Divide and Congruence: From Decomposition of Modal Formulas to Preservation of Branching and η-Bisimilarity

Wan Fokkink[1], Rob van Glabbeek[2,3], and Paulien de Wind[1]

[1] VU University Amsterdam, Department of Computer Science, Amsterdam
[2] National ICT Australia, Sydney
[3] University of New South Wales, School of Computer Sc. and Engineering, Sydney
w.j.fokkink@vu.nl, rvg@cs.stanford.edu, p.dewind@tiscali.nl

Abstract. We present a method for decomposing modal formulas for processes with the internal action τ. To decide whether a process algebra term satisfies a modal formula, one can check whether its subterms satisfy formulas that are obtained by decomposing the original formula. The decomposition uses the structural operational semantics that underlies the process algebra. We use this decomposition method to derive congruence formats for two weak and rooted weak semantics: branching and η-bisimilarity.

1 Introduction

Structural operational semantics [30] provides process algebras and specification languages with an interpretation. It generates a labelled transition system, in which states are the closed terms over a (single-sorted, first-order) signature, and transitions between states may be supplied with labels. The transitions between states are obtained from a transition system specification, which consists of a set of proof rules called transition rules.

Labelled transition systems can be distinguished from each other by a wide range of behavioural equivalences, based on e.g. branching structure or decorated versions of execution sequences. Van Glabbeek [17] classified so-called weak semantics, which take into account the internal action τ. Here we focus on two such equivalences which, to different degrees, abstract away from internal actions: branching bisimilarity [20] and η-bisimilarity [2]. Also we consider the rooted counterparts of these equivalences, which were introduced because unlike the unrooted versions they are congruences for basic process algebras, notably for the alternative composition operator.

In general a behavioural equivalence induced by a transition system specification 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. Syntactic formats for transition rules have been developed with respect to several behavioural equivalences, to ensure that such an equivalence is a congruence. These formats help to avoid repetitive congruence proofs. Several congruence formats were introduced for bisimilarity, such as the De Simone format [31], the GSOS format [7], the tyft/tyxt format [22], and the ntyft/ntyxt format [21]. Bloom [5] introduced congruence formats for branching bisimilarity and for rooted branching bisimilarity. These formats include so-called patience rules for arguments i of function symbols f, which imply that a term $f(p_1, \ldots, p_n)$ inherits the τ-transitions of its argument p_i. Furthermore, arguments of function symbols that contain running processes are marked, and this marking is used to restrict occurrences of variables in transition rules.

Behavioural equivalences can be characterised in terms of the observations that an experimenter could make during a session with a process. Modal logic captures such observations. 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 [23] is a modal characterisation of (strong) bisimilarity.

Larsen & Liu [26] introduced a method for decomposing formulas from Hennessy-Milner logic for τ-free processes, with respect to terms from a process algebra with a structural operational semantics in the De Simone format. To decide whether a process algebra term satisfies a modal formula, one can check whether its subterms satisfy certain other formulas, obtained by decomposing the original formula. This method was extended by Bloom, Fokkink & van Glabbeek [6] to the ntyft/ntyxt format without lookahead, and by Fokkink, van Glabbeek & de Wind [12] to the tyft/tyxt format. In [6], the decomposition method was applied to obtain congruence formats for a range of behavioural equivalences. The idea is that given an equivalence and its modal characterisation C, the congruence format for this equivalence must ensure that decomposing a formula in C always produces formulas in C.

Here we extend the work of [6] to processes with τ-transitions. We present a method for decomposing modal formulas for processes with τ-transitions, and use this decomposition method to obtain congruence formats for two weak and rooted weak semantics: branching and η-bisimilarity. In contrast to the ad hoc construction of congruence formats from the past, we can now systematically derive expressive congruence formats from the modal characterisations of behavioural equivalences. The congruence formats that we obtain are more liberal and more elegant than existing congruence formats for these semantics. In Sect. 8 we will present an in-depth comparison with congruence formats from the literature.

Our formats use two predicates $\aleph$ and Λ on arguments of function symbols: $\aleph$ marks processes that can execute immediately, and Λ marks processes that have started executing (but may currently be unable to execute). The predicate $\aleph$ is new, whereas Λ originates from [10]. The two formats for weak semantics (branching and η-bisimilarity) can be expressed as a subset of the formats for the corresponding rooted weak semantics, by imposing one extra restriction: Λ must be universal, meaning that it holds for all arguments of function symbols.

Preliminary versions of this paper, focusing on branching and η-bisimilarity, respectively, were published as [13,14].

2 Preliminaries

This section recalls the basic notions of labelled transition systems and weak semantics (Sect. 2.1), and presents modal characterisations of the semantic equivalences that are studied in this paper (Sect. 2.2). Then follows a brief introduction to structural operational semantics and the notion of a well-supported proof (Sect. 2.3). Next we recall some syntactic restrictions on transition rules (Sect. 2.4). Finally, we recall a basic result from [6], Prop. 1, regarding so-called ruloids (Sect. 2.5), and introduce two predicates Λ and $\aleph$ on arguments of function symbols (Sect. 2.6).

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 *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 $p \overset{\alpha}{\not\longrightarrow}$ for $\neg\exists q \in \mathbb{P} : p \xrightarrow{\alpha} q$. Furthermore, $\overset{\epsilon}{\Longrightarrow}$ denotes the transitive-reflexive closure of $\xrightarrow{\tau}$.

The following two versions of bisimilarity abstract away, to different degrees, from the internal action τ.

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

– B is a *branching bisimulation* if pBq and $p \xrightarrow{\alpha} p'$ implies that either $\alpha = \tau$ and $p'\, B\, q$, or $q \overset{\epsilon}{\Longrightarrow} q' \xrightarrow{\alpha} q''$ for some q' and q'' with pBq' and $p'Bq''$.
  Processes p, q are *branching bisimilar*, denoted $p \leftrightarroweq_b q$, if there exists a branching bisimulation B with pBq.
– B is an *η-bisimulation* if pBq and $p \xrightarrow{\alpha} p'$ implies that either $\alpha = \tau$ and $p'\, B\, q$, or $q \overset{\epsilon}{\Longrightarrow} q' \xrightarrow{\alpha}\overset{\epsilon}{\Longrightarrow} q''$ for some q' and q'' with pBq' and $p'Bq''$.
  Processes p, q are *η-bisimilar*, denoted $p \leftrightarroweq_\eta q$, if there exists an η-bisimulation B with pBq.

Clearly, branching bisimilarity is included in η-bisimilarity. A typical example of two processes that are η- but not branching bisimilar is: Let $p \xrightarrow{a} r$ be the only transition of p, $q \xrightarrow{a} r$ and $q \xrightarrow{a} s$ the only two transitions of q, and $r \xrightarrow{\tau} s$. Then, independent of the transitions of r and s, always $p \leftrightarroweq_\eta q$, because $q \xrightarrow{a} s$ can be mimicked by $p \xrightarrow{a} r \xrightarrow{\tau} s$. But in general $p \not\leftrightarroweq_b q$.

It is well-known that branching and η-bisimilarity constitute equivalence relations [3,2]. However, these two weak semantics 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 $p_1, \ldots, p_n$. A rootedness condition generally remedies this imperfection.

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

- R is a *rooted branching bisimulation* if pRq and $p \xrightarrow{\alpha} p'$ implies that $q \xrightarrow{\alpha} q'$ for some q' with $p' \leftrightarroweq_b q'$.
  Processes p, q are *rooted branching bisimilar*, denoted $p \leftrightarroweq_{rb} q$, if there exists a rooted branching bisimulation R with pRq.
- R is a *rooted η-bisimulation* if pRq and $p \xrightarrow{\alpha} p'$ implies that $q \xrightarrow{\alpha}\overset{\epsilon}{\Longrightarrow} q'$ for some q' with $p' \leftrightarroweq_\eta q'$.
  Processes p, q are *rooted η-bisimilar*, denoted $p \leftrightarroweq_{r\eta} q$, if there exists a rooted η-bisimulation R with pRq.

Our main aim is to develop congruence formats for both the rooted and the unrooted versions of the two weak semantics defined in this section. These congruence formats will impose syntactic restrictions on the transition rules (see Sect. 2.3) that are used to generate the underlying LTS. The congruence formats will be determined using the characterising modal logics for these two weak semantics, which are presented in the next section.

2.2 Modal logic

Modal logic aims to formulate properties of processes in an LTS. Following [17], we extend Hennessy-Milner logic [23] with the modal 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 state where φ holds.

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

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

$p \models \varphi$ denotes that p satisfies φ. By definition, $p \models \langle\alpha\rangle\varphi$ if $p \xrightarrow{\alpha} p'$ for some p' with $p' \models \varphi$, $p \models \langle\epsilon\rangle\varphi$ if $p \overset{\epsilon}{\Longrightarrow} p'$ 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 $p' \models \varphi$. We use abbreviations $\top$ 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 $\varphi \wedge \langle\hat{\tau}\rangle\varphi'$. We write $\varphi \equiv \varphi'$ if $p \models \varphi \Leftrightarrow p \models \varphi'$ for any process p in any LTS.

Definition 4. The subclasses $\mathbb{O}_e$ and $\mathbb{O}_{re}$ of $\mathbb{O}$, for $e \in \{b, \eta\}$, are defined as follows:

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

In these definitions, a ranges over A and α over A_τ. The classes $\mathbb{O}_e^\equiv$ and $\mathbb{O}_{re}^\equiv$ denote the closures of $\mathbb{O}_e$, respectively $\mathbb{O}_{re}$, under $\equiv$.

The last clause in the definition of $\mathbb{O}_{re}$ guarantees that $\mathbb{O}_e \subseteq \mathbb{O}_{re}$, which will be needed in the proof of Prop. 4. If this clause were omitted, it would still follow that $\mathbb{O}_e^{\equiv} \subseteq \mathbb{O}_{re}^{\equiv}$, using structural induction together with $\langle\epsilon\rangle\langle\epsilon\rangle\varphi \equiv \langle\epsilon\rangle\varphi$, $\langle\tau\rangle\langle\epsilon\rangle\varphi \equiv \langle\epsilon\rangle\langle\tau\rangle\varphi$, $\langle\epsilon\rangle\varphi \equiv \varphi \vee \langle\tau\rangle\langle\epsilon\rangle\varphi$ and (for $e = b$) $\langle\hat{\tau}\rangle\varphi \equiv \varphi \vee \langle\tau\rangle\varphi$. Note that if $\varphi \in \mathbb{O}_b^{\equiv}$, then $\langle\epsilon\rangle\varphi \equiv \langle\epsilon\rangle\varphi\langle\hat{\tau}\rangle\varphi \in \mathbb{O}_b^{\equiv}$.

For $L \subseteq \mathbb{O}$, we write $p \sim_L q$ if p and q satisfy the same formulas in L. Note that, trivially, $p \sim_{\mathbb{O}_e} q \Leftrightarrow p \sim_{\mathbb{O}_e^{\equiv}} q$ and $p \sim_{\mathbb{O}_{re}} q \Leftrightarrow p \sim_{\mathbb{O}_{re}^{\equiv}} q$.

Theorem 1. *$p \,\underline{\leftrightarrow}_e\, q \Leftrightarrow p \sim_{\mathbb{O}_e} q$ and $p \,\underline{\leftrightarrow}_{re}\, q \Leftrightarrow p \sim_{\mathbb{O}_{re}} q$, for all $p, q \in \mathbb{P}$, and $e \in \{b, \eta\}$.*

A proof of this theorem for the case $e = b$ is presented in the appendix. The proof for the case $e = \eta$ is similar.

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, with typical elements x, y, z; we always take $|\Sigma|, |A| \leq |V|$. 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 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 [30] generates a labelled transition system, in which the processes are the closed terms, and transitions between processes are supplied with labels. The transitions between processes 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 $t \stackrel{\alpha}{\not\longrightarrow}$. 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* of the rule, respectively. With $rhs(H)$ we denote the set of right-hand sides of the positive premises 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)*, written (Σ, R), consists of a signature Σ and a collection R of transition rules over Σ. A TSS is *standard* if all its rules are.

The concept of a standard TSS with only positive premises was introduced in [22]; negative premises were added in [21]. The resulting notion constitutes the first formalisation of *structural operational semantics* [30] that is sufficiently general to cover many of its applications. TSSs with negative conclusions were introduced in [6] because they are needed as intermediate steps in our proofs for standard TSSs.

The following definition tells when a literal is provable from a TSS. It generalises the standard definition (see e.g. [22]) by allowing the derivation of transition rules. The derivation of a literal λ corresponds to the derivation of the transition rule $\frac{H}{\lambda}$ with $H = \emptyset$. The case $H \neq \emptyset$ corresponds to the derivation of λ under the assumptions H.

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 [6] because H must equal (instead of include) the set of labels of the hypotheses. Irredundancy will be crucial for the preservation under provability of our congruence formats; see Sect. 4.2. Namely, in a 'redundant' proof one can freely add premises to the derived rule, so also a premise that violates a syntactic restriction of the congruence format under consideration.

A TSS is meant to specify an LTS in which the transitions are closed positive literals. A standard TSS with only positive premises specifies an LTS in a straightforward way, but it is not so easy to associate an LTS to a TSS with negative premises. From [18] we adopt the notion of a well-supported proof of a closed literal. Literals $t \xrightarrow{\alpha} u$ and $t \overset{\alpha}{\not\rightarrow}$ are said to *deny* each other.

Definition 7. 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:

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 $P \vdash_{ws} p \overset{\alpha}{\not\rightarrow}$ or $P \vdash_{ws} p \xrightarrow{\alpha} p'$ for some p'.

In [18] 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.

Example 1. Let $A = \{a\}$ and $P = (\Sigma, R)$, where $\Sigma = \{a\}$ and R consists of the rule $\frac{a \overset{a}{\not\rightarrow}}{a \xrightarrow{a} a}$. The standard TSS P is not complete, because neither $P \vdash_{ws} a \overset{a}{\not\rightarrow}$ nor $P \vdash_{ws} a \xrightarrow{a} a$. For this reason P can be considered not to specify an LTS.

2.4 Syntactic restrictions on transition rules

In this section we present terminology for syntactic restrictions on rules, originating from [6,21,22], where congruence formats are presented for a range of concrete semantics (which do not take into account the internal action τ).

