trace preorder, the readiness preorder, the failure trace preorder and the failure preorder. The precongruence formats for the last two preorders coincide. Following [4, 12] these three precongruence formats distinguish between 'frozen' and 'liquid' arguments of function symbols. This distinction is used in posing restrictions on occurrences of variables in transition rules. The ready simulation format of [16] is more liberal than the format for the ready trace preorder, which is more liberal than the format for the readiness preorder, which is more liberal than the format for the failure trace preorder, which in turn is more liberal than de Simone's format.

The three precongruence formats introduced in this paper apply to incomplete TSSs as well. For this purpose the definitions of the corresponding preorders are extended to 3-valued transition relations.

The precongruence formats put forward in this paper were obtained by a careful study of the modal characterizations of the preorders in question. The outline of the proof underlying each of our precongruence results is as follows (detailed proofs can be found in [6]). First, any TSS in the ready simulation format is transformed into an equivalent TSS—equivalent in the sense that is proves the same transitions and negated transitions—of a special form, in which the left-hand sides of positive premises are single variables. Next, any such TSS is extended with a number of transition rules with negative conclusions, in such a way that a (negated) transition has a well-supported proof from the original TSS if and only if it has a standard proof from the extended TSS. In the extended TSS, the left-hand sides of positive and negative premises can further be reduced to single variables. It is shown for each of the precongruence formats in this paper that its syntactic criteria are preserved under these transformations. Finally, the resulting transition rules are captured by means of logical formulas that are within the modal characterization of the preorder in question. This implies the desired precongruence result.

This paper is set up as follows. Section 2 presents the basics of structural operational semantics and defines the preorders based on decorated traces w.r.t. 3-valued transition relations. Section 3 recalls the ready simulation format of [16] and formulates extra requirements to obtain new precongruence formats for the decorated trace preorders. Section 4 sketches the proofs of the precongruence results, using the definitions of preorders in terms of observations. Section 5 contains full abstraction properties of the various precongruence formats: for each format we determine the coarsest congruence with respect to all operators in that format that is finer than partial or completed trace equivalence. In Section 6, counterexamples are given to show that all syntactic restrictions are essential for the obtained precongruence results. Finally, Section 7 presents some applications of the precongruence formats to TSSs from the literature.

The reader is referred to [6] for detailed proofs and more applications.

2. Preliminaries

V and A are two sets of *variables* and *actions*. In our proofs (see [6]) it is used that V is infinite and at least at large as A (i.e. $|V| \geq |A|$). Many concepts that will appear later on are parameterized by the choice of V and A, but as in this paper this choice is fixed, a corresponding index is suppressed. A syntactic object is *closed* if it does not contain any variables from V.

Definition 1 A *signature* is a collection Σ of *function symbols* $f \notin V$, with $|\Sigma| \leq |V|$, equipped with a function $ar : \Sigma \to \mathbb{N}$. The set $\mathbb{T}(\Sigma)$ of *terms* over a signature Σ is defined recursively by:

- $V \subseteq \mathbb{T}(\Sigma)$,
- $f(t_1, \ldots, t_{ar(f)}) \in \mathbb{T}(\Sigma)$ for all functions symbols $f \in \Sigma$ and terms $t_1, \ldots, t_{ar(f)} \in \mathbb{T}(\Sigma)$.

A term $c()$ is abbreviated as c. For $t \in \mathbb{T}(\Sigma)$, $var(t)$ denotes the set of variables that occur in t. $T(\Sigma)$ is

the set of closed terms over Σ, i.e. the terms $t \in \mathbb{T}(\Sigma)$ with $var(t) = \emptyset$. A Σ-*substitution* σ is a partial function from V to $\mathbb{T}(\Sigma)$. If σ is a substitution and S is any syntactic object, then $\sigma(S)$ denotes the object obtained from S by replacing, for x in the domain of σ, every occurrence of x in S by $\sigma(x)$. In that case $\sigma(S)$ is called a *substitution instance* of S. A Σ-substitution is *closed* if it is a total function from V to $T(\Sigma)$.

Definition 2 Let Σ be a signature. A *positive* Σ-*literal* is an expression $t \xrightarrow{a} t'$ and a *negative* Σ-*literal* an expression $t \xrightarrow{a}\!\!\!\!\!\!/\;\;$ with $t, t' \in \mathbb{T}(\Sigma)$ and $a \in A$. For $t, t' \in \mathbb{T}(\Sigma)$ the literals $t \xrightarrow{a} t'$ and $t \xrightarrow{a}\!\!\!\!\!\!/\;\;$ are said to *deny* each other. A *transition rule* over Σ is an expression of the form $\frac{H}{\alpha}$ with H a set of Σ-literals (the *premises* of the the rule) and α a Σ-literal (the *conclusion*). The left- and right-hand side (if any) of α are called the *source* and the *target* of the rule, respectively. A rule $\frac{H}{\alpha}$ with $H = \emptyset$ is also written α. A *transition system specification (TSS)* over Σ is a collection of transition rules over Σ. A TSS is *standard* if all its rules have positive conclusions, and *positive* if moreover all premises of its rules are positive.

The concept of a positive TSS was introduced by Groote & Vaandrager [19]; negative premises were added by Groote [18]. The resulting notion constitutes the first formalization of Plotkin's *structural operational semantics (SOS)* [24] that is sufficiently general to cover most of its applications. TSSs with negative conclusions are introduced here, because they are needed as intermediate steps in our proofs for standard TSSs.

The following definition tells when a literal is provable from a TSS. It generalizes the standard definition (see e.g. [19]) by allowing the derivation of transition rules. The derivation of a literal α corresponds to the derivation of the transition rule $\frac{H}{\alpha}$ with $H = \emptyset$. The case $H \neq \emptyset$ corresponds to the derivation of α under the assumptions H.

Definition 3 Let R be a TSS over a signature Σ. An *irredundant proof* of a transition rule $\frac{H}{\alpha}$ from R is a well-founded, upwardly branching tree of which the nodes are labelled by Σ-literals, and some of the leaves are marked "hypothesis", such that:

- the root is labelled by α,
- H is the set of labels of the hypotheses, and
- if β is the label of a node q which is not an hypothesis and K is the set of labels of the nodes directly above q, then $\frac{K}{\beta}$ is a substitution instance of a transition rule in R.

A *proof* of $\frac{K}{\alpha}$ from R is an irredundant proof of $\frac{H}{\alpha}$ from R with $H \subseteq K$. If an (irredundant) proof of $\frac{K}{\alpha}$ from R exists, then $\frac{K}{\alpha}$ is *(irredundantly) provable* from R, notation $R \vdash \frac{K}{\alpha}$ (resp. $R \vdash_{\mathrm{irr}} \frac{K}{\alpha}$).

The main purpose of a transition system specification (over a signature Σ) is to specify a transition relation (over Σ). Here a *transition relation* over Σ can be defined as a set of closed positive Σ-literals (*transitions*). A positive TSS specifies a transition relation in a straightforward way as the set of all provable transitions. But as pointed out by Groote [18], it is much less trivial to associate a transition relation to a TSS with negative premises; in particular there are TSSs that appear not to specify a transition relation in a meaningful way at all. In van Glabbeek [17] eleven answers to the questions "*Which TSSs are meaningful, and which transition relations do they specify?*" are reviewed. The "most general solution without undesirable properties" is due to van Gelder, Ross & Schlipf [14] in the setting of logic programming, and has been adapted to TSSs by Bol & Groote [8]. In [17] it has been reformulated in terms of completeness w.r.t. a notion of provability of closed literals that incorporates a form of *negation as failure*.

