Divide and Congruence II: Delay and Weak Bisimilarity

Wan Fokkink
Vrije Universiteit Amsterdam, The Netherlands
w.j.fokkink@vu.nl

Rob van Glabbeek
NICTA/Data61, CSIRO, Sydney, Australia
University of New South Wales, Sydney, Australia
rvg@cs.stanford.edu

Abstract

Earlier we presented a method to decompose modal formulas for processes with the internal action τ; congruence formats for branching and η-bisimilarity were derived on the basis of this decomposition method. The idea is that a congruence format for a semantics must ensure that formulas in the modal characterisation of this semantics are always decomposed into formulas in this modal characterisation. Here the decomposition method is enhanced to deal with modal characterisations that contain a modality $\langle\epsilon\rangle\langle a\rangle\varphi$, to derive congruence formats for delay and weak bisimilarity.

1. Introduction

In [2] a method was developed to generate congruence formats for (concrete) process semantics from their modal characterisation. It crosses the borders between process algebra, structural operational semantics, process semantics, and modal logic. Cornerstone is the work in [18] to decompose formulas from Hennessy-Milner logic [17] with respect to a structural operational semantics in the De Simone format [21]. It was extended to the ntyft format [15] without lookahead in [2], and to the tyft format [16] in [8].

An equivalence is a congruence for a term algebra if the equivalence class of any term $f(p_1,\ldots,p_n)$ is determined by the equivalence classes of $p_1,\ldots,p_n$. Being a congruence is an important property, e.g. to fit a process semantics into an axiomatic framework. Syntactic formats for structural operational semantics have been developed for several process semantics, to ensure that such a semantics is a congruence; notably for unrooted and rooted weak bisimilarity in [1] and for unrooted and rooted delay bisimilarity in [14].

Key idea in [2] is that a congruence format for a process semantics must ensure that the formulas in a modal characterisation of this semantics are always decomposed into formulas that are again in this modal characterisation. This yielded liberal and elegant congruence formats for all known concrete process semantics in a convenient way. In [9] this method was extended to weak process semantics, which take into account the internal action τ. As a result, congruence formats for rooted branching and η-bisimilarity were derived. Two predicates $\aleph$ and Λ on arguments of function symbols are used: $\aleph$ marks processes that can execute immediately, and Λ processes that have started executing. Formats for unrooted branching and η-bisimilarity are obtained by imposing one restriction on top of the format for the corresponding rooted semantics: Λ holds universally.

The framework from [9] covers only a small part of the spectrum of weak semantics [12]. In particular, it does not readily extend to delay and weak bisimilarity [19, 20]. The reason is that in these semantics, in contrast to branching and η-bisimilarity, a process q that mimics an a-transition from a process p, does not need to be related to p at the moment that q performs the a-transition. This implies that in the modal characterisation of delay and weak bisimilarity, a modality $\langle a\rangle\varphi$ stating that an a-transition to a process where φ holds, is always preceded by a modality $\langle\epsilon\rangle$ allowing any number of τ-transitions. As a consequence, devising congruence formats for delay and weak bisimilarity is notoriously difficult, see e.g. [1, 14]. Here we show how this technical obstacle can be overcome by the semantic notion *delay resistance*, which ensures that modalities $\langle\epsilon\rangle\langle a\rangle\varphi$ are decomposed into formulas that again have this form. Thus congruence formats can be derived for semantics with a modal characterisation containing such modalities. We derive congruence formats for rooted delay and weak bisimilarity. Congruence formats for the unrooted counterparts of these semantics are again obtained by the extra requirement that Λ is universal.

We moreover provide syntactic restrictions which imply delay resistance, leading to the first entirely syntactic congruence formats for rooted delay and weak bisimilarity. The congruence formats we obtain are more liberal and elegant than existing congruence formats for these semantics. In particular, in [1] it is stated that the RWB format put forward in that paper has a "horrible definition", and that "negative rules seem incompatible with weak process equivalences." Here we show how negative premises can be included in congruence formats for rooted delay and weak bisimilarity.

NICTA is funded by the Australian Government through the Department of Communications and the Australian Research Council through the ICT Centre of Excellence Program.

In: Proc. LICS 2016, pp. 778–787.

2. Preliminaries

2.1 Equivalences on labelled transition systems

A *labelled transition system (LTS)* is a pair $(\mathbb{P}, \rightarrow)$, with $\mathbb{P}$ a set of *processes* and $\rightarrow \subseteq \mathbb{P} \times (A \cup \{\tau\}) \times \mathbb{P}$, where τ is an *internal action* and A a set of *concrete actions* not containing τ. We use p, q to denote processes, α, β, γ for elements of $A \cup \{\tau\}$, and a, b for elements of A. We write $p \xrightarrow{\alpha} q$ for $(p, \alpha, q) \in \rightarrow$ and $p \not\xrightarrow{\alpha}$ for $\neg(\exists q \in \mathbb{P} : p \xrightarrow{\alpha} q)$. The transitive-reflexive closure of $\xrightarrow{\tau}$ is denoted by $\stackrel{\epsilon}{\Longrightarrow}$.

Processes can be distinguished by a wide range of semantics, based on e.g. branching structure or decorated execution sequences. Weak semantics, classified in [12], abstract away from τ's to different degrees. Here we focus on the two weak semantics that are employed most often: delay bisimilarity [19] and weak bisimilarity [20].

Definition 1 Let $B \subseteq \mathbb{P} \times \mathbb{P}$ be a symmetric relation.

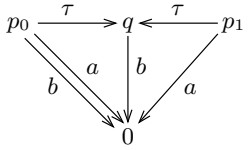
- B is a *delay bisimulation* if pBq and $p \xrightarrow{\alpha} p'$ implies that either $\alpha = \tau$ and $p' \, B \, q$, or $q \stackrel{\epsilon}{\Longrightarrow} \xrightarrow{\alpha} q''$ for some q'' with $p' B q''$. Processes p, q are *delay bisimilar*, denoted $p \leftrightarrows_d q$, if there exists a delay bisimulation B with pBq.
- B is a *weak bisimulation* if pBq and $p \xrightarrow{\alpha} p'$ implies that either $\alpha = \tau$ and $p' \, B \, q$, or $q \stackrel{\epsilon}{\Longrightarrow} \xrightarrow{\alpha} \stackrel{\epsilon}{\Longrightarrow} q''$ for some q'' with $p' B q''$. Processes p, q are *weakly bisimilar*, denoted $p \leftrightarrows_w q$, if there exists a weak bisimulation B with pBq.

$\leftrightarrows_d$ and $\leftrightarrows_w$ are equivalence relations. However, they are not congruences for most process algebras. Rooted variants of these semantics, which require for the pair of initial states that a τ-transition needs to be matched by at least one τ-transition, are congruences for basic process algebras, notably for the alternative composition operator.

Definition 2 Let $R \subseteq \mathbb{P} \times \mathbb{P}$ be a symmetric relation.

- R is a *rooted delay bisimulation* if pRq and $p \xrightarrow{\alpha} p'$ implies that $q \stackrel{\epsilon}{\Longrightarrow} \xrightarrow{\alpha} q'$ for some q' with $p' \leftrightarrows_d q'$. Processes p, q are *rooted delay bisimilar*, $p \leftrightarrows_{rd} q$, if there exists a rooted delay bisimulation R with pRq.
- R is a *rooted weak bisimulation* if pRq and $p \xrightarrow{\alpha} p'$ implies that $q \stackrel{\epsilon}{\Longrightarrow} \xrightarrow{\alpha} \stackrel{\epsilon}{\Longrightarrow} q'$ for some q' with $p' \leftrightarrows_w q'$. Processes p, q are *rooted weakly bisimilar*, $p \leftrightarrows_{rw} q$, if there exists a rooted weak bisimulation R with pRq.

Example 1 Processes p_0 and p_1 in the LTS below are rooted delay bisimilar but not η-bisimilar. In an η-bisimulation the transition $p_0 \xrightarrow{b} 0$ cannot be mimicked by p_1; the only candidate $p_1 \xrightarrow{\tau} q \xrightarrow{b} 0$ fails because q cannot be related to p_0, while this would be required for an η-bisimulation.



2.2 Modal logic

Hennessy-Milner logic [17] is extended with the modal connective $\langle\epsilon\rangle\varphi$, expressing that a process can perform zero or more τ-transitions to a state where φ holds.

Definition 3 [12] The class $\mathbb{O}$ of *modal formulas* is defined as follows, where I ranges over all index sets:

$$\mathbb{O} \qquad \varphi ::= \textstyle\bigwedge_{i\in I}\varphi_i \mid \neg\varphi \mid \langle\alpha\rangle\varphi \mid \langle\epsilon\rangle\varphi \ .$$

$p \models \varphi$ denotes that p satisfies φ. By definition, $p \models \langle\alpha\rangle\varphi$ if $p \xrightarrow{\alpha} p'$ for some p' with $p' \models \varphi$, and $p \models \langle\epsilon\rangle\varphi$ if $p \stackrel{\epsilon}{\Longrightarrow} p'$ for some p' with $p' \models \varphi$. We use abbreviations $\top$ for the empty conjunction and $\varphi_1 \wedge \varphi_2$ for $\bigwedge_{i\in\{1,2\}} \varphi_i$. We write $\varphi \equiv \varphi'$ if $p \models \varphi \Leftrightarrow p \models \varphi'$ for any process p in any LTS.