Definition 8. 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.

The idea behind the names of the rules is that the 'n' in front refers to the presence of negative premises, and the following four letters refer to the allowed forms of left- and right-hand sides of premises and of the conclusion, respectively. For example, ntyft means a rule with negative premises (n), where left-hand sides of premises are general terms (t), right-hand sides of positive premises are variables (y), the source contains exactly one function symbol (f), and the target, if present at all, is a general term (t).

Definition 9. 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 transition rules induces a corresponding syntactic format for TSSs of the same name. For instance, a TSS is in decent ntyft format if it contains decent ntyft rules only.

The following lemma, on the preservation of decency under irredundant provability, is proved in [6].

Lemma 1. *Let P be a TSS in decent ntytt format. Then any ntytt rule irredundantly provable from P is decent.*

We proceed to define further syntactic formats for TSSs. The ntyft/ntyxt and ready simulation formats [21,6] were originally introduced to guarantee congruence for (strong) bisimilarity and ready simulation.

Definition 10. A TSS is in *ntyft/ntyxt format* if it consists of ntyft and ntyxt rules, and in *ready simulation format* if moreover its rules have no lookahead.

2.5 Ruloids

To decompose modal formulas, we use a result from [6], 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 [6].

First P is converted to a standard TSS in decent ntyft format. In this conversion from [22], 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 n-ary function symbol f in the signature of P, where the variables $x_1, \ldots, x_{ar(f)}$ are fresh.

Next, using a construction from [11], left-hand sides of positive premises in rules of P 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, all of whose rules are irredundantly provable from P.

In the final transformation step, non-standard rules with a negative conclusion $t \overset{\alpha}{\not\longrightarrow}$ 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 $\frac{H}{f(x_1,\ldots,x_n)\overset{\alpha}{\not\longrightarrow}}$ is obtained by picking one premise from each standard rule with a conclusion of the form $f(x_1,\ldots,x_n) \xrightarrow{\alpha} t$, and including the denial of each of the selected premises as a premise in H. For this last transformation it is essential that rules have no lookahead.

The resulting TSS, which is in decent ntyft format, is denoted by P^+. In [6] it is established, for all closed literals μ, that $P \vdash_{ws} \mu$ if and only if μ is irredundantly provable from P^+. The P-ruloids are those decent nxytt rules that are irredundantly provable from P^+.

The following correspondence result from [6] between a TSS and its ruloids plays a crucial role in the decomposition method employed here. It says 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.

Proposition 1. *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 $P \vdash_{ws} \rho(t) \xrightarrow{\alpha} q$ if and only if there are a P-ruloid $\frac{H}{t\xrightarrow{\alpha}u}$ and a closed substitution ρ' such that $P \vdash_{ws} \rho'(\mu)$ for all $\mu \in H$, $\rho'(t) = \rho(t)$ and $\rho'(u) = q$.*

2.6 The predicates Λ and $\aleph$

In Sect. 4, we will assume two predicates on arguments of function symbols. The predicate Λ marks arguments that contain processes that have started executing (but may currently be unable to execute). It stems from [10] and fine-tunes the predicate "tame" from Bloom [5], which is a predicate on function symbols rather than arguments of function symbols. The predicate $\aleph$ is new and 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). Finally, Λ does not hold and $\aleph$ holds for the arguments of alternative composition $t_1 + t_2$ (see Ex. 2 in Sect. 3); they contain processes that did not yet start to execute, but that can start executing immediately. In

Sections 6.6 and 6.7 we will see examples of arguments for which Λ holds and $\aleph$ does not.

We proceed to introduce some terminology from [5,6] for predicates on arguments of function symbols.

Definition 11. Let Γ be a unary 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*. 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*.

Note that an occurrence of a variable x in a term $t \in \mathbb{T}(\Sigma)$ is Γ-frozen if and only if t contains a subterm $f(t_1, \ldots, t_{ar(f)})$ such that the occurrence of x is in t_i for a Γ-frozen argument i of f.

In Sect. 3 we will present 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. In Sect. 4 we will instantiate Γ with $\aleph \cap \Lambda$.

Definition 12. 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 called 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 called *Γ-impatient*.

A patience rule for an argument i of a function symbol f expresses that terms $f(p_1, \ldots, p_n)$ can mimic the τ-transitions of argument p_i (cf. [5,10]). Typically, in process algebra, there are patience rules for the arguments of the merge and for the first argument of sequential composition, but not for the arguments of the alternative composition $+$ or for the second argument of sequential composition.

Remark 1. A standard ntytt rule is Γ-patient if and only if it has the form $\frac{x \xrightarrow{\tau} y}{C[x] \xrightarrow{\tau} C[y]}$ for some Γ-liquid context $C[\ ]$.

3 Decomposition of Modal Formulas

In this section we show how one can decompose formulas from $\mathbb{O}$. To each term t and formula φ we assign a set $t^{-1}(\varphi)$ of decomposition mappings $\psi : V \to \mathbb{O}$. Each of these mappings $\psi \in t^{-1}(\varphi)$ guarantees that for closed substitutions 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 $x \in var(t)$. This is formalised in Thm. 2.

Definition 13. 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 six conditions. Let t denote a univariate term, i.e. without multiple occurrences of the same variable.

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
$$\psi(x) = \bigwedge_{i\in I} \psi_i(x) \qquad \text{for all } x \in V$$
2. $\psi \in t^{-1}(\neg\varphi)$ iff there is a function $h : t^{-1}(\varphi) \to \mathit{var}(t)$ such that
$$\psi(x) = \begin{cases} \displaystyle\bigwedge_{\chi \in h^{-1}(x)} \neg\chi(x) & \text{if } x \in \mathit{var}(t) \\ \top & \text{if } x \notin \mathit{var}(t) \end{cases}$$
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
$$\psi(x) = \begin{cases} \chi(x) \ \wedge \displaystyle\bigwedge_{x \xrightarrow{\beta} y \in H} \langle\beta\rangle\chi(y) \ \wedge \bigwedge_{x \stackrel{\gamma}{\nrightarrow} \in H} \neg\langle\gamma\rangle\top & \text{if } x \in \mathit{var}(t) \\ \top & \text{if } x \notin \mathit{var}(t) \end{cases}$$
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
$$\psi(x) = \begin{cases} \langle\epsilon\rangle\chi(x) & \text{if } x \text{ occurs } \Gamma\text{-liquid in } t \\ \chi(x) & \text{otherwise} \end{cases}$$
   (b) or there is a non-Γ-patient P-ruloid $\frac{H}{t \xrightarrow{\tau} u}$ and a $\chi \in u^{-1}(\langle\epsilon\rangle\varphi)$ such that
$$\psi(x) = \begin{cases} \top & \text{if } x \notin \mathit{var}(t) \\ \langle\epsilon\rangle\left(\chi(x) \ \wedge \displaystyle\bigwedge_{x \xrightarrow{\beta} y \in H} \langle\beta\rangle\chi(y) \ \wedge \bigwedge_{x \stackrel{\gamma}{\nrightarrow} \in H} \neg\langle\gamma\rangle\top\right) & \text{if } x \text{ occurs } \Gamma\text{-liquid in } t \\ \chi(x) \wedge \displaystyle\bigwedge_{x \xrightarrow{\beta} y \in H} \langle\beta\rangle\chi(y) \ \wedge \bigwedge_{x \stackrel{\gamma}{\nrightarrow} \in H} \neg\langle\gamma\rangle\top & \text{otherwise} \end{cases}$$
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
$$\psi(x) = \begin{cases} \langle\hat{\tau}\rangle\chi(x) & \text{if } x = x_0 \\ \chi(x) & \text{otherwise} \end{cases}$$

(c) or there is a Γ-impatient P-ruloid $\frac{H}{t \xrightarrow{\tau} u}$ and a $\chi \in u^{-1}(\varphi)$ such that

$$\psi(x) = \begin{cases} \chi(x) \wedge \bigwedge_{x \xrightarrow{\beta} y \in H} \langle\beta\rangle\chi(y) \ \wedge \bigwedge_{x \xrightarrow{\gamma}\!\!\!\!\!/\ \in H} \neg\langle\gamma\rangle\top & \text{if } x \in var(t) \\ \top & \text{otherwise} \end{cases}$$

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

$$\psi(x) = \bigwedge_{z \in \sigma^{-1}(x)} \chi(z) \qquad \text{for all } x \in V$$

To explain the idea behind Def. 13, we expand on two of its cases. Consider $t^{-1}(\neg\varphi)$, and let ρ be a closed substitution. We have $\rho(t) \not\models \varphi$ if and only if there is no $\chi \in t^{-1}(\varphi)$ such that $\rho(x) \models \chi(x)$ for all $x \in var(t)$. In other words, for each $\chi \in t^{-1}(\varphi)$, $\psi(x)$ must contain a conjunct $\neg\chi(x)$, for some $x \in var(t)$.

Consider $t^{-1}(\langle\alpha\rangle\varphi)$, and let ρ be a closed substitution. The question is under which conditions $\psi(x) \in \mathbb{O}$ on $\rho(x)$, for each $x \in var(t)$, there is a transition $\rho(t) \xrightarrow{\alpha} q$ with $q \models \varphi$. According to Prop. 1, there is such a transition if and only if there is a closed substitution ρ' with $\rho'(t) = \rho(t)$ and a P-ruloid $\frac{H}{t \xrightarrow{\alpha} u}$ such that (1) the premises in $\rho'(H)$ are satisfied and (2) $\rho'(u) \models \varphi$. The first condition is covered if for each $x \in var(t)$, $\psi(x)$ contains conjuncts $\langle\beta\rangle\top$ for $x \xrightarrow{\beta} y \in H$ and conjuncts $\neg\langle\gamma\rangle\top$ for $x \xrightarrow{\gamma}\!\!\!\!\!/\ \in H$. By adding a conjunct $\chi(x)$, and replacing each conjunct $\langle\beta\rangle\top$ by $\langle\beta\rangle\chi(y)$, for some $\chi \in u^{-1}(\varphi)$, the second condition is covered as well.

The following three lemmas state basic properties of formulas $\psi(x)$.

Lemma 2. *Let $\psi \in t^{-1}(\varphi)$, for some term t and formula φ. If $x \notin var(t)$, then $\psi(x) \equiv \top$.*

Proof. This can be derived in a straightforward fashion from Def. 13, by induction on the construction of ψ. □

The following lemma states that $\cdot^{-1}$ is invariant under α-conversion up to $\equiv$.

Lemma 3. *Let $\psi \in \sigma(t)^{-1}(\varphi)$ for $\sigma : V \to V$ a bijective renaming of variables. Then there is a $\psi' \in t^{-1}(\varphi)$ satisfying $\psi'(x) \equiv \psi(\sigma(x))$ for all $x \in V$.*

Proof. Again by induction on the construction of ψ. □

Lemma 4. *Let $\psi \in t^{-1}(\langle\epsilon\rangle\varphi)$, for some term t and formula φ. If x occurs only Γ-liquid in t, then $\psi(x) = \langle\epsilon\rangle\varphi'$ for some formula φ'.*

Proof. Immediate from Def. 13.4. □

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

Theorem 2. *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 $\varphi \in \mathbb{O}$:*

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

Proof. By simultaneous induction on the structure of φ and the construction of ψ. First we treat the case where t is univariate.

- $\varphi = \bigwedge_{i \in I} \varphi_i$
  $\rho(t) \models \bigwedge_{i\in I} \varphi_i \;\Leftrightarrow\; \forall i \in I : \rho(t) \models \varphi_i \Leftrightarrow\; \forall i \in I\ \exists \psi_i \in t^{-1}(\varphi_i)\ \forall x \in var(t) : \rho(x) \models \psi_i(x) \Leftrightarrow\; \exists \psi \in t^{-1}(\bigwedge_{i \in I} \varphi_i)\ \forall x \in var(t) : \rho(x) \models \psi(x)$.

- $\varphi = \neg\varphi'$
  $\rho(t) \models \neg\varphi' \Leftrightarrow \rho(t) \not\models \varphi' \Leftrightarrow \exists h : t^{-1}(\varphi') \to var(t)\ \forall \chi \in t^{-1}(\varphi') : \rho(h(\chi)) \not\models \chi(h(\chi)) \Leftrightarrow \exists h : t^{-1}(\varphi') \to var(t)\ \forall x \in var(t) : \rho(x) \models \bigwedge_{\chi \in h^{-1}(x)} \neg\chi(x) \Leftrightarrow \exists \psi \in t^{-1}(\neg\varphi')\ \forall x \in var(t) : \rho(x) \models \psi(x)$.

- $\varphi = \langle\alpha\rangle\varphi'$
  ($\Rightarrow$) Let $\rho(t) \models \langle\alpha\rangle\varphi'$. Then $P \vdash_{ws} \rho(t) \xrightarrow{\alpha} p$ with $p \models \varphi'$. By Prop. 1 there is a P-ruloid $\frac{H}{t \xrightarrow{\alpha} u}$ and a closed substitution ρ' with $P \vdash_{ws} \rho'(\mu)$ for $\mu \in H$, $\rho'(t) = \rho(t)$, i.e. $\rho'(x) = \rho(x)$ for all $x \in var(t)$, and $\rho'(u) = p$. Since $\rho'(u) \models \varphi'$, by induction on formula size there is a $\chi \in u^{-1}(\varphi')$ with $\rho'(z) \models \chi(z)$ for each $z \in var(u)$. Moreover, by Lem. 2, $\rho'(z) \models \chi(z) \equiv \top$ for each $z \notin var(u)$. Define $\psi \in t^{-1}(\langle\alpha\rangle\varphi')$ as in Def. 13.3, using $\frac{H}{t \xrightarrow{\alpha} u}$ and χ. Let $x \in var(t)$. For $x \xrightarrow{\beta} y \in H$, $P \vdash_{ws} \rho'(x) \xrightarrow{\beta} \rho'(y) \models \chi(y)$, so $\rho'(x) \models \langle\beta\rangle\chi(y)$. Moreover, for $x \stackrel{\gamma}{\not\longrightarrow} \in H$, $P \vdash_{ws} \rho'(x) \stackrel{\gamma}{\not\longrightarrow}$, so the consistency of $\vdash_{ws}$ yields $P \not\vdash_{ws} \rho'(x) \xrightarrow{\gamma} q$ for all q, and thus $\rho'(x) \models \neg\langle\gamma\rangle\top$. Hence $\rho(x) = \rho'(x) \models \psi(x)$.