Definition 4 Let R be a standard TSS over a signature Σ. A *well-supported proof* of a closed literal α from R is a well-founded, upwardly branching tree of which the nodes are labelled by closed Σ-literals, such that:

- the root is labelled by α, and
- if β is the label of a node q and K is the set of labels of the nodes directly above q, then
  1. either $\frac{K}{\beta}$ is a closed substitution instance of a transition rule in R
  2. or β is negative and for every set N of negative closed literals such that $R \vdash \frac{N}{\gamma}$ for γ a closed literal denying β, a literal in K denies one in N.

α is *ws-provable* from R, notation $R \vdash_{ws} \alpha$, if a well-supported proof of α from R exists.

Note that the proof-steps 1 and 2 establish the validity of β when K is the set of literals established earlier. Step 2 allows to infer $t \xrightarrow{a}\!\!\!\!\!\!/\;\;$ whenever it is manifestly impossible to infer $t \xrightarrow{a} t'$ for some term t' (because every conceivable proof of $t \xrightarrow{a} t'$ involves a premise that has already been refuted). This practice is sometimes referred to as *negation as failure* [10].

Definition 5 A standard TSS R is *complete* if for any closed literal $t \xrightarrow{a}\!\!\!\!\!\!/\;\;$ either $R \vdash_{ws} t \xrightarrow{a} t'$ for some closed term t' or $R \vdash_{ws} t \xrightarrow{a}\!\!\!\!\!\!/\;\;$.

Now a standard TSS is meaningful, in the sense that it specifies a transition relation, iff it is complete. The specified transition relation is then the set of all *ws*-provable transitions.

In the present paper this solution is extended by considering all standard TSSs to be meaningful. However, following VAN GELDER, ROSS & SCHLIPF [14], the meaning of an incomplete TSS is now not given by a two-valued transition relation as defined above, but by a three-valued transition relation, in which a potential transition can be *true*, *false* or *unknown*. In fact, a slight abstraction of this notion will suffice, in which a transition relation is simply defined as a set of closed, positive and negative, literals.

Definition 6 Let Σ be a signature. A *(3-valued) transition relation* over Σ is a set of closed Σ-literals, not containing literals that deny each other. A transition relation $\rightarrow$ is *2-valued* if it satisfies

$$(t \stackrel{a}{\not\rightarrow}) \in \rightarrow \;\; \Leftrightarrow \;\; \forall t' \in T(\Sigma) : (t \stackrel{a}{\longrightarrow} t') \notin \rightarrow .$$

The transition relation *associated* to a standard TSS over Σ is the set of closed Σ-literals that are *ws*-provable from that TSS.

In [17] is has been shown that $\vdash_{ws}$ is consistent, in the sense that no standard TSS admits well-supported proofs of two literals that deny each other. Thus the transition relation associated to a standard TSS is indeed a transition relation as defined above. Note that if a standard TSS R is complete, its associated transition relation is 2-valued. This means that the negative literals in its associated transition relation are completely determined by the positive ones; hence the transition relation can be simply given by its positive part.

In the literature several preorders have been defined in terms of (2-valued) transition relations, and provided with modal characterizations. Below we employ these modal characterizations as the definitions of these preorders, and simultaneously extend these definitions to 3-valued transition relations.

Definition 7 Assume an action set A. The set $\mathbb{O}$ of *potential observations* or *modal formulas* is defined inductively by:

$\top \in \mathbb{O}$. The trivial observation, obtained by terminating the session.

$a\varphi \in \mathbb{O}$ if $\varphi \in \mathbb{O}$ and $a \in A$. The observation of an action a, followed by the observation φ.

$\widetilde{a} \in \mathbb{O}$ for $a \in A$. The observation that the process cannot perform the action a.

$\bigwedge_{i\in I} \varphi_i \in \mathbb{O}$ if $\varphi_i \in \mathbb{O}$ for all $i \in I$. The process admits each of the observations φ_i.

Definition 8 Let R be a standard TSS over a signature Σ. The *satisfaction relation* $\models_R \subseteq T(\Sigma) \times \mathbb{O}$, telling which observations are possible for which process, is inductively defined by the clauses below (in which $p, q \in T(\Sigma)$).

$$\begin{array}{ll} p \models_R \top & \\ p \models_R a\varphi & \text{if } \exists q : R \vdash_{ws} p \stackrel{a}{\longrightarrow} q \wedge q \models_R \varphi \\ p \models_R \widetilde{a} & \text{if } R \vdash_{ws} p \stackrel{a}{\not\rightarrow} \\ p \models_R \bigwedge_{i\in I} \varphi_i & \text{if } p \models_R \varphi_i \text{ for all } i \in I \end{array}$$

We will use the binary conjunction $\varphi_1 \wedge \varphi_2$ as an abbreviation of $\bigwedge_{i\in\{1,2\}} \varphi_i$, whereas $\top$ is identified with the empty conjunction. We identify formulas that are logically equivalent using the laws for conjunction $T \wedge \varphi \cong \varphi$ and $\bigwedge_{i\in I}(\bigwedge_{j\in J_i} a_{ij}) \cong \bigwedge_{k\in K} a_k$ where $K = \{ij \mid i \in I \wedge j \in J_i\}$. This is justified because $\varphi \cong \psi$ implies $p \models_R \varphi \Leftrightarrow p \models_R \psi$.

Definition 9 Below, several sublanguages of the set $\mathbb{O}$ of observations are defined.

$$\begin{array}{lll} \mathbb{O}_T & \varphi ::= \top \mid a\varphi' & \textit{(partial) trace} \text{ observations} \\ \mathbb{O}_{CT} & \varphi ::= \top \mid a\varphi' \mid \bigwedge_{a\in A} \widetilde{a} & \textit{completed trace} \text{ observations} \\ \mathbb{O}_F & \varphi ::= \top \mid a\varphi' \mid \bigwedge_{i\in I} \widetilde{a}_i & \textit{failure} \text{ observations} \\ \mathbb{O}_R & \varphi ::= \top \mid a\varphi' \mid \bigwedge_{i\in I} \widetilde{a}_i \wedge \bigwedge_{j\in J} b_j\top & \textit{readiness} \text{ observations} \\ \mathbb{O}_{FT} & \varphi ::= \top \mid a\varphi' \mid \bigwedge_{i\in I} \widetilde{a}_i \wedge \varphi' & \textit{failure trace} \text{ observations} \\ \mathbb{O}_{RT} & \varphi ::= \top \mid a\varphi' \mid \bigwedge_{i\in I} \widetilde{a}_i \wedge \bigwedge_{j\in J} b_j\top \wedge \varphi' & \textit{ready trace} \text{ observations} \end{array}$$

For each of these notions the *N-observations of* $p \in T(\Sigma)$ are given by $\mathcal{O}_N^R(p) := \{\varphi \in \mathbb{O}_N \mid p \models_R \varphi\}$. The *N-preorder induced by* R is defined by $p \sqsubseteq_N^R q$ if $\mathcal{O}_N^R(p) \subseteq \mathcal{O}_N^R(q)$. When clear from the context, the superscript R will be omitted. In fact a slight reformulation of this definition will be needed in this paper.

Definition 10 For $N \in \{T, CT, F, R, FT, RT\}$ let $\mathbb{O}_N^\wedge$ consist of all formulas $\bigwedge_{i\in I} \varphi_i$ with $\varphi_i \in \mathbb{O}_N$. Let $\mathcal{O}_N^\wedge(p) := \{\varphi \in \mathbb{O}_N^\wedge \mid p \models \varphi\}$.

