Divide and Congruence III: Stability & Divergence

Divide and Congruence III: Stability & Divergence

Wan Fokkink[1], Rob van Glabbeek[2], and Bas Luttik[3]

1 **Vrije Universiteit Amsterdam, The Netherlands**
2 **Data61, CSIRO, Sydney, Australia and**
University of New South Wales, Sydney, Australia
3 **Eindhoven University of Technology, The Netherlands**

Abstract

In two earlier papers we derived congruence formats for weak semantics on the basis of a decomposition method for modal formulas. The idea is that a congruence format for a semantics must ensure that the formulas in the modal characterisation of this semantics are always decomposed into formulas that are again in this modal characterisation. Here this work is extended with important stability and divergence requirements. Stability refers to the absence of a τ-transition. We show, using the decomposition method, how congruence formats can be relaxed for weak semantics that are stability-respecting. Divergence, which refers to the presence of an infinite sequence of τ-transitions, escapes the inductive decomposition method. We circumvent this problem by proving that a congruence format for a stability-respecting weak semantics is also a congruence format for its divergence-preserving counterpart.

1998 ACM Subject Classification F.3.2 Operational Semantics, F.4.1 Modal Logic

Keywords and phrases Structural Operational Semantics, Weak Semantics, Modal Logic

Digital Object Identifier 10.4230/LIPIcs.CONCUR.2017.15

1 Introduction

Structural operational semantics generates a labelled transition system, in which states are the closed terms over a signature, and transitions between states carry labels. Transitions are obtained from a transition system specification (TSS), consisting of proof rules called transition rules. States in labelled transition systems can be identified by a wide range of behavioural equivalences, based on e.g. branching structure or decorated versions of execution sequences. Weak semantics, which take into account the internal action τ, are classified in [11]. A significant number of the weak semantics based on a bisimulation relation carry a stability or divergence requirement. Stability refers to the absence of a τ-transition and divergence to the presence of an infinite sequence of τ-transitions.

In general a behavioural equivalence induced by a TSS is not guaranteed to be a congruence, i.e. the equivalence class of a term $f(p_1, \ldots, p_n)$ need not be determined by f and the equivalence classes of its arguments $p_1, \ldots, p_n$. Being a congruence is an important property, for instance in order to fit the equivalence into an axiomatic framework. Respecting stability or preserving divergence sometimes needs to be imposed in order to obtain a congruence relation, for example in case of the priority operator [1].

Modal logic captures observations an experimenter can make during a session with a process. 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. For instance, Hennessy-Milner logic [17] constitutes a modal characterisation of (strong) bisimilarity. A cornerstone for the current paper is the work in [3] to decompose formulas from Hennessy-Milner logic with respect to a structural operational semantics in

the ntyft format [16] without lookahead. Here the *decomposition* of a modal formula φ w.r.t. a term $t = f(p_1, \ldots, p_n)$ is a selection of n-tuples of modal formulas, one of which needs to be satisfied by the processes p_i in order for t to satisfy φ. Based on this method, formats for behavioural equivalences can be generated from their modal characterisation, ensuring that they are congruences. Such formats help to avoid repetitive congruence proofs and obtain insight into the congruence property. Key idea is that congruence is ensured if formulas from the modal characterisation C of an equivalence are always decomposed into formulas that are again in C. This approach was extended to weak semantics in [10, 7].

Here we expand the latter work to weak semantics that respect stability or preserve divergence. We focus on *branching bisimilarity* and *rooted branching bisimilarity* [15] and consider for each a stability-respecting and two divergence-preserving variants. Divergence-preserving branching bisimilarity [15] is the coarsest congruence relation for the parallel composition operator that only equates processes satisfying the same formulas from the well-known temporal logic CTL* minus the next-time operator X [14]. With regard to stability the expansion is relatively straightforward: we extend the modal characterisation of the semantics with one clause to capture that a semantics is stability-respecting, and study the decomposition of this additional clause. Next we show how the congruence formats for branching bisimilarity and rooted branching bisimilarity from [10] can be relaxed, owing to the extended modal characterisation for stability-respecting branching bisimilarity. Notably, the transition rules for the priority operator are within the more relaxed formats.

The divergence preservation property escapes the inductive decomposition method, as it concerns an infinite sequence of τ-transitions. We overcome this problem by a general framework for lifting congruence formats from a weak semantics $\approx$ to a finer semantics $\sim$. We show four applications of this method. In two cases $\approx$ is stability-respecting and in two cases rooted stability-respecting branching bisimilarity, while in $\sim$ stability is replaced by two different forms of divergence. Hence the congruence format for stability-respecting branching bisimilarity is also applicable to divergence-preserving as well as weakly divergence-preserving branching bisimilarity; and likewise for their rooted counterparts.

2 Preliminaries

2.1 Stability-respecting / divergence-preserving branching bisimilarity

A *labelled transition system (LTS)* is a triple $(\mathbb{P}, Act, \rightarrow)$, with $\mathbb{P}$ a set of *processes*, Act a set of *actions*, and $\rightarrow \subseteq \mathbb{P} \times Act \times \mathbb{P}$. We normally let $Act = A \cup \{\tau\}$ where τ is an *internal action* and A some set of external or observable actions not containing τ. We write A_τ for $A \cup \{\tau\}$. We use p, q to denote processes, α, β, γ for elements of A_τ, and a, b for elements of A. We write $p \xrightarrow{\alpha} q$ for $(p, \alpha, q) \in \rightarrow$, $p \xrightarrow{\alpha}$ for $\exists q \in \mathbb{P} : p \xrightarrow{\alpha} q$, and $p \not\xrightarrow{\alpha}$ for $\neg(p \xrightarrow{\alpha})$. Furthermore, $\stackrel{\epsilon}{\Longrightarrow}$ denotes the transitive-reflexive closure of $\xrightarrow{\tau}$.

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

- $\mathcal{B}$ is a *branching bisimulation* if $p \mathcal{B} q$ and $p \xrightarrow{\alpha} p'$ implies either $\alpha = \tau$ and $p' \mathcal{B} q$, or $q \stackrel{\epsilon}{\Longrightarrow} q' \xrightarrow{\alpha} q''$ for some q' and q'' with $p \mathcal{B} q'$ and $p' \mathcal{B} q''$.
- $\mathcal{B}$ is *stability-respecting* if $p \mathcal{B} q$ and $p \not\xrightarrow{\tau}$ implies $q \stackrel{\epsilon}{\Longrightarrow} q' \not\xrightarrow{\tau}$ for some q' with $p \mathcal{B} q'$.
- $\mathcal{B}$ is *divergence-preserving* if it satisfies the following condition:

  (D) if $p \mathcal{B} q$ and there is an infinite sequence of processes $(p_k)_{k \in \mathbb{N}}$ with $p = p_0$, $p_k \xrightarrow{\tau} p_{k+1}$ and $p_k \mathcal{B} q$ for all $k \in \mathbb{N}$, then there exists an infinite sequence of processes $(q_\ell)_{\ell \in \mathbb{N}}$ with $q = q_0$, $q_\ell \xrightarrow{\tau} q_{\ell+1}$ for all $\ell \in \mathbb{N}$, and $p_k \mathcal{B} q_\ell$ for all $k, \ell \in \mathbb{N}$.

  The definition of a *weakly divergence-preserving* relation is obtained by omitting the condition "and $p_k \mathcal{B} q_\ell$ for all $k, \ell \in \mathbb{N}$."

Processes p, q are *branching bisimilar*, denoted $p \leftrightarrows_b q$, if there exists a branching bisimulation $\mathcal{B}$ with $p \mathcal{B} q$. They are *stability-respecting*, *divergence-preserving* or *weakly divergence-preserving* branching bisimilar, denoted $p \leftrightarrows_b^s q$, $p \leftrightarrows_b^\Delta q$ or $p \leftrightarrows_b^{\Delta^\top} q$, if moreover $\mathcal{B}$ is stability-respecting, divergence-preserving or weakly divergence-preserving, respectively.

We have $\leftrightarrows_b \supset \leftrightarrows_b^s \supset \leftrightarrows_b^{\Delta^\top} \supset \leftrightarrows_b^\Delta$. The relations $\leftrightarrows_b$, $\leftrightarrows_b^s$, $\leftrightarrows_b^\Delta$ and $\leftrightarrows_b^{\Delta^\top}$ are equivalences [2, 11, 13]. However, they are not *congruences* with respect to most process algebras from the literature, meaning that the equivalence class of a process $f(p_1, \ldots, p_n)$, with f an n-ary function symbol, is not always determined by the equivalence classes of its arguments, i.e. the processes $p_1, \ldots, p_n$. Therefore an additional rootedness condition is imposed.

▶ **Definition 2.** *Rooted* branching bisimilarity, $\leftrightarrows_{rb}$, is the largest symmetric relation on $\mathbb{P}$ with $p \leftrightarrows_{rb} q$ and $p \xrightarrow{\alpha} p'$ implies $q \xrightarrow{\alpha} q'$ for some q' with $p' \leftrightarrows_b q'$. Likewise, *rooted* stability-respecting, divergence-preserving or weakly divergence-preserving branching bisimilarity, denoted by $p \leftrightarrows_{rb}^s q$, $p \leftrightarrows_{rb}^\Delta q$ or $p \leftrightarrows_{rb}^{\Delta^\top} q$, is the largest symmetric relation $\mathcal{R}$ on $\mathbb{P}$ with $p \mathcal{R} q$ and $p \xrightarrow{\alpha} p'$ implies $q \xrightarrow{\alpha} q'$ for some q' with $p' \leftrightarrows_b^s q'$, $p' \leftrightarrows_b^\Delta q'$ or $p' \leftrightarrows_b^{\Delta^\top} q'$.