  ($\Leftarrow$) Let $\psi \in t^{-1}(\langle\alpha\rangle\varphi')$ with $\rho(x) \models \psi(x)$ for all $x \in var(t)$. According to Def. 13.3, there is a P-ruloid

  $$\frac{\{x \xrightarrow{\beta_i} y_i \mid i \in I_x,\ x \in var(t)\} \cup \{x \stackrel{\gamma_j}{\not\longrightarrow} \mid j \in J_x,\ x \in var(t)\}}{t \xrightarrow{\alpha} u}$$

  and a $\chi \in u^{-1}(\varphi')$ with $\psi(x) = \chi(x) \wedge \bigwedge_{i \in I_x} \langle\beta_i\rangle\chi(y_i) \wedge \bigwedge_{j \in J_x} \neg\langle\gamma_j\rangle\top$ for all $x \in var(t)$. For each $x \in var(t)$, $\rho(x) \models \psi(x)$ yields, for each $i \in I_x$, $P \vdash_{ws} \rho(x) \xrightarrow{\beta_i} p_i \models \chi(y_i)$ for some p_i; moreover, for each $j \in J_x$ we have $P \not\vdash_{ws} \rho(x) \xrightarrow{\gamma_j} q$ for all q, so by the completeness of P, $P \vdash_{ws} \rho(x) \stackrel{\gamma_j}{\not\longrightarrow}$. Define $\rho'(x) = \rho(x)$ and $\rho'(y_i) = p_i$ for all $x \in var(t)$ and $i \in I_x$. (Here we use that the y_i are all different and do not occur in t.) Then $\rho'(z) \models \chi(z)$ for all $z \in var(u)$, because $var(u) \subseteq \{x, y_i \mid x \in var(t), i \in I_x\}$. So by induction on formula size, $\rho'(u) \models \varphi'$. Moreover, for each $x \in var(t)$, $P \vdash_{ws} \rho'(x) \xrightarrow{\beta_i} \rho'(y_i)$ for each $i \in I_x$, and $P \vdash_{ws} \rho'(x) \stackrel{\gamma_j}{\not\longrightarrow}$ for each $j \in J_x$, so by Prop. 1, $P \vdash_{ws} \rho'(t) \xrightarrow{\alpha} \rho'(u)$. Hence $\rho(t) = \rho'(t) \models \langle\alpha\rangle\varphi'$.

- $\varphi = \langle\epsilon\rangle\varphi'$
  ($\Rightarrow$) We prove by induction on n: if $P \vdash_{ws} p_i \xrightarrow{\tau} p_{i+1}$ for all $i \in \{0, \ldots, n-1\}$, with $\rho(t) = p_0$ and $p_n \models \varphi'$, then there is a $\psi \in t^{-1}(\langle\epsilon\rangle\varphi')$ with $\rho(x) \models \psi(x)$ for all $x \in var(t)$.

$n = 0$ Since $\rho(t) = p_0 \models \varphi'$, by induction on formula size, there is a $\chi \in t^{-1}(\varphi')$ with $\rho(x) \models \chi(x)$ for all $x \in var(t)$. Define $\psi \in t^{-1}(\langle\epsilon\rangle\varphi')$ as in Def. 13.4a, using χ. Then clearly $\rho(x) \models \psi(x)$ for all $x \in var(t)$.

$n > 0$ Since $P \vdash_{ws} \rho(t) \xrightarrow{\tau} p_1$, by Prop. 1 there is a P-ruloid $\frac{H}{t \xrightarrow{\tau} u}$ and a closed substitution ρ' with $P \vdash_{ws} \rho'(\mu)$ for all $\mu \in H$, $\rho'(t) = \rho(t)$, i.e. $\rho'(x) = \rho(x)$ for all $x \in var(t)$, and $\rho'(u) = p_1$. Since $P \vdash_{ws} \rho'(u) = p_1 \xrightarrow{\tau} \cdots \xrightarrow{\tau} p_n \models \varphi'$, by induction on n there is a $\chi \in u^{-1}(\langle\epsilon\rangle\varphi')$ with $\rho'(z) \models \chi(z)$ for each $z \in var(u)$. Moreover, by Lem. 2, $\rho'(z) \models \chi(z) \equiv \top$ for each $z \notin var(u)$. We distinguish two cases.

Case 1: $\frac{H}{t \xrightarrow{\tau} u}$ is Γ-patient. By Remark 1, using that t is univariate, H must be of the form $\{x_0 \xrightarrow{\tau} y_0\}$, with $u = t[y_0/x_0]$, and the unique occurrence of y_0 in u being Γ-liquid. Let $\sigma : V \to V$ be the bijection that swaps x_0 and y_0, so that $u = \sigma(t)$. According to Lem. 3, there is a $\chi' \in t^{-1}(\langle\epsilon\rangle\varphi')$ satisfying $\chi'(x) \equiv \chi(\sigma(x))$ for all $x \in V$.
For each $x \in var(t)\backslash\{x_0\}$, $\rho(x) = \rho'(x) \models \chi(x) \equiv \chi'(x)$, so $\rho(x) \models \chi'(x)$. Furthermore, $P \vdash_{ws} \rho'(x_0) \xrightarrow{\tau} \rho'(y_0) \models \chi(y_0)$. By Lem. 4, $\chi(y_0) \equiv \langle\epsilon\rangle\varphi''$ for some φ''. Hence $\rho(x_0) = \rho'(x_0) \models \chi(y_0) \equiv \chi'(x_0)$, so $\rho(x_0) \models \chi'(x_0)$.

Case 2: $\frac{H}{t \xrightarrow{\tau} u}$ is Γ-impatient. Define $\psi \in t^{-1}(\langle\epsilon\rangle\varphi')$ as in Def. 13.4b, using $\frac{H}{t \xrightarrow{\tau} u}$ and χ. Let $x \in var(t)$. For each $x \xrightarrow{\beta} y \in H$, $P \vdash_{ws} \rho'(x) \xrightarrow{\beta} \rho'(y)$ and $\rho'(y) \models \chi(y)$, so $\rho'(x) \models \langle\beta\rangle\chi(y)$. Moreover, for each $x \stackrel{\gamma}{\not\rightarrow} \in H$, $P \vdash_{ws} \rho'(x) \stackrel{\gamma}{\not\rightarrow}$, so the consistency of $\vdash_{ws}$ yields $P \not\vdash_{ws} \rho'(x) \xrightarrow{\gamma} q$ for all q, and thus $\rho'(x) \models \neg\langle\gamma\rangle\top$. Hence $\rho(x) = \rho'(x) \models \psi(x)$. (In case the occurrence of x in t is Γ-liquid, note that if $p \models \xi$ then certainly $p \models \langle\epsilon\rangle\xi$.)

($\Leftarrow$) Let $\psi \in t^{-1}(\langle\epsilon\rangle\varphi')$ with $\rho(x) \models \psi(x)$ for all $x \in var(t)$. According to Def. 13.4 we can distinguish two cases.

Case 1: There is a $\chi \in t^{-1}(\varphi')$ with $\psi(x) = \langle\epsilon\rangle\chi(x)$ if x occurs Γ-liquid in t, and $\psi(x) = \chi(x)$ otherwise. Then $\rho(x) \models \chi(x)$ for each x that occurs Γ-frozen in t. Furthermore, for each x that occurs Γ-liquid in t, $\rho(x) \models \langle\epsilon\rangle\chi(x)$, i.e. $P \vdash_{ws} \rho(x) \stackrel{\epsilon}{\Longrightarrow} p_x \models \chi(x)$ for some p_x. Define $\rho'(x) = p_x$ if x occurs Γ-liquid in t, and $\rho'(x) = \rho(x)$ otherwise. Since P is Γ-patient, $P \vdash_{ws} \rho(t) \stackrel{\epsilon}{\Longrightarrow} \rho'(t)$. We have $\rho'(x) \models \chi(x)$ for all $x \in var(t)$, so by induction on formula size, $\rho'(t) \models \varphi'$. Hence $\rho(t) \models \langle\epsilon\rangle\varphi'$.

Case 2: There is a Γ-impatient P-ruloid $\frac{H}{t \xrightarrow{\tau} u}$, and a $\chi \in u^{-1}(\langle\epsilon\rangle\varphi')$ with $\psi(x) = \langle\epsilon\rangle\psi'(x)$ if x occurs Γ-liquid in t, and $\psi(x) = \psi'(x)$ otherwise, where

$$\psi'(x) = \begin{cases} \chi(x) \ \wedge \displaystyle\bigwedge_{x \xrightarrow{\beta} y \in H} \langle\beta\rangle\chi(y) \ \wedge \displaystyle\bigwedge_{x \stackrel{\gamma}{\not\rightarrow} \in H} \neg\langle\gamma\rangle\top & \text{if } x \in var(t) \\ \top & \text{if } x \notin var(t) \end{cases}$$

For each x that occurs Γ-liquid in t, $P \vdash_{ws} \rho(x) \stackrel{\epsilon}{\Longrightarrow} p_x \models \psi'(x)$ for some p_x. Let $\rho'(x) = p_x$ if x occurs Γ-liquid in t, and $\rho'(x) = \rho(x)$ otherwise. Since P is Γ-patient, $P \vdash_{ws} \rho(t) \stackrel{\epsilon}{\Longrightarrow} \rho'(t)$. Furthermore $\rho'(x) \models \psi'(x)$ for all $x \in var(t)$. Utilising the case $\varphi = \langle\tau\rangle\varphi'$ of this proof, which we obtained above, it follows that $\rho'(t) \models \langle\tau\rangle\varphi'$. Hence $\rho(t) \models \langle\epsilon\rangle\varphi'$.

– $\varphi = \langle\hat{\tau}\rangle\varphi'$
($\Rightarrow$) Suppose $\rho(t) \models \langle\hat{\tau}\rangle\varphi'$. Then either $\rho(t) \models \varphi'$, or $P \vdash_{ws} \rho(t) \xrightarrow{\tau} p \models \varphi'$ for some p. In the first case, by induction on formula size, there is a $\psi{\in}t^{-1}(\varphi')$ such that $\rho(x) \models \psi(x)$ for all $x \in \mathit{var}(t)$; by Def. 13.5a, $\psi \in t^{-1}(\langle\hat{\tau}\rangle\varphi')$, and we are done. In the second case, by Prop. 1 there is a P-ruloid $\frac{H}{t\xrightarrow{\tau}u}$ and a closed substitution ρ' with $P \vdash_{ws} \rho'(\mu)$ for all $\mu \in H$, $\rho'(t) = \rho(t)$, i.e. $\rho'(x) = \rho(x)$ for all $x \in \mathit{var}(t)$, and $\rho'(u) = p$. Since $\rho'(u) \models \varphi'$, by induction on formula size, there is a $\chi \in u^{-1}(\varphi')$ such that $\rho'(z) \models \chi(z)$ for each $z \in \mathit{var}(u)$. Furthermore, by Lem. 2, $\rho'(z) \models \chi(z) \equiv \top$ for each $z \notin \mathit{var}(u)$. We distinguish two cases.

CASE 1: $\frac{H}{t\xrightarrow{\tau}u}$ is Γ-patient. By Remark 1, using that t is univariate, H must be of the form $\{x_0 \xrightarrow{\tau} y_0\}$, with $u{=}t[y_0/x_0]$, and the unique occurrence of y_0 in u being Γ-liquid. Let $\sigma : V \to V$ be the bijection that swaps x_0 and y_0, so that $u = \sigma(t)$. Let $\psi(y_0) = \langle\hat{\tau}\rangle\chi(y_0)$ and $\psi(x) = \chi(x)$ for $x \neq y_0$. By Def. 13.5b, $\psi \in u^{-1}(\langle\hat{\tau}\rangle\varphi)$. According to Lem. 3, there is a $\psi' \in t^{-1}(\langle\epsilon\rangle\varphi')$ satisfying $\psi'(x) \equiv \psi(\sigma(x))$ for all $x \in V$.
For each $x \in \mathit{var}(t)\backslash\{x_0\}$, $\rho(x) = \rho'(x) \models \chi(x) = \psi(x) \equiv \psi'(x)$, so $\rho(x) \models \psi'(x)$. Furthermore, $P \vdash_{ws} \rho'(x_0) \xrightarrow{\tau} \rho'(y_0) \models \chi(y_0)$, hence $\rho(x_0) = \rho'(x_0) \models \langle\hat{\tau}\rangle\chi(y_0) = \psi(y_0) \equiv \psi'(x_0)$, so $\rho(x_0) \models \psi'(x_0)$.

CASE 2: $\frac{H}{t\xrightarrow{\tau}u}$ is Γ-impatient. Define $\psi \in t^{-1}(\langle\hat{\tau}\rangle\varphi')$ as in Def. 13.5c, using $\frac{H}{t\xrightarrow{\tau}u}$ and χ. Let $x \in \mathit{var}(t)$. For each $x \xrightarrow{\beta} y \in H$, $P \vdash_{ws} \rho'(x) \xrightarrow{\beta} \rho'(y)$ and $\rho'(y) \models \chi(y)$, so $\rho'(x) \models \langle\beta\rangle\chi(y)$. Moreover, for each $x \overset{\gamma}{\not\longrightarrow} \in H$, $P \vdash_{ws} \rho'(x) \overset{\gamma}{\not\longrightarrow}$, so the consistency of $\vdash_{ws}$ yields $P \not\vdash_{ws} \rho'(x) \xrightarrow{\gamma} q$ for all q, and thus $\rho'(x) \models \neg\langle\gamma\rangle\top$. Hence $\rho(x) = \rho'(x) \models \psi(x)$.

($\Leftarrow$) Suppose $\psi \in t^{-1}(\langle\hat{\tau}\rangle\varphi')$ with $\rho(x) \models \psi(x)$ for all $x \in \mathit{var}(t)$. According to Def. 13.5 we can distinguish three cases.