Clearly $\mathcal{O}_N(p) \subseteq \mathcal{O}_N(q) \Leftrightarrow \mathcal{O}_N^\wedge(p) \subseteq \mathcal{O}_N^\wedge(q)$.

Proposition 1 For $N \in \{T, CT, F, R, FT, RT\}$ we have $p \sqsubseteq_N q$ iff $\mathcal{O}_N^\wedge(p) \subseteq \mathcal{O}_N^\wedge(q)$.

Definition 11 Let Σ be a signature. A preorder $\sqsubseteq$ on $T(\Sigma)$ is a *precongruence* if for all $f \in \Sigma$

$$\begin{array}{rl} & p_i \sqsubseteq q_i \text{ for } i = 1, \ldots, ar(f) \\ \Rightarrow & f(p_1, \ldots, p_{ar(f)}) \sqsubseteq f(q_1, \ldots, q_{ar(f)}). \end{array}$$

This is equivalent to the requirement that for all $t \in \mathbb{T}(\Sigma)$ and closed substitutions $\sigma, \sigma' : V \rightarrow T(\Sigma)$

$$\sigma(x) \sqsubseteq \sigma'(x) \text{ for } x \in var(t) \;\Rightarrow\; \sigma(t) \sqsubseteq \sigma'(t).$$

For every preorder $\sqsubseteq_N$ defined above there exists an associated equivalence $=_N$ (the *kernel* of $\sqsubseteq_N$) given by $p =_N q$ iff $p \sqsubseteq_N q \wedge q \sqsubseteq_N p$. Obviously $p =_N q$ iff $\mathcal{O}_N(p) = \mathcal{O}_N(p)$ iff $\mathcal{O}_N^\wedge(p) = \mathcal{O}_N^\wedge(p)$. In case a relation is an equivalence as well as a precongruence, it is called a *congruence*. Note that if $\sqsubseteq_N$ is a precongruence, its kernel is a congruence. Thus by establishing precongruence results for the preorders $\sqsubseteq_N$, we also obtain congruence results for the associated equivalences $=_N$.

3. Precongruence formats

In this section we define the formats for TSSs that play a rôle in this paper and state the precongruence results that we have established.

Definition 12 An *ntytt rule* is a transition rule in which the right-hand sides of positive premises are variables that are all distinct, and that do not occur in the source. An ntytt rule is an *ntyxt rule* if its source is a variable, and an *ntyft rule* if its source contains exactly one function symbol and no multiple occurrences of variables. An ntytt rule (resp. ntyft rule) is an *nxytt rule* (resp. *nxyft rule*) if the left-hand sides of its premises are variables.

Definition 13 A transition rule has *no lookahead* if the variables occurring in the right-hand sides of its positive premises do not occur in the left-hand sides of its premises. A variable occurring in a transition rule is *free* if it does not occur in the source nor in the right-hand sides of the positive premises of this rule. We say that a transition rule is *decent* if it has no lookahead and does not contain free variables.

Each combination of syntactic restrictions on transition rules induces a corresponding syntactic format for TSSs of the same name. For instance, a TSS is in *decent ntyft* format if it consists of decent ntyft rules. We proceed to define further syntactic formats for TSSs.

Definition 14 A TSS is in *ntyft/ntyxt format* if it contains only ntyft and ntyxt rules. A TSS is in *ready simulation format* if it is in ntyft/ntyxt format and its transition rules have no lookahead.

Definition 15 An occurrence of a variable in an ntytt rule is *propagated* if the occurrence is either in the target, or in the left-hand side of a positive premise whose right-hand side occurs in the target. An occurrence of a variable in an ntytt rule is *polled* if the occurrence is in the left-hand side of a premise that does not have a right-hand side occurring in the target.

Consider for instance the transition rules of Example 2 in Section 6. In the second rule both occurrences of x in the premises are propagated, i.e. the variable x is propagated twice. In the third rule the variables x_1 and x_2 are polled once each. We can think of a process, represented by a variable in a transition rule, as being *copied* if the variable is propagated more than once. The process is *tested* if the variable is either propagated or polled.

Our precongruence formats operate by keeping track of which variables represent running processes, and which do not. For example, it is semantically reasonable to copy a process before it starts, effectively getting information about all the conjuncts in a $\phi \in \mathbb{O}^\wedge$. However, copying a running process would give information about the branching structure of the process, which is incompatible with any form of decorated trace semantics. We introduce a predicate Λ as the basis for determining the Λ-*floating* variables, which represent processes that may be running.

Definition 16 Let Σ be a signature, and Λ a unary predicate on $\{(f,i) \mid 1 \leq i \leq ar(f),\ f \in \Sigma\}$. If $\Lambda(f,i)$, then we say that argument i of f is *liquid*; otherwise it is *frozen*. An occurrence of a variable x in a term $t \in \mathbb{T}(\Sigma)$ is Λ-*liquid* if either $t = x$, or $t = f(t_1, \ldots, t_{ar(f)})$ and the occurrence of x is Λ-liquid in t_i for some liquid argument i of f. A variable in an ntytt rule over Σ is Λ-*floating* if either it occurs as the right-hand side of a positive premise, or it occurs exactly once in the source, at a Λ-liquid position.

Note that an occurrence of a variable x in a term $t \in \mathbb{T}(\Sigma)$ is Λ-liquid iff t does not contain a subterm $f(t_1, \ldots, t_{ar(f)})$ such that the occurrence of x is in t_i for a frozen argument i of f.

Definition 17 Let Λ be a unary predicate on arguments of function symbols. A standard ntytt rule is Λ-*ready trace safe* if

- it has no lookahead, and
- each Λ-floating variable is propagated at most once, and at a Λ-liquid position.

The rule is Λ-*readiness safe* if

- it is Λ-ready trace safe, and
- each Λ-floating variable is not both propagated and polled.

The rule is Λ-*failure trace safe* if

- it is Λ-readiness safe, and
- each Λ-floating variable is polled at most once, at a Λ-liquid position in a positive premise.

The second restriction on "Λ-ready trace safe" guarantees that a running process is never copied, and continued to be marked as running after it has executed. The "Λ-readiness safe" restriction ensures that only at the end of its execution a running process is tested multiple times. The "Λ-failure trace safe" restriction further limits to a positive test on a single action.

Definition 18 A standard TSS is in *ready trace format* if it is in ntyft/ntyxt format and its rules are Λ-ready trace safe w.r.t. some Λ. A standard TSS is in *readiness format* if it is in ntyft/ntyxt format and its rules are Λ-readiness safe w.r.t. some Λ. A standard TSS is in *failure trace format* if it is in ntyft/ntyxt format and its rules are Λ-failure trace safe w.r.t. some Λ.

If a standard TSS R is in ready trace (resp. readiness or failure trace) format, then there is a smallest predicate Λ_0 such that the rules in R are Λ_0-ready trace (resp. Λ_0-readiness or Λ_0-failure trace) safe. In the context of the ready trace format, for instance, Λ_0 can be defined as the smallest predicate Λ such that for all rules of R each Λ-floating variable is propagated at Λ-liquid positions only. Now R is in ready trace format iff it has no lookahead and in all of its rules each Λ_0-floating variable is propagated at most once. Therefore, in the context of a given standard TSS and a given format, positions can be called *liquid* and variables *floating* without mentioning a specific predicate Λ; in such a case Λ_0 may be assumed.