A modal characterisation of an equivalence on processes consists of a class C of modal formulas such that two processes are equivalent if and only if they satisfy the same formulas in C. Hennessy-Milner logic is a modal characterisation of bisimilarity. We now introduce modal characterisations for (unrooted and rooted) delay and weak bisimilarity.

Definition 4 Let a range over A and α over $A \cup \{\tau\}$. The subclasses $\mathbb{O}_e$ and $\mathbb{O}_{re}$ of $\mathbb{O}$, for $e \in \{d, w\}$, are defined by:

$$\begin{array}{ll}
\mathbb{O}_d & \varphi ::= \textstyle\bigwedge_{i\in I}\varphi_i \mid \neg\varphi \mid \langle\epsilon\rangle\varphi \mid \langle\epsilon\rangle\langle a\rangle\varphi \\
\mathbb{O}_{rd} & \varphi ::= \textstyle\bigwedge_{i\in I}\varphi_i \mid \neg\varphi \mid \langle\epsilon\rangle\langle\alpha\rangle\hat{\varphi} \mid \hat{\varphi}\ (\hat{\varphi} \in \mathbb{O}_d) \\
\mathbb{O}_w & \varphi ::= \textstyle\bigwedge_{i\in I}\varphi_i \mid \neg\varphi \mid \langle\epsilon\rangle\varphi \mid \langle\epsilon\rangle\langle a\rangle\langle\epsilon\rangle\varphi \\
\mathbb{O}_{rw} & \varphi ::= \textstyle\bigwedge_{i\in I}\varphi_i \mid \neg\varphi \mid \langle\epsilon\rangle\langle\alpha\rangle\langle\epsilon\rangle\hat{\varphi} \mid \hat{\varphi}\ (\hat{\varphi} \in \mathbb{O}_w).
\end{array}$$

For $L \subseteq \mathbb{O}$, we write $p \sim_L q$ if p and q satisfy the same formulas in L. The class $L^{\equiv}$ denotes the closure of L under $\equiv$. Trivially, $p \sim_L q \Leftrightarrow p \sim_{L^{\equiv}} q$.

Theorem 1 $p \leftrightarrows_e q \Leftrightarrow p \sim_{\mathbb{O}_e} q$ and $p \leftrightarrows_{re} q \Leftrightarrow p \sim_{\mathbb{O}_{re}} q$, for all $p, q \in \mathbb{P}$, where $e \in \{d, w\}$.

2.3 Structural operational semantics

A *signature* is a set Σ of function symbols f with arity $ar(f)$. Let V be an infinite set of variables x, y, z, with $|\Sigma|, |A| \leq |V|$. A syntactic object is *closed* if it does not contain any variables. The set $\mathbb{T}(\Sigma)$ of terms over Σ and V is defined as usual; t, u, v, w denote terms and $var(t)$ is the set of variables that occur in term t. A term is *univariate* if it is without multiple occurrences of the same variable. A substitution σ is a partial function from V to $\mathbb{T}(\Sigma)$. A closed substitution is a total function from V to closed terms.

Structural operational semantics provides process algebras and specification languages with an interpretation. It generates an LTS, in which processes are the closed terms over a (single-sorted, first-order) signature, and transitions between processes may be supplied with labels. The transitions are obtained from a transition system specification, which consists of a set of proof rules called transition rules.

Definition 5 A (*positive* or *negative*) *literal* is an expression $t \xrightarrow{\alpha} u$ or $t \not\xrightarrow{\alpha}$. A *(transition) rule* is of the form $\frac{H}{\lambda}$ with H a set of literals called the *premises*, and λ a literal called the *conclusion*; the term at the left-hand side of λ is called the *source*. H^+ denotes the positive premises and H^{s-} the *stable* negative premises in H: those $t \not\xrightarrow{\alpha}$ for which also $t \not\xrightarrow{\tau}$ is in H. A rule $\frac{\emptyset}{\lambda}$ is also written λ. A rule is *standard* if it has a positive conclusion. A *transition system specification (TSS)* consists of a signature Σ and a set R of rules over Σ. A TSS is *standard* if all its rules are.

Definition 6 Let $P = (\Sigma, R)$ be a TSS. An *irredundant proof* from P of a rule $\frac{H}{\lambda}$ is a well-founded tree with the nodes labelled by literals and some of the leaves marked "hypothesis", such that the root has label λ, H is the set of labels of the hypotheses, and if μ is the label of a node that is not a hypothesis and K is the set of labels of the children of this node then $\frac{K}{\mu}$ is a substitution instance of a rule in R.

The proof of $\frac{H}{\lambda}$ is called irredundant [2] because H must equal (instead of include) the set of labels of the hypotheses. Irredundancy is crucial for the preservation under provability of our congruence formats. Namely, in a 'redundant' proof one can freely add any premises to the derived rule.

Literals $t \xrightarrow{\alpha} u$ and $t \not\xrightarrow{\alpha}$ are said to *deny* each other.

Definition 7 [13] Let $P = (\Sigma, R)$ be standard TSS. A *well-supported proof* from P of a closed literal λ is a well-founded tree with the nodes labelled by closed literals, such that the root is labelled by λ, and if μ is the label of a node and K is the set of labels of the children of this node, then: either μ is positive and $\frac{K}{\mu}$ is a closed substitution instance of a rule in R; or μ is negative and for each set N of closed negative literals with $\frac{N}{\nu}$ irredundantly provable from P and ν a closed positive literal that denies μ, a literal in K denies one in N. $P \vdash_{ws} \lambda$ denotes that a well-supported proof from P of λ exists. A standard TSS P is *complete* if for each p and α, either $P \vdash_{ws} p \not\xrightarrow{\alpha}$ or $P \vdash_{ws} p \xrightarrow{\alpha} p'$ for some p'.

In [13] it was shown that $\vdash_{ws}$ is consistent, in the sense that no standard TSS admits well-supported proofs of two literals that deny each other. A complete TSS specifies an LTS, consisting of the *ws*-provable closed positive literals.

2.4 Syntactic restrictions on transition rules

We present terminology for syntactic restrictions on rules from [2, 15, 16]. An *ntytt rule* is a 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, an *ntyft rule* if its source contains exactly one function symbol and no multiple occurrences of variables, and an *nxytt rule* if the left-hand sides of its premises are variables. An *xynft rule* is ntyft and the left-hand sides of its positive premises are variables.

A variable in a rule is *free* if it occurs neither in the source nor in right-hand sides of premises. A rule has *lookahead* if some variable occurs in the right-hand side of a premise and in the left-hand side of a premise. A *decent* rule has no lookahead and no free variables. A TSS in *ready simulation format* consists of ntyft and ntyxt rules without lookahead.

2.5 Patience rules

Let Γ be a unary predicate on arguments of function symbols. If $\Gamma(f, i)$, then argument i of f is Γ-*liquid*; otherwise it is Γ-*frozen*. An occurrence of x in t is Γ-*liquid* if $t = x$ or $t = f(t_1, \ldots, t_{ar(f)})$ and the occurrence is Γ-liquid in t_i with $\Gamma(f, i)$; otherwise the occurrence is Γ-*frozen*.

Sect. 2.8 presents a method for decomposing modal formulas that gives a special treatment to arguments of function symbols that are deemed *patient*; we will use a predicate Γ to mark the arguments that get this special treatment.

Definition 8 [1, 5] A standard ntyft rule is a *patience rule* for argument i of f if it is of the form

$$\frac{x_i \xrightarrow{\tau} y}{f(x_1, \ldots, x_{ar(f)}) \xrightarrow{\tau} f(x_1, \ldots, x_{i-1}, y, x_{i+1}, \ldots, x_{ar(f)})}$$

Given a predicate Γ, the rule above is a Γ-*patience rule* if $\Gamma(f, i)$. A TSS is Γ-*patient* if it contains all Γ-patience rules. A standard ntytt rule is Γ-*patient* if it is irredundantly provable from the Γ-patience rules; else it is Γ-*impatient*.

Typically, in process algebra, there are patience rules for both arguments of the merge operator and for the first argument of sequential composition, as they can contain running processes, but not for the arguments of alternative composition or for the second argument of sequential composition.

2.6 Ruloids

To decompose modal formulas, we use a result from [2], where for any standard TSS P in ready simulation format a collection of decent nxytt rules, called P-*ruloids*, is constructed. We explain this construction at a rather superficial level; the precise transformation can be found in [2].

First P is converted to a standard TSS $P^\dagger$ in decent ntyft format. In this conversion from [16], free variables in a rule are replaced by arbitrary closed terms, and if the source is of the form x, then this variable is replaced by a term $f(x_1, \ldots, x_{ar(f)})$ for each function symbol f in the signature of P, where the variables $x_1, \ldots, x_{ar(f)}$ are fresh.