CASE 1: $\psi{\in}t^{-1}(\varphi')$. By induction on formula size, $\rho(t){\models}\varphi'$, so $\rho(t){\models}\langle\hat{\tau}\rangle\varphi'$.

CASE 2: Some x_0 occurs Γ-liquid in t, and there is a $\chi \in t^{-1}(\varphi')$ with $\psi(x_0) = \langle\hat{\tau}\rangle\chi(x_0)$, and $\psi(x) = \chi(x)$ otherwise. Then $\rho(x) \models \chi(x)$ for each $x \in \mathit{var}(t)\backslash\{x_0\}$. Furthermore, $\rho(x_0) \models \langle\hat{\tau}\rangle\chi(x_0)$, so either $\rho(x_0) \models \chi(x_0)$ or $P \vdash_{ws} \rho(x_0) \xrightarrow{\tau} p \models \chi(x_0)$ for some p. In the first case, by induction on formula size, $\rho(t) \models \varphi'$, so $\rho(t) \models \langle\hat{\tau}\rangle\varphi'$, and we are done. In the second case, by the presence of all Γ-patience rules, $P \vdash_{ws} \rho(t) \xrightarrow{\tau} \rho(t[p/x_0])$. Furthermore, by induction on formula size, $\rho(t[p/x_0]) \models \varphi'$. Hence $\rho(t) \models \langle\hat{\tau}\rangle\varphi'$.

CASE 3: There is a Γ-impatient P-ruloid $\frac{H}{t\xrightarrow{\tau}u}$, and a $\chi \in u^{-1}(\langle\epsilon\rangle\varphi')$ with $\psi(x)$ as given in Def. 13.5c. Utilising the case $\varphi = \langle\tau\rangle\varphi'$ of this proof, obtained above, it follows that $\rho(t) \models \langle\tau\rangle\varphi'$. Hence $\rho(t) \models \langle\hat{\tau}\rangle\varphi'$.

Finally, suppose t is not univariate. Let $t = \sigma(u)$ for some univariate u and non-injective substitution $\sigma : \mathit{var}(u) \to V$. Then $\rho(\sigma(u)) \models \varphi \Leftrightarrow \exists\chi \in u^{-1}(\varphi)\ \forall z \in \mathit{var}(u) : \rho(\sigma(z)) \models \chi(z) \Leftrightarrow \exists\chi \in u^{-1}(\varphi)\ \forall x \in \mathit{var}(t) : \rho(x) \models \bigwedge_{z\in\sigma^{-1}(x)} \chi(z) \Leftrightarrow \exists\psi \in t^{-1}(\varphi)\ \forall x \in \mathit{var}(t) : \rho(x) \models \psi(x)$. □

The part of Thm. 2 that deals with the modalities $\bigwedge_{i\in I}$, $\neg$ and $\langle\alpha\rangle$ only, has already been established in [12]. There, a few examples are given showing how Def. 13 can be used to decompose a modal formula, as well as a counterexample showing that the completeness requirement in Thm. 2 cannot simply be skipped. The inclusion of the modalities $\langle\epsilon\rangle$ and $\langle\hat{\tau}\rangle$ is new. The following example illustrates the use of the decomposition method on a formula with the modality $\langle\epsilon\rangle$.

Example 2. Consider basic CCS, consisting of the inaction constant $\mathbf{0}$, the prefix operator αt where α ranges over some set of actions containing the internal action τ, alternative composition t_1+t_2, and the merge $t_1\|t_2$. The transition rules are:

$$\frac{}{\alpha x \xrightarrow{\alpha} x} \qquad \frac{x_1 \xrightarrow{\alpha} y}{x_1+x_2 \xrightarrow{\alpha} y} \qquad \frac{x_2 \xrightarrow{\alpha} y}{x_1+x_2 \xrightarrow{\alpha} y}$$

$$\frac{x_1 \xrightarrow{\alpha} y}{x_1\|x_2 \xrightarrow{\alpha} y\|x_2} \qquad \frac{x_2 \xrightarrow{\alpha} y}{x_1\|x_2 \xrightarrow{\alpha} x_1\|y}$$

This standard TSS is complete and in ready simulation format.

Let Γ be defined to hold only for the two arguments of the merge. The rules $\frac{x_1 \xrightarrow{\tau} y}{x_1\|x_2 \xrightarrow{\tau} y\|x_2}$ and $\frac{x_2 \xrightarrow{\tau} y}{x_1\|x_2 \xrightarrow{\tau} x_1\|y}$ are Γ-patience rules. They make the TSS Γ-patient.

We compute $(x_1\|x_2)^{-1}(\langle\epsilon\rangle\langle a\rangle\top)$. As there is no Γ-impatient ruloid $\frac{H}{x_1\|x_2 \xrightarrow{\tau} u}$, Def. 13.4b is vacuous. By Def. 13.4a, for each $\psi\in(x_1\|x_2)^{-1}(\langle\epsilon\rangle\langle a\rangle\top)$ we have $\psi(x_1)=\langle\epsilon\rangle\chi(x_1)$ and $\psi(x_2)=\langle\epsilon\rangle\chi(x_2)$ for some $\chi\in(x_1\|x_2)^{-1}(\langle a\rangle\top)$. According to Def. 13.3, we have $(x_1\|x_2)^{-1}(\langle a\rangle\top)=\{\chi_1,\chi_2\}$, where χ_1 and χ_2 are constructed from the only P-ruloids with a conclusion $x_1\|x_2 \xrightarrow{a} \_$, namely the two rules in the TSS themselves, together with $\xi_1\in(y\|x_2)^{-1}(\top)$ resp. $\xi_2\in(x_1\|y)^{-1}(\top)$:

$$\begin{array}{ll} \chi_1(x_1)=\xi_1(x_1)\langle a\rangle\xi_1(y)\equiv\langle a\rangle\top & \chi_2(x_1)=\top \\ \chi_1(x_2)=\top & \chi_2(x_2)=\xi_2(x_2)\langle a\rangle\xi_2(y)\equiv\langle a\rangle\top \end{array}$$

Hence $(x_1\|x_2)^{-1}(\langle\epsilon\rangle\langle a\rangle\top)=\{\psi_1,\psi_2\}$ with ψ_1 and ψ_2 defined as follows:

$$\begin{array}{ll} \psi_1(x_1)=\langle\epsilon\rangle\chi_1(x_1)\equiv\langle\epsilon\rangle\langle a\rangle\top & \psi_2(x_1)=\langle\epsilon\rangle\chi_2(x_1)=\langle\epsilon\rangle\top\equiv\top \\ \psi_1(x_2)=\langle\epsilon\rangle\chi_1(x_2)=\langle\epsilon\rangle\top\equiv\top & \psi_2(x_2)=\langle\epsilon\rangle\chi_2(x_2)\equiv\langle\epsilon\rangle\langle a\rangle\top \end{array}$$

4 Branching Bisimilarity as a Congruence

A behavioural equivalence $\sim$ is a *congruence* for a function symbol f defined on an LTS 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 call $\sim$ a congruence *for a TSS* (Σ,R), if it is a congruence for all function symbols from the signature Σ with respect to the LTS generated by (Σ,R). This is the case if for any open term $t\in\mathbb{T}(\Sigma)$ and any closed substitutions $\rho,\rho':V\to\mathbb{T}$ we have that

$$\forall x\in var(t).\ \rho(x)\sim\rho'(x) \quad\Rightarrow\quad \rho(t)\sim\rho'(t)\ .$$

A *congruence format* for $\sim$ is a list of syntactic restrictions on TSSs, such that $\sim$ is guaranteed to be a congruence for any TSS satisfying these restrictions.

We proceed to apply the decomposition method from the previous section to derive congruence formats for weak and rooted weak semantics. We start, in this section, by considering branching and rooted 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. We prove that the branching bisimulation format guarantees that a formula from $\mathbb{O}_b$ is always decomposed into formulas from $\mathbb{O}_b^{\equiv}$ (see Prop. 3). Likewise, the rooted branching bisimulation format guarantees that a formula from $\mathbb{O}_{rb}$ is always decomposed into formulas from $\mathbb{O}_{rb}^{\equiv}$ (see Prop. 4). This implies the desired congruence results (see Thm. 3 and Thm. 4, respectively).

4.1 Congruence format

We formulate a syntactic format for standard TSSs, called the rooted branching bisimulation format. The branching bisimulation format is defined by means of one simple restriction (namely, Λ is universal) on top of the rooted branching bisimulation format. Our aim for the rest of this section will be to prove that the (rooted) branching bisimulation format guarantees that (rooted) branching bisimilarity is a congruence.

We give some intuition for the conditions below, using process algebraic notations. Recall that Λ marks running processes; to maintain this marking, Λ-liquid arguments of the source and right-hand sides of premises are only allowed to occur Λ-liquid in a rule (conditions 1,2). Furthermore, since $\aleph$ marks the processes that can execute immediately, only $\aleph$-liquid arguments of the source are allowed to be *tested*, i.e. to occur $\aleph$-liquid in left-hand sides of premises (condition 3). In arguments containing running processes, the rootedness property of $\leftrightarroweq_{rb}$ has been lost; so in view of branching bisimilar processes like $a\mathbf{0}$ and $\tau a\mathbf{0}$ (cf. Ex. 2), $\aleph\cap\Lambda$-liquid arguments of the source are not allowed to be tested in negative premises (condition 4b). Likewise, consider for example processes p and q with no other outgoing transitions then $p \xrightarrow{\tau} q$, $q \xrightarrow{\tau} p$, $p \xrightarrow{a} \mathbf{0}$ and $q \xrightarrow{b} \mathbf{0}$. We have $p \leftrightarroweq_b a\mathbf{0} + b\mathbf{0}$, which implies that $\aleph\cap\Lambda$-liquid arguments of the source cannot be tested multiple times in positive premises (condition 4a). Finally, in view of branching bisimilar processes like $a\mathbf{0}$ and $\tau a\mathbf{0}$, testing for τ-transitions from $\aleph\cap\Lambda$-liquid arguments is only allowed when applying patience rules (condition 4c).

Definition 14. A standard 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 $x \in var(t)$.[4]

1. Right-hand sides of positive premises occur only Λ-liquid in u.

[4] For the rooted branching bisimulation format in Def. 15, only the requirements for rules in which t is univariate matter. The formulation of Def. 14 for general terms t paves the way for Lem. 5 and Prop. 2.

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.

Lookahead must be forbidden in view of rooted branching bisimilar processes like $ab\mathbf{0}$ and $a\tau b\mathbf{0}$. Therefore the rooted branching bisimulation format is subsumed by the ready simulation format.

Definition 15. A standard TSS is in *rooted branching bisimulation format* if it is in ready simulation format and, for some $\aleph$ and Λ, it is $\aleph \cap \Lambda$-patient and only contains rules that are rooted branching bisimulation safe w.r.t. $\aleph$ and Λ.

This TSS is in *branching bisimulation format* if moreover Λ is universal.

Remark 2. If a standard TSS P is in rooted branching bisimulation format, then there are smallest predicates $\aleph_0$ and Λ_0 such that P is in rooted branching bisimulation format w.r.t. $\aleph_0$ and Λ_0. Namely the Λ_0-liquid arguments are *generated* by conditions 1 and 2 of Def. 14; they are the smallest collection of arguments such that these two requirements are satisfied. Likewise the $\aleph_0$-liquid arguments are generated by condition 3, which can be read as "If x occurs $\aleph$-liquid in H, then the unique occurrence of x in t is $\aleph$-liquid." For any standard TSS P in ready simulation format, $\aleph_0$ and Λ_0 are determined in this way, and whether P is in rooted branching bisimulation format then depends solely on whether it is $\aleph_0 \cap \Lambda_0$-patient, and condition 4 of Def. 14 is satisfied by all rules in P.

4.2 Preservation of syntactic restrictions

In the definition of modal decomposition, we did not use the rules from the original standard TSS P, but the P-ruloids. Therefore we must verify that if P is in rooted branching bisimulation format, then the P-ruloids are rooted branching bisimulation safe (Prop. 2). The key part of the proof is to show that the syntactic restriction of decent rooted branching bisimulation safety is preserved under irredundant provability (Lem. 5). The adjective irredundant is essential here; this preservation result would clearly fail if "junk" could be added to the premises of derived rules.

In the proofs of the preservation lemma below, rules with a negative conclusion will play an important role. For this reason, the notion of rooted branching bisimulation safety first needs to be extended to non-standard rules. In Sect. 2.5 it was explained that in the construction of P-ruloids, a non-standard rule $\frac{H}{f(x_1,\ldots,x_n) \stackrel{\alpha}{\nrightarrow}}$ is obtained by picking one premise from each standard rule with a conclusion of the form $f(x_1,\ldots,x_n) \xrightarrow{\alpha} t$, and including the denial of each of the selected premises as a premise in H. The following definition is tailored in such a way that applying this procedure to standard ntytt rules that are rooted branching bisimulation safe gives rise to non-standard ntytt rules that are again rooted branching bisimulation safe.

Definition 16. An ntytt rule $r = \frac{H}{t \stackrel{\alpha}{\nrightarrow}}$ is *rooted branching bisimulation safe* w.r.t. $\aleph$ and Λ if it satisfies conditions 2 and 3 of Def. 14.[5]

Lemma 5. *Let P be a TSS in decent ntyft format, in which each transition rule is rooted branching bisimulation safe w.r.t. $\aleph$ and Λ. Then any ntytt rule irredundantly provable from P is rooted branching bisimulation safe w.r.t. $\aleph$ and Λ.*

Proof. Let an ntytt rule $\frac{H}{t \xrightarrow{\alpha} u}$ or $\frac{H}{t \stackrel{\alpha}{\nrightarrow}}$ be irredundantly provable from P, by means of a proof π. We prove, using structural induction with respect to π, that this rule is rooted branching bisimulation safe w.r.t. $\aleph$ and Λ.

Induction basis: Suppose π has only one node. Then $\frac{H}{t \xrightarrow{\alpha} u}$ equals $\frac{t \xrightarrow{\alpha} u}{t \xrightarrow{\alpha} u}$ (so u is a variable), or $\frac{H}{t \stackrel{\alpha}{\nrightarrow}}$ equals $\frac{t \stackrel{\alpha}{\nrightarrow}}{t \stackrel{\alpha}{\nrightarrow}}$. Both rules are rooted branching bisimulation safe w.r.t. $\aleph$ and Λ.