The various notions of bisimilarity defined above are examples of so-called behavioural equivalences. For a general formulation of the results in Section 4, it is convenient to formally define a notion of behavioural equivalence that includes at least the examples above. Note that a common feature of their definitions is that they associate with every LTS $\mathrm{G} = (\mathbb{P}_\mathrm{G}, Act_\mathrm{G}, \rightarrow_\mathrm{G})$ a binary relation $\sim_\mathrm{G}$. (For instance, in the case of branching bisimilarity, the relation $\sim_\mathrm{G}$ associated with G is defined as the binary relation $\leftrightarrows_b \subseteq \mathbb{P}_\mathrm{G} \times \mathbb{P}_G$ such that $p \leftrightarrows_b q$ if (and only if) there exists a branching bisimulation $\mathcal{B} \subseteq \mathbb{P}_\mathrm{G} \times \mathbb{P}_G$ with $p \mathcal{B} q$.) One may, thus, think of a behavioural equivalence as a family of binary relations indexed by LTSs. It turns out that we need to impose just one extra condition on such families to arrive at a suitable formalisation of the notion of behavioural equivalence. The condition states that the relation associated with the disjoint union of two LTSs restricted to one of the components coincides with the relation associated with that component.

Two LTSs $\mathrm{G} = (\mathbb{P}_\mathrm{G}, Act_\mathrm{G}, \rightarrow_\mathrm{G})$ and $\mathrm{H} = (\mathbb{P}_\mathrm{H}, Act_\mathrm{H}, \rightarrow_\mathrm{H})$ are called *disjoint* if $\mathbb{P}_\mathrm{G} \cap \mathbb{P}_\mathrm{H} = \emptyset$. In that case $\mathrm{G} \uplus \mathrm{H}$ denotes their union $(\mathbb{P}_\mathrm{G} \cup \mathbb{P}_\mathrm{H}, Act_\mathrm{G} \cup Act_\mathrm{H}, \rightarrow_\mathrm{G} \cup \rightarrow_\mathrm{H})$.

A *behavioural equivalence* $\sim$ on LTSs is a family of equivalence relations $\sim_\mathrm{G}$, one for every LTS G, such that for each pair of disjoint LTSs $\mathrm{G} = (\mathbb{P}_\mathrm{G}, Act_\mathrm{G}, \rightarrow_\mathrm{G})$ and H we have $g \sim_\mathrm{G} g' \Leftrightarrow g \sim_{\mathrm{G}\uplus\mathrm{H}} g'$ for any $g, g' \in \mathbb{P}_\mathrm{G}$. The notions of bisimilarity defined above clearly qualify as behavioural equivalences. We write $\sim \subseteq \approx$ iff $\sim_\mathrm{G} \subseteq \approx_\mathrm{G}$ for each LTS G.

2.2 Modal logic

Modal logic formulas express behavioural properties of processes. Following [11], we extend Hennessy-Milner logic [17] with connectives $\langle\epsilon\rangle\varphi$ and $\langle\hat{\tau}\rangle\varphi$, expressing that a process can perform zero or more, respectively zero or one, τ-transitions to a process where φ holds.

▶ **Definition 3.** The class $\mathbb{O}$ of *modal formulas* is defined as follows, where I ranges over all index sets and α over A_τ: $\varphi ::= \bigwedge_{i\in I} \varphi_i \mid \neg\varphi \mid \langle\alpha\rangle\varphi \mid \langle\epsilon\rangle\varphi \mid \langle\hat{\tau}\rangle\varphi$. We write $\top$ for the empty conjunction, $\varphi_1 \wedge \varphi_2$ for $\bigwedge_{i\in\{1,2\}} \varphi_i$, $\varphi\langle\alpha\rangle\varphi'$ for $\varphi \wedge \langle\alpha\rangle\varphi'$, and $\varphi\langle\hat{\tau}\rangle\varphi'$ for $\varphi \wedge \langle\hat{\tau}\rangle\varphi'$.

$p \models \varphi$ denotes that process p satisfies formula φ. The first two operators represent the standard Boolean operators conjunction and negation. By definition, $p \models \langle\alpha\rangle\varphi$ if $p \xrightarrow{\alpha} p'$ for some p' with $p' \models \varphi$, $p \models \langle\epsilon\rangle\varphi$ if $p \stackrel{\epsilon}{\Longrightarrow} p'$ for some p' with $p' \models \varphi$, and $p \models \langle\hat{\tau}\rangle\varphi$ if either $p \models \varphi$ or $p \xrightarrow{\tau} p'$ for some p' with $p' \models \varphi$.

For each $L \subseteq \mathbb{O}$, we write $p \sim_L q$ if p and q satisfy the same formulas in L. We say that L is a *modal characterisation* of some behavioural equivalence $\sim$ if $\sim_L$ coincides with $\sim$. We write $\varphi \equiv \varphi'$ if $p \models \varphi \Leftrightarrow p \models \varphi'$ for all processes p. The class $L^{\equiv}$ denotes the closure of $L \subseteq \mathbb{O}$ under $\equiv$. Trivially, $p \sim_L q \Leftrightarrow p \sim_{L^{\equiv}} q$.

▶ **Definition 4** ([11]). The subclasses $\mathbb{O}_b$ and $\mathbb{O}_{rb}$ of $\mathbb{O}$ are defined as follows, where a ranges over A and α over A_τ:

$$\begin{array}{lllll} \mathbb{O}_b & \varphi & ::= & \bigwedge_{i\in I} \varphi_i \mid \neg\varphi \mid \langle\epsilon\rangle(\varphi\langle\hat{\tau}\rangle\varphi) \mid \langle\epsilon\rangle(\varphi\langle a\rangle\varphi) & \\ \mathbb{O}_{rb} & \overline{\varphi} & ::= & \bigwedge_{i\in I} \overline{\varphi}_i \mid \neg\overline{\varphi} \mid \langle\alpha\rangle\varphi \mid \varphi & (\varphi \in \mathbb{O}_b) \end{array}$$

$\mathbb{O}_b$ and $\mathbb{O}_{rb}$ are modal characterisations of $\leftrightarrows_b$ and $\leftrightarrows_{rb}$, respectively (see [10]).

The idea behind stability is: (I) if $p \leftrightarrows_b^s q$ and $p \stackrel{\epsilon}{\Longrightarrow} p' \stackrel{\tau}{\not\longrightarrow}$ with $p \leftrightarrows_b^s p'$, then $q \stackrel{\epsilon}{\Longrightarrow} q' \stackrel{\tau}{\not\longrightarrow}$ with $q \leftrightarrows_b^s q'$. In the definition of $\leftrightarrows_b^s$ this was formulated more weakly: (II) if $p \leftrightarrows_b^s q$ and $p \stackrel{\tau}{\not\longrightarrow}$, then $q \stackrel{\epsilon}{\Longrightarrow} q' \stackrel{\tau}{\not\longrightarrow}$ with $q \leftrightarrows_b^s q'$. It is easy to see that formulations (I) and (II) are equivalent. The additional clause in the following modal characterisation of stability-respecting branching bisimilarity is based on (I).

▶ **Definition 5** ([11]). The subclasses $\mathbb{O}_b^s$ and $\mathbb{O}_{rb}^s$ of $\mathbb{O}$ are defined as follows:

$$\begin{array}{lllll} \mathbb{O}_b^s & \varphi & ::= & \bigwedge_{i\in I} \varphi_i \mid \neg\varphi \mid \langle\epsilon\rangle(\varphi\langle\hat{\tau}\rangle\varphi) \mid \langle\epsilon\rangle(\varphi\langle a\rangle\varphi) \mid \langle\epsilon\rangle(\neg\langle\tau\rangle\top \wedge \overline{\varphi}) & (\overline{\varphi} \in \mathbb{O}_{rb}^s) \\ \mathbb{O}_{rb}^s & \overline{\varphi} & ::= & \bigwedge_{i\in I} \overline{\varphi}_i \mid \neg\overline{\varphi} \mid \langle\alpha\rangle\varphi \mid \varphi & (\varphi \in \mathbb{O}_b^s) \end{array}$$

The additional clause $\langle\epsilon\rangle(\neg\langle\tau\rangle\top \wedge \overline{\varphi})$ in $\mathbb{O}_b^s$ expresses stability. The first part $\langle\epsilon\rangle(\neg\langle\tau\rangle\top \ldots)$ captures $p \stackrel{\epsilon}{\Longrightarrow} p' \stackrel{\tau}{\not\longrightarrow}$, while the second part $\ldots \wedge \overline{\varphi}$ captures the stability-respecting branching bisimulation class of p'. Since $p' \stackrel{\tau}{\not\longrightarrow}$, unrooted and rooted stability-respecting branching bisimilarity coincide for p', which allows us to take the second part from $\mathbb{O}_{rb}^s$.

▶ **Theorem 6.** *$p \leftrightarrows_b^s q \Leftrightarrow p \sim_{\mathbb{O}_b^s} q$ and $p \leftrightarrows_{rb}^s q \Leftrightarrow p \sim_{\mathbb{O}_{rb}^s} q$, for all $p, q \in \mathbb{P}$.*

All proofs omitted from the paper can be found in [8].