Theorem 1 If a standard TSS is in ready trace format, then the ready trace preorder that it induces is a precongruence.

Theorem 2 If a standard TSS is in readiness format, then the readiness preorder that it induces is a precongruence.

Theorem 3 If a standard TSS is in failure trace format, then the failure trace and failure preorders that it induces are precongruences.

See [6] for detailed proofs, or the next section for an outline, of Theorems 1–3. Section 6 presents a series of counterexamples showing that the syntactic restrictions formulated above are essential for the claimed precongruence results. These counterexamples also help in motivating the definitions above.

For comparison with the literature we point out that a standard TSS is in GSOS format [7] iff it is in nxyft format and its transition rules have no lookahead. A standard TSS is in de Simone's format iff it is positive, in nxyft format, and its rules are Λ-failure trace safe with Λ the universal predicate (making all arguments of function symbols liquid).

4. Proof sketch

This section outlines the proofs of Theorems 1–3. The reader is referred to [6] for detailed proofs of the propositions in this section. The proofs use transition rules with negative conclusions. For this reason, the ready trace, readiness and failure trace formats need to be extended to non-standard TSSs.

Definition 19 Let Λ be a unary predicate on arguments of function symbols. A ntytt rule with a negative conclusion is Λ-*ready trace safe* or Λ-*readiness safe* if it has no lookahead. The rule is Λ-*failure trace safe* if

- it is Λ-readiness safe, and
- Λ-floating variables are polled only at Λ-liquid positions and only in negative premises.

Now Definition 18 applies to non-standard TSSs as well. Note that for Λ-ready trace and Λ-readiness safety the requirements are the same as in the standard case, for in a rule with a negative conclusion no variable is propagated. In the definition of Λ-failure trace safety, however, rules with positive and negative conclusions are treated differently.

Theorems 1–3 deal with preorders induced by standard TSSs through the notion of well-founded provability (of Definition 4). The next proposition states that without loss of generality we may use the classical notion of provability (of Definition 3) instead.

Proposition 2 Let R be a standard TSS in ready simulation format. Then there is a TSS R^+ in decent ntyft format such that $R^+ \vdash \alpha \Leftrightarrow R \vdash_{ws} \alpha$ for all closed literals α, and if R is in ready trace (resp. readiness or failure trace) format then so is R^+.

In general R^+ is not a standard TSS. It is for this reason that rules with a negative conclusion have been introduced in Definition 2, and that the precongruence formats were extended to non-standard TSSs in Definition 19.

Definition 20 Let R be a standard TSS in ready simulation format. An R-*ruloid* is a decent nxytt rule, irredundantly provable from R^+.

Proposition 3 Let R be a standard TSS in ready trace (resp. readiness or failure trace) format. All R-ruloids are Λ-ready trace (resp. Λ-readiness or Λ-failure trace) safe for some Λ.

The next proposition says, for R a standard TSS in decent ntyft format, that for any function symbol f there are a number of R-ruloids with source $f(x_1, \ldots, x_{ar(f)})$ that are "just right" for f. Here "just right" means

that for any (closed) literal $f(t_1,\ldots,t_{ar(f)}) \stackrel{a}{\not\longrightarrow}$ or $f(t_1,\ldots,t_{ar(f)}) \stackrel{a}{\longrightarrow} t'$ that is ws-provable from R there is a (closed) proof using one of those rules as the last step. Moreover, the same holds not only for function symbols f, but for arbitrary open terms. If for an open term t with $var(t) = \{x_1,\ldots,x_n\}$ we would introduce an n-ary function symbol f_t such that $f_t(x_1,\ldots,x_n)$ is just a shorthand for t, then there exists a collection of R-ruloids that is just right for f_t.

Proposition 4 Let R be a standard TSS in ready simulation format. Then $R \vdash_{ws} \sigma(t) \stackrel{a}{\not\longrightarrow}$ [respectively $R \vdash_{ws} \sigma(t) \stackrel{a}{\longrightarrow} t'$] for t a term, [t' a closed term] and σ a closed substitution, iff there are an R-ruloid $\frac{H}{t\stackrel{a}{\not\longrightarrow}}$ [resp. $\frac{H}{t\stackrel{a}{\longrightarrow}u}$] and a closed substitution σ' with $R \vdash_{ws} \sigma'(\alpha)$ for $\alpha \in H$, $\sigma'(t) = \sigma(t)$ [and $\sigma'(u) = t'$].

The following definition assigns to each term and each observation in $\mathbb{O}$ a collection of mappings from variables to $\mathbb{O}$. This construct will play a crucial rôle in the proof of Corollary 1. Intuitively, $t_R^{-1}(\varphi)$ consists of observational reformulations of the R-ruloids with source t that validate the observation φ for the term $\sigma(t)$, for any closed substitution σ, in terms of the observations that can be made for the terms $\sigma(x)$ for $x \in var(t)$.

Definition 21 Let Σ be a signature, and let R be a standard TSS over Σ in ready simulation format. Then $\cdot_R^{-1} : \mathbb{T}(\Sigma) \to (\mathbb{O} \to \mathcal{P}(V \to \mathbb{O}))$ is defined by:

- $t_R^{-1}(\top) = \{\psi\}$ with $\psi(x) = \top$ for $x \in V$.
- $\psi \in t_R^{-1}(\widetilde{a})$ iff there is an R-ruloid $\frac{H}{t\stackrel{a}{\not\longrightarrow}}$ and $\psi : V \to \mathbb{O}$ is given by

$$\psi(x) = \bigwedge_{(x\stackrel{b}{\not\longrightarrow})\in H} \widetilde{b} \;\wedge \bigwedge_{(x\stackrel{c}{\longrightarrow}y)\in H} c\top \quad \text{for } x \in var(t)$$
$$\psi(x) = \top \quad \text{for } x \notin var(t).$$

- $\psi \in t_R^{-1}(a\varphi)$ iff there are an R-ruloid $\frac{H}{t\stackrel{a}{\longrightarrow}u}$ and a $\chi \in u_R^{-1}(\varphi)$ and $\psi : V \to \mathbb{O}$ is given by

$$\psi(x) = \chi(x) \wedge \bigwedge_{(x\stackrel{b}{\not\longrightarrow})\in H} \widetilde{b} \;\wedge \bigwedge_{(x\stackrel{c}{\longrightarrow}y)\in H} c\chi(y)$$

  for $x \in var(t)$ and $\psi(x) = \top$ otherwise.
- $t_R^{-1}(\bigwedge_{i\in I}\varphi_i) = \{\bigwedge_{i\in I}\psi_i \mid \psi_i \in t_R^{-1}(\varphi_i) \text{ for } i \in I\}$.

The next proposition can be proved using Proposition 4.

Proposition 5 Let Σ be a signature, and let R be a standard TSS over Σ in ready simulation format. Let $\varphi \in \mathbb{O}$. For any term $t \in \mathbb{T}(\Sigma)$ and closed substitution $\sigma : V \to T(\Sigma)$ one has

$$\begin{array}{ll} & \sigma(t) \models_R \varphi \\ \Leftrightarrow & \exists \psi \in t_R^{-1}(\varphi)\ \forall x \in var(t) : \sigma(x) \models_R \psi(x). \end{array}$$

In order to arrive at the desired precongruence results we need to know that if a standard TSS is in the desired format N, and φ is a potential N-observation of a closed term $\sigma(t)$, then the observations of $\sigma(x)$ for $x \in var(t)$ that determine whether or not φ is an N-observation of $\sigma(t)$ are also N-observations. This is established in the following two propositions. In fact, our formats have been found by investigating what was needed to make these propositions hold.