Induction step: Let $r \in R$ be the rule and σ the substitution used at the bottom of π. By assumption, r is decent, ntyft, and rooted branching bisimulation safe w.r.t. $\aleph$ and Λ. Let

$$\{v_k \xrightarrow{\beta_k} y_k \mid k \in K\} \cup \{w_\ell \stackrel{\gamma_\ell}{\nrightarrow} \mid \ell \in L\}$$

be the set of premises of r, and

$$f(x_1, \ldots, x_{ar(f)}) \xrightarrow{\alpha} v \qquad \text{or} \qquad f(x_1, \ldots, x_{ar(f)}) \stackrel{\alpha}{\nrightarrow}$$

the conclusion of r. Then $\sigma(f(x_1, \ldots, x_{ar(f)})) = t$ and, in the first case, $\sigma(v) = u$. Moreover, rules $r_k = \frac{H_k}{\sigma(v_k) \xrightarrow{\beta_k} \sigma(y_k)}$ for each $k \in K$ and $r_\ell = \frac{H_\ell}{\sigma(w_\ell) \stackrel{\gamma_\ell}{\nrightarrow}}$ for each $\ell \in L$ are irredundantly provable from P by means of strict subproofs of π, where $H = \bigcup_{k \in K} H_k \cup \bigcup_{\ell \in L} H_\ell$.

As r is decent, $var(v_k) \subseteq \{x_1, \ldots, x_{ar(f)}\}$, so $var(\sigma(v_k)) \subseteq var(t)$ for each $k \in K$. Likewise, $var(\sigma(w_\ell)) \subseteq var(t)$ for each $\ell \in L$. From $rhs(H) \cap var(t) = \emptyset$ it follows that $rhs(H_k) \cap var(\sigma(v_k)) = \emptyset$ for each $k \in K$, and $rhs(H_\ell) \cap var(\sigma(w_\ell)) = \emptyset$ for each $\ell \in L$. So for each $k \in K$ and $\ell \in L$, the rules r_k and r_ℓ are ntytt rules. By Lem. 1, they are decent. And by induction, they are rooted branching bisimulation safe w.r.t. $\aleph$ and Λ.

We prove that $\frac{H}{t \xrightarrow{\alpha} u}$ satisfies conditions 1–4 of Def. 14, and that $\frac{H}{t \stackrel{\alpha}{\nrightarrow}}$ satisfies conditions 2 and 3 of Def. 16.

1. Consider $\frac{H}{t \xrightarrow{\alpha} u}$. Let $z \in rhs(H)$. Then $z \notin var(t)$, so $z \notin var(\sigma(x_i))$ for each $i \in \{1, \ldots, ar(f)\}$. As r is decent, $var(v_k) \subseteq \{x_1, \ldots, x_{ar(f)}\}$, so $z \notin var(\sigma(v_k))$ for each $k \in K$. Hence, if $z \in var(\sigma(y_{k_0}))$ for some $k_0 \in K$,

[5] The syntactic restrictions on non-standard rules are usually not simply a subset of the ones on standard rules; typically, requirements on positive premises become requirements on negative premises, and vice versa. See for example the failure trace format in [6].

then by the decency of r_{k_0}, $z \in \mathit{rhs}(H_{k_0})$. Since r_{k_0} is rooted branching bisimulation safe w.r.t. $\aleph$ and Λ, by condition 1 of Def. 14, z occurs only Λ-liquid in $\sigma(y_{k_0})$. Furthermore, since r is rooted branching bisimulation safe w.r.t. $\aleph$ and Λ, by condition 1 of Def. 14, y_{k_0} occurs only Λ-liquid in v. By the decency of r, $\mathit{var}(v) \subseteq \{x_1, \ldots, x_{ar(f)}\} \cup \{y_k \mid k \in K\}$. Concluding, z occurs only Λ-liquid in $\sigma(v) = u$.

2. Let $x \in \mathit{var}(t)$ occur only Λ-liquid in t. Let I denote $\{i \in \{1, \ldots, ar(f)\} \mid x \in \mathit{var}(\sigma(x_i))\}$. Since $t = \sigma(f(x_1, \ldots, x_{ar(f)}))$, for each $i \in I$, $\Lambda(f, i)$ and x occurs only Λ-liquid in $\sigma(x_i)$. Since r is rooted branching bisimulation safe w.r.t. $\aleph$ and Λ, by condition 2 of Def. 14 or Def. 16, the x_i for all $i \in I$ occur only Λ-liquid in r. Hence x occurs only Λ-liquid in $\sigma(v_k)$ for all $k \in K$ and in $\sigma(w_\ell)$ for all $\ell \in L$. Since the r_k for all $k \in K$ and r_ℓ for all $\ell \in L$ are rooted branching bisimulation safe w.r.t. $\aleph$ and Λ, and decent, it follows using condition 2 that x occurs only Λ-liquid in these rules. (The reference to the decency of r_k and r_ℓ is needed in case x does not occur in their source.) So x occurs only Λ-liquid in H. Moreover, x occurs only Λ-liquid in $\sigma(y_k)$ for all $k \in K$. Since r is rooted branching bisimulation safe w.r.t. $\aleph$ and Λ, by condition 1 of Def. 14, the y_k for all $k \in K$ occur only Λ-liquid in v. And for each $i \in I$, $\Lambda(f, i)$ implies that x_i occurs only Λ-liquid in v. We already noted that x occurs only Λ-liquid in $\sigma(x_i)$ for each $i \in \{1, \ldots, ar(f)\}$. By the decency of r, $\mathit{var}(v) \subseteq \{x_1, \ldots, x_{ar(f)}\} \cup \{y_k \mid k \in K\}$. Hence x occurs only Λ-liquid in $\sigma(v) = u$. Concluding, x occurs only Λ-liquid in $\frac{H}{t \xrightarrow{\alpha} u}$ or $\frac{H}{t \xrightarrow{\alpha}\!\!\!\!\!/\ \ }$.

3. Suppose that x occurs only $\aleph$-frozen in t. Then, for each $i \in \{1, \ldots, ar(f)\}$, either $\neg\aleph(f, i)$, or x occurs only $\aleph$-frozen in $\sigma(x_i)$. In the first case, since r is rooted branching bisimulation safe w.r.t. $\aleph$ and Λ, by condition 3 of Def. 14 or Def. 16, x_i occurs only $\aleph$-frozen in v_k for all $k \in K$ and w_ℓ for all $\ell \in L$. So x occurs only $\aleph$-frozen in $\sigma(v_k)$ for all $k \in K$ and $\sigma(w_\ell)$ for all $\ell \in L$. Since r_k for all $k \in K$ and r_ℓ for all $\ell \in L$ are rooted branching bisimulation safe w.r.t. $\aleph$ and Λ, and decent, it follows using conditions 3 of Def. 14 and Def. 16 that x occurs only $\aleph$-frozen in H.

4. Consider $\frac{H}{t \xrightarrow{\alpha} u}$. Suppose that x has exactly one $\aleph$-liquid occurrence in t, which is also Λ-liquid. Then there is an $i_0 \in \{1, \ldots, ar(f)\}$ with $\aleph(f, i_0)$ and $\Lambda(f, i_0)$ such that x has exactly one $\aleph$-liquid occurrence in $\sigma(x_{i_0})$, which is also Λ-liquid. Furthermore, for each $i \in \{1, \ldots, ar(f)\} \backslash \{i_0\}$, either $\neg\aleph(f, i)$, or x occurs only $\aleph$-frozen in $\sigma(x_i)$. Since r is rooted branching bisimulation safe w.r.t. $\aleph$ and Λ, by condition 3 of Def. 14, if $\neg\aleph(f, i)$, then x_i occurs only $\aleph$-frozen in v_k for all $k \in K$, as well as in w_ℓ for all $\ell \in L$. And by condition 4 of Def. 14, there is a $K' \subseteq K$ containing at most one element such that x_{i_0} has exactly one $\aleph$-liquid occurrence in v_k if $k \in K'$, and occurs only $\aleph$-frozen in v_k for all $k \in K \backslash K'$, as well as in w_ℓ for all $\ell \in L$. Moreover, by condition 2 of Def. 14, the unique $\aleph$-liquid occurrence of x_{i_0} in v_k must be Λ-liquid. Hence x has exactly one $\aleph$-liquid occurrence in $\sigma(v_k)$ if $k \in K'$, which is also Λ-liquid, and occurs only $\aleph$-frozen in $\sigma(v_k)$ for all $k \in K \backslash K'$, as well

as in w_ℓ for all $\ell \in L$. Since the r_k for all $k \in K$ and r_ℓ for all $\ell \in L$ are rooted branching bisimulation safe w.r.t. $\aleph$ and Λ, by condition 4 of Def. 14, x has at most one $\aleph$-liquid occurrence in (the left-hand side of) one premise in H_k, which must be positive, if $k \in K'$. And by conditions 3 of Def. 14 and Def. 16, x occurs only $\aleph$-frozen in the premises in H_k for all $k \in K \backslash K'$, and in those in H_ℓ for all $\ell \in L$. Concluding, since $H = \bigcup_{k\in K} H_k \cup \bigcup_{\ell \in L} H_\ell$, and K' contains at most one element, x has at most one $\aleph$-liquid occurrence in (the left-hand side of) one premise in H, which must be positive.

Suppose that x has an $\aleph$-liquid occurrence in a positive premise in H with label τ. Then clearly $K' = \{k_0\}$, where x has exactly one $\aleph$-liquid occurrence in $\sigma(v_{k_0})$, which is also Λ-liquid, and in a positive premise in H_{k_0} with label τ. Since $r_{k_0} = \frac{H_{k_0}}{\sigma(v_{k_0}) \xrightarrow{\beta_{k_0}} \sigma(y_{k_0})}$ is rooted branching bisimulation safe w.r.t. $\aleph$ and Λ, by condition 4 of Def. 14, r_{k_0} must be $\aleph \cap \Lambda$-patient. In particular, $\beta_{k_0} = \tau$. Since moreover $\aleph \cap \Lambda(f, i_0)$, x_{i_0} has an $\aleph \cap \Lambda$-liquid occurrence in v_{k_0}, and r is rooted branching bisimulation safe w.r.t. $\aleph$ and Λ, by condition 4 of Def. 14, r must be an $\aleph \cap \Lambda$-patience rule. This implies that $K = \{k_0\}$ and $L = \emptyset$; so $H = H_{k_0}$. It follows that $\frac{H}{t \xrightarrow{\alpha} u}$ is $\aleph \cap \Lambda$-patient.

Hence $\frac{H}{t \xrightarrow{\alpha} u}$ or $\frac{H}{t \stackrel{\alpha}{\not\longrightarrow}}$ is rooted branching bisimulation safe w.r.t. $\aleph$ and Λ. □

Proposition 2. *Let P be a 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.*

Proof. We recall from Sect. 2.5, that the standard TSS P can be transformed into a TSS P^+ in decent ntyft format; the P-ruloids are those decent nxytt rules that are irredundantly provable from P^+.

As the rules of P are rooted branching bisimulation safe w.r.t. $\aleph$ and Λ, then so are the rules in P^+. Namely, as described in Sect. 2.5, the rules in P^+ are constructed in three steps. The first step (the conversion of P to decent ntyft format) clearly preserves the rooted branching bisimulation format. The second step (the construction to reduce left-hand sides of positive premises to variables) yields an intermediate TSS, all of whose rules are irredundantly provable from P, and thus is covered by Lem. 5. Regarding the final step (constructing non-standard rules with negative conclusions), as said before Def. 16, this definition is tailored in such a way that applying this procedure to standard ntytt rules that are rooted branching bisimulation safe gives rise to non-standard ntytt rules that are again rooted branching bisimulation safe.

Since the rules in P^+ are rooted branching bisimulation safe w.r.t. $\aleph$ and Λ, by Lem. 5, each P-ruloid is rooted branching bisimulation safe w.r.t. $\aleph$ and Λ. □

4.3 Preservation of modal characterisations

Consider a standard TSS in rooted branching bisimulation format, w.r.t. some $\aleph$ and Λ. Def. 13 yields decomposition mappings $\psi \in t^{-1}(\varphi)$, with $\Gamma := \aleph \cap \Lambda$.

In this section we will first prove that if $\varphi \in \mathbb{O}_b$, then $\psi(x) \in \mathbb{O}_b^{\equiv}$ if x occurs only Λ-liquid in t. (That is why in the branching bisimulation format, Λ must be universal.) Next we will prove that if $\varphi \in \mathbb{O}_{rb}$, then $\psi(x) \in \mathbb{O}_{rb}^{\equiv}$ for all variables x. From these preservation results we will, in Sect. 4.4, deduce the promised congruence results for branching bisimilarity and rooted branching bisimilarity, respectively.

Proposition 3. *Let P be an $\aleph \cap \Lambda$-patient standard TSS in ready simulation format, in which each transition 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:*

$$\varphi \in \mathbb{O}_b \;\Rightarrow\; \forall \psi \in t^{-1}(\varphi) : \psi(x) \in \mathbb{O}_b^{\equiv}$$

Proof. We apply simultaneous induction on the structure of $\varphi \in \mathbb{O}_b$ and the construction of ψ. Let $\psi \in t^{-1}(\varphi)$, and let x occur only Λ-liquid in t. First we treat the case where t is univariate. If $x \notin var(t)$, then by Lem. 2, $\psi(x) \equiv \top \in \mathbb{O}_b^{\equiv}$. So suppose x has exactly one, Λ-liquid occurrence in t.