2.3 Structural operational semantics

A *signature* is a set Σ of function symbols f with arity $ar(f)$. A function symbol of arity 0 is called a *constant*. Let V be an infinite set of variables; we assume $|\Sigma|, |A| \leq |V|$. A syntactic object is *closed* if it does not contain any variables. The sets $\mathbb{T}(\Sigma)$ and $\mathrm{T}(\Sigma)$ of terms over Σ and V and closed terms over Σ, respectively, are defined as usual; t, u, v, w denote terms, p, q denote closed terms, and $var(t)$ is the set of variables that occur in term t. 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. The domain of substitutions is extended to $\mathbb{T}(\Sigma)$ as usual.

Structural operational semantics generates an LTS in which the processes are the closed terms. The labelled transitions between processes are obtained from a transition system specification, which consists of a set of proof rules called transition rules.

A (*positive* or *negative*) *literal* is an expression $t \xrightarrow{\alpha} u$ or $t \stackrel{\alpha}{\not\longrightarrow}$. 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 terms at the left- and right-hand side of λ are called the *source* and *target*. A rule $\frac{\emptyset}{\lambda}$ is also written λ. A rule is *standard* if it has a positive conclusion. A *transition system specification (TSS)*, written (Σ, Act, R), consists of a signature Σ, a set of actions Act, and a set R of transition rules over Σ. A TSS is *standard* if all its rules are.

A TSS specifies an LTS in which the transitions are the closed positive literals that can be proved using the rules of the TSS. Since rules may have negative premises, consistency is a concern. Literals $t \xrightarrow{\alpha} u$ and $t \stackrel{\alpha}{\not\longrightarrow}$ are said to *deny* each other; a notion of provability

associated with TSSs is *consistent* if it is not possible to prove two literals that deny each other. To arrive at a consistent notion of provability, we proceed in two steps: first we define the notion of an *irredundant proof*, which on a standard TSS does not allow the derivation of negative literals at all, and then arrive at a notion of *well-supported proof* that allows the derivation of negative literals whose denials are manifestly underivable by irredundant proofs. In [12] it was shown that the notion of well-supported provability is consistent.

▶ **Definition 7** ([3]). Let $P = (\Sigma, Act, R)$ 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 rule $\frac{H}{\lambda}$ is irredundantly provable from P, notation $P \vdash_{irr} \frac{H}{\lambda}$, if such a proof exists.

▶ **Definition 8** ([12]). Let $P = (\Sigma, Act, R)$ be a 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:

1. either μ is positive and $\frac{K}{\mu}$ is a closed substitution instance of a rule in R;
2. 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 denying 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 $P \vdash_{ws} p \stackrel{\alpha}{\not\longrightarrow}$ or $P \vdash_{ws} p \stackrel{\alpha}{\longrightarrow} q$ for some closed term q.

If $P = (\Sigma, Act, R)$ is a complete TSS, then the LTS *associated with* P is $(\mathrm{T}(\Sigma), Act, \rightarrow)$ with $\rightarrow = \{(p, \alpha, q) \mid P \vdash_{ws} p \stackrel{\alpha}{\longrightarrow} q\}$. We do not associate an LTS with an incomplete TSS.

2.4 Congruence formats

Let $P = (\Sigma, Act, R)$ be a transition system specification, and let $\sim_P$ be an equivalence relation defined on the set of closed terms $\mathrm{T}(\Sigma)$. Then $\sim_P$ is a *congruence* for P if, for each $f \in \Sigma$, we have that $p_i \sim_P q_i$ implies $f(p_1, \ldots, p_{ar(f)}) \sim_P f(q_1, \ldots, q_{ar(f)})$. Note that this is the case if for each open term $t \in \mathbb{T}(\Sigma)$ and each pair of closed substitutions $\rho, \rho' : V \to \mathrm{T}(\Sigma)$ we have $(\forall x \in var(t).\ \rho(x) \sim_P \rho'(x)) \Rightarrow \rho(t) \sim_P \rho'(t)$.

Every complete TSS generates an LTS of which the states are the closed terms of the TSS. Thus each behavioural equivalence $\sim$ associates with every TSS P an equivalence $\sim_P$ on its set of closed terms. By a *congruence format* for $\sim$ we mean a class of TSSs such that for every TSS P in the class the equivalence $\sim_P$ is a congruence. Usually, a congruence format is defined by means of a list of syntactic restrictions on the rules of TSSs.

In an *ntytt rule*, right-hand sides of positive premises are distinct variables 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. 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 rule is *decent* if it has no lookahead and does not contain free variables. Each combination of syntactic restrictions on rules induces a format for TSSs of the same name. For instance, a TSS is in decent ntyft format if it contains decent ntyft rules only. A TSS is in *ready simulation format* if it consists of ntyft and ntyxt rules that have no lookahead.

In congruence formats for weak semantics, lookahead must be forbidden. To see this, consider CCS [18] extended with an operator f defined by $\frac{x \xrightarrow{a} y \quad y \xrightarrow{b} z}{f(x) \xrightarrow{c} z}$. Then $ab\mathbf{0} \leftrightarrows_{rb} a\tau b\mathbf{0}$, whereas $f(ab\mathbf{0}) \not\leftrightarrows_{rb} f(a\tau b\mathbf{0})$. Therefore congruence formats for weak semantics are generally obtained by imposing additional restrictions on the ready simulation format.

2.5 Decomposition of modal formulas

The decomposition method from [10] gives a special treatment to arguments of function symbols that are deemed *patient*; a predicate marks these arguments. Let Γ be a predicate on arguments of function symbols. A standard ntyft rule is a Γ-*patience rule* [6] if it is of the form $\frac{x_i \xrightarrow{\tau} y}{f(x_1,\ldots,x_{ar(f)}) \xrightarrow{\tau} f(x_1,\ldots,x_{i-1},y,x_{i+1},\ldots,x_{ar(f)})}$ with $\Gamma(f,i)$. 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 called Γ-*impatient*. Let Γ be a predicate on $\{(f,i) \mid 1 \leq i \leq ar(f),\ f \in \Sigma\}$. If $\Gamma(f,i)$, then argument i of f is Γ-*liquid*; otherwise it is Γ-*frozen* [3]. An occurrence of x in t is Γ-*liquid* if either $t = x$, or $t = f(t_1,\ldots,t_{ar(f)})$ and the occurrence is Γ-liquid in t_i for a liquid argument i of f; otherwise the occurrence is Γ-*frozen*.

To each term t and formula φ we assign a set $t^{-1}(\varphi)$ of decomposition mappings $\psi : V \to \mathbb{O}$, such that $\rho(t) \models \varphi$ iff there is a $\psi \in t^{-1}(\varphi)$ with $\rho(x) \models \psi(x)$ for all $x \in var(t)$. To define $t^{-1}(\varphi)$, we use a result from [3], where for each standard TSS P in ready simulation format a collection of decent nxytt rules, called P-*ruloids*, is constructed. First, P is converted into a non-standard TSS P^+ with the property that, for all closed literals μ, we have $P \vdash_{ws} \mu$ if and only if μ is irredundantly provable from P^+. The P-ruloids are the decent nxytt rules irredundantly provable from P^+. In [3] is was proved that there is a well-supported proof from P of a transition $\rho(t) \xrightarrow{a} q$, with ρ a closed substitution, if and only if there is a proof of this transition that uses at the root a P-ruloid with source t.

▶ **Definition 9** ([10])**.** Let $P = (\Sigma, A_\tau, R)$ 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 smallest set $t^{-1}(\varphi) \in \mathcal{P}(V \to \mathbb{O})$ of decomposition mappings $\psi : V \to \mathbb{O}$ satisfying the following six conditions. Let t denote a univariate term, i.e. without multiple occurrences of the same variable. (Cases 1–5 associate with every univariate term t a set $t^{-1}(\varphi)$. In Case 6, the definition is generalised to terms that are not univariate, using that every term can be obtained by applying a non-injective substitution to 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 $i \in I$ with $\psi(x) = \bigwedge_{i\in I} \psi_i(x)$ for $x \in V$;
2. $\psi \in t^{-1}(\neg\varphi)$ iff there is a function $h : t^{-1}(\varphi) \to var(t)$ such that $\psi(x) = \bigwedge_{\chi\in h^{-1}(x)} \neg\chi(x)$ if $x \in var(t)$, and $\psi(x) = \top$ otherwise;
3. $\psi \in t^{-1}(\langle\alpha\rangle\varphi)$ iff there is a P-ruloid $\frac{H}{t \xrightarrow{\alpha} u}$ and a $\chi \in u^{-1}(\varphi)$ such that $\psi(x) = \chi(x) \wedge \bigwedge_{x \xrightarrow{\beta} y\in H} \langle\beta\rangle\chi(y) \wedge \bigwedge_{x \xrightarrow{\gamma}\not\in H} \neg\langle\gamma\rangle\top$ if $x \in var(t)$, and $\psi(x) = \top$ otherwise;
4. $\psi \in t^{-1}(\langle\epsilon\rangle\varphi)$ iff one of the following holds:
   a. either there is a $\chi \in t^{-1}(\varphi)$ such that $\psi(x) = \langle\epsilon\rangle\chi(x)$ if x occurs Γ-liquid in t, and $\psi(x) = \chi(x)$ otherwise;
   b. or there is a Γ-impatient P-ruloid $\frac{H}{t \xrightarrow{\tau} u}$ and a $\chi \in u^{-1}(\langle\epsilon\rangle\varphi)$ such that $\psi(x) = \top$ if $x \notin var(t)$, $\psi(x) = \langle\epsilon\rangle\Big(\chi(x) \wedge \bigwedge_{x \xrightarrow{\beta} y\in H} \langle\beta\rangle\chi(y) \wedge \bigwedge_{x \xrightarrow{\gamma}\not\in H} \neg\langle\gamma\rangle\top\Big)$ if x occurs Γ-liquid in t, and $\psi(x) = \chi(x) \wedge \bigwedge_{x \xrightarrow{\beta} y\in H} \langle\beta\rangle\chi(y) \wedge \bigwedge_{x \xrightarrow{\gamma}\not\in H} \neg\langle\gamma\rangle\top$ otherwise;
5. $\psi \in t^{-1}(\langle\hat{\tau}\rangle\varphi)$ iff one of the following holds:
   a. either $\psi \in t^{-1}(\varphi)$;

b. or there is an x_0 that occurs Γ-liquid in t, and a $\chi \in t^{-1}(\varphi)$ such that $\psi(x) = \langle\hat{\tau}\rangle\chi(x)$ if $x = x_0$, and $\psi(x) = \chi(x)$ otherwise;
c. or there is a Γ-impatient P-ruloid $\frac{H}{t \xrightarrow{\tau} u}$ and a $\chi \in u^{-1}(\varphi)$ such that $\psi(x) = \chi(x) \wedge \bigwedge_{x \xrightarrow{\beta} y \in H} \langle\beta\rangle\chi(y) \wedge \bigwedge_{x \xrightarrow{\gamma} \not\in H} \neg\langle\gamma\rangle\top$ if $x \in var(t)$, and $\psi(x) = \top$ otherwise;

6. $\psi \in \sigma(t)^{-1}(\varphi)$ 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 $x \in V$.