Next, using a construction from [6], left-hand sides of positive premises are reduced to variables. Roughly the idea is, given a premise $f(t_1, \ldots, t_n) \xrightarrow{\alpha} y$ in a rule r, and another rule $\frac{H}{f(x_1,\ldots,x_n)\xrightarrow{\alpha} t}$, to transform r by replacing the aforementioned premise by H, y by t, and the x_i by the t_i; this is repeated (transfinitely) until all positive premises with a non-variable term as left-hand side have disappeared. This yields an intermediate standard TSS $P^\ddagger$ in xynft format, of

which all the rules are irredundantly provable from P. In fact, the rules of $P^\ddagger$ are exactly the xynft rules irredundantly provable from $P^\dagger$. The motivation for this transformation step is that for TSSs in xynft format the semantic phrase "for each set N of closed negative literals with $\frac{N}{\nu}$ irredundantly provable from P" in the second clause of Def. 7 of a well-supported proof can be replaced by a syntactic phrase: "for each closed substitution instance $\frac{N}{\nu}$ of a rule in R".

Finally, non-standard rules with a negative conclusion $t \stackrel{\alpha}{\not\longrightarrow}$ are introduced. The motivation is that instead of the notion of well-founded provability of Def. 7, we want a more constructive notion like Def. 6, by making it possible that a negative premise is matched with a negative conclusion. A non-standard rule $\frac{H}{f(x_1,\ldots,x_n)\stackrel{\alpha}{\not\longrightarrow}}$ is obtained by picking one premise from each xynft rule in $P^\ddagger$ with a conclusion of the form $f(x_1,\ldots,x_n) \stackrel{\alpha}{\longrightarrow} t$, and including the denial of each of the selected premises as a premise in H.

The resulting TSS, which is in decent ntyft format, is denoted by P^+. The above construction implies that if P is Γ-patient, then so is P^+. In [2] it was established, for all closed literals μ, that $P \vdash_{ws} \mu$ if and only if μ is irredundantly provable from P^+. By definition, the *P-ruloids* are the (decent) nxytt rules irredundantly provable from P^+.

There is a well-supported proof from a TSS P of a transition $\rho(t) \stackrel{a}{\longrightarrow} q$ if and only if there is a derivation of this transition that uses at the root a P-ruloid with source t. This result underlies the decomposition method in Sect. 2.8.

Proposition 1 [2] Let $P = (\Sigma, R)$ be a standard TSS in ready simulation format, $t \in \mathbb{T}(\Sigma)$ and $\rho : V \to \mathbb{T}(\Sigma)$ a closed substitution. Then $P \vdash_{ws} \rho(t) \stackrel{\alpha}{\longrightarrow} q$ if and only if there are a P-ruloid $\frac{H}{t\stackrel{\alpha}{\longrightarrow}u}$ and a closed substitution ρ' such that $P \vdash_{ws} \rho'(\mu)$ for all $\mu \in H$, $\rho'(t) = \rho(t)$ and $\rho'(u) = q$.

2.7 Linear proofs

Definition 9 An irredundant proof of a rule $\frac{H}{\lambda}$ is called *linear* if no two hypotheses in the proof tree of Def 6 are labelled with the same positive premise.

Clearly, each ntytt rule provable from a TSS is a substitution instance of an ntytt rule that has a linear proof. In Def. 7 it does not make any difference whether in clause 2 we quantify over rules $\frac{N}{\nu}$ that are provable (as in [13, 2, 8]), irredundantly provable (as in [9]), or linearly provable.

In Sect. 2.6 a non-standard TSS P^+ is constructed out of a given TSS P in ready simulation format, via the intermediate stages $P^\dagger$ and $P^\ddagger$. Here $P^\ddagger$ consists of all xynft rules irredundantly provable from $P^\dagger$. With $\hat{P}^\ddagger$ we denote the TSS consisting of all xynft rules linearly provable from $P^\dagger$. The TSS P^+ is obtained by augmenting $P^\ddagger$ with non-standard rules; with $\hat{P}^+$ we denote the corresponding augmentation of $\hat{P}^\ddagger$. For the construction of the non-standard rules in the augmentation, it makes no difference whether we start from $P^\ddagger$ or $\hat{P}^\ddagger$, since the difference disappears when abstracting from the right-hand sides of positive literals. Thus, $\hat{P}^+ \subseteq P^+$ and each rule in P^+ is a substitution instance of a rule in $\hat{P}^+$. Hence, an ntytt rule is irredundantly provably from $\hat{P}^+$ iff it is irredundantly provable from P^+.

A *linear* P-ruloid has a linear proof from $\hat{P}^+$. Clearly, each P-ruloid is a substitution instance of a linear P-ruloid. So Prop. 1 still holds if we only consider linear ruloids.

2.8 Decomposition of modal formulas

In [9] it was shown how one can decompose formulas from $\mathbb{O}$. To each term t and formula $\varphi \in \mathbb{O}$, a set $t^{-1}(\varphi)$ of decomposition mappings $\psi : V \to \mathbb{O}$ is assigned. Each $\psi \in t^{-1}(\varphi)$ guarantees that for any closed substitution ρ, $\rho(t) \models \varphi$ if $\rho(x) \models \psi(x)$ for all $x \in var(t)$. Vice versa, whenever $\rho(t) \models \varphi$, there is a decomposition mapping $\psi \in t^{-1}(\varphi)$ with $\rho(x) \models \psi(x)$ for all $x \in var(t)$.

Definition 10 [9] Let $P = (\Sigma, R)$ be a Γ-patient standard TSS in ready simulation format. We define $\cdot^{-1} : \mathbb{T}(\Sigma) \times \mathbb{O} \to \mathcal{P}(V \to \mathbb{O})$ as the function that for each $t \in \mathbb{T}(\Sigma)$ and $\varphi \in \mathbb{O}$ returns the set $t^{-1}(\varphi) \in \mathcal{P}(V \to \mathbb{O})$ of decomposition mappings $\psi : V \to \mathbb{O}$ generated by following five conditions. In the remainder of this definition, t is a univariate term.

1. $\psi \in t^{-1}(\bigwedge_{i\in I} \varphi_i)$ iff there are $\psi_i \in t^{-1}(\varphi_i)$ for each $i \in I$ such that $\psi(x) = \bigwedge_{i\in I} \psi_i(x)$ for all $x \in V$.
2. $\psi \in t^{-1}(\neg\varphi)$ iff there is a function $h : t^{-1}(\varphi) \to var(t)$ such that $\psi(x)$ equals either $\bigwedge_{\chi\in h^{-1}(x)} \neg\chi(x)$ if $x \in var(t)$, or $\top$ if $x \notin var(t)$.
3. $\psi \in t^{-1}(\langle\alpha\rangle\varphi)$ iff there is a P-ruloid $\frac{H}{t\stackrel{\alpha}{\longrightarrow}u}$ and a $\chi \in u^{-1}(\varphi)$ such that $\psi(x)$ equals either
$$\chi(x) \wedge \bigwedge_{x\stackrel{\beta}{\longrightarrow}y\in H} \langle\beta\rangle\chi(y) \wedge \bigwedge_{x\stackrel{\gamma}{\not\longrightarrow}\in H} \neg\langle\gamma\rangle\top$$
if $x \in var(t)$, or $\top$ if $x \notin var(t)$.
4. $\psi \in t^{-1}(\langle\epsilon\rangle\varphi)$ iff one of the following holds.
   (a) There is a $\chi \in t^{-1}(\varphi)$ such that $\psi(x)$ equals either $\langle\epsilon\rangle\chi(x)$ if x occurs Γ-liquid in t, or $\chi(x)$ otherwise.
   (b) There is a Γ-impatient P-ruloid $\frac{H}{t\stackrel{\tau}{\longrightarrow}u}$ and a $\chi \in u^{-1}(\langle\epsilon\rangle\varphi)$ such that $\psi(x)$ equals either
$$\langle\epsilon\rangle\Big(\chi(x) \wedge \bigwedge_{x\stackrel{\beta}{\longrightarrow}y\in H} \langle\beta\rangle\chi(y) \wedge \bigwedge_{x\stackrel{\gamma}{\not\longrightarrow}\in H} \neg\langle\gamma\rangle\top\Big)$$
if x occurs Γ-liquid in t, or
$$\chi(x) \wedge \bigwedge_{x\stackrel{\beta}{\longrightarrow}y\in H} \langle\beta\rangle\chi(y) \wedge \bigwedge_{x\stackrel{\gamma}{\not\longrightarrow}\in H} \neg\langle\gamma\rangle\top$$
if x occurs Γ-frozen in t, or $\top$ if $x \notin var(t)$.
5. $\psi \in \sigma(t)^{-1}(\varphi)$ for a non-injective substitution $\sigma : var(t) \to V$ iff there is a $\chi \in t^{-1}(\varphi)$ such that $\psi(x) = \bigwedge_{z\in\sigma^{-1}(x)} \chi(z)$ for all $x \in V$.