– $\varphi = \bigwedge_{i\in I} \varphi_i$ with $\varphi_i \in \mathbb{O}_b$ for each $i \in I$. By Def. 13.1, $\psi(x) = \bigwedge_{i\in I} \psi_i(x)$ with $\psi_i \in t^{-1}(\varphi_i)$ for each $i \in I$. By induction on formula size, $\psi_i(x) \in \mathbb{O}_b^{\equiv}$ for each $i \in I$, so $\psi(x) \in \mathbb{O}_b^{\equiv}$.
– $\varphi = \neg\varphi'$ with $\varphi' \in \mathbb{O}_b$. By Def. 13.2, there is a function $h : t^{-1}(\varphi') \to var(t)$ such that $\psi(x) = \bigwedge_{\chi \in h^{-1}(x)} \neg\chi(x)$. By induction on formula size, $\chi(x) \in \mathbb{O}_b^{\equiv}$ for each $\chi \in h^{-1}(x)$, so $\psi(x) \in \mathbb{O}_b^{\equiv}$.
– $\varphi = \langle\epsilon\rangle(\varphi_1\langle\hat{\tau}\rangle\varphi_2)$ with $\varphi_1, \varphi_2 \in \mathbb{O}_b$. According to Def. 13.4, we can distinguish two cases.

Case 1: $\psi(x)$ is defined on the basis of Def. 13.4a. Then $\psi(x) = \langle\epsilon\rangle\chi(x)$ if x occurs $\aleph$-liquid in t, or $\psi(x) = \chi(x)$ if x occurs $\aleph$-frozen in t, for some $\chi \in t^{-1}(\varphi_1\langle\hat{\tau}\rangle\varphi_2)$. By Def. 13.1, $\chi(x) = \chi_1(x) \wedge \chi_2(x)$ with $\chi_1 \in t^{-1}(\varphi_1)$ and $\chi_2 \in t^{-1}(\langle\hat{\tau}\rangle\varphi_2)$. By induction on formula size, $\chi_1(x) \in \mathbb{O}_b^{\equiv}$. For $\chi_2(x)$, according to Def. 13.5, we can distinguish three cases.

Case 1.1: $\chi_2(x)$ is defined on the basis of Def. 13.5a. Then $\chi_2 \in t^{-1}(\varphi_2)$. By induction on formula size, $\chi_2(x) \in \mathbb{O}_b^{\equiv}$. Since $\psi(x)$ is of the form $\langle\epsilon\rangle(\chi_1(x) \wedge \chi_2(x))$ or $\chi_1(x) \wedge \chi_2(x)$, it follows that $\psi(x) \in \mathbb{O}_b^{\equiv}$.

Case 1.2: $\chi_2(x)$ is defined on the basis of Def. 13.5b. Then x_0 occurs $\aleph\cap\Lambda$-liquid in t, and either $\chi_2(x) = \langle\hat{\tau}\rangle\xi(x)$ and x occurs $\aleph$-liquid in t (if $x = x_0$) or $\chi_2(x) = \xi(x)$, for some $\xi \in t^{-1}(\varphi_2)$. By induction on formula size, $\xi(x) \in \mathbb{O}_b^{\equiv}$. Since $\psi(x)$ is of the form $\langle\epsilon\rangle(\chi_1(x)\langle\hat{\tau}\rangle\xi(x))$, $\langle\epsilon\rangle(\chi_1(x) \wedge \xi(x))$ or $\chi_1(x) \wedge \xi(x)$, it follows that $\psi(x) \in \mathbb{O}_b^{\equiv}$.

Case 1.3: $\chi_2(x)$ is defined on the basis of Def. 13.5c, employing an $\aleph\cap\Lambda$-impatient P-ruloid $\frac{H}{t \xrightarrow{\tau} u}$ and a $\xi \in u^{-1}(\varphi_2)$. So

$$\chi_2(x) = \xi(x) \;\wedge \bigwedge_{x \xrightarrow{\beta} y \in H} \langle\beta\rangle\xi(y) \;\wedge \bigwedge_{x \overset{\gamma}{\not\rightarrow} \in H} \neg\langle\gamma\rangle\top \,.$$

By Prop. 2, $\frac{H}{t \xrightarrow{\tau} u}$ is rooted branching bisimulation safe. Since the occurrence of x in t is Λ-liquid, by condition 2 of Def. 14, x occurs only Λ-liquid in u. Therefore, by induction on formula size, $\xi(x) \in \mathbb{O}_b^{\equiv}$.

Case 1.3.1: The occurrence of x in t is $\aleph$-frozen. By condition 3 of Def. 14, x does not occur in H. Hence $\chi_2(x) = \xi(x) \in \mathbb{O}_b^{\equiv}$ and thus $\psi(x) \in \mathbb{O}_b^{\equiv}$.

Case 1.3.2: The occurrence of x in t is $\aleph$-liquid. By condition 4 of Def. 14, H has at most one premise of the form $x \xrightarrow{\beta} y$, for which $\beta \neq \tau$, and none of the form $x \overset{\gamma}{\not\longrightarrow}$. Thus either $\chi_2(x) = \xi(x)$—and we are done—or $\chi_2(x) = \xi(x)\langle b\rangle\xi(y)$ with $b \in A$ and $x \xrightarrow{b} y \in H$. In the latter case $\psi(x) \equiv \langle\epsilon\rangle((\chi_1(x) \wedge \xi(x))\langle b\rangle\xi(y))$. By condition 1 of Def. 14, y occurs only Λ-liquid in u, so by induction $\xi(y) \in \mathbb{O}_b^{\equiv}$. It follows that $\psi(x) \in \mathbb{O}_b^{\equiv}$.

Case 2: $\psi(x)$ is defined on the basis of Def. 13.4b, employing an $\aleph\cap\Lambda$-impatient P-ruloid $\frac{H}{t\xrightarrow{\tau}u}$ and a $\chi \in u^{-1}(\langle\epsilon\rangle(\varphi_1\langle\hat{\tau}\rangle\varphi_2))$. By Prop. 2, $\frac{H}{t\xrightarrow{\tau}u}$ is rooted branching bisimulation safe. Since the occurrence of x in t is Λ-liquid, by condition 2 of Def. 14, x occurs only Λ-liquid in u. Therefore, by induction on the construction of ψ, $\chi(x) \in \mathbb{O}_b^{\equiv}$.

Case 2.1: The occurrence of x in t is $\aleph$-frozen. Then

$$\psi(x) = \chi(x) \;\wedge \bigwedge_{x\xrightarrow{\beta}y\in H} \langle\beta\rangle\chi(y) \;\wedge \bigwedge_{x\overset{\gamma}{\not\longrightarrow}\in H} \neg\langle\gamma\rangle\top.$$

By condition 3 of Def. 14, x does not occur in H. So $\psi(x) = \chi(x) \in \mathbb{O}_b^{\equiv}$.

Case 2.2: The occurrence of x in t is $\aleph$-liquid. Then

$$\psi(x) = \langle\epsilon\rangle\left(\chi(x) \;\wedge \bigwedge_{x\xrightarrow{\beta}y\in H} \langle\beta\rangle\chi(y) \;\wedge \bigwedge_{x\overset{\gamma}{\not\longrightarrow}\in H} \neg\langle\gamma\rangle\top\right).$$

By condition 4 of Def. 14, H has at most one premise of the form $x \xrightarrow{\beta} y$, for which $\beta \neq \tau$, and none of the form $x \overset{\gamma}{\not\longrightarrow}$. Thus either $\psi(x) = \langle\epsilon\rangle\chi(x)$—and we are done—or $\psi(x) = \langle\epsilon\rangle\chi(x)\langle b\rangle\chi(y)$ with $b \in A$ and $x \xrightarrow{b} y \in H$. By condition 1 of Def. 14, y occurs only Λ-liquid in u, so by induction $\chi(y) \in \mathbb{O}_b^{\equiv}$. It follows that $\psi(x) \in \mathbb{O}_b^{\equiv}$.

– $\varphi = \langle\epsilon\rangle(\varphi_1\langle a\rangle\varphi_2)$ with $\varphi_1, \varphi_2 \in \mathbb{O}_b$. According to Def. 13.4, we can distinguish two cases.

Case 1: $\psi(x)$ is defined on the basis of Def. 13.4a. Then $\psi(x) = \langle\epsilon\rangle\chi(x)$ if x occurs $\aleph$-liquid in t, or $\psi(x) = \chi(x)$ if x occurs $\aleph$-frozen in t, for some $\chi \in t^{-1}(\varphi_1\langle a\rangle\varphi_2)$. By Def. 13.1, $\chi(x) = \chi_1(x) \wedge \chi_2(x)$ with $\chi_1 \in t^{-1}(\varphi_1)$ and $\chi_2 \in t^{-1}(\langle a\rangle\varphi_2)$. By induction on formula size, $\chi_1(x) \in \mathbb{O}_b^{\equiv}$. And by Def. 13.3,

$$\chi_2(x) = \xi(x) \wedge \bigwedge_{x\xrightarrow{\beta}y\in H} \langle\beta\rangle\xi(y) \wedge \bigwedge_{x\overset{\gamma}{\not\longrightarrow}\in H} \neg\langle\gamma\rangle\top$$

for some P-ruloid $\frac{H}{t\xrightarrow{a}u}$ and $\xi \in u^{-1}(\varphi_2)$. By Prop. 2, $\frac{H}{t\xrightarrow{a}u}$ is rooted branching bisimulation safe w.r.t. $\aleph$ and Λ. Since the occurrence of x in t is Λ-liquid, by condition 2 of Def. 14, x occurs only Λ-liquid in u. Moreover, by condition 1 of Def. 14, variables in $rhs(H)$ occur only Λ-liquid in u. So by induction on formula size, $\xi(x) \in \mathbb{O}_b^{\equiv}$, and $\xi(y) \in \mathbb{O}_b^{\equiv}$ for $x \xrightarrow{\beta} y \in H$. We distinguish two cases.

Case 1.1: The occurrence of x in t is $\aleph$-liquid. Then $\psi(x) = \langle\epsilon\rangle(\chi_1(x) \wedge \chi_2(x))$. The rule $\frac{H}{t \xrightarrow{a} u}$ is rooted branching bisimulation safe w.r.t. $\aleph$ and Λ, and an nxytt rule. By condition 4 of Def. 14, x is the left-hand side of at most one premise in H, which must be positive. Hence either $\chi_2(x) = \xi(x)$, or $\chi_2(x) = \xi(x)\langle\beta\rangle\xi(y)$ with $x \xrightarrow{\beta} y \in H$. Since $a \neq \tau$, by condition 4 of Def. 14, $\beta \neq \tau$. Since $\psi(x)$ is of the form $\langle\epsilon\rangle(\chi_1(x) \wedge \xi(x))$ or $\langle\epsilon\rangle((\chi_1(x) \wedge \xi(x))\langle\beta\rangle\xi(y))$, it follows that $\psi(x) \in \mathbb{O}_b^{\equiv}$.

Case 1.2: The occurrence of x in t is $\aleph$-frozen. Then $\psi(x) = \chi_1(x) \wedge \chi_2(x)$. The rule $\frac{H}{t \xrightarrow{a} u}$ is rooted branching bisimulation safe w.r.t. $\aleph$ and Λ, and an nxytt rule. By condition 3 of Def. 14, x does not occur in H. So $\chi_2(x) = \xi(x)$, and thus $\psi(x) = \chi_1(x) \wedge \xi(x) \in \mathbb{O}_b^{\equiv}$.

Case 2: $\psi(x)$ is defined on the basis of Def. 13.4b. This case proceeds in the same way as case 2 of $\varphi = \langle\epsilon\rangle(\varphi_1\langle\hat{\tau}\rangle\varphi_2)$.

Finally, suppose t is not univariate. Then $t = \sigma(u)$ for some univariate term u and $\sigma : \mathit{var}(u) \to V$ not injective. By Def. 13.6, $\psi(x) = \bigwedge_{z\in\sigma^{-1}(x)} \chi(z)$ for some $\chi \in u^{-1}(\varphi)$. Since u is univariate, and for each $z \in \sigma^{-1}(x)$ the occurrence in u is Λ-liquid, $\chi(z) \in \mathbb{O}_b^{\equiv}$ for all $z \in \sigma^{-1}(x)$. Hence $\psi(x) \in \mathbb{O}_b^{\equiv}$. □

Proposition 4. *Let P be an $\aleph\cap\Lambda$-patient standard TSS in ready simulation format, in which each transition rule is rooted branching bisimulation safe w.r.t. $\aleph$ and Λ. For any term t and variable x:*

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

Proof. We apply simultaneous induction on the structure of $\varphi \in \mathbb{O}_{rb}$ and the construction of ψ. Let $\psi \in t^{-1}(\varphi)$. We restrict attention to the case where t is univariate; the general case then follows just as at the end of the proof of Prop. 3. If $x \notin \mathit{var}(t)$, then by Lem. 2, $\psi(x) \equiv \top \in \mathbb{O}_{rb}^{\equiv}$. So suppose x occurs once in t.

- The cases $\varphi = \bigwedge_{i\in I} \varphi_i$ and $\varphi = \neg\varphi'$ proceed as in the proof of Prop. 3.
- $\varphi = \langle\alpha\rangle\varphi'$ with $\varphi' \in \mathbb{O}_b$. By Def. 13.3,

$$\psi(x) = \chi(x) \wedge \bigwedge_{x \xrightarrow{\beta} y \in H} \langle\beta\rangle\chi(y) \wedge \bigwedge_{x \overset{\gamma}{\not\rightarrow} \in H} \neg\langle\gamma\rangle\top$$

  for some P-ruloid $\frac{H}{t \xrightarrow{\alpha} u}$ and $\chi \in u^{-1}(\varphi')$. By induction on formula size, $\chi(x) \in \mathbb{O}_{rb}^{\equiv}$. (Induction may be applied because $\varphi' \in \mathbb{O}_b \subseteq \mathbb{O}_{rb}$.) By Prop. 2, $\frac{H}{t \xrightarrow{\alpha} u}$ is rooted branching bisimulation safe w.r.t. $\aleph$ and Λ, so by condition 1 of Def. 14, variables in $\mathit{rhs}(H)$ occur only Λ-liquid in u. Hence by Prop. 3, $\chi(y) \in \mathbb{O}_b^{\equiv}$, and thus $\langle\beta\rangle\chi(y) \in \mathbb{O}_{rb}^{\equiv}$, for all $x \xrightarrow{\beta} y \in H$. Moreover, $\neg\langle\gamma\rangle\top \in \mathbb{O}_{rb}^{\equiv}$ for all $x \overset{\gamma}{\not\rightarrow} \in H$. Hence $\psi(x) \in \mathbb{O}_{rb}^{\equiv}$.
- $\varphi \in \mathbb{O}_b$. The cases $\varphi = \bigwedge_{i\in I} \varphi_i$ and $\varphi = \neg\varphi'$ proceed as in the proof of Prop. 3. We therefore focus on the other two cases. If the occurrence of x in t is Λ-liquid, then $\psi(x) \in \mathbb{O}_{rb}^{\equiv}$ follows from Prop. 3. So we can assume that this occurrence is Λ-frozen.