The following theorem will be the key to the forthcoming congruence results.

▶ **Theorem 10** ([10]). *Let $P = (\Sigma, A_\tau, R)$ be a Γ-patient complete standard TSS in ready simulation format. For each term $t \in \mathbb{T}(\Sigma)$, closed substitution ρ, and $\varphi \in \mathbb{O}$:*

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

3 Stability-respecting branching bisimilarity as a congruence

We proceed to apply the decomposition method from the previous section to derive congruence formats for stability-respecting branching bisimilarity and rooted stability-respecting branching bisimilarity. The idea behind the construction of these congruence formats is that the format must guarantee that a formula from the characterising logic of the equivalence under consideration is always decomposed into formulas from this same logic (see Proposition 14). This implies the desired congruence results (see Theorem 15 and Theorem 16).

The definitions of the congruence formats for (rooted) stability-respecting branching bisimilarity, below, presuppose two predicates Λ and $\aleph$ on the arguments of the function symbols of a TSS: Λ marks arguments that contain processes that have started executing, while $\aleph$ marks arguments that contain processes that can execute immediately. For example, in process algebra, Λ and $\aleph$ typically hold for the arguments of the merge $t_1 \| t_2$, and for the first argument of sequential composition $t_1 \cdot t_2$, and do not hold for the second argument of sequential composition. Λ does not hold and $\aleph$ holds for the arguments of alternative composition $t_1 + t_2$. We will instantiate Γ (from Section 2.5) with $\aleph \cap \Lambda$.

We define when a standard ntytt rule is rooted stability-respecting branching bisimulation safe, and base the rooted stability-respecting branching bisimulation format on that notion. The stability-respecting branching bisimulation format is defined by adding one additional restriction to its rooted counterpart: Λ holds for all arguments.

▶ **Definition 11.** A standard ntytt rule $r = \frac{H}{t \xrightarrow{\alpha} u}$ is *rooted stability-respecting branching bisimulation safe* w.r.t. predicates $\aleph$ and Λ if it satisfies the following conditions.

1. Right-hand sides of positive premises occur only Λ-liquid in u.
2. If $x \in var(t)$ occurs only Λ-liquid in t, then x occurs only Λ-liquid in r.
3. If $x \in var(t)$ occurs only $\aleph$-frozen in t, then x occurs only $\aleph$-frozen in H.
4. Suppose x has exactly one $\aleph$-liquid occurrence in t, and this occurrence is also Λ-liquid.
a. If x has an $\aleph$-liquid occurrence in a negative premise in H or more than one $\aleph$-liquid occurrence in the positive premises in H, then there is a premise $v \xrightarrow{\tau} \not\in H$ such that x occurs $\aleph$-liquid in v.
b. If there is a premise $w \xrightarrow{\tau} y$ in H and x occurs $\aleph$-liquid in w, then r is $\aleph \cap \Lambda$-patient.

Conditions 1–3 were copied from the definition of rooted branching bisimulation safeness from [10], and condition 4b is part of condition 4 in that definition. Condition 4a, however, establishes a relaxation of condition 4 from the definition in [10], where it is required that x has at most one $\aleph$-liquid occurrence in H, which must be in a positive premise. Here, owing to stability, we can be more tolerant, as long as $x \xrightarrow{\tau} \not\in$ can be derived. As a consequence

of this relaxation the rule for the priority operator is rooted stability-respecting branching bisimulation safe, while it is not rooted branching bisimulation safe.

▶ **Definition 12.** A standard TSS is in *rooted stability-respecting branching bisimulation format* if it is in ready simulation format and, for some $\aleph$ and Λ, it is $\aleph\cap\Lambda$-patient and its rules are all rooted stability-respecting branching bisimulation safe w.r.t. $\aleph$ and Λ. This TSS is in *stability-respecting branching bisimulation format* if moreover Λ is universal.

Since the definition of modal decomposition is based on the P-ruloids, we must verify that if P is in rooted stability-respecting branching bisimulation format, then P-ruloids are rooted stability-respecting branching bisimulation safe.

▶ **Proposition 13.** *Let P be an $\aleph\cap\Lambda$-patient TSS in ready simulation format, in which each rule is rooted stability-respecting branching bisimulation safe w.r.t. $\aleph$ and Λ. Then each P-ruloid is rooted stability-respecting branching bisimulation safe w.r.t. $\aleph$ and Λ.*

Consider a standard TSS in rooted stability-respecting branching bisimulation format, w.r.t. some $\aleph$ and Λ. Definition 9 yields decomposition mappings $\psi \in t^{-1}(\varphi)$, with $\Gamma := \aleph\cap\Lambda$. We now prove that if $\varphi \in \mathbb{O}_b^s$, then $\psi(x) \in \mathbb{O}_b^{s\equiv}$ if x occurs only Λ-liquid in t. (That is why in the stability-respecting branching bisimulation format, Λ must be universal.) Furthermore, we prove that if $\varphi \in \mathbb{O}_{rb}^s$, then $\psi(x) \in \mathbb{O}_{rb}^{s\equiv}$ for all x.

▶ **Proposition 14.** *Let P be an $\aleph\cap\Lambda$-patient standard TSS in ready simulation format, in which each rule is rooted stability-respecting branching bisimulation safe w.r.t. $\aleph$ and Λ.*
1. *For each term t and x that occurs only Λ-liquid in t: $\varphi \in \mathbb{O}_b^s \Rightarrow \forall \psi \in t^{-1}(\varphi) : \psi(x) \in \mathbb{O}_b^{s\equiv}$.*
2. *For each term t and variable x: $\overline{\varphi} \in \mathbb{O}_{rb}^s \;\Rightarrow\; \forall \psi \in t^{-1}(\overline{\varphi}) : \psi(x) \in \mathbb{O}_{rb}^{s\equiv}$.*

Now the promised congruence results for $\leftrightarrows_b^s$ and $\leftrightarrows_{rb}^s$ can be proved in the same way as their counterparts for $\leftrightarrows_b$ and $\leftrightarrows_{rb}$ in [10].

▶ **Theorem 15.** *Let P be a complete standard TSS in stability-respecting branching bisimulation format. Then $\leftrightarrows_b^s$ is a congruence for P.*

▶ **Theorem 16.** *Let P be a complete standard TSS in rooted stability-respecting branching bisimulation format. Then $\leftrightarrows_{rb}^s$ is a congruence for P.*

The *priority* operator [1] is a unary function the definition of which is based on an ordering $<$ on atomic actions. The term $\Theta(p)$ executes the transitions of the term p, with the restriction that a transition $p \xrightarrow{a} q$ only gives rise to a transition $\Theta(p) \xrightarrow{a} \Theta(q)$ if there does not exist a transition $p \xrightarrow{b} q'$ with $b > a$. This intuition is captured by the rule $\dfrac{x \xrightarrow{a} y \quad x \xrightarrow{b}\!\!\!\!\!\!/\;\;\; \text{for all } b>a}{\Theta(x) \xrightarrow{a} \Theta(y)}$.

In view of the target $\Theta(y)$, by condition 1 of Definition 11, the argument of Θ must be chosen Λ-liquid. And in view of condition 3 of Definition 11, the argument of Θ must be $\aleph$-liquid. The rule above is rooted stability-respecting branching bisimulation safe, if the following condition on the ordering on atomic actions is satisfied: if $b > a$, then $\tau > a$. Namely, this guarantees condition 4a of Definition 11: if there is a premise $x \xrightarrow{b}\!\!\!\!\!\!/\;\;\;$, then there is also a premise $x \xrightarrow{\tau}\!\!\!\!\!\!/\;\;\;$.

▶ **Corollary 17.** *$\leftrightarrows_b^s$ and $\leftrightarrows_{rb}^s$ are congruences for the priority operator.*

The priority operator Θ does not preserve $\leftrightarrows_{rb}$ (cf. [22, pp. 130–132]). So inevitably, as observed in [10], the rule for Θ is not in the rooted branching bisimulation format. Namely, the $\aleph\cap\Lambda$-liquid argument x in the source occurs $\aleph$-liquid in the negative premises, which violates the more restrictive condition 4 of the rooted branching bisimulation format.