Theorem 2 [9] Let $P = (\Sigma, R)$ be a Γ-patient complete standard TSS in ready simulation format. For any term $t \in \mathbb{T}(\Sigma)$, closed substitution ρ, and $\varphi \in \mathbb{O}$:

$$\rho(t) \models \varphi \;\Leftrightarrow\; \exists \psi \in t^{-1}(\varphi)\, \forall x \in var(t) : \rho(x) \models \psi(x) \;\; .$$

3. Delay resistant TSSs

In the next section the decomposition method from Sec. 2.8 is applied to obtain congruence formats for (rooted) delay and weak bisimilarity. However, Def. 10 needs to be refined in the case $t^{-1}(\langle\epsilon\rangle\varphi)$, because in the modal logics for delay and weak bisimilarity, occurrences of subformulas $\langle\beta\rangle\varphi'$ are always preceded by $\langle\epsilon\rangle$, while in Def. 10 this is not always the case. The refinement of Def. 10, which is presented in Def. 15, is only valid for so-called delay resistant TSSs.

Def. 14 of delay resistance is inspired by a requirement in the RDB and RWB cool formats, see [14, Def. 15(3)]. It is crafted in such a way that Prop. 2 holds: if for a premise $x \xrightarrow{\beta} y$ in a ruloid $r = \frac{H}{t \xrightarrow{\alpha} u}$ the execution of β is delayed by a τ-step, i.e. for some σ we have $\sigma(x) \xrightarrow{\tau} \xrightarrow{\beta} \sigma(y)$, then the conclusion $\sigma(t) \xrightarrow{\alpha} \sigma(u)$ of the σ-instance of r is unaffected, or merely delayed by a τ-step as well.

Example 2 Consider the rooted delay bisimilar processes p_0 and p_1 from Ex. 1. The ruloid r may apply when substituting p_0 for x and b for β, given that $p_0 \xrightarrow{b} 0$. If instead of p_0 we substitute p_1 for x, to safeguard congruence, it is necessary that the (possibly delayed) conclusion of r can still be derived, even though we only have $p_1 \xrightarrow{\tau} q \xrightarrow{b} 0$.

In Def. 12 we allow two possible implementations of this idea. Each premise $x \xrightarrow{\beta} y$ of r is either *delayable* (Def. 11), in which case $\sigma(x) \xrightarrow{\tau} \xrightarrow{\beta} \sigma(y)$ induces $\sigma(t) \xrightarrow{\tau} \xrightarrow{\alpha} \sigma(u)$ by two ruloids that can be used in place of r; or *τ-pollable*, meaning that r remains valid if this premise is replaced by $x \xrightarrow{\tau} z$ for a fresh variable z, so that the premise $\sigma(x) \xrightarrow{\tau} \sigma(z)$ takes over the role of $\sigma(x) \xrightarrow{\beta} \sigma(y)$. Def. 12 and Prop. 2 allow only finitely many delayable positive premises, but infinitely many τ-pollable ones.

Definition 11 A premise $w \xrightarrow{\beta} y$ of an ntytt rule $r = \frac{H}{t \xrightarrow{\alpha} u}$ is *delayable* in a TSS P if there are ntytt rules $\frac{H_1}{t \xrightarrow{\tau} v}$ and $\frac{H_2}{v \xrightarrow{\alpha} u}$, linearly provable from P, with $H_1 \subseteq (H \setminus \{w \xrightarrow{\beta} y\}) \cup \{w \xrightarrow{\tau} z\}$ and $H_2 \subseteq (H \setminus \{w \xrightarrow{\beta} y\}) \cup \{z \xrightarrow{\beta} y\}$ for some term v and fresh variable z.

Suppose that $\frac{H}{t \xrightarrow{\alpha} u}$ in Def. 11 is a ruloid, so that w is a variable x. The intuition behind this definition is that the argument x of t may not be able to perform a β-transition to a term y immediately, but only after a τ-transition to z. The ruloid $\frac{H_1}{t \xrightarrow{\tau} v}$ then allows t to postpone its α-transition to u, by first performing a τ-transition to v. The ruloid $\frac{H_2}{v \xrightarrow{\alpha} u}$ guarantees that the postponed β-transition from z to y still gives rise to an α-transition from v to u.

Linearity is needed to make sure that in the construction of ruloids, distinct delayable positive premises are never collapsed to a single non-delayable premise.

Def. 12 requires that each positive premise is delayable (i.e., in the finite set H^d), τ-pollable, or redundant.

Definition 12 An ntytt rule $\frac{H}{t \xrightarrow{\alpha} u}$ is *positive delay resistant* w.r.t. a TSS P if there exists a finite set $H^d \subseteq H^+$ of delayable positive premises such that for each set $M \subseteq H^+ \setminus H^d$ there is a rule $r_M = \frac{H_M}{t \xrightarrow{\alpha} u}$, linearly provable from P, where $H_M \subseteq (H \setminus M) \cup M_\tau$ with $M_\tau = \{w \xrightarrow{\tau} z_y \mid (w \xrightarrow{\beta} y) \in M,\ z_y \text{ fresh}\}$.

The intuition behind Def. 13 is closely related to Def. 11. If a ruloid $\frac{H}{t \xrightarrow{\alpha} u}$ has a premise $x \xrightarrow{\gamma}\!\!\!\!\!\!/$, we want it not to apply in case $\sigma(x) \xrightarrow{\tau} \xrightarrow{\gamma}$, even if $\sigma(x) \xrightarrow{\gamma}\!\!\!\!\!\!/$. We therefore require that for each premise $x \xrightarrow{\gamma}\!\!\!\!\!\!/$ there must also be a premise $x \xrightarrow{\tau}\!\!\!\!\!\!/$. We make an exception for redundant premises $x \xrightarrow{\gamma}\!\!\!\!\!\!/$.

Definition 13 A rule $\frac{H}{t \xrightarrow{\alpha} u}$ is *negative delay resistant* w.r.t. a TSS P if there is a rule $\frac{H'}{t \xrightarrow{\alpha} u}$, linearly provable from P, with $H' \subseteq H^+ \cup H^{s-}$.

Definition 14 An ntytt rule $\frac{H}{t \xrightarrow{\alpha} u}$ is *delay resistant* w.r.t. a TSS P if it is positive delay resistant as well as negative delay resistant. A standard TSS P in ready simulation format is *delay resistant* if all its linear ruloids with a positive conclusion are delay resistant w.r.t. $\hat{P}^+$.

The following proposition is key to the notion of delay resistance. It will allow us to adapt the definition of modal decomposition for delay resistant TSSs, so that it becomes applicable for generating congruence formats for weak semantics, like delay and weak bisimilarity, with a modal characterisation in which $\langle\beta\rangle\varphi$ is always preceded by $\langle\epsilon\rangle$.

Proposition 2 Let P be a delay resistant standard TSS in ready simulation format. Let $\frac{H}{t \xrightarrow{\alpha} u}$ be a P-ruloid and ρ a closed substitution with $P \vdash_{ws} \rho(x) \stackrel{\epsilon}{\Longrightarrow} \xrightarrow{\beta} \rho(y)$ for each premise $x \xrightarrow{\beta} y$ in H^+ and $P \vdash_{ws} \rho(x) \xrightarrow{\gamma}\!\!\!\!\!\!/$ for each premise $x \xrightarrow{\gamma}\!\!\!\!\!\!/$ in H^{s-}. Then $P \vdash_{ws} \rho(t) \stackrel{\epsilon}{\Longrightarrow} \xrightarrow{\alpha} \rho(u)$.

For delay resistant TSSs case 4b of Def. 10, $t^{-1}(\langle\epsilon\rangle\varphi)$, needs to be adapted, to ensure that in the modal logics for delay and weak bisimilarity, occurrences of subformulas $\langle\beta\rangle\varphi''$ are always preceded by $\langle\epsilon\rangle$. Moreover, case 4a is provided with the restriction that φ is not of the form $\langle\alpha\rangle\varphi'$. Else decompositions of formulas $\langle\epsilon\rangle\langle\alpha\rangle\varphi'$ in $\mathbb{O}_d$ would be defined in terms of formulas $\chi \in t^{-1}(\langle\alpha\rangle\varphi')$, while $\langle\alpha\rangle\varphi'$ is not in $\mathbb{O}_d$. Instead, if φ is of the form $\langle\alpha\rangle\varphi'$, cases 3 and 4 of Def. 10 are combined, as can be seen in case 4b(ii) below.

Definition 15 Let P be a delay resistant Γ-patient standard TSS in ready simulation format. We define $\cdot^{-1}_{\mathrm{dr}} : \mathbb{T}(\Sigma) \times \mathbb{O} \to \mathcal{P}(V \to \mathbb{O})$ exactly as $\cdot^{-1}$ in Def. 10, except for case 4: $t^{-1}_{\mathrm{dr}}(\langle\epsilon\rangle\varphi)$ (with t univariate).

Erratum, 11 March 2017: With case 4b(i) as formulated here, Prop. 5 does not hold. "$\chi \in u_{\rm dr}^{-1}(\langle\epsilon\rangle\varphi)$" should be "$\chi \in u_{\rm dr}^{-1}(\langle\epsilon\rangle\varphi')$ in case φ is of the form $\langle\tau\rangle\varphi'$, and $\chi \in u_{\rm dr}^{-1}(\langle\epsilon\rangle\varphi)$ otherwise".