- $\varphi = \langle\epsilon\rangle(\varphi_1\langle\hat{\tau}\rangle\varphi_2)$ with $\varphi_1, \varphi_2 \in \mathbb{O}_b$. According to Def. 13.4, we can distinguish two cases.

  CASE 1: $\psi(x)$ is defined on the basis of Def. 13.4a. Then, since x occurs Λ-frozen in t, $\psi(x) = \chi(x)$ for some $\chi \in t^{-1}(\varphi_1\langle\hat{\tau}\rangle\varphi_2)$. By Def. 13.1, $\chi(x) = \chi_1(x) \wedge \chi_2(x)$ with $\chi_1 \in t^{-1}(\varphi_1)$ and $\chi_2 \in t^{-1}(\langle\hat{\tau}\rangle\varphi_2)$. By induction on formula size, $\chi_1(x) \in \mathbb{O}_{rb}^{\equiv}$. So to prove $\psi(x) \in \mathbb{O}_{rb}^{\equiv}$, it suffices to prove $\chi_2(x) \in \mathbb{O}_{rb}^{\equiv}$. According to Def. 13.5, we can distinguish three cases.

  CASE 1.1: $\chi_2(x)$ is defined on the basis of Def. 13.5a. Then $\chi_2 \in t^{-1}(\varphi_2)$. By induction on formula size, $\chi_2(x) \in \mathbb{O}_{rb}^{\equiv}$.

  CASE 1.2: $\chi_2(x)$ is defined on the basis of Def. 13.5b. Then, since x occurs Λ-frozen in t, $\chi_2(x) = \xi(x)$ for some $\xi \in t^{-1}(\varphi_2)$. By induction on formula size, $\chi_2(x) = \xi(x) \in \mathbb{O}_{rb}^{\equiv}$.

  CASE 1.3: $\chi_2(x)$ is defined on the basis of Def. 13.5c. Then $\chi_2(x) \in \mathbb{O}_{rb}^{\equiv}$ follows in exactly the same way as $\psi(x) \in \mathbb{O}_{rb}^{\equiv}$ in the case $\varphi = \langle\alpha\rangle\varphi'$ of this proof.

  CASE 2: $\psi(x)$ is defined on the basis of Def. 13.4b, using a P-ruloid $\frac{H}{t \xrightarrow{\tau} u}$ and a $\chi \in u^{-1}(\langle\epsilon\rangle(\varphi_1\langle\hat{\tau}\rangle\varphi_1))$. As the occurrence of x in t is Λ-frozen,

  $$\psi(x) = \chi(x) \wedge \bigwedge_{x \xrightarrow{\beta} y \in H} \langle\beta\rangle\chi(y) \wedge \bigwedge_{x \overset{\gamma}{\not\rightarrow} \in H} \neg\langle\gamma\rangle\top$$

  By induction on the construction of ψ, $\chi(x) \in \mathbb{O}_b^{\equiv}$. The rest of the argument proceeds as in the case $\varphi = \langle\alpha\rangle\varphi'$ above.

- $\varphi = \langle\epsilon\rangle(\varphi_1\langle a\rangle\varphi_2)$ with $\varphi_1, \varphi_2 \in \mathbb{O}_b$. According to Def. 13.4, we can distinguish two cases.

  CASE 1: $\psi(x)$ is defined on the basis of Def. 13.4a. Then, since x occurs Λ-frozen in t, $\psi(x) = \chi(x)$ for some $\chi \in t^{-1}(\varphi_1\langle a\rangle\varphi_2)$. By Def. 13.1, $\chi(x) = \chi_1(x) \wedge \chi_2(x)$ with $\chi_1 \in t^{-1}(\varphi_1)$ and $\chi_2 \in t^{-1}(\langle a\rangle\varphi_2)$. By induction on formula size, $\chi_1(x), \chi_2(x) \in \mathbb{O}_{rb}^{\equiv}$. So $\psi(x) \in \mathbb{O}_{rb}^{\equiv}$.

  CASE 2: $\psi(x)$ is defined on the basis of Def. 13.4b. This case proceeds in the same way as case 2 of $\varphi = \langle\epsilon\rangle\varphi_1\langle\hat{\tau}\rangle\varphi_2$. □

4.4 Congruence results

Now we are in a position to prove the promised congruence results for $\leftrightarroweq_b$ and $\leftrightarroweq_{rb}$.

Theorem 3. *Let P be a complete standard TSS in branching bisimulation format. Then $\leftrightarroweq_b$ is a congruence for P.*

Proof. Let ρ, ρ' be closed substitutions and t a term. Suppose that $\rho(x) \leftrightarroweq_b \rho'(x)$ for all $x \in var(t)$; we need to prove that then $\rho(t) \leftrightarroweq_b \rho'(t)$.

By Def. 15 each transition rule in P is rooted branching bisimulation safe w.r.t some $\aleph$ and the universal predicate Λ, and P is $\aleph \cap \Lambda$-patient and in ready simulation format. Let $\rho(t) \models \varphi \in \mathbb{O}_b$. By Thm. 2, taking $\Gamma := \aleph \cap \Lambda$, there is a $\psi \in t^{-1}(\varphi)$ 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. 3, $\psi(x) \in \mathbb{O}_b^{\equiv}$ for all $x \in var(t)$. By Thm. 1, $\rho(x) \leftrightarrows_b \rho'(x)$ implies $\rho(x) \sim_{\mathbb{O}_b^{\equiv}} \rho'(x)$ for all $x \in var(t)$. So $\rho'(x) \models \psi(x)$ for all $x \in var(t)$. Therefore, by Thm. 2, $\rho'(t) \models \varphi$. Likewise, $\rho'(t) \models \varphi \in \mathbb{O}_b$ implies $\rho(t) \models \varphi$. So $\rho(t) \sim_{\mathbb{O}_b} \rho'(t)$. Hence, by Thm. 1, $\rho(t) \leftrightarrows_b \rho'(t)$. □

We can follow the same approach to prove that the rooted branching bisimulation format guarantees that $\leftrightarrows_{rb}$ is a congruence.

Theorem 4. *Let P be a complete standard TSS in rooted branching bisimulation format. Then $\leftrightarrows_{rb}$ is a congruence for P.*

The proof is similar to the one of Thm. 3, except that Prop. 4 is applied instead of Prop. 3; therefore x needs not occur Λ-liquid in t, which is why universality of Λ can be dropped.

5 η-Bisimilarity as a Congruence

We now proceed to derive a congruence format for rooted η-bisimilarity. This format can be formulated by adding one extra syntactic restriction to the rooted branching bisimulation format. The proofs that the resulting format is preserved under the transformation to ruloids, and that it guarantees that rooted η-bisimilarity is a congruence, are for a large part similar to these proofs for the rooted branching bisimulation format. We will therefore only explain how these proofs deviate from the proofs for the rooted branching bisimulation format.

The notion of rooted η-bisimulation safeness is obtained by strengthening condition 1 in the definition of rooted branching bisimulation safeness. The action refinement operator exemplifies that this strengthening is essential; see Sect. 6.6.

Definition 17. A standard ntytt rule $r = \frac{H}{t \xrightarrow{\alpha} u}$ is *rooted η-bisimulation safe* w.r.t. $\aleph$ and Λ if it satisfies conditions 2–4 of Def. 14, together with:

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

Definition 18. A standard TSS is in *rooted η-bisimulation format* if it is in ready simulation format 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 *η-bisimulation format* if moreover Λ is universal.

For non-standard ntytt rules, the notion of rooted η-bisimulation safeness coincides with the notion of rooted branching bisimulation safeness (see Def. 16).

Lemma 6. *Let P be a TSS in decent ntyft format, in which each transition rule is rooted η-bisimulation safe w.r.t. $\aleph$ and Λ. Then any ntytt rule irredundantly provable from P is rooted η-bisimulation safe w.r.t. $\aleph$ and Λ.*

Proof. Since rooted η-bisimulation safeness is stricter than rooted branching bisimulation safeness, preservation of conditions 2–4 of rooted η-bisimulation safeness follows directly from Lem. 5.

We only need to prove that condition $1'$ of Def. 17 is preserved. For this, the part within the proof of Lem. 5 that is devoted to the preservation of condition 1 in Def. 14 can be copied almost literally. The only difference is that to obtain a preservation proof for condition $1'$, the three occurrences of "Λ-liquid" in the preservation proof for condition 1 have to be replaced by "$\aleph\cap\Lambda$-liquid". $\square$

Now the following proposition can be proved in the same way as the corresponding Prop. 2 for the rooted branching bisimulation format.

Proposition 5. *Let P be a TSS in ready simulation format, in which each transition 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 6. *Let P be an $\aleph\cap\Lambda$-patient standard TSS in ready simulation format, in which each transition 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:*

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

Proof. Again, the proof is very similar to the proof of the corresponding Prop. 3 for the rooted branching bisimulation format. We spell out (part of) the only two cases where the proofs really differ. The differences are underlined. It is here that the stronger condition $1'$ will be needed. We recall that it is assumed that t is univariate, and that x has exactly one, Λ-liquid occurrence in t.

- $\varphi = \langle\epsilon\rangle\underline{\varphi'}$ with $\varphi' \in \mathbb{O}_\eta$. According to Def. 13.4, we can distinguish two cases.
  CASE 1: $\psi(x)$ is defined on the basis of Def. 13.4a.
  $\underline{\text{Then either } \psi(x) = \langle\epsilon\rangle\chi(x) \text{ or } \psi(x) = \chi(x) \text{ for some } \chi \in t^{-1}(\varphi').}$
  $\underline{\text{By induction on formula size, } \chi(x) \in \mathbb{O}_\eta^{\equiv}\text{. So } \psi(x) \in \mathbb{O}_\eta^{\equiv}.}$
  Case 2 proceeds as in the proof of Prop. 3 (in that proof the two occurrences of "CASE 2" proceed in the same way). However, CASE 2.2 now ends by: By condition $\underline{1' \text{ of Def. 17}}$, y occurs only $\underline{\aleph\cap\Lambda}$-liquid in u, so by induction $\chi(y) \in \mathbb{O}_\eta^{\equiv}$, $\underline{\text{and according to Lem. 4, } \chi(y) \equiv \langle\epsilon\rangle\chi(y).}$ It follows that $\psi(x) \in \mathbb{O}_\eta^{\equiv}$.
- $\varphi = \langle\epsilon\rangle\varphi_1\langle a\rangle\underline{\langle\epsilon\rangle}\varphi_2$ with $\varphi_1, \varphi_2 \in \mathbb{O}_\eta$. According to Def. 13.4, we can distinguish two cases.
  CASE 1: $\psi(x)$ is defined on the basis of Def. 13.4a. Then $\psi(x) = \langle\epsilon\rangle\chi(x)$ if x occurs $\aleph$-liquid in t, or $\psi(x) = \chi(x)$ if x occurs $\aleph$-frozen in t, for some $\chi \in t^{-1}(\varphi_1\langle a\rangle\underline{\langle\epsilon\rangle}\varphi_2)$. By Def. 13.1, $\chi(x) = \chi_1(x) \wedge \chi_2(x)$ with $\chi_1 \in t^{-1}(\varphi_1)$ and $\chi_2 \in t^{-1}(\langle a\rangle\underline{\langle\epsilon\rangle}\varphi_2)$. By induction on formula size, $\chi_1(x) \in \mathbb{O}_\eta^{\equiv}$. And by Def. 13.3,

$$\chi_2(x) = \xi(x) \wedge \bigwedge_{x \xrightarrow{\beta} y \in H} \langle\beta\rangle\xi(y) \wedge \bigwedge_{x \stackrel{\gamma}{\not\rightarrow} \in H} \neg\langle\gamma\rangle\top$$

for some P-ruloid $\frac{H}{t\xrightarrow{a}u}$ and $\xi \in u^{-1}(\underline{\langle\epsilon\rangle}\varphi_2)$. By Prop. 5, $\frac{H}{t\xrightarrow{a}u}$ is rooted η-bisimulation safe w.r.t. $\aleph$ and Λ. Since the occurrence of x in t is Λ-liquid, by condition 2 of Def. 14, x occurs only Λ-liquid in u. Moreover, $\underline{\text{by condition } 1' \text{ of Def. 17}}$, variables in $rhs(H)$ occur only $\underline{\aleph\cap\Lambda}$-liquid in u. So by induction on formula size, $\xi(x) \in \mathbb{O}_\eta^{\equiv}$, and $\xi(y) \in \mathbb{O}_\eta^{\equiv}$ for each $x \xrightarrow{\beta} y \in H$. $\underline{\text{According to Lem. 4, } \xi(y) \equiv \langle\epsilon\rangle\xi(y).}$ We distinguish two cases.

Case 1.1: The occurrence of x in t is $\aleph$-liquid. Then $\psi(x) = \langle\epsilon\rangle(\chi_1(x) \wedge \chi_2(x))$. The rule $\frac{H}{t\xrightarrow{a}u}$ is rooted η-bisimulation safe w.r.t. $\aleph$ and Λ, and an nxytt rule. By condition 4 of Def. 14, x is the left-hand side of at most one premise in H, which must be positive. Hence either $\chi_2(x) = \xi(x)$, or $\chi_2(x) = \xi(x)\langle\beta\rangle\xi(y)$ with $x \xrightarrow{\beta} y \in H$. Since $a \neq \tau$, by condition 4 of Def. 14, $\beta \neq \tau$. Since $\psi(x)$ is of the form $\langle\epsilon\rangle(\chi_1(x) \wedge \xi(x))$ or $\langle\epsilon\rangle(\chi_1(x) \wedge \xi(x))\langle\beta\rangle\xi(y) \underline{\equiv \langle\epsilon\rangle(\chi_1(x) \wedge \xi(x))\langle\beta\rangle\langle\epsilon\rangle\xi(y)}$, it follows that $\psi(x) \in \mathbb{O}_\eta^{\equiv}$.