4 Divergence-preserving branching bisimilarity as a congruence

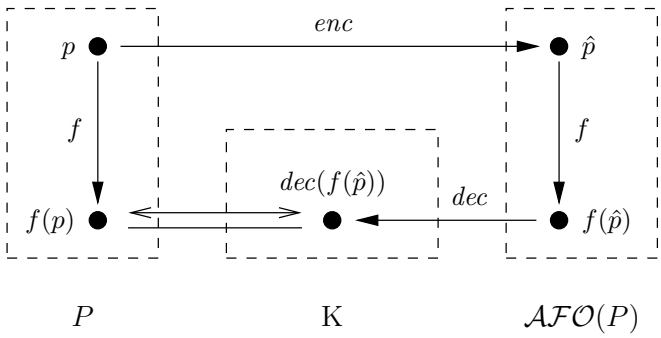
A modal characterisation of divergence-preserving branching bisimilarity is obtained by adding a unary modality Δ to the modal logic for branching bisimilarity: $p \models \Delta\varphi$ if there is an infinite trace $p = p_0 \xrightarrow{\tau} p_1 \xrightarrow{\tau} p_2 \xrightarrow{\tau} \cdots$ such that $p_i \models \varphi$ for all $i \in \mathbb{N}$. Modal formulas $\Delta\varphi$ however elude the inductive decomposition method from Definition 9, because they ask for the existence of an infinite sequence of τ-transitions, as shown by the following example.

▶ **Example 18.** Let $A = \{a_i, b_i \mid i \in \mathbb{N}\} \cup \{c\}$. Parallel composition $\|$ is defined by the rules $\frac{x_1 \xrightarrow{\alpha} y}{x_1 \| x_2 \xrightarrow{\alpha} y \| x_2}$ and $\frac{x_2 \xrightarrow{\alpha} y}{x_1 \| x_2 \xrightarrow{\alpha} x_1 \| y}$, where α ranges over A. We extend its operational semantics with asymmetric communication rules $\frac{x_1 \xrightarrow{a_i} y_1 \quad x_2 \xrightarrow{b_i} y_2}{x_1 \| x_2 \xrightarrow{c} y_1 \| y_2}$ for all $i \in \mathbb{N}$. We define $p_i \xrightarrow{\tau} p_{i+1}$ and $p_i \xrightarrow{a_i} \mathbf{0}$ and $q_i \xrightarrow{\tau} q_{i+1}$ and $q_i \xrightarrow{b_i} \mathbf{0}$ for all $i \in \mathbb{N}$. Then $p_0 \| q_0 \models \Delta(\langle\varepsilon\rangle\langle c\rangle\top)$. There is no obvious way to decompose this into modal properties of its arguments p_0 and q_0.

We circumvent this problem by introducing so-called *abstraction-free* TSSs that allow only patience rules and rules without premises to carry a conclusion with the label τ, and by introducing *oracle transitions* $p \xrightarrow{\omega} \surd$, where the transition label ω reveals some pertinent information on the behaviour of the process p, such as whether p can diverge. On abstraction-free TSSs with appropriate oracle transitions the equivalences $\leftrightarrows_b^s$ and $\leftrightarrows_b^\Delta$ coincide; so there our congruence format for $\leftrightarrows_b^s$ is also a congruence format for $\leftrightarrows_b^\Delta$. We extend this observation to general TSSs by encoding any given TSS into an abstraction-free TSS with oracle transitions, and conclude that on any TSS in the stability-respecting branching bisimulation format $\leftrightarrows_b^\Delta$ is a congruence. The same proof strategy shows that on any such TSS also $\leftrightarrows_b^{\Delta\top}$ is a congruence, and it extends to the rooted case. In Section 4.1 we present this method in more detail. We refrain from introducing the needed machinery, but state exactly which properties of this machinery we require, and prove our main congruence result based on these properties. Instead of dealing with specific equivalences $\leftrightarrows_b^\Delta$ and $\leftrightarrows_b^s$, we work with parametric equivalences $\sim$ and $\approx$ where $\sim$ is finer than $\approx$; they will later be instantiated with $\leftrightarrows_b^\Delta$ and $\leftrightarrows_b^s$. This allows a reuse of our work with $\leftrightarrows_b^{\Delta\top}$ and $\leftrightarrows_b^s$ in the roles of $\sim$ and $\approx$, as well as with $\leftrightarrows_{rb}^{\Delta\top}$ and $\leftrightarrows_{rb}^\Delta$ in the role of $\sim$ and $\leftrightarrows_{rb}^s$ in the role of $\approx$.

4.1 A framework for lifting congruence formats to finer equivalences

Our general proof idea is illustrated below, where P is a TSS in a congruence format for $\approx$. We show that also $\sim$ is a congruence on P. Consider an operator f, for simplicity depicted as unary. Given two closed terms p and q in P with $p \sim q$, we show that $f(p) \sim f(q)$. The picture shows a roundabout trajectory from p to $f(p)$. Imagine a similar trajectory from q to $f(q)$ – not depicted but hovering above the page.



First we apply a transformation $\mathcal{AFO}$ on P, yielding the abstraction-free TSS with oracle transitions $\mathcal{AFO}(P)$, depicted on the right. For each closed term p of P we introduce a constant $\hat{p}$ in $\mathcal{AFO}(P)$, in such away that $p \sim q$ implies $\hat{p} \sim \hat{q}$. Each n-ary operator f of P remains an n-ary operator f of $\mathcal{AFO}(P)$. Since $\sim \subseteq \approx$ we have $\hat{p} \approx \hat{q}$. We argue that if P is within our congruence format for $\approx$, then the TSS $\mathcal{AFO}(P)$ is also within this congruence format, and conclude from $\hat{p} \approx \hat{q}$ that $f(\hat{p}) \approx f(\hat{q})$. An important result, deferred to Section 4.3, is that on $\mathcal{AFO}(P)$ the equivalences $\sim$ and $\approx$ coincide. Hence $f(\hat{p}) \sim f(\hat{q})$.

Finally, we decode the processes $f(\hat{p})$ and $f(\hat{q})$, aiming to return to the LTS generated by P, but actually ending up in another LTS K. Our decoding function *dec* exactly undoes the effects of the encoding *enc*, that sent p to $\hat{p}$, so that $dec(f(\hat{p}))$ is strongly bisimilar with $f(p)$. A crucial property of the function *dec*, also deferred to Section 4.3, is that it is compositional for $\sim$, meaning that from $f(\hat{p}) \sim f(\hat{q})$ we may conclude $dec(f(\hat{p})) \sim dec(f(\hat{q}))$. By imposing the requirement that $\sim$ contains strong bisimilarity ($\leftrightarroweq$), this implies that $f(p) \sim f(q)$.

We now formalise this proof idea. If P is a complete TSS and G is an LTS disjoint from the LTS associated with P, then, in our formalisation, it will sometimes be convenient to write $P \uplus G$ for the disjoint union of the LTSs associated with P and G.

▶ **Theorem 19.** *Let $\sim$ and $\approx$ be behavioural equivalences on LTSs, with $\leftrightarroweq \subseteq \sim \subseteq \approx$. Let $\mathfrak{F}$ be a congruence format for $\approx$, included in the decent ntyft format, and let $\mathcal{AFO}$ be an operation on standard TSSs, where for each TSS $P = (\Sigma, Act, R)$ the signature $\hat{\Sigma}$ of $\mathcal{AFO}(P)$ contains Σ enriched by a fresh constant $\hat{p}$ for each closed term p in $T(\Sigma)$, such that, for each complete standard TSS P in decent ntyft format:*

1. *also $\mathcal{AFO}(P)$ is a complete standard TSS,*
2. *if P is in $\mathfrak{F}$-format then so is $\mathcal{AFO}(P)$,*
3. *$p \sim_P q \;\Rightarrow\; \hat{p} \sim_{\mathcal{AFO}(P)} \hat{q}$,*
4. *$\sim_{\mathcal{AFO}(P)}$ and $\approx_{\mathcal{AFO}(P)}$ coincide, and*

there is an LTS $\mathrm{K} = (\mathbb{P}_\mathrm{K}, Act_\mathrm{K}, \rightarrow_\mathrm{K})$, disjoint from P, and a $dec : \mathrm{T}(\hat{\Sigma}) \to \mathbb{P}_\mathrm{K}$ such that:

5. *$p \sim_{\mathcal{AFO}(P)} q \;\Rightarrow\; dec(p) \sim_\mathrm{K} dec(q)$, and*
6. *$f(p_1, \ldots, p_n) \leftrightarroweq_{P \uplus \mathrm{K}} dec(f(\hat{p}_1, \ldots, \hat{p}_n))$ for any n-ary $f \in \Sigma$ and $p_1, \ldots, p_n \in T(\Sigma)$.*

Then $\mathfrak{F}$ is also a congruence format for $\sim$.

Proof. Let $P = (\Sigma, Act, R)$ be a complete standard TSS in $\mathfrak{F}$-format. We will show that $\sim_P$ is a congruence for P. So let $f \in \Sigma$ be an n-ary function symbol, and let $p_i, q_i \in \mathrm{T}(\Sigma)$ with $p_i \sim_P q_i$ for $i = 1, \ldots, n$. We need to show that $f(p_1, \ldots, p_n) \sim_P f(q_1, \ldots, q_n)$.