4. $\psi \in t_{\rm dr}^{-1}(\langle\epsilon\rangle\varphi)$ iff one of the following holds.
   (a) φ is not of the form $\langle\alpha\rangle\varphi'$, and there is a $\chi \in t_{\rm dr}^{-1}(\varphi)$ such that $\psi(x)$ equals either $\langle\epsilon\rangle\chi(x)$ if x occurs Γ-liquid in t, or $\chi(x)$ otherwise.
   (b) (i) There is a Γ-impatient P-ruloid $\frac{H}{t\xrightarrow{\tau}u}$ and a $\chi \in u_{\rm dr}^{-1}(\langle\epsilon\rangle\varphi)$,
   (ii) or φ is of the form $\langle\alpha\rangle\varphi'$, and there is a P-ruloid $\frac{H}{t\xrightarrow{\alpha}u}$ and a $\chi \in u_{\rm dr}^{-1}(\varphi')$,

   such that $\psi(x)$ equals either

$$\langle\epsilon\rangle\Big(\chi(x) \;\wedge \bigwedge_{x\xrightarrow{\beta}y\in H^+} \langle\epsilon\rangle\langle\beta\rangle\chi(y) \;\wedge \bigwedge_{x\xrightarrow{\gamma}\in H^{s-}} \neg\langle\gamma\rangle\top\Big)$$

   if x occurs Γ-liquid in t, or

$$\chi(x) \;\wedge \bigwedge_{x\xrightarrow{\beta}y\in H^+} \langle\epsilon\rangle\langle\beta\rangle\chi(y) \;\wedge \bigwedge_{x\xrightarrow{\gamma}\in H^{s-}} \neg\langle\gamma\rangle\top$$

   if x occurs Γ-frozen in t, or $\top$ if $x \notin var(t)$.

The counterpart of Thm. 2 for delay resistant TSSs is:

Theorem 3 Let $P = (\Sigma, R)$ be a delay resistant Γ-patient complete standard TSS in ready simulation format. For any $t \in \mathbb{T}(\Sigma)$, closed substitution ρ, and $\varphi \in \mathbb{O}$:

$$\rho(t) \models \varphi \;\Leftrightarrow\; \exists\psi \in t_{\rm dr}^{-1}(\varphi)\, \forall x \in var(t) : \rho(x) \models \psi(x) \;\; .$$

4. Congruence results

A behavioural equivalence $\sim$ is a *congruence* for a function symbol f if $p_i \sim q_i$ for all $i \in \{1,\ldots,ar(f)\}$ implies that $f(p_1,\ldots,p_{ar(f)}) \sim f(q_1,\ldots,q_{ar(f)})$. We apply the decomposition method from the previous section to derive congruence formats for delay bisimulation and rooted delay bisimulation semantics. The idea behind the construction of these formats is that a formula from the characterising logic of the equivalence under consideration must always be decomposed into formulas from this same logic. The delay bisimulation format guarantees that a formula from $\mathbb{O}_d$ is always decomposed into formulas from $\mathbb{O}_d^{\equiv}$ (Prop. 4). Likewise, the rooted delay bisimulation format guarantees that a formula from $\mathbb{O}_{rd}$ is always decomposed into formulas from $\mathbb{O}_{rd}^{\equiv}$ (Prop. 5). This implies the desired congruence results (Thm. 4 resp. Thm. 5). These results are transposed to (rooted) weak bisimilarity by adding one condition to the congruence format for (rooted) delay bisimilarity.

4.1 Congruence format for rooted delay bisimilarity

We recall the notion of a rooted branching bisimulation safe rule, which underlies the rooted branching bisimulation format from [9]. The congruence format for rooted delay bisimilarity is obtained by additionally requiring delay resistance.

We assume two predicates on arguments of function symbols from [5, 9]. The predicate Λ marks arguments that contain processes that have started executing (but may currently be unable to execute). The predicate $\aleph$ marks arguments that contain processes that can execute immediately. For example, in process algebra, Λ and $\aleph$ hold for the arguments of the merge $t_1\|t_2$, and for the first argument of sequential composition $t_1\cdot t_2$; they can contain processes that started to execute in the past, and these processes can continue their execution immediately. On the other hand, Λ and $\aleph$ typically do not hold for the second argument of sequential composition; it contains a process that did not yet start to execute, and cannot execute immediately (in absence of the empty process). Λ does not hold and $\aleph$ holds for the arguments of alternative composition $t_1 + t_2$; they contain processes that did not yet start to execute, but that can start executing immediately.

Definition 16 [9] An ntytt rule $r = \frac{H}{t\xrightarrow{\alpha}u}$ is *rooted branching bisimulation safe* w.r.t. $\aleph$ and Λ if it satisfies the following conditions. Let $x \in var(t)$.

1. Right-hand sides of positive premises occur only Λ-liquid in u.
2. If x occurs only Λ-liquid in t, then x occurs only Λ-liquid in r.
3. If x occurs only $\aleph$-frozen in t, then x occurs only $\aleph$-frozen in H.
4. If x has exactly one $\aleph$-liquid occurrence in t, which is also Λ-liquid, then x has at most one $\aleph$-liquid occurrence in H, which must be in a positive premise. If moreover this premise is labelled τ, then r must be $\aleph\cap\Lambda$-patient.

Definition 17 A standard TSS P is in *rooted delay bisimulation format* if it is in ready simulation format and delay resistant, and, for some $\aleph$ and Λ, it is $\aleph\cap\Lambda$-patient and all its rules are rooted branching bisimulation safe w.r.t. $\aleph$ and Λ.

This TSS is in *delay bisimulation format* if Λ is universal.

Proposition 3 [9] Let P be a standard TSS in ready simulation format, in which each transition rule is rooted branching bisimulation safe w.r.t. $\aleph$ and Λ. Then each P-ruloid is rooted branching bisimulation safe w.r.t. $\aleph$ and Λ.

4.2 Congruence for rooted delay bisimilarity

Consider a standard TSS that is in rooted delay bisimulation format, w.r.t. some $\aleph$ and Λ. Def. 15 yields decomposition mappings $\psi \in t_{\rm dr}^{-1}(\varphi)$, with $\Gamma = \aleph\cap\Lambda$. If $\varphi \in \mathbb{O}_d$, then $\psi(x) \in \mathbb{O}_d^{\equiv}$ if x occurs only Λ-liquid in t. (That is why in the delay bisimulation format, Λ must be universal.) If $\varphi \in \mathbb{O}_{rd}$, then $\psi(x) \in \mathbb{O}_{rd}^{\equiv}$ for all variables x. These preservation results induce the promised congruence results for delay bisimilarity and rooted delay bisimilarity, respectively.

Proposition 4 Let P be a delay resistant, $\aleph\cap\Lambda$-patient standard TSS in ready simulation format, in which each rule is rooted branching bisimulation safe w.r.t. $\aleph$ and Λ. For any term t and variable x that occurs only Λ-liquid in t:

$$\varphi \in \mathbb{O}_d \;\Rightarrow\; \forall\psi \in t_{\rm dr}^{-1}(\varphi) : \psi(x) \in \mathbb{O}_d^{\equiv} \;\; .$$

Proposition 5 Let P be a delay resistant, $\aleph \cap \Lambda$-patient standard TSS in ready simulation format, in which each rule is branching delay bisimulation safe w.r.t. $\aleph$ and Λ. For any term t and variable x:

$$\varphi \in \mathbb{O}_{rd} \;\Rightarrow\; \forall \psi \in t^{-1}_{\rm dr}(\varphi) : \psi(x) \in \mathbb{O}^{\equiv}_{rd} \;.$$

Theorem 4 If P is a complete TSS in delay bisimulation format, then $\leftrightarrows_d$ is a congruence for P.

Proof By Def. 17 each rule of P is rooted branching bisimulation safe w.r.t some $\aleph$ and the universal predicate Λ, and P is delay resistant, $\aleph \cap \Lambda$-patient and in ready simulation format. Let ρ, ρ' be closed substitutions and t a term with $\rho(x) \leftrightarrows_d \rho'(x)$ for all $x \in var(t)$.

Let $\rho(t) \models \varphi \in \mathbb{O}_d$. By Thm. 3, taking $\Gamma = \aleph \cap \Lambda$, there is a $\psi \in t^{-1}_{\rm dr}(\varphi)$ with $\rho(x) \models \psi(x)$ for all $x \in var(t)$. Since x occurs Λ-liquid in t (because Λ is universal), by Prop. 4, $\psi(x) \in \mathbb{O}^{\equiv}_d$ for all $x \in var(t)$. By Thm. 1, $\rho(x) \leftrightarrows_d \rho'(x)$ implies $\rho(x) \sim_{\mathbb{O}^{\equiv}_d} \rho'(x)$ for all $x \in var(t)$. So $\rho'(x) \models \psi(x)$ for all $x \in var(t)$. Therefore, by Thm. 3, $\rho'(t) \models \varphi$. Likewise, $\rho'(t) \models \varphi \in \mathbb{O}_d$ implies $\rho(t) \models \varphi$. So $\rho(t) \sim_{\mathbb{O}_d} \rho'(t)$. Hence, by Thm. 1, $\rho(t) \leftrightarrows_d \rho'(t)$. ∎

Theorem 5 If P is a complete TSS in rooted delay bisimulation format, then $\leftrightarrows_{rd}$ is a congruence for P.