The work is divided over two propositions. Proposition 6 deals with those variables of t that are floating in the R-ruloids with source t. As the proof inductively refers to terms that can be thought of as successors of t after performing a number of actions, and as these terms may contain variables y representing successors of the arguments of t after performing several actions, the observations employed should be the ones from Definition 9. Proposition 7 extends this to arbitrary variables. As non-floating variables represent processes in their initial state only, that proposition may use the richer language of observations employed in Definition 10, which is much less cumbersome. This enables the absence of restrictions on non-floating variables in the definitions of the formats.

Proposition 6 Let R be a standard TSS in ready simulation format, and let Λ be an unary predicate on arguments of function symbols. Let $t \in \mathbb{T}(\Sigma)$, $\varphi \in \mathbb{O}$, $\psi \in t_R^{-1}(\varphi)$ and $x \in var(t)$, such that x occurs only once in t, and at a Λ-liquid position.

- If the transition rules in R^+ are Λ-ready trace safe and $\varphi \in \mathbb{O}_{RT}$ then $\psi(x) \in \mathbb{O}_{RT}$.
- If the transition rules in R^+ are Λ-readiness safe and $\varphi \in \mathbb{O}_R$ then $\psi(x) \in \mathbb{O}_R$.
- If the transition rules in R^+ are Λ-failure trace safe and $\varphi \in \mathbb{O}_{FT}$ then $\psi(x) \in \mathbb{O}_{FT}$.
- If the transition rules in R^+ are Λ-failure trace safe and $\varphi \in \mathbb{O}_F$ then $\psi(x) \in \mathbb{O}_F$.

Proof: We only prove the last statement here. The other three can be proven in a similar fashion; see [6].

Let the transition rules in R^+ be Λ-failure trace safe and $\varphi \in \mathbb{O}_F$. We apply structural induction on φ. Take $t \in \mathbb{T}(t)$, $\psi \in t_R^{-1}(\varphi)$ and $x \in var(t)$, such that x occurs only once in t, and at a Λ-liquid position.

– In case $\varphi = \top$ we have $\psi(x) = \top \in \mathbb{O}_F$.
– Let $\varphi = \bigwedge_{i\in I} \widetilde{a}_i$. Then $\psi(x) = \bigwedge_{i\in I} \psi_i(x)$ where $\psi_i(x) \in t_R^{-1}(\widetilde{a}_i)$ for $i \in I$. For $i \in I$ there is an R-ruloid $\frac{H}{t \xrightarrow{a_i}}$ such that

$$\psi_i(x) = \bigwedge_{(x \overset{b}{\not\longrightarrow}) \in H} \widetilde{b} \;\wedge \bigwedge_{(x \xrightarrow{c} y) \in H} c\top.$$

By Proposition 3, H has no premises of the form $x \xrightarrow{c} y$. Therefore $\psi_i(x) \cong \bigwedge_{j\in J} \widetilde{b_j} \in \mathbb{O}_F$.
– Let $\varphi = a\varphi'$ with $\varphi' \in \mathbb{O}_F$. Then there are an R-ruloid $\frac{H}{t \xrightarrow{a} u}$ and $\chi \in u_R^{-1}(\varphi')$ such that

$$\psi(x) = \chi(x) \wedge \bigwedge_{(x \overset{b}{\not\longrightarrow}) \in H} \widetilde{b} \;\wedge \bigwedge_{(x \xrightarrow{c} y) \in H} c\chi(y).$$

By Proposition 3, x is propagated at most once in $\frac{H}{t \xrightarrow{a} u}$, and only at a Λ-liquid position. Moreover, x is not both propagated and polled in $\frac{H}{t \xrightarrow{a} u}$. We consider three cases.
* Suppose $x \in var(u)$. Then x is propagated, so it occurs only once in u, and at a Λ-liquid position. By induction $\chi(x) \in \mathbb{O}_F$. Furthermore, H has no premises of the form $x \overset{b}{\not\longrightarrow}$ or $x \xrightarrow{c} y$. Hence $\psi(x) \cong \chi(x) \in \mathbb{O}_F$.
* Suppose x is propagated, but does not occur in u. Then $\chi(x) \cong \top$. Furthermore, H contains no premises of the form $x \overset{b}{\not\longrightarrow}$ and exactly one of the form $x \xrightarrow{c} y$, where y occurs in u. So $\psi(x) \cong c\chi(y)$. As y is Λ-floating in $\frac{H}{t \xrightarrow{a} u}$ and does not occur in t, Proposition 3 guarantees that y occurs only once in u, and at a Λ-liquid position. By induction $\chi(y) \in \mathbb{O}_F$, so $\psi(x) \in \mathbb{O}_F$.
* Suppose x is not propagated. Then $x \notin var(u)$ so $\chi(x) \cong \top$. By Proposition 3, x is polled at most once in $\frac{H}{t \xrightarrow{a} u}$, and in a positive premise. Hence H contains no literals of the from $x \overset{b}{\not\longrightarrow}$, and no more than one literal $x \xrightarrow{c} y$. If there is such a literal, y does not occur in u, so $\chi(y) \cong \top$. Hence $\psi(x) \cong \top \in \mathbb{O}_F$ or $\psi(x) \cong c\top \in \mathbb{O}_F$. □

Let N range over {ready trace, readiness, failure trace, failure}, where *failure format* means failure trace format.

Proposition 7 Let R be a standard TSS in N format, $t \in \mathbb{T}(\Sigma)$, $\varphi \in \mathbb{O}$, $\psi \in t_R^{-1}(\varphi)$ and $x \in var(t)$. If $\varphi \in \mathbb{O}_N^\wedge$ then $\psi(x) \in \mathbb{O}_N^\wedge$.

After Proposition 6, the proof of Proposition 7 is rather simple; see [6]. This is sufficient to prove Theorems 1–3. In the light of Definition 11 these theorems can be reformulated as in the following corollary.

Corollary 1 Let R be a standard TSS in N format, $t \in \mathbb{T}(\Sigma)$ and σ, σ' closed substitutions. If $\sigma(x) \sqsubseteq_N^R \sigma'(x)$ for $x \in var(t)$, then $\sigma(t) \sqsubseteq_N^R \sigma'(t)$.

Proof: By Proposition 1 it suffices to show that $\mathcal{O}_N^\wedge(\sigma(x)) \subseteq \mathcal{O}_N^\wedge(\sigma'(x))$ for all $x \in var(t)$ implies $\mathcal{O}_N(\sigma(t)) \subseteq \mathcal{O}_N(\sigma'(t))$. Suppose that, for $x \in var(t)$, $\mathcal{O}_N^\wedge(\sigma(x)) \subseteq \mathcal{O}_N^\wedge(\sigma'(x))$. Let $\varphi \in \mathcal{O}_N(\sigma(t))$, i.e. $\varphi \in \mathbb{O}_N$ and $\sigma(t) \models_R \varphi$. By Proposition 5

$$\exists \psi \in t_R^{-1}(\varphi) \forall x \in var(t) : \sigma(x) \models_R \psi(x).$$

By Proposition 7 $\forall x \in var(t) : \psi(x) \in \mathbb{O}_N^\wedge$. Thus $\forall x \in var(t) : \psi(x) \in \mathcal{O}_N^\wedge(\sigma(x)) \subseteq \mathcal{O}_N^\wedge(\sigma'(x))$. Hence $\forall x \in var(t) : \sigma'(x) \models_R \psi(x)$. So by Proposition 5 $\sigma'(t) \models_R \varphi$. It follows that $\varphi \in \mathcal{O}_N(\sigma'(t))$, which had to be proved. □

5. Full abstraction

We say that an equivalence on processes is *fully abstract* w.r.t. a syntactic format for TSSs and an equivalence $=_{obs}$ on processes if it is the coarsest congruence w.r.t. all operators specifiable by a TSS in that format that is finer than $=_{obs}$. The proofs of the following full abstraction results can be found in [6].