By requirements 1 and 2, $\mathcal{AFO}(P)$ is a complete standard TSS in $\mathfrak{F}$-format; hence $\approx_{\mathcal{AFO}(P)}$ is a congruence for $\mathcal{AFO}(P)$. By requirement 3 $\hat{p}_i \sim_{\mathcal{AFO}(P)} \hat{q}_i$ for $i = 1, \ldots, n$. By requirement 4 also $\sim_{\mathcal{AFO}(P)}$ is a congruence for $\mathcal{AFO}(P)$. So $f(\hat{p}_1, \ldots, \hat{p}_n) \sim_{\mathcal{AFO}(P)} f(\hat{q}_1, \ldots, \hat{q}_n)$. Hence, by requirement 5, $dec(f(\hat{p}_1, \ldots, \hat{p}_n)) \sim_\mathrm{K} dec(f(\hat{q}_1, \ldots, \hat{q}_n))$. Therefore, by two applications of requirement 6, and the definition of a behavioural equivalence, $f(p_1, \ldots, p_n) \sim_{P \uplus \mathrm{K}} f(q_1, \ldots, q_n)$ and consequently $f(p_1, \ldots, p_n) \sim_P f(q_1, \ldots, q_n)$. ◀

4.2 Abstraction-freeness

In this section we introduce the machinery needed for Theorem 19, namely the conversion $\mathcal{AFO}$ on TSSs and the function *dec* into the LTS K. We also establish requirements 1 and 6 of Theorem 19, leaving 2–5 to the applications of Theorem 19 in Section 4.3 for specific instances of $\sim$ and $\approx$. We here take the set of actions *Act* used in Section 4.1 to be A_τ.

Again Γ denotes a predicate that marks arguments of function symbols. In [9] we called a standard TSS *abstraction-free* w.r.t. Γ if only its Γ-patience rules carry the label τ in their

conclusion. Here we use a more liberal definition of abstraction-freeness that also allows rules that have no premises, and a conclusion of the form $c \xrightarrow{\tau} d$ for constants c and d.

The next conversion turns a Γ-patient standard TSS $P = (\Sigma, A_\tau, R)$ into a Γ-patient and abstraction-free TSS $\mathcal{AFO}_\Gamma^{O,\zeta}(P)$. It is parameterised by the choice of a fresh set of actions O, so $O \cap A_\tau = \emptyset$, and a partial function $\zeta : \mathrm{T}(\Sigma) \rightharpoonup O$ called an *oracle*. The choice of O and ζ varies for different applications of Theorem 19. This choice will be made in Section 4.3 in such a way that requirements 3 and 4 of Theorem 19 are met, for specific instances of $\sim$ and $\approx$.

▶ **Definition 20.** Given a Γ-patient standard TSS $P = (\Sigma, A_\tau, R)$. Let $\hat{\Sigma}$ be the signature Σ, enriched with a fresh constant $\surd$ and a fresh constant $\hat{p}$ for each closed term $p \in \mathrm{T}(\Sigma)$. Pick a fresh action $\iota \notin A_\tau \cup O$. We define the TSS $\mathcal{AFO}_\Gamma^{O,\zeta}(P)$ as $(\hat{\Sigma}, A_\tau \cup O \cup \{\iota\}, R')$ where the rules in R' are obtained from the rules in R as follows:

1. R_1 is obtained from R by adding for each rule r and each non-empty subset S of positive τ-premises of r, a copy of r in which the labels τ in the premises in S are replaced by ι;
2. R_2 is obtained from R_1 by replacing, in every rule that has a conclusion with the label τ and is not a Γ-patience rule, the τ-label in the conclusion by ι;
3. R_3 is obtained from R_2 by adding the premise $v \xrightarrow{\iota}\!\!\!\!\!\!/\;$ to each rule with a premise $v \xrightarrow{\tau}\!\!\!\!\!\!/\;$;
4. R_4 is obtained from R_3 by the addition of a rule without premises $\hat{p} \xrightarrow{\alpha} \hat{q}$ for each transition $p \xrightarrow{\alpha} q$ *ws*-provable from P;
5. R_5 is obtained from R_4 by adding a rule $\hat{p} \xrightarrow{\zeta(p)} \surd$ for each $p \in \mathrm{T}(\Sigma)$ with $\zeta(p)$ defined;
6. R' adds to R_5 a rule $\dfrac{x_k \xrightarrow{\omega} y}{f(x_1,\ldots,x_n) \xrightarrow{\omega} y}$ for each $\omega \in O$, $f \in \Sigma$ and argument k with $\Gamma(f,k)$.

Step 2 above makes the resulting TSS abstraction-free by renaming τ-labels in conclusions of non-patience rules into ι. To ensure that still the same transitions are derived, modulo the conversion of some τ into ι-labels, step 1 above allows positive premises labelled ι to be used instead of τ in all rules, and step 3 achieves the same purpose for negative premises. These three steps result in a Γ-patient and abstraction-free TSS that could be called $\mathcal{AF}_\Gamma(P)$.

For convenience we consider an auxiliary LTS $\mathrm{G} = (\mathbb{P}_\mathrm{G}, A_\tau, \rightarrow_\mathrm{G})$ with $\mathbb{P}_\mathrm{G} = \{\hat{p} \mid p \in \mathrm{T}(\Sigma)\}$ and $\hat{p} \xrightarrow{\alpha}_\mathrm{G} \hat{q}$ iff $P \vdash_{ws} p \xrightarrow{\alpha} q$, and an auxiliary LTS $\mathrm{H} = (\mathbb{P}_\mathrm{H}, A_\tau \cup O, \rightarrow_\mathrm{H})$ with $\mathbb{P}_\mathrm{H} = \mathbb{P}_\mathrm{G} \cup \{\surd\}$ and $\rightarrow_\mathrm{H} := \rightarrow_\mathrm{G} \cup \{\hat{p} \xrightarrow{\zeta(p)} \surd \mid \zeta(p) \text{ defined}\}$. The LTS G is simply a disjoint copy of the LTS generated by P.

▶ **Lemma 21.** *If $p \sim_P q$ for some $p, q \in T(\Sigma)$, then $\hat{p} \sim_\mathrm{G} \hat{q}$.*

The LTS H adds *oracle transitions* to G. The idea is that $\zeta(p)$ is particular for the $\sim$-equivalence class of $p \in \mathrm{T}(\Sigma)$, which on the one hand ensures that $\hat{p} \sim_\mathrm{H} \hat{q}$ iff $\hat{p} \sim_\mathrm{G} \hat{q}$, and on the other hand enforces that $\sim$ and $\approx$ coincide on H. Namely, if $\hat{p} \approx_\mathrm{H} \hat{q}$ then the oracle action of $\hat{p}$ can be matched by $\hat{q}$ (and vice versa), which implies $\hat{p} \sim_\mathrm{H} \hat{q}$.

Steps 4 and 5 of Definition 20 incorporate the entire LTS H into $\mathcal{AF}_\Gamma(P)$: each state appears as a constant and each transition appears as rule without premises. The operators from Σ can now be applied to arguments of the form $\hat{p}$. Finally, step 6 lets any term $f(x_1, \ldots, x_n)$ inherit the oracle transitions from its Γ-liquid arguments. Steps 4, 5 and 6 preserve abstraction-freeness; for step 4 this uses the relaxed definition of abstraction-freeness that allows to incorporate τ-transitions between constants as rules without premises.

▶ **Example 22.** Let P have the rules

$$\frac{x_1 \xrightarrow{\tau} y}{g(x_1,x_2,x_3) \xrightarrow{\tau} g(y,x_2,x_3)} \qquad \frac{x_1 \xrightarrow{a} y_1 \quad x_1 \xrightarrow{\tau} y_2 \quad x_3 \xrightarrow{\tau} y_3}{g(x_1,x_2,x_3) \xrightarrow{\tau} x_2} \qquad \frac{x_2 \xrightarrow{\tau} y \quad x_3 \xrightarrow{\tau}\!\!\!\!\!\!/\;}{g(x_1,x_2,x_3) \xrightarrow{a} y}$$

where $\Gamma(g,1)$. Then $\mathcal{AFO}_\Gamma^{O,\zeta}(P)$ has the rules

$$\frac{x_1 \xrightarrow{\tau} y}{g(x_1,x_2,x_3) \xrightarrow{\tau} g(y,x_2,x_3)} \qquad \frac{x_1 \xrightarrow{a} y_1 \quad x_1 \xrightarrow{\tau} y_2 \quad x_3 \xrightarrow{\tau} y_3}{g(x_1,x_2,x_3) \xrightarrow{\iota} x_2} \qquad \frac{x_2 \xrightarrow{\tau} y \quad x_3 \not\xrightarrow{\tau} \quad x_3 \not\xrightarrow{\iota}}{g(x_1,x_2,x_3) \xrightarrow{a} y}$$

$$\frac{x_1 \xrightarrow{\iota} y}{g(x_1,x_2,x_3) \xrightarrow{\iota} g(y,x_2,x_3)} \qquad \frac{x_1 \xrightarrow{a} y_1 \quad x_1 \xrightarrow{\iota} y_2 \quad x_3 \xrightarrow{\iota} y_3}{g(x_1,x_2,x_3) \xrightarrow{\iota} x_2} \qquad \frac{x_2 \xrightarrow{\iota} y \quad x_3 \not\xrightarrow{\tau} \quad x_3 \not\xrightarrow{\iota}}{g(x_1,x_2,x_3) \xrightarrow{a} y}$$

$$\frac{x_1 \xrightarrow{a} y_1 \quad x_1 \xrightarrow{\iota} y_2 \quad x_3 \xrightarrow{\tau} y_3}{g(x_1,x_2,x_3) \xrightarrow{\iota} x_2} \qquad \frac{x_1 \xrightarrow{a} y_1 \quad x_1 \xrightarrow{\tau} y_2 \quad x_3 \xrightarrow{\iota} y_3}{g(x_1,x_2,x_3) \xrightarrow{\iota} x_2} \qquad \frac{x_1 \xrightarrow{\omega} y}{g(x_1,x_2,x_3) \xrightarrow{\omega} y}\ (\omega\in O)$$

This illustrates steps 1,2,3,6 of Definition 20. Since there are no closed terms, steps 4,5 are void.