The proof of Thm. 5 is similar to the one of Thm. 4, except that Prop. 5 is applied instead of Prop. 4; therefore x need not occur Λ-liquid in t, so that universality of Λ can be dropped.

4.3 Rooted weak bisimilarity as a congruence

We now proceed to derive a congruence format for rooted weak bisimilarity. It is obtained from the congruence format from [9] for rooted η-bisimilarity by additionally requiring delay resistance. The format for rooted η-bisimilarity in turn is obtained by strengthening condition 1 in the definition of rooted branching bisimulation safeness.

Definition 18 [9] An ntytt rule $\frac{H}{t \xrightarrow{\alpha} u}$ is *rooted η-bisimulation safe* w.r.t. $\aleph$ and Λ if it satisfies conditions 2–4 of Def. 16, together with:

1′. Right-hand sides of positive premises occur only $\aleph \cap \Lambda$-liquid in u.

Definition 19 A standard TSS is in *rooted weak bisimulation format* if it is in ready simulation format and delay resistant, and, for some $\aleph$ and Λ, it is $\aleph \cap \Lambda$-patient and contains only rules that are rooted η-bisimulation safe w.r.t. $\aleph$ and Λ.

This TSS is in *weak bisimulation format* if Λ is universal.

Proposition 6 [9] Let P be a TSS in ready simulation format, in which each rule is rooted η-bisimulation safe w.r.t. $\aleph$ and Λ. Then each P-ruloid is rooted η-bisimulation safe w.r.t. $\aleph$ and Λ.

Proposition 7 Let P be a delay resistant, $\aleph \cap \Lambda$-patient standard TSS in ready simulation format, in which each rule is rooted η-bisimulation safe w.r.t. $\aleph$ and Λ. For any term t and variable x that occurs only Λ-liquid in t:

$$\varphi \in \mathbb{O}_w \;\Rightarrow\; \forall \psi \in t^{-1}_{\rm dr}(\varphi) : \psi(x) \in \mathbb{O}^{\equiv}_w \;.$$

Proposition 8 Let P be a delay resistant, $\aleph \cap \Lambda$-patient standard TSS in ready simulation format, in which each rule is rooted η-bisimulation safe w.r.t. $\aleph$ and Λ. For any term t and variable x:

$$\varphi \in \mathbb{O}_{rw} \;\Rightarrow\; \forall \psi \in t^{-1}_{\rm dr}(\varphi) : \psi(x) \in \mathbb{O}^{\equiv}_{rw} \;.$$

Theorem 6 If P is a complete TSS in weak bisimulation format, then $\leftrightarrows_w$ is a congruence for P.

Theorem 7 If P is a complete TSS in rooted weak bisimulation format, then $\leftrightarrows_{rw}$ is a congruence for P.

4.4 Counterexamples

In [9] it was shown that none of the syntactic requirements of the rooted branching bisimulation format in Def. 16 can be omitted, and that the presence of $\aleph \cap \Lambda$-patience rules is crucial. Here we present examples to show that none of the requirements that make up delay resistance is redundant.

All TSSs in this section are standard, complete, in ready-simulation format and $\aleph \cap \Lambda$-patient, and their rules are rooted η-bisimulation safe.

Example 3 Let f be a unary function with an $\aleph$-liquid, Λ-frozen argument, defined by $\frac{x \xrightarrow{a}\!\!\!\!\!\!\rightarrow}{f(x) \xrightarrow{b} 0}$. This rule is positive delay resistant, but not negative delay resistant.

$p_0, p_1, q, 0$ are constants with $p_0 \xrightarrow{\tau} p_0$, $p_0 \xrightarrow{a} 0$, $p_1 \xrightarrow{\tau} q$ and $q \xrightarrow{a} 0$. Note that $p_0 \leftrightarrows_{rd} p_1$. However, $f(p_0)$ exhibits no transitions, while $f(p_1) \xrightarrow{b} 0$. So $f(p_0) \not\leftrightarrows_{rd} f(p_1)$. Hence $\leftrightarrows_{rd}$ is not a congruence.

Example 4 Let f be a unary function with an $\aleph$-liquid, Λ-frozen argument, defined by $\frac{x \xrightarrow{a} y}{f(x) \xrightarrow{b} 0}$. This TSS is negative delay resistant, but not positive delay resistant.

Consider the LTS from Ex. 3. We have $f(p_0) \xrightarrow{b} 0$, while $f(p_1)$ does not exhibit any transitions. So $f(p_0) \not\leftrightarrows_{rd} f(p_1)$. Hence $\leftrightarrows_{rd}$ is not a congruence.

The following example shows that the requirement that H^d is finite in Def. 12 of positive delay resistance is essential.

Example 5 $A = \{a_k \mid k \in \mathbb{Z}_{>0}\} \cup \{b\}$. Binary functions f_k for $k \in \mathbb{Z}_{>0}$, with both arguments $\aleph$-liquid and only the second argument Λ-liquid, are defined by

$$\frac{x_1 \xrightarrow{\tau} y}{f_k(x_1, x_2) \xrightarrow{\tau} f_\ell(x_1, y)} \text{ (for all } \ell > k)$$

$$\frac{x_2 \xrightarrow{\tau} y}{f_k(x_1, x_2) \xrightarrow{\tau} f_k(x_1, y)} \qquad \frac{\{x_1 \xrightarrow{a_\ell} y_\ell \mid \ell > k\} \cup \{x_2 \xrightarrow{a_k} y_k\}}{f_k(x_1, x_2) \xrightarrow{b} 0}$$

ω_1, ω_2, ω_3 are constants with $\omega_2 \xrightarrow{a_k} 0$ and $\omega_3 \xrightarrow{a_k} 0$ for all $k \geq 1$, $\omega_1 \xrightarrow{\tau} \omega_3$ and $\omega_2 \xrightarrow{\tau} \omega_3$. Clearly, $\omega_1 \leftrightarrows_{rd} \omega_2$. The last rule is not delay resistant. With regard to Def. 12 it violates the requirement that H^d needs to be chosen finite.

Prop. 2 is violated: although $\omega_1 \stackrel{\epsilon}{\Longrightarrow} \xrightarrow{a_\ell}$ for $\ell \geq 1$, there is no sequence $f_k(\omega_1, \omega_1) \stackrel{\epsilon}{\Longrightarrow} \xrightarrow{b}$ for any $k \geq 1$. Since $f_k(\omega_2, \omega_2) \xrightarrow{b} 0$ for all $k \geq 1$, $\leftrightarrows_{rd}$ is not a congruence.

5. Semi-syntactic criteria for delay resistance

Delay resistance of a TSS P means that each P-ruloid has to satisfy a property, and a non-trivial TSS has infinitely many ruloids. We now introduce requirements on the *rules* of P that imply delay resistance. This yields *semi-syntactic* congruence formats. They are not purely syntactic, as one of the conditions requires the existence of linearly provable rules.

Definition 20 A rule $\frac{H}{t \xrightarrow{\alpha} u}$ is *negative-stable* if for every premise $w \stackrel{\alpha}{\not\longrightarrow}$ in H, also $w \stackrel{\tau}{\not\longrightarrow}$ is in H.

The difference with Def. 13 is that here the requirement also applies to redundant premises $w \stackrel{\alpha}{\not\longrightarrow}$.

Definition 21 An ntytt rule is *manifestly delay resistant* w.r.t. a TSS P if it is negative-stable, as well as positive delay resistant w.r.t. P where the rules r_M from Def. 12 and $\frac{H_1}{t \xrightarrow{\tau} v}$ from Def. 11 are, up to bijective renaming of variables, rules of P, rather than just linearly provable from P.

Theorem 8 Let P be a standard TSS in decent ntyft format, in which each rule is manifestly delay resistant w.r.t. P. Then P is delay resistant.

6. Delay resistance w.r.t. Λ

The notion "delay resistant w.r.t. Λ" relaxes the notion of delay resistance: the conditions of Defs. 11, 12 and 13 need to be checked only for (sets of) premises that contain a variable that occurs only Λ-frozen in the source. This relaxation is possible if the following condition is added to our formats.

Definition 22 Given a standard ntytt rule $r = \frac{H}{t \xrightarrow{\alpha} u}$, we define the following syntactic condition:

5. If x has exactly one occurrence in t, which is Λ-liquid, and an $\aleph$-liquid occurrence in H, then these are the only two occurrences of x in r.

Proposition 9 Let P be a standard TSS in ready simulation format, in which each rule is rooted branching bisimulation safe and satisfies condition 5 of Def. 22 w.r.t. $\aleph$ and Λ. Then any P-ruloid satisfies condition 5 of Def. 22 w.r.t. $\aleph$ and Λ.

Theorem 9 Let P be a standard TSS in ready simulation format, in which each transition rule is rooted branching bisimulation safe and satisfies condition 5 of Def. 22 w.r.t. $\aleph$ and Λ. Let moreover P be $\aleph \cap \Lambda$-patient and delay resistant w.r.t. Λ. Then P is delay resistant.[1]

[1] Using a notion of *manifest delay resistance w.r.t. Λ and P*, Thms. 8 and 9 can be combined in the obvious way.

Hence, by adding condition 5 of Def. 22, the delay and weak bisimulation formats do not require delay resistance at all, since with Λ universal there are no Λ-frozen occurrences.