Theorem 4 Ready trace equivalence is fully abstract for the ready trace format and trace equivalence. Readiness equivalence is fully abstract for the readiness format and trace equivalence. Trace equivalence is fully abstract for the failure trace format and trace equivalence. Failure equivalence is fully abstract for the failure trace format and completed trace equivalence.

6. Counterexamples

This section presents a string of counterexamples of complete standard TSSs in ntyft/ntyxt format, to show that the syntactic restrictions of our precongruence formats are essential. In [19] a series of counterexamples can be found showing that the syntactic restrictions of the ntyft/ntyxt format are essential as well.

6.1. Basic process algebra

The examples in this section assume basic process algebra [3]. We assume a collection A of constants, called *atomic actions*, representing indivisible behaviour, and two special constants: the *deadlock* δ does not display any behaviour, while the *empty process* ε [28] terminates successfully. Basic process algebra moreover includes function symbols $_+_$ and $_\cdot_$ of arity two, called *alternative composition* and *sequential composition*, respectively. Intuitively, $t_1 + t_2$ executes either t_1 or t_2,

while $t_1 \cdot t_2$ first executes t_1 and upon successful termination executes t_2. These intuitions are made precise by means of the standard TSS for $\mathrm{BPA}_{\delta\varepsilon}$ presented in Table 1, where the label a ranges over the set A of atomic actions together with a special label $\surd$, representing successful termination. Note that this TSS is positive and in ready simulation format. It is not hard to check that the TSS is in failure trace format (and so by default in ready trace and readiness format), if we take at least the first argument of sequential composition to be liquid. In particular, in the first rule for sequential composition the floating variable y occurs in a liquid argument of the target, and in the second rule for sequential composition the floating variable x_1 is polled only once and not propagated.

Table 1. TSS for $\mathrm{BPA}_{\delta\varepsilon}$

$$a \xrightarrow{a} \varepsilon \ (a \neq \surd) \qquad \varepsilon \xrightarrow{\surd} \delta$$

$$\frac{x_1 \xrightarrow{a} y}{x_1 + x_2 \xrightarrow{a} y} \qquad \frac{x_2 \xrightarrow{a} y}{x_1 + x_2 \xrightarrow{a} y}$$

$$\frac{x_1 \xrightarrow{a} y}{x_1 \cdot x_2 \xrightarrow{a} y \cdot x_2} \ (a \neq \surd) \qquad \frac{x_1 \xrightarrow{\surd} y_1 \quad x_2 \xrightarrow{a} y_2}{x_1 \cdot x_2 \xrightarrow{a} y_2}$$

Terms $t_1 \cdot t_2$ are abbreviated to $t_1 t_2$. Brackets are used for disambiguation only, assuming associativity of $+$ and $\cdot$, and letting $\cdot$ bind stronger than $+$. In the remainder of this section we assume that $A = \{a, b, c, d\}$. Moreover, we assume unary function symbols f and h and a binary function symbol g.

6.2. Lookahead

The following counterexample shows that the ready trace format (and its more restrictive analogues) cannot allow lookahead.

Example 1 We extend $\mathrm{BPA}_{\delta\varepsilon}$ with the following rule, containing lookahead:

$$\frac{x \xrightarrow{b} y_1 \quad y_1 \xrightarrow{c} y_2}{f(x) \xrightarrow{a} \delta}$$

It is easy to see that $bd \sqsubseteq_{RT} bc + bd$ (so *a fortiori* $bd \sqsubseteq_N bc + bd$ for $N \in \{R, FT, F\}$). The empty trace is a completed trace of $f(bd)$ but not of $f(bc + bd)$ (as $f(bc + bd) \xrightarrow{a} \delta$). Hence, $f(bd) \not\sqsubseteq_{CT} f(bc + bd)$ (and *a fortiori* $f(bd) \not\sqsubseteq_N f(bc + bd)$ for $N \in \{RT, R, FT, F\}$).

6.3. Multiple propagations

The following counterexample shows that the ready trace format (and its more restrictive analogues) cannot allow a liquid argument of the source to be propagated more than once in the left-hand sides of the positive premises.

Example 2 Let the arguments of f and g be liquid. We extend $\mathrm{BPA}_{\delta\varepsilon}$ with the rules:

$$\frac{x \xrightarrow{a} y}{f(x) \xrightarrow{a} f(y)} \qquad \frac{x \xrightarrow{b} y_1 \quad x \xrightarrow{b} y_2}{f(x) \xrightarrow{b} g(y_1, y_2)}$$

$$\frac{x_1 \xrightarrow{c} y_1 \quad x_2 \xrightarrow{d} y_2}{g(x_1, x_2) \xrightarrow{d} \delta}$$

In the second rule, the liquid argument x of the source is propagated in the left-hand sides of both premises.

It is easy to see that $a(bc + bd) \sqsubseteq_{RT} abc + abd$ (so *a fortiori* $a(bc + bd) \sqsubseteq_N abc + abd$ for $N \in \{R, FT, F\}$). Note that abd is a trace of $f(a(bc + bd))$ (as $f(a(bc + bd)) \xrightarrow{a} f(bc + bd) \xrightarrow{b} g(c, d) \xrightarrow{d} \delta$), but not of $f(abc + abd)$. Hence, $f(a(bc + bd)) \not\sqsubseteq_T f(abc + abd)$ (and *a fortiori* $f(a(bc + bd)) \not\sqsubseteq_N f(abc + abd)$ for $N \in \{RT, R, FT, F\}$).

A similar example can be given to show that the ready trace format cannot allow a liquid argument of the source or a right-hand side of a positive premise to occur more than once in the target. Likewise, an example can be given to show that the ready trace format cannot allow a liquid argument of the source to be propagated in the left-hand side of a positive premise and at the same time to occur in the target.

6.4. Propagation at a non-liquid position

If in the example above the argument of f were defined to be frozen, then in the first rule the right-hand side y of the premise would occur in a non-liquid position in the target. This shows that the ready trace format cannot allow right-hand sides of positive premises to occur at non-liquid positions in the target. Variants of Example 2 show that the ready trace format cannot allow liquid arguments of the source to be propagated at non-liquid positions either.

Example 3 Replace the second rule in Example 2 by the two rules

$$\frac{h(x) \xrightarrow{b} y}{f(x) \xrightarrow{b} y} \qquad \frac{x \xrightarrow{b} y_1 \quad x \xrightarrow{b} y_2}{h(x) \xrightarrow{b} g(y_1, y_2)}$$

Taking the arguments of f en g liquid, but that of h frozen, the resulting TSS sins against the ready trace

format only in that in the first rule above the floating variable x is propagated at a non-liquid position, namely as argument of h. Clearly the same mishap as in Example 2 ensues. The same argument applies when replacing the second rule in Example 2 by the two rules

$$f(x) \xrightarrow{a} h(x) \qquad \frac{x \xrightarrow{b} y_1 \quad x \xrightarrow{b} y_2}{h(x) \xrightarrow{b} g(y_1, y_2)}$$

The examples above show that if a floating variable x is propagated in a term $f(x)$, then the argument of f should be classified as liquid. Furthermore, if x is propagated in a term $f(h(x))$, then *both* the argument of f and that of h should be classified as liquid. Namely, if only the argument of f would be liquid, a rule with conclusion $h(x) \xrightarrow{b} g(x,x)$ could have fatal consequences, and if only the argument of h would be liquid, a rule with conclusion $f(x) \xrightarrow{b} g(x,x)$ could be fatal. (It is left to the reader to fill in the details.) This justifies the definition of Λ-liquid in Section 3.