Clearly, for any Γ-patient standard TSS P, the standard TSS $\mathcal{AFO}_\Gamma^{O,\zeta}(P)$ is Γ-patient and abstraction-free w.r.t. Γ. We drop superscripts O and ζ, and $\mathcal{AFO}(P)$ denotes $\mathcal{AFO}_\Gamma(P)$ for the largest Γ for which P is Γ-patient. The signature of $\mathcal{AFO}(P)$ contains the signature of P enriched by a fresh constant $\hat{p}$ for each closed term p in P, as required in Theorem 19.

▶ **Lemma 23.** *Let P be a complete standard TSS in ntyft format. If $\hat{p} \sim_{\mathrm{H}} \hat{q}$ for some $p,q \in T(\Sigma)$, then $\hat{p} \sim_{\mathcal{AFO}(P)} \hat{q}$.*

As an immediate consequence of Lemmas 21 and 23 we have the following corollary.

▶ **Corollary 24.** *Requirement 3 of Theorem 19 is met if $\hat{p} \sim_{\mathrm{G}} \hat{q}$ implies $\hat{p} \sim_{\mathrm{H}} \hat{q}$.* ◀

The inference $\hat{p} \sim_{\mathrm{G}} \hat{q} \Rightarrow \hat{p} \sim_{\mathrm{H}} \hat{q}$ depends on the choice of O and ζ; it is deferred to Section 4.3.

The oracle inheritance rules in $\mathcal{AFO}_\Gamma(P)$ – introduced in step 6 of Definition 20 – ensure that a closed term $f(p_1,\ldots,p_n)$ has an outgoing ω-transition, for $\omega \in O$, iff one of its Γ-liquid arguments p_k has such a transition. Ultimately, all such oracle transitions stem from H. Using that in $\mathcal{AFO}_\Gamma(P)$ any term $p \in \mathrm{T}(\hat{\Sigma})$ can uniquely be written as $\rho(t)$ with $t \in \mathbb{T}(\Sigma)$ and $\rho : var(t) \to \mathbb{P}_{\mathrm{H}}$, this fact can be phrased as follows. Let $t \in \mathbb{T}(\Sigma)$ and $\rho : var(t) \to \mathbb{P}_{\mathrm{H}}$; then $\mathcal{AFO}_\Gamma(P) \vdash_{ws} \rho(t) \xrightarrow{\omega}$ iff t has a Γ-liquid occurrence of a variable x with $\rho(x) \xrightarrow{\omega}_{\mathrm{H}}$. Using this, we now verify requirement 1 of Theorem 19:

▶ **Lemma 25.** *If a standard TSS P in decent ntyft format is complete, then so is $\mathcal{AFO}(P)$.*

The LTS $\mathrm{K} = (\mathbb{P}_{\mathrm{K}}, A_\tau \cup O \cup \{\iota\}, \to_{\mathrm{K}})$ has as states $\mathbb{P}_{\mathrm{K}} = \{dec(p) \mid p \in \mathrm{T}(\hat{\Sigma})\}$, and its transitions are the ones generated by the rules $\frac{x \xrightarrow{\alpha} y}{dec(x) \xrightarrow{\alpha} dec(y)}$ and $\frac{x \xrightarrow{\iota} y}{dec(x) \xrightarrow{\tau} dec(y)}$, where α ranges over A_τ. The operator $dec : \mathrm{T}(\hat{\Sigma}) \to \mathbb{P}_{\mathrm{K}}$ erases all transitions with labels from O and renames labels ι into τ. All other transitions are preserved.

We end this section by verifying requirement 6 of Theorem 19. Intuitively, the behaviour of a process $f(p_1,\ldots,p_n)$ in P is the same as that of $f(\hat{p}_1,\ldots,\hat{p}_n)$ in $\mathcal{AFO}(P)$, except that some τ-transitions of the former are turned into ι-transitions of the latter process, and some oracle transitions may have been added in the latter. Since any rule in $\mathcal{AFO}(P)$ with a conclusion labelled by $A_\tau \cup \{\iota\}$ has positive and negative premises with labels from $A_\tau \cup \{\iota\}$ only, these oracle transitions have no influence on the derivation of any transitions from $\mathcal{AFO}(P)$ with labels in $A_\tau \cup \{\iota\}$. The operator dec removes all oracle transitions and renames ι into τ, thereby returning the behaviour of $f(\hat{p}_1,\ldots,\hat{p}_n)$ to match that of $f(p_1,\ldots,p_n)$ exactly.

▶ **Proposition 26.** *Let P be a complete standard TSS in decent ntyft format. Then $f(p_1,\ldots,p_n) \leftrightarrows_{P \uplus \mathrm{K}} dec(f(\hat{p}_1,\ldots,\hat{p}_n))$ for any n-ary $f \in \Sigma$ and $p_1,\ldots,p_n \in T(\Sigma)$.*

4.3 Application of the framework to divergence-preserving semantics

We apply Theorem 19 to show that the stability-respecting branching bisimulation format and its rooted variant are congruence formats for $\leftrightarrows_b^{\Delta\top}$ and $\leftrightarrows_b^{\Delta}$, and for $\leftrightarrows_{rb}^{\Delta\top}$ and $\leftrightarrows_{rb}^{\Delta}$.

As congruence format in Theorem 19 we take the (rooted) stability-respecting branching bisimulation format intersected with the decent ntyft format. It is straightforward to check that the conversion $\mathcal{AFO}$ on standard TSSs defined in Section 4.2 preserves the (rooted) stability-respecting branching bisimulation format, and thus satisfies requirement 2 of Theorem 19. Here it is important that in step 6 of Definition 20 a term $f(x_1,\ldots,x_n)$ inherits oracle transitions only from its Γ-liquid arguments – else condition 3 of Definition 11 would be violated. Furthermore, condition 4a of Definition 11 is preserved because premises $v\stackrel{\tau}{\not\longrightarrow}$ are kept in place in step 3 of Definition 20; and condition 4b of Definition 11 is preserved because the transformation in Definition 20 does not introduce new positive τ-premises.

We first apply Theorem 19 with $\leftrightarrows_b^{\Delta\top}$ and $\leftrightarrows_b^{s}$ in the roles of $\sim$ and $\approx$. Let us say that a process p in an LTS is *divergent* if there exists an infinite sequence of processes $(p_k)_{k\in\mathbb{N}}$ such that $p_k \stackrel{\tau}{\longrightarrow} p_{k+1}$ for all $k\in\mathbb{N}$, i.e. if $p\models\Delta\top$. In the construction of the LTS H out of G (cf. Section 4.2) we take $O=\{\Delta\top\}$ and let $\zeta(g)=\Delta\top$ iff g is divergent. Thus in H all divergent states of G have a fresh outgoing transition labelled $\Delta\top$.

With this definition of H, we have $\hat{p}\sim_{\mathrm{H}}\hat{q}$ iff $\hat{p}\sim_{\mathrm{G}}\hat{q}$: any weakly divergence-preserving branching bisimulation $\mathcal{B}$ on G relates divergent states with divergent states only, and thus is also a weakly divergence-preserving branching bisimulation on H (adding $\surd\,\mathcal{B}\,\surd$.) Hence, by Corollary 24, requirement 3 of Theorem 19 is satisfied. Requirement 4 is also satisfied:

▶ **Proposition 27.** *On $\mathcal{AFO}(P)$ the equivalences $\leftrightarrows_b^{\Delta\top}$ and $\leftrightarrows_b^{s}$ coincide.*

The next example shows that Proposition 27 would not hold if we had skipped step 6 of Definition 20, inheriting oracle transitions for Γ-liquid arguments, or had not used oracle transitions at all.

▶ **Example 28.** Let $p\in\mathrm{T}(\Sigma)$ have no outgoing transitions, while $q\in\mathrm{T}(\Sigma)$ has only a τ-transition to itself and a τ-transition to p. Then in the LTS G we have $\hat{p}\leftrightarrows_{b\ \mathrm{G}}^{s}\hat{q}$ but $\hat{p}\not\leftrightarrows_{b\ \mathrm{G}}^{\Delta\top}\hat{q}$. After translation to H, the processes $\hat{p}$ and $\hat{q}$ are distinguished by means of oracle transitions, so that we have $\hat{p}\not\leftrightarrows_{b\ \mathrm{H}}^{s}\hat{q}$ and $\hat{p}\not\leftrightarrows_{b\ \mathrm{H}}^{\Delta\top}\hat{q}$. Now let Σ feature a unary operator f with as only rule $\frac{x\stackrel{\tau}{\longrightarrow}y}{f(x)\stackrel{\tau}{\longrightarrow}f(y)}$. If oracle transitions would not be inherited in $\mathcal{AFO}(P)$, then we would have $f(\hat{p})\leftrightarrows_{b\ \mathcal{AFO}(P)}^{s}f(\hat{q})$ but $f(\hat{p})\not\leftrightarrows_{b\ \mathcal{AFO}(P)}^{\Delta\top}f(\hat{q})$.

Also requirement 5 of Theorem 19 holds.

▶ **Proposition 29.** $p\leftrightarrows_{b\ \mathcal{AFO}(P)}^{\Delta\top}q \;\Rightarrow\; dec(p)\leftrightarrows_{b\ \mathrm{K}}^{\Delta\top}dec(q)$.