7. Syntactic criteria for delay resistance

We show how delay resistance can be replaced by additional syntactic requirements.

Definition 23 A standard TSS $P = (\Sigma, R)$ is in *syntactic rooted delay bisimulation format* if, for some $\aleph$ and Λ and predicates $\Delta_\alpha \subseteq \aleph \cap \Lambda$ where α ranges over $A \cup \{\tau\}$:

1. P is in decent nxytt format and $\aleph \cap \Lambda$-patient.
2. Each rule in R is rooted branching bisimulation safe and negative-stable, satisfies condition 5 of Def. 22 w.r.t. $\aleph$ and Λ, and has finitely many positive premises.
3. If R contains $\frac{H \uplus \{x_i \xrightarrow{\beta} y\}}{f(x_1,\ldots,x_{ar(f)}) \xrightarrow{\alpha} u}$ where $\neg\Lambda(f,i)$, then:
   (a) $\beta = \alpha$;
   (b) R contains $\frac{H' \cup \{x_i \xrightarrow{\tau} y\}}{f(x_1,\ldots,x_{ar(f)}) \xrightarrow{\tau} u}$ for some $H' \subseteq H$; and
   (c) y has exactly one, Δ_α-liquid occurrence in u.
4. R contains $\frac{x_i \xrightarrow{\alpha} y}{f(x_1,\ldots,x_i,\ldots,x_{ar(f)}) \xrightarrow{\alpha} f(x_1,\ldots,y,\ldots,x_{ar(f)})}$ for all $\Delta_\alpha(f,i)$.

P is in *syntactic rooted weak bisimulation format* if its rules moreover satisfy condition 1′ of Def. 22.

Theorem 10 If a standard TSS is in syntactic rooted delay bisimulation format, then it is delay resistant.

Proof Consider a rule $\frac{H \uplus \{x_i \xrightarrow{\beta} y\}}{f(x_1,\ldots,x_n) \xrightarrow{\alpha} u}$ of P with $\neg\Lambda(f,i)$. By condition 3a of Def. 23, $\beta = \alpha$. And by condition 3c of Def. 23, y has exactly one, Δ_α-liquid occurrence in u. By the combination of Thms. 8 and 9, it suffices to show that, for some term v and fresh variable z, there is a rule $\frac{H_1}{f(x_1,\ldots,x_n) \xrightarrow{\tau} v}$ in R with $H_1 \subseteq H \cup \{x_i \xrightarrow{\tau} z\}$ and a rule $\frac{H_2}{v \xrightarrow{\alpha} u}$ linearly provable from P with $H_2 \subseteq H \cup \{z \xrightarrow{\beta} y\}$. Let v be obtained by substituting z for y in u. The first of these rules exists by condition 3b of Def. 23, substituting z for y. The second is the rule $\frac{z \xrightarrow{\beta} y}{v \xrightarrow{\alpha} u}$, which can be derived by condition 4 of Def. 23. Here we use that $\beta = \alpha$ and y has exactly one, Δ_α-liquid occurrence in u. ∎

The introduction of predicates $\Delta_\alpha \subseteq \aleph \cap \Lambda$ is of practical importance. If in Def. 23 one would replace the occurrences of Δ_α by $\aleph \cap \Lambda$, then for instance the *encapsulation* operator ∂_H, which blocks all actions in the set H, would violate condition 4 of Def. 23. Namely, the argument of ∂_H is $\aleph \cap \Lambda$-liquid, but there is no rule $\frac{x \xrightarrow{a} y}{\partial_H(x) \xrightarrow{a} \partial_H(y)}$ if $a \in H$. Actually, in many applications Δ_α can be empty, as a rule that tests an $\aleph$-liquid, Λ-frozen argument of the source in practice tends to have a single y as right-hand side of the conclusion, so that condition 3c of Def. 23 is trivially satisfied; a notable example is the rule $\frac{x_1 \xrightarrow{\alpha} y}{x_1 + x_2 \xrightarrow{\alpha} y}$ for alternative composition.

Corollary 1 If a complete standard TSS P is in syntactic rooted delay (resp. weak) bisimulation format, then $\leftrightarrows_{rd}$ (resp. $\leftrightarrows_{rw}$) is a congruence for P.

8. Applications

We revisit some applications of our congruence formats that were considered in [9]: the basic process algebra $\text{BPA}_{\varepsilon\delta\tau}$, extended with binary Kleene star as an example where the predicates Δ_α from Def. 23 are non-empty, and initial priority which includes negative premises. We also consider a deadlock test outside the syntactic rooted delay bisimulation format. In all these cases our formats yield congruence results for $\leftrightarrows_{rd}$ and $\leftrightarrows_{rw}$, while they are outside the congruence formats for these semantics from [1, 14] (which are within the positive GSOS format [3]). The TSSs in this section are $\aleph\cap\Lambda$-patient and in decent xynft format.

Basic process algebra $\text{BPA}_{\varepsilon\delta\tau}$ consists of: actions from an alphabet $Act\cup\{\tau\}$; empty process ε; deadlock δ; alternative composition $t_1 + t_2$; sequential composition $t_1 \cdot t_2$. Let ℓ range over $Act \cup \{\tau\}$ and α over $Act \cup \{\tau, \surd\}$. Rules are:

$$\frac{}{\ell \xrightarrow{\ell} \varepsilon} \qquad \frac{}{\varepsilon \xrightarrow{\surd} \delta} \qquad \frac{x_1 \xrightarrow{\alpha} y}{x_1 + x_2 \xrightarrow{\alpha} y} \qquad \frac{x_2 \xrightarrow{\alpha} y}{x_1 + x_2 \xrightarrow{\alpha} y}$$

$$\frac{x_1 \xrightarrow{\ell} y}{x_1 \cdot x_2 \xrightarrow{\ell} y \cdot x_2} \qquad \frac{x_1 \xrightarrow{\surd} y_1 \quad x_2 \xrightarrow{\alpha} y_2}{x_1 \cdot x_2 \xrightarrow{\alpha} y_2}$$

To show that $\leftrightarrows_{rd}$ and $\leftrightarrows_{rw}$ are congruences, we argue that this TSS satisfies the conditions of Def. 23. In [9] it was shown that it is in rooted η-bisimulation format, with $\aleph$ and Λ defined as follows. Since the arguments of alternative and sequential composition can all execute immediately, $\aleph$ holds for all these arguments. Since only the first argument of sequential composition can contain running processes, it is the only argument for which Λ holds. With regard to condition 2 of Def. 23, we need to check that the rules satisfy condition 5 of Def. 22: only the two rules for sequential composition contain a Λ-liquid occurrence of a variable, x_1, in their source; and in both cases x_1 has only one other occurrence in the rule, in the left-hand side of a premise. Condition 3 of Def. 23 needs to be verified with regard to the two rules for alternative composition and the second rule for sequential composition, since in these rules a Λ-frozen argument of the source is tested in a premise. Condition 3 is satisfied for these rules, where we can take $\Delta_\gamma = \emptyset$ for all γ. Hence condition 4 is trivially satisfied. Concluding, by Cor. 1 $\leftrightarrows_{rd}$ and $\leftrightarrows_{rw}$ are congruences for $\text{BPA}_{\varepsilon\delta\tau}$.

Binary Kleene star ${t_1}^*t_2$ repeatedly executes t_1 until it executes t_2. This operational behaviour is captured by the following rules, which are added to the rules for $\text{BPA}_{\varepsilon\delta\tau}$.

$$\frac{x_1 \xrightarrow{\ell} y}{{x_1}^*x_2 \xrightarrow{\ell} y\cdot({x_1}^*x_2)} \qquad \frac{x_2 \xrightarrow{\alpha} y}{{x_1}^*x_2 \xrightarrow{\alpha} y}$$

Again, to show that $\leftrightarrows_{rd}$ and $\leftrightarrows_{rw}$ are congruences, we argue that the resulting TSS satisfies the conditions of Def. 23. In [9] it was shown that it is in rooted η-bisimulation format, if we take the arguments of binary Kleene star to be Λ-frozen and $\aleph$-liquid. Since these arguments are Λ-frozen, condition condition 5 of Def. 22 is trivially satisfied. Conditions 3(a,b) of Def. 23 are satisfied by both rules, and the second rule for binary Kleene star trivially satisfies condition 3(c). In view of this condition with regard to the first rule for binary Kleene star, we mark the first argument of sequential composition by Δ_ℓ for all $\ell \in Act \cup \{\tau\}$. No other arguments are marked by the Δ_γ. Condition 4 of Def. 23 is satisfied with respect to the Δ_γ. (For this last condition it is essential that the first argument of sequential composition is not marked by $\Delta_\surd$.) Concluding, by Cor. 1 $\leftrightarrows_{rd}$ and $\leftrightarrows_{rw}$ are congruences for $\text{BPA}_{\varepsilon\delta\tau}$ with binary Kleene star.