6.5. Propagation together with polling

The following counterexample shows that the readiness format cannot allow that a liquid argument of the source is both propagated and polled. (The TSS in this example is in a flawed congruence format for failure equivalence from [16].)

Example 4 Let the arguments of f and h be liquid. We extend $\mathrm{BPA}_{\delta\varepsilon}$ with the rules:

$$\frac{x \xrightarrow{a} y}{f(x) \xrightarrow{a} f(y)} \qquad \frac{x \xrightarrow{b} y}{f(x) \xrightarrow{b} h(x)}$$

$$\frac{x \xrightarrow{c} y}{h(x) \xrightarrow{c} h(y)} \qquad \frac{x \xrightarrow{d} y}{h(x) \xrightarrow{d} \delta}$$

In the second rule, the liquid argument x of the source is both propagated and polled.

It is not hard to see that $a(b + cd) + ac$ and $a(b + c) + acd$ are readiness and failure equivalent (but not ready trace or failure trace equivalent). Nevertheless, $abcd$ is a trace of $f(a(b + cd) + ac)$ (as $f(a(b + cd) + ac) \xrightarrow{a} f(b + cd) \xrightarrow{b} h(b + cd) \xrightarrow{c} h(d) \xrightarrow{d} \delta$), but not of $f(a(b + c) + acd)$. Hence, $f(a(b + cd) + ac)$ and $f(a(b + c) + acd)$ are not even trace equivalent.

It is easy to see that $a(b + c) + ab + ac$ and $ab + ac$ are failure trace and failure equivalent (but not ready trace or readiness equivalent). Note that abc is a trace of $f(a(b + c) + ab + ac)$ (as $f(a(b + c) + ab + ac) \xrightarrow{a} f(b + c) \xrightarrow{b} h(b + c) \xrightarrow{c} h(\varepsilon)$), but not of $f(ab + ac)$. Hence, $f(a(b + c) + ab + ac)$ and $f(ab + ac)$ are not even trace equivalent.

6.6. Multiple pollings

The following counterexample shows that the failure trace format cannot allow that a liquid argument of the source is polled more than once.

Example 5 Let the argument of f be liquid. We extend $\mathrm{BPA}_{\delta\varepsilon}$ with the rules:

$$\frac{x \xrightarrow{a} y}{f(x) \xrightarrow{a} f(y)} \qquad \frac{x \xrightarrow{b} y_1 \quad x \xrightarrow{c} y_2}{f(x) \xrightarrow{d} \delta}$$

In the second rule, the liquid argument x of the source is polled in the two positive premises.

We recall that $a(b + c) + ab + ac$ and $ab + ac$ are failure trace and failure equivalent. Nevertheless, ad is a trace of $f(a(b + c) + ab + ac)$ (as $f(a(b + c) + ab + ac) \xrightarrow{a} f(b + c) \xrightarrow{d} \delta$), but not of $f(ab + ac)$. Hence, $f(a(b + c) + ab + ac)$ and $f(ab + ac)$ are not even trace equivalent.

6.7. Polling at a non-liquid position

The following variant of Example 5 shows that the failure trace format cannot allow that a liquid argument of the source is polled at non-liquid positions.

Example 6 Let the argument of f be liquid and the argument of h be frozen. We extend $\mathrm{BPA}_{\delta\varepsilon}$ with the rules:

$$\frac{x \xrightarrow{a} y}{f(x) \xrightarrow{a} f(y)} \qquad \frac{h(x) \xrightarrow{b} y}{f(x) \xrightarrow{b} \delta} \qquad \frac{x \xrightarrow{b} y_1 \quad x \xrightarrow{c} y_2}{h(x) \xrightarrow{d} \delta}$$

In the second rule, the liquid argument x of the source is polled at a non-liquid position. Clearly the same mishap as in Example 5 ensues.

6.8. Polling in a negative premise

The following counterexample shows that the failure trace format cannot allow that a liquid argument of the source is polled in a negative premise.

Example 7 Let the argument of f be liquid. We extend $\mathrm{BPA}_{\delta\varepsilon}$ with the rules:

$$\frac{x \xrightarrow{a} y}{f(x) \xrightarrow{a} f(y)} \qquad \frac{x \not\xrightarrow{b}}{f(x) \xrightarrow{d} \delta} \qquad \frac{x \not\xrightarrow{c}}{f(x) \xrightarrow{d} \delta}$$

In the second and third rule, the liquid argument x of the source is polled in the negative premise.

We recall that $a(b + c) + ab + ac$ and $ab + ac$ are failure trace and failure equivalent. Note that a is a completed trace of $f(a(b + c) + ab + ac)$ (as $f(a(b + c) + ab + ac) \xrightarrow{a} f(b + c) \not\xrightarrow{d}$), but not of $f(ab + ac)$. Hence, $f(a(b + c) + ab + ac)$ and $f(ab + ac)$ are not even completed trace equivalent.

7. Applications

This section contains some applications of our precongruence formats to TSSs from the literature.

7.1. Priority

Priority [1] is a unary function symbol that assumes an ordering on atomic actions. The term $\Theta(t)$ executes the transitions of t, with the restriction that a transition $t \xrightarrow{a} t'$ only gives rise to a transition $\Theta(t) \xrightarrow{a} \Theta(t')$ if there does not exist a transition $t \xrightarrow{b} t''$ with $a < b$. This intuition is captured by the rule for the priority operator in Table 2, which is added to the TSS for $\mathrm{BPA}_{\delta\varepsilon}$ in Table 1. The resulting standard TSS is in ready simulation format.

Table 2. Transition rule for priority

$$\frac{x \xrightarrow{a} y \quad x \not\xrightarrow{b} \text{ for } a < b}{\Theta(x) \xrightarrow{a} \Theta(y)}$$

As the floating variable y in the rule for priority is propagated in $\Theta(y)$, the argument of Θ has to be liquid. The floating variable x is propagated only once (and trivially at a liquid position). Hence the TSS for $\mathrm{BPA}_{\delta\varepsilon}$ with priority is in ready trace format.

Corollary 2 The ready trace preorder is a precongruence w.r.t. $\mathrm{BPA}_{\delta\varepsilon}$ with priority.

The TSS for $\mathrm{BPA}_{\delta\varepsilon}$ with priority is not in readiness format. Namely, in the case of a non-trivial ordering on atomic actions, the floating variable x in the rule for priority is both propagated and polled. In general the readiness, failure trace and failure preorders are not precongruences w.r.t. $\mathrm{BPA}_{\delta\varepsilon}$ with priority.

7.2. Binary Kleene star

The *binary Kleene star* ${t_1}^* t_2$ [20] repeatedly executes t_1 until it executes t_2. This operational behaviour is captured by the rules in Table 3, which are added to the TSS for $\mathrm{BPA}_{\delta\varepsilon}$ in Table 1. The resulting positive TSS is in failure trace format if we take the first argument of sequential composition to be liquid and the first argument of the binary Kleene star to be frozen. Note that in the first rule for the binary Kleene star the floating variable y is propagated only once, at a liquid position, and not polled. Moreover in this rule the variable x_1, which is propagated twice, is non-floating.