▶ **Corollary 30.** *The stability-respecting branching bisimulation format intersected with the decent ntyft format is a congruence format for $\leftrightarrows_b^{\Delta\top}$.*

Each standard TSS P in ready simulation format can be converted to a TSS P' in decent ntyft format, preserving the set of ws-provable closed literals [3]. Moreover, if P is in stability-respecting branching bisimulation format, then so is P'. Thus we obtain:

▶ **Theorem 31.** *Let P be a complete standard TSS in stability-respecting branching bisimulation format. Then $\leftrightarrows_b^{\Delta\top}$ is a congruence for P.* □

In a similar fashion it can be proved that the stability-respecting branching bisimulation format is a congruence format for $\leftrightarrows_b^{\Delta}$, and that the rooted stability-respecting branching bisimulation format is a congruence format for $\leftrightarrows_{rb}^{\Delta\top}$ as well as $\leftrightarrows_{rb}^{\Delta}$.

5 Related work

Ulidowski [19, 20, 21] proposed congruence formats, inside GSOS [4], for weak semantics that take into account non-divergence, called *convergence* in [11]. In [19] he introduces the *ISOS* format, and shows that the weak convergent *refusal simulation* preorder is a precongruence for all TSSs in the ISOS format. The GSOS format – in our terminology the *decent nxyft* format – allows only decent ntyft rules with variables as the left-hand sides of premises. The ISOS format is contained in the intersection of the GSOS format and our stability-preserving branching bisimulation format. Its additional restriction is that no variable may occur multiple times as the left-hand side of a positive premise, or both as the left-hand side of a positive premise and in the conclusion of a rule. In [20, 21] he employs *Ordered SOS* (OSOS) TSSs [20]. An OSOS TSS allows no negative premises, but includes priorities between rules: $r < r'$ means that r can only be applied if r' cannot. An OSOS specification can be seen as, or translated into, a GSOS specification with negative premises. Each rule r with exactly one higher-priority rule $r' > r$ is replaced by a number of rules, one for each (positive) premise of r'; in the copy of r, this premise is negated. For a rule r with multiple higher-priority rules r', this replacement is carried out for each such r'.

The `ebo` and `bbo` formats from [20] target convergent *delay* and branching bisimulation equivalence, respectively, whereas the `rebo` and `rbbo` formats from [21] target their rooted counterparts. These rooted formats are more liberal than their unrooted counterparts, and the `(r)bbo` format is more liberal than the `(r)ebo` format. If patience rules are not allowed to have a lower priority than other rules, then the `(r)bbo` format, upon translation from OSOS to GSOS, can be seen as a subformat of our (rooted) stability-respecting branching bisimulation format. Patience rules are in the `(r)bbo` format however, under strict conditions, allowed to be dominated by other rules, which in our setting gives rise to patience rules with negative premises. This is outside the realm of our rooted stability-respecting branching bisimulation format. On the other hand, the TSSs of the process algebra $\mathrm{BPA}_{\varepsilon\delta\tau}$, the binary Kleene star and deadlock testing (see [7]), for which rooted convergent branching bisimulation equivalence is a congruence, are outside `rbbo` but within the rooted stability-respecting branching bisimulation format.

6 Conclusions

We showed how the method from [10] for deriving congruence formats through modal decomposition can be applied to weak semantics that are stability-respecting. We used (rooted and unrooted) stability-respecting branching bisimulation equivalence as a notable example. Moreover, we developed a general method for lifting congruence formats from a weak semantics to a finer semantics, and used it to show that congruence formats for $\leftrightarrows^s_{rb}$ and $\leftrightarrows^s_b$ are also congruence formats for their divergence-preserving counterparts. This research provides a deeper insight into the link between modal logic and congruence formats, and strengthens the framework from [10] for the derivation of congruence formats for weak semantics.

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 [12]; the ntyft/ntyxt format [16, 5]; the transformation to ruloids; and the work on modal decomposition and congruence formats from [3]. In spite of these technicalities, the resulting framework for deriving congruence formats for weak semantics is relatively straightforward. For this one only needs to: (1) provide a modal characterisation of the weak semantics under consideration; (2) study the class of modal formulas that result from decomposing this modal characterisation, and formulate syntactic restrictions on TSSs to bring this class of modal formulas within the

original modal characterisation; and (3) check that these syntactic restrictions are preserved under the transformation to ruloids. Steps (2) and (3) are very similar in structure for different weak semantics, as exemplified by the way we obtained a congruence format for stability-respecting branching bisimulation equivalence. And the resulting congruence formats tend to be more liberal and elegant than existing congruence formats in the literature.

Our intention is to carve out congruence formats for all weak semantics in the spectrum from [11] that have reasonable congruence properties. At first we expected that the current third instalment would allow us to do so. However, it turns out that convergent weak semantics as considered in for instance [20, 21, 23] still need extra work. The modal characterisations of these semantics are three-valued [11], which requires an extension of the modal decomposition technique to a three-valued setting.

References

1 J.C.M. Baeten, J.A. Bergstra, and J.W. Klop. Syntax and defining equations for an interrupt mechanism in process algebra. *Fundamenta Informaticae*, 9(2):127–167, 1986.

2 T. Basten. Branching bisimulation is an equivalence indeed! *Information Processing Letters*, 58(3):141–147, 1996. doi:10.1016/0020-0190(96)00034-8.

3 B. Bloom, W.J. Fokkink, and R.J. van Glabbeek. Precongruence formats for decorated trace semantics. *Transactions on Computational Logic*, 5(1):26–78, 2004. doi:10.1145/963927.963929.

4 B. Bloom, S. Istrail, and A.R. Meyer. Bisimulation can't be traced. *Journal of the ACM*, 42(1):232–268, 1995. doi:10.1145/200836.200876.

5 R. Bol and J.F. Groote. The meaning of negative premises in transition system specifications. *Journal of the ACM*, 43(5):863–914, 1996. doi:10.1145/234752.234756.

6 W.J. Fokkink. Rooted branching bisimulation as a congruence. *Journal of Computer and System Sciences*, 60(1):13–37, 2000. doi:10.1006/jcss.1999.1663.

7 W.J. Fokkink and R.J. van Glabbeek. Divide and congruence II: Delay and weak bisimilarity. In *Proc. LICS 2016*, pages 778–787. ACM/IEEE, 2016. doi:10.1145/2933575.2933590.

8 W.J. Fokkink, R.J. van Glabbeek, and B. Luttik. Divide and congruence III: From decomposition of modal formulas to preservation of stability and divergence. Full version. http://theory.stanford.edu/~rvg/abstracts.html#125.

9 W.J. Fokkink, R.J. van Glabbeek, and P. de Wind. Divide and congruence: From decomposition of modalities to preservation of branching bisimulation. In *Proc. FMCO 2005*, volume 4111 of *LNCS*, pages 195–218, 2006. doi:10.1007/11804192_10.

10 W.J. Fokkink, R.J. van Glabbeek, and P. de Wind. Divide and congruence: From decomposition of modal formulas to preservation of branching and η-bisimilarity. *Information and Computation*, 214:59–85, 2012. doi:10.1016/j.ic.2011.10.011.

11 R.J. van Glabbeek. The linear time-branching time spectrum II: The semantics of sequential systems with silent moves. In *Proc. CONCUR 1993*, volume 715 of *LNCS*, pages 66–81. Springer, 1993. doi:10.1007/3-540-57208-2_6.

12 R.J. van Glabbeek. The meaning of negative premises in transition system specifications II. *Journal of Logic and Algebraic Programming*, 60/61:229–258, 2004. doi:10.1016/j.jlap.2004.03.007.

13 R.J. van Glabbeek, B. Luttik, and N. Trčka. Branching bisimilarity with explicit divergence. *Fundamenta Informaticae*, 93(4):371–392, 2009. doi:10.3233/FI-2009-109.

14 R.J. van Glabbeek, B. Luttik, and N. Trčka. Computation tree logic with deadlock detection. *Logical Methods in Computer Science*, 5(4), 2009.

15 R.J. van Glabbeek and W.P. Weijland. Branching time and abstraction in bisimulation semantics. *Journal of the ACM*, 43(3):555–600, 1996. doi:10.1145/233551.233556.

16 J.F. Groote. Transition system specifications with negative premises. *Theoretical Computer Science*, 118(2):263–299, 1993. doi:10.1016/0304-3975(93)90111-6.

17 M. Hennessy and R. Milner. Algebraic laws for nondeterminism and concurrency. *Journal of the ACM*, 32(1):137–161, 1985. doi:10.1145/2455.2460.

18 R. Milner. *Communication and Concurrency*. Prentice-Hall, 1989.

19 I. Ulidowski. Equivalences on observable processes. In *Proc. LICS 1992*, pages 148–159. IEEE, 1992. doi:10.1109/LICS.1992.185529.

20 I. Ulidowski and I. Phillips. Ordered SOS rules and process languages for branching and eager bisimulations. *Information and Computation*, 178(1):180–213, 2002. doi:10.1006/inco.2002.3161.

21 I. Ulidowski and S. Yuen. Process languages for rooted eager bisimulation. In *Proc. CONCUR 2000*, volume 1877 of *LNCS*, pages 275–289. Springer, 2000. doi:10.1007/3-540-44618-4_21.

22 F.W. Vaandrager. *Algebraic Techniques for Concurrency and their Application*. PhD thesis, University of Amsterdam, 1990.

23 D. Walker. Bisimulation and divergence. *Information and Computation*, 85(2):202–241, 1990. doi:10.1016/0890-5401(90)90048-M.