Initial priority Assume an ordering on actions. $\theta(t)$ executes the transitions of t, except that an initial transition $t \xrightarrow{\ell} t_1$ only gives rise to an initial transition $\theta(t) \xrightarrow{\ell} t_1$ if there does not exist an initial transition $t \xrightarrow{\ell'} t_2$ with $\ell < \ell'$.

$$\frac{x \xrightarrow{\ell} y \qquad x \not\xrightarrow{\ell'} \text{ for all } \ell' > \ell}{\theta(x) \xrightarrow{\ell} y} \qquad \frac{x \xrightarrow{\surd} y}{\theta(x) \xrightarrow{\surd} y}$$

The argument of initial priority is Λ-frozen and $\aleph$-liquid. In [9] it was observed that this TSS is in rooted η-bisimulation format, irrespective of the ordering on actions. If τ is greater than all actions in Act, then both rules are negative-stable, because instances of the first rule with a premise $x \not\xrightarrow{a}$ for some $a \in Act$ are guaranteed to also contain $x \not\xrightarrow{\tau}$. In fact it is sufficient to require $\forall\ell : (\exists\ell' : \ell' > \ell) \Rightarrow \tau > \ell$.

To show that $\leftrightarrows_{rd}$ and $\leftrightarrows_{rw}$ are congruences, we argue that the TSS satisfies the conditions of Def. 23. Condition 5 of Def. 22 is trivially satisfied by the rules for initial priority, because its argument is Λ-frozen. Condition 3 of Def. 23 is satisfied by both rules for initial priority, where we can take $\Delta_\gamma = \emptyset$ for all γ. In particular, condition 3(b) is satisfied by the first rule for initial priority, because this rule with $\ell = \tau$ contains no negative premises. Since the Δ_γ are empty, condition 4 of Def. 23 is trivially satisfied. Concluding, if τ is greater than all actions in Act, $\leftrightarrows_{rd}$ and $\leftrightarrows_{rw}$ are congruences for $\text{BPA}_{\epsilon\delta\tau}$ with initial priority.

If τ is smaller than some a in Act, then $\leftrightarrows_{rd}$ and $\leftrightarrows_{rw}$ are not congruences for $\text{BPA}_{\epsilon\delta\tau}$ with initial priority. For example, $\tau\cdot a \leftrightarrows_{rd} (\tau\cdot a)+a$. However, $\theta(\tau\cdot a) \xrightarrow{\tau} a$ cannot be mimicked by $\theta((\tau \cdot a) + a)$, as the latter term can only perform an a-transition to ε. So $\theta(\tau \cdot a) \not\leftrightarrows_{rw} \theta((\tau \cdot a) + a)$.

Deadlock testing Finally we give an example outside the format from Def. 23, but within the more general format of Thm. 5. The operator f tests if its argument is a deadlock.

$$\frac{x \xrightarrow{\alpha} y}{f(x) \xrightarrow{no} \delta} \qquad \frac{x \not\xrightarrow{\alpha} y \text{ for all } \alpha}{f(x) \xrightarrow{yes} \delta}$$

The argument of f is Λ-frozen and $\aleph$-liquid. Clearly the first rule violates condition 3 of Def. 23.

The TSS is in rooted η-bisimulation format. For both rules, we can take $H^d := \emptyset$, while $r_\emptyset$ is the rule itself. Furthermore, for the first rule, $r_{\{x \xrightarrow{\alpha} y\}}$ is $\frac{x \xrightarrow{\tau} y}{f(x) \xrightarrow{no} \delta}$. By Thm. 8 this suffices to conclude that the TSS is delay resistant. Hence, by Thms. 5 and 7 $\leftrightarrows_{rd}$ and $\leftrightarrows_{rw}$ are congruences for $\mathrm{BPA}_{\epsilon\delta\tau}$ extended with f.

9. Conclusions

In [14, 9] a method was presented to generalise any congruence format within ntyft format into a *two-tiered* variant. The two-tiered versions of the congruence formats we presented generalise the formats in [1, 14]. Ulidowski [22, 23, 24] proposed congruence formats for weak semantics with a different treatment of divergence; these formats cover the (non-initial) priority operator, for which $\leftrightarrows_{rw}$ is not a congruence. The TSSs of $\mathrm{BPA}_{\epsilon\delta\tau}$, binary Kleene star and deadlock testing in Sect. 8 are however outside those formats.

This research shows that it is worthwhile to study the interplay of structural operational semantics and modal logic. The modal characterisation of a process semantics turns out to be fundamental for its congruence properties. Admittedly, the whole story is quite technical and intricate. Partly this is because we build on a rich body of earlier work in the realm of structural operational semantics: the notions of well-supported proofs and complete TSSs from [13] (or actually [11] in logic programming); the ntyft format from [15, 4]; the transformation to ruloids, which for the main part goes back to [6]; and the work on modal decomposition and congruence formats from [2] and [9]. Moreover, the proofs underlying the technical developments, which have been omitted from this extended abstract, are quite intricate. They are included in the full version of this paper [7].

However, the bulk of this work can be reused in the context of other weak process semantics. This is witnessed by the fact that the congruence results for rooted delay and weak bisimilarity are obtained in an almost identical fashion, and build upon the congruence proof for rooted branching bisimilarity in [9]. The door is now open to derive congruence formats for a wide range of weak semantics. Further work is needed to cover the entire spectrum in [12]. Specifically, it would be interesting to extend the framework to divergence-sensitive semantics. For future research, it would also be interesting to see whether the bridge between modal logic and congruence formats could be employed in the realm of logics and semantics for e.g. probabilities and security. As a first step in this direction, in [10] the decomposition method for Hennessy-Milner logic was lifted to probabilistic systems.

References

[1] B. Bloom. *Structural operational semantics for weak bisimulations. Theor. Comput. Sci.* 146:25-68, 1995.

[2] B. Bloom, W. Fokkink, R. van Glabbeek. *Precongruence formats for decorated trace semantics. ACM TOCL* 5:26-78, 2004.

[3] B. Bloom, S. Istrail, A. Meyer. *Bisimulation can't be traced. J. ACM* 42:232-268, 1995.

[4] R. Bol, J.F. Groote. *The meaning of negative premises in transition system specifications. J. ACM* 43:863-914, 1996

[5] W. Fokkink. *Rooted branching bisimulation as a congruence. J. Comput. Syst. Sci.* 60:13-37, 2000.

[6] W. Fokkink, R. van Glabbeek. *Ntyft/ntyxt rules reduce to ntree rules. Inform. Comput.* 126:1-10, 1996.

[7] W. Fokkink, R. van Glabbeek. *Divide and congruence II: From decomposition of modal formulas to preservation of delay and weak bisimilarity.* Technical report 9351, NICTA, 2016. `http://arxiv.org/abs/1604.07530`.

[8] W. Fokkink, R. van Glabbeek, P. de Wind. *Compositionality of Hennessy-Milner logic by structural operational semantics. Theor. Comput. Sci.* 354:421-440, 2006.

[9] W. Fokkink, R. van Glabbeek, P. de Wind. *Divide and congruence: From decomposition of modal formulas to preservation of branching and η-bisimilarity. Inf. Comp.* 214:59-85, 2012.

[10] D. Gebler, W. Fokkink. *Compositionality of probabilistic Hennessy-Milner logic through structural operational semantics.* In *CONCUR*, LNCS 7454, pp. 395-409. Springer, 2012.

[11] A. van Gelder, K. Ross, J.S. Schlipf. *The well-founded semantics for general logic programs. J. ACM* 38:620-650, 1991.

[12] R. van Glabbeek. *The linear time-branching time spectrum II: The semantics of sequential systems with silent moves.* In *CONCUR*, LNCS 715, pp. 66-81. Springer, 1993.

[13] R. van Glabbeek. *The meaning of negative premises in transition system specifications II. J. Logic Algebr. Progr.* 60/61:229-258, 2004.

[14] R. van Glabbeek. *On cool congruence formats for weak bisimulations. Theor. Comput. Sci.*, 412:3283-3302, 2011.

[15] J.F. Groote. *Transition system specifications with negative premises. Theor. Comput. Sci.* 118:263-299, 1993.

[16] J.F. Groote, F. Vaandrager. *Structured operational semantics and bisimulation as a congruence. Inform. Comput.* 100:202-260, 1992.

[17] M. Hennessy, R. Milner. *Algebraic laws for non-determinism and concurrency. J. ACM* 32:137-161, 1985.

[18] K. Larsen, X. Liu. *Compositionality through an operational semantics of contexts. J. Logic Comput.* 1:761-795, 1991.

[19] R. Milner. *A modal characterisation of observable machine-behaviour.* In *CAAP*, LNCS 112, pp. 25-34. Springer, 1981.

[20] R. Milner. *Communication and Concurrency.* Prentice Hall, 1989.

[21] R. de Simone. *Higher-level synchronising devices in* MEIJE-*SCCS. Theor. Comput. Sci.* 37:245-267, 1985.

[22] I. Ulidowski. *Equivalences on observable processes.* In *LICS*, pp. 148-159, IEEE, 1992.

[23] I. Ulidowski, I. Phillips. *Ordered SOS rules and process languages for branching and eager bisimulations. Inform. Comput.*, 178:180-213, 2002.

[24] I. Ulidowski, S. Yuen. *Process languages for rooted eager bisimulation.* In *CONCUR*, LNCS 1877, pp. 275-289, Springer, 2000.