Corollary 3 The ready trace, readiness, failure trace and failure preorders are precongruences w.r.t. $\mathrm{BPA}_{\delta\varepsilon}$ with the binary Kleene star.

Table 3. Transition rules for the binary Kleene star

$$\frac{x_1 \xrightarrow{a} y}{{x_1}^* x_2 \xrightarrow{a} y \cdot ({x_1}^* x_2)} \; (a \neq \surd) \qquad \frac{x_2 \xrightarrow{a} y}{{x_1}^* x_2 \xrightarrow{a} y}$$

The second rule for the binary Kleene star does not fit the congruence format for ready trace equivalence from [16]. Namely, in this rule the variables x_1 and y in the target are connected by the premise. Neither does this rule fit de Simone's format, due to the fact that x_1 occurs in the left-hand side of the premise and in the target.

7.3. Sequencing

Sequencing $t_1; t_2$ executes t_1 until it can do no further transitions, after which it starts executing t_2. Basic process algebra with sequencing contains the atomic actions in A, alternative composition and sequencing, while the deadlock and the empty process are fused to a single constant $\mathbf{0}$. The TSS for $\mathrm{BPA}^{;}_{\mathbf{0}}$ is presented in Table 4, where a and b range over A. This standard TSS is in ready simulation format. The rules for sequencing were taken from [5].

Table 4. TSS for $\mathrm{BPA}^{;}_{\mathbf{0}}$

$$a \xrightarrow{a} \mathbf{0}$$

$$\frac{x_1 \xrightarrow{a} y}{x_1 + x_2 \xrightarrow{a} y} \qquad \frac{x_2 \xrightarrow{a} y}{x_1 + x_2 \xrightarrow{a} y}$$

$$\frac{x_1 \xrightarrow{a} y}{x_1; x_2 \xrightarrow{a} y; x_2} \qquad \frac{x_1 \not\xrightarrow{a} \text{ for } a \in A \quad x_2 \xrightarrow{b} y}{x_1; x_2 \xrightarrow{b} y}$$

As the floating variable y in the first rule for sequencing is propagated in $y; x_2$, the first argument of sequencing has to be liquid. In the first rule for sequencing the floating variable x_1 is propagated only once (and trivially at a liquid position), and not polled. In the second rule for sequencing the floating variable x_1 is not propagated, while the floating variable y is propagated only once (and trivially at a liquid position), and not polled. Hence the TSS for $\mathrm{BPA}^{;}_{\mathbf{0}}$ is in readiness format.

Corollary 4 The ready trace and readiness preorders are precongruences w.r.t. $\mathrm{BPA}^{;}_{\mathbf{0}}$.

The TSS for $\mathrm{BPA}^{\dot{}}_{\mathbf{0}}$ is not in failure trace format. Namely, in the second rule for sequencing the floating variable x_1 is polled in negative premises. The failure preorder is not a precongruence w.r.t. $\mathrm{BPA}^{\dot{}}_{\mathbf{0}}$. However, the failure trace preorder does constitute a precongruence w.r.t. $\mathrm{BPA}^{\dot{}}_{\mathbf{0}}$. Hence sequencing is an example of an operator from the literature that preserves failure traces but that lies outside the scope of the failure trace format.

References

[1] J. Baeten, J. Bergstra & J.W. Klop (1986): *Syntax and defining equations for an interrupt mechanism in process algebra. Fundamenta Informaticae* IX, pp. 127–168.

[2] J. Baeten, J. Bergstra & J.W. Klop (1987): *Ready-trace semantics for concrete process algebra with the priority operator. The Computer Journal* 30, pp. 498–506.

[3] J. Bergstra & J.W. Klop (1984): *Process algebra for synchronous communication. Information and Control* 60, pp. 109–137.

[4] B. Bloom (1993): *Ready, set, go: structural operational semantics for linear-time process algebras.* Report TR 93-1372, Cornell University.

[5] B. Bloom (1994): *When is partial trace equivalence adequate? Formal Aspects of Computing* 6, pp. 317–338.

[6] B. Bloom, W. Fokkink & R. van Glabbeek (2000): *Full abstraction in structural operational semantics*, full version of the present paper. Available at `http://boole.stanford.edu/pub/winter.ps.gz`.

[7] B. Bloom, S. Istrail & A. Meyer (1995): *Bisimulation can't be traced, Journal of the ACM* 42, pp. 232–268.

[8] R. Bol & J.F. Groote (1996): *The meaning of negative premises in transition system specifications. Journal of the ACM* 43, pp. 863–914.

[9] S.D. Brookes, C.A.R. Hoare & A.W. Roscoe (1984): *A theory of communicating sequential processes. Journal of the ACM* 31, pp. 560–599.

[10] K. Clark (1978): *Negation as failure.* In H. Gallaire and J. Minker, eds., *Logic and Databases.* Plenum Press, New York.

[11] R. De Nicola & M. Hennessy (1984): *Testing equivalences for processes. Theoretical Computer Science* 34, pp. 83–133.

[12] W. Fokkink: *Language preorder as a precongruence. Theoretical Computer Science*, to appear.

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

[14] A. van Gelder, K. Ross & J. Schlipf (1991): *The well-founded semantics for general logic programs. Journal of the ACM* 38, pp. 620–650.

[15] R. van Glabbeek (1990): *The linear time – branching time spectrum (extended abstract).* In *Proceedings 1st Conference on Concurrency Theory*, LNCS 458, Springer, pp. 278–297.

[16] R. van Glabbeek (1993): *Full abstraction in structural operational semantics (extended abstract).* In *Proceedings 3rd Conference on Algebraic Methodology and Software Technology*, Workshops in Computing, Springer, pp. 77–84.

[17] R. van Glabbeek (1995): *The meaning of negative premises in transition system specifications II.* Report STAN-CS-TN-95-16, Department of Computer Science, Stanford University. Extended abstract in *Proceedings 23rd Colloquium on Automata, Languages and Programming*, LNCS 1099, Springer, 1996, pp. 502–513.

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

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

[20] S. Kleene (1956): *Representation of events in nerve nets and finite automata.* In *Automata Studies*, Princeton University Press, pp. 3–41.

[21] R. Milner (1989): *Communication and Concurrency.* Prentice Hall, Englewood Cliffs.

[22] E.-R. Olderog & C.A.R. Hoare (1986): *Specification-oriented semantics for communicating processes. Acta Informatica* 23, pp. 9–66.

[23] I. Phillips (1987): *Refusal testing. Theoretical Computer Science* 50(3), pp. 241–284.

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

[25] A. Pnueli (1985): *Linear and branching structures in the semantics and logics of reactive systems.* In *Proceedings 12th Colloquium on Automata, Languages and Programming*, LNCS 194, Springer, pp. 15–32.

[26] R. de Simone (1985): *Higher-level synchronising devices in* Meije-SCCS. *Theoretical Computer Science* 37, pp. 245–267.

[27] F. Vaandrager (1991): *On the relationship between process algebra and input/output automata (extended abstract).* In *Proceedings 6th Symposium on Logic in Computer Science*, IEEE Computer Society Press, pp. 387–398.

[28] J. Vrancken (1997): *The algebra of communicating processes with empty process. Theoretical Computer Science* 177, pp. 287–328.