Case 1.2 and case 2 proceed as in the proof of Prop. 3. □

Proposition 7. *Let P be an $\aleph\cap\Lambda$-patient standard TSS in ready simulation format, in which each transition rule is rooted η-bisimulation safe w.r.t. $\aleph$ and Λ. For any term t and variable x:*

$$\varphi \in \mathbb{O}_{r\eta} \;\Rightarrow\; \forall\psi \in t^{-1}(\varphi) : \psi(x) \in \mathbb{O}_{r\eta}^{\equiv}$$

Proof. Again, the proof is very similar to the proof of the corresponding Prop. 4 for the rooted branching bisimulation format. As in the previous proof, the only real difference is that we have to exploit the stronger condition $1'$ of Def. 17: for each P-ruloid $\frac{H}{t\xrightarrow{\alpha}u}$, each $y \in rhs(H)$ can occur only $\aleph\cap\Lambda$-liquid in u; so by Lem. 4, if $\chi \in u^{-1}(\langle\epsilon\rangle\varphi)$, then $\chi(y) \equiv \langle\epsilon\rangle\chi(y)$. Moreover, we need to observe that $\neg\langle\gamma\rangle\top \equiv \neg\langle\gamma\rangle\langle\epsilon\rangle\top \in \mathbb{O}_{r\eta}$. □

The proofs of the following congruence theorems for (rooted) η-bisimulation are omitted, as they are almost identical to the proofs of the corresponding congruence theorems for (rooted) branching bisimilarity.

Theorem 5. *Let P be a complete standard TSS in η-bisimulation format. Then $\underline{\leftrightarrow}_\eta$ is a congruence for P.*

Theorem 6. *Let P be a complete standard TSS in rooted η-bisimulation format. Then $\underline{\leftrightarrow}_{r\eta}$ is a congruence for P.*

6 Examples

In this section we present some applications of our congruence formats, as well as some counterexamples to show the need for the requirements in our congruence formats.

6.1 Basic process algebra

Basic process algebra BPA [4] assumes a collection *Act* of constants, called *atomic actions*, which upon execution terminate successfully. The signature of BPA moreover includes function symbols $\_+\_$ and $\_\cdot\_$ of arity two, called *alternative composition* and *sequential composition*, respectively. Intuitively, $t_1 + t_2$ executes either t_1 or t_2, while $t_1 \cdot t_2$ first executes t_1 and upon successful termination executes t_2. In addition to τ, we assume two special constants outside *Act*: ε represents successful termination, while the deadlock δ does not display any behaviour. These intuitions are made precise by means of the transition rules for $\mathrm{BPA}_{\varepsilon\delta\tau}$ presented below. In these rules, ℓ ranges over $Act \cup \{\tau\}$, and α over $Act \cup \{\tau, \surd\}$.

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

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

The TSS above is in ready simulation format: it consists of ntyft rules that do not contain lookahead. In view of condition 1 of Def. 14, we make the first argument of sequential composition Λ-liquid, and the two arguments of alternative composition and the second argument of sequential composition Λ-frozen. This is in line with the intuition that Λ-liquid arguments can contain running processes. The two rules for sequential composition clearly satisfy condition 2 of Def. 14: the variable x_1 only occurs Λ-liquid in both rules. We make the two arguments of both alternative and sequential composition $\aleph$-liquid, because these four arguments can all start executing immediately (in the case of the second argument of sequential composition this is due to the presence of ε). Since all arguments of function symbols are $\aleph$-liquid, condition 3 of Def. 14 is satisfied trivially. Finally, condition 4 of Def. 14 needs to be checked for the two rules for sequential composition, as the first argument of this operator is $\aleph \cap \Lambda$-liquid. In both rules, x_1 has only one $\aleph$-liquid occurrence in the premises, and this occurrence is in a positive premise. In the second rule for sequential composition, the label of this premise is $\surd \neq \tau$. In the first rule, if $\ell = \tau$, then this is the patience rule for the first argument of sequential composition. Concluding, the TSS for $\mathrm{BPA}_{\varepsilon\delta\tau}$ is in rooted branching bisimulation format.

The rules also satisfy the strengthened condition $1'$ of Def. 17. So the TSS for $\mathrm{BPA}_{\varepsilon\delta\tau}$ is in rooted η-bisimulation format.

Corollary 1. *$\leftrightarroweq_{rb}$ and $\leftrightarroweq_{r\eta}$ are congruences for $BPA_{\varepsilon\delta\tau}$.*

6.2 Recursion

Given a signature Σ, a recursive specification E is a finite set of equations $\{X_i = t_i \mid i = 1, \ldots, n\}$, where the X_i are recursion variables, and the t_i are open terms over Σ, with possible occurrences of recursion variables. Intuitively, the syntactic construct $\langle X|E\rangle$ denotes a solution of X with respect to E. The precise meaning of this construct is given by the transition rules for recursion below, which originate from [16]. The expression E in these rules represents a recursive specification, which contains an equation $X = t$. Furthermore, $\langle t|E\rangle$ denotes the term t with occurrences of recursion variables Y replaced by $\langle Y|E\rangle$. We consider the expressions $\langle X|E\rangle$ as constants.

$$\frac{\langle t|E\rangle \xrightarrow{\alpha} y}{\langle X|E\rangle \xrightarrow{\alpha} y}$$

The resulting TSS is in rooted η-bisimulation format.

Corollary 2. *$\underline{\leftrightarrow}_{rb}$ and $\underline{\leftrightarrow}_{r\eta}$ are congruences for BPA$_{\varepsilon\delta\tau}$ extended with recursion constants $\langle X|E\rangle$.*

6.3 Replication

Let $\gamma : Act \times Act \rightharpoonup Act$ be a partial *communication function*, normally required to be commutative and associative. Then the ACP parallel composition [4] is given by the following rules, where ℓ ranges over $Act \cup \{\tau\}$, and a, b over Act.

$$\frac{x_1 \xrightarrow{\ell} y}{x_1\|x_2 \xrightarrow{\ell} y\|x_2} \qquad \frac{x_2 \xrightarrow{\ell} y}{x_1\|x_2 \xrightarrow{\ell} x_1\|y}$$

$$\frac{x_1 \xrightarrow{a} y_1 \quad x_2 \xrightarrow{b} y_2}{x_1\|x_2 \xrightarrow{\gamma(a,b)} y_1\|y_2}\ (\gamma(a,b)\text{ defined}) \qquad \frac{x_1 \xrightarrow{\surd} y_1 \quad x_2 \xrightarrow{\surd} y_2}{x_1\|x_2 \xrightarrow{\surd} y_1\|y_2}$$

Here $\gamma(a,b)$ is defined if the actions a and b can communicate, say because one of them is a send action and the other a corresponding receive action. In that case $\gamma(a,b) \in Act$ is the action that results from the synchronisation of a and b.

The arguments of $\|$ need to be chosen Λ-liquid as well as $\aleph$-liquid. The TSS for BPA$_{\varepsilon\delta\tau}$ augmented with parallel composition is in rooted η-bisimulation format, so $\underline{\leftrightarrow}_{rb}$ and $\underline{\leftrightarrow}_{r\eta}$ are congruences for this TSS. If we leave out the operators for alternative and sequential composition, the resulting TSS is even in η-bisimulation format, and also $\underline{\leftrightarrow}_{b}$ and $\underline{\leftrightarrow}_{\eta}$ become congruences.

The following unary *replication* operator ! stems from the π-calculus [29]. Intuitively it can be regarded as a parallel composition of infinitely many copies of its argument process.

$$\frac{x\|!x \xrightarrow{\ell} y}{!x \xrightarrow{\ell} y}$$

The argument of ! is Λ-frozen and $\aleph$-liquid, and the rule is in rooted η-bisimulation format.

Corollary 3. *$\leftrightarroweq_{rb}$ and $\leftrightarroweq_{r\eta}$ are congruences for BPA$_{\varepsilon\delta\tau}$ with parallel composition and replication.*

We have included this example because it requires a non-variable term in the premise of its rule. It is tempting to rewrite the rule above as

$$\frac{x \xrightarrow{\ell} y}{!x \xrightarrow{\ell} y \| !x}$$

but this would not capture initial communications between two copies of the replicated process.

6.4 Binary Kleene star

The *binary Kleene star* $t_1{}^*t_2$ [25] 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 BPA$_{\varepsilon\delta\tau}$.

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

We take the arguments of the binary Kleene star to be Λ-frozen (they do not contain running processes) and $\aleph$-liquid (they can start executing immediately). The resulting TSS is in rooted η-bisimulation format.

Corollary 4. *$\leftrightarroweq_{rb}$ and $\leftrightarroweq_{r\eta}$ are congruences for BPA$_{\varepsilon\delta\tau}$ with the binary Kleene star.*

6.5 Initial priority

Initial priority is a unary function that assumes an ordering on atomic actions. The term $\theta(t)$ executes the transitions of t, with the restriction 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 $\ell < \ell'$. This intuition is captured by the first rule for the initial priority operator below, which is added to the rules for BPA$_{\varepsilon\delta\tau}$.

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

We take the argument of initial priority to be Λ-frozen (it does not contain running processes) and $\aleph$-liquid (it can start executing immediately). The resulting TSS is in rooted η-bisimulation format.

Corollary 5. *$\leftrightarroweq_{rb}$ and $\leftrightarroweq_{r\eta}$ are congruences for BPA$_{\varepsilon\delta\tau}$ with initial priority.*

Initial priority is derived from the priority operator Θ [1]: in $\Theta(t)$ all transitions of t (so not only the initial ones) are blocked by simultaneous transitions of t with a greater label.

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

Consider the first rule above. In view of the target $\Theta(y)$, by condition 1 of Def. 14, the argument of Θ must be chosen Λ-liquid. And in view of condition 3 of Def. 14, the argument of Θ must be chosen $\aleph$-liquid. The $\aleph\cap\Lambda$-liquid argument x occurs $\aleph$-liquid in the negative premise, which violates condition 4 of Def. 14. In general, the priority operator Θ does not preserve rooted branching bisimilarity (cf. [32, pp. 130–132]).

6.6 Action refinement

The binary *action refinement* operator $t_1[b \rightsquigarrow t_2]$, for some $b \in Act$, replaces each b-transition in t_1 by the behaviour of t_2.

$$\frac{x_1 \xrightarrow{\alpha} y}{x_1[b \rightsquigarrow x_2] \xrightarrow{\alpha} y[b \rightsquigarrow x_2]}\ (\alpha \neq b) \qquad\qquad \frac{x_1 \xrightarrow{b} y_1 \qquad x_2 \xrightarrow{\ell} y_2}{x_1[b \rightsquigarrow x_2] \xrightarrow{\ell} y_2{\cdot}(y_1[b \rightsquigarrow x_2])}$$

Note that an initial successful termination of t_2 is ignored, as else we would get a transition rule with lookahead, which would violate the rooted branching bisimulation format.

For the second rule of action refinement to be rooted branching bisimulation safe, it is essential that the second argument of sequential composition is Λ-liquid, for else it would violate condition 1 of Def. 14. But then, since in $\mathrm{BPA}_{\varepsilon\delta\tau}$ this argument is also $\aleph$-liquid, the rule $\frac{x_1 \xrightarrow{\surd} y_1 \quad x_2 \xrightarrow{\tau} y_2}{x_1 \cdot x_2 \xrightarrow{\tau} y_2}$ violates condition 4 of Def. 14. Namely, the occurrence of x_2 in the source is at an $\aleph\cap\Lambda$-position, and x_2 occurs at an $\aleph$-liquid position in a positive premise with the label τ, but this is not an $\aleph\cap\Lambda$-patience rule. Thus the TSS for $\mathrm{BPA}_{\varepsilon\delta\tau}$ with the rules for action refinement above is not in rooted branching bisimulation format. And it should not be, as $\leftrightarrow_{rb}$ is not a congruence for this process algebra, due to the presence of the empty process. For example, $b{\cdot}(\tau{\cdot}c) \leftrightarrow_{rb} b{\cdot}c$, but the terms $b{\cdot}(\tau{\cdot}c)[b \rightsquigarrow b{\cdot}(d{\cdot}\delta + \varepsilon)] \leftrightarrow_{rb} b{\cdot}(d{\cdot}\delta + \tau{\cdot}c)$ and $b{\cdot}c[b \rightsquigarrow b{\cdot}(d{\cdot}\delta + \varepsilon)] \leftrightarrow_{rb} b{\cdot}(d{\cdot}\delta + c)$ are not rooted branching bisimilar.

Let us now consider the action refinement operator in the context of $\mathrm{BPA}_{\delta\tau}$, so without the empty process. Since ε is no longer present, we adapt the TSS for this basic process algebra as follows, where a ranges over Act. Whereas in $\mathrm{BPA}_{\varepsilon\delta\tau}$ a transition $p \xrightarrow{\surd} q$ can be thought of as statement that immediate termination is possible in state p, here it indicates an internal action resulting in successful termination.

$$a \xrightarrow{a} \tau \qquad \tau \xrightarrow{\surd} \delta \qquad \frac{x_1 \xrightarrow{\alpha} y}{x_1 + x_2 \xrightarrow{\alpha} y} \qquad \frac{x_2 \xrightarrow{\alpha} y}{x_1 + x_2 \xrightarrow{\alpha} y}$$

$$\frac{x_1 \xrightarrow{\ell} y}{x_1 \cdot x_2 \xrightarrow{\ell} y \cdot x_2} \qquad \frac{x_1 \xrightarrow{\surd} y}{x_1 \cdot x_2 \xrightarrow{\tau} x_2}$$

The TSS for BPA$_{\delta\tau}$ with action refinement is in rooted branching bisimulation format, if we mark the arguments of the BPA operators as before, except that the second argument of sequential composition is Λ-liquid, in view of the second rule for action refinement. Moreover, the first argument of action refinement is made Λ-liquid and $\aleph$-liquid, while the second argument of action refinement is made Λ-frozen and $\aleph$-liquid.

Corollary 6. *$\leftrightarrow_{rb}$ is a congruence for BPA$_{\delta\tau}$ with the action refinement operator.*

In the congruence format for rooted η-bisimilarity, condition 1 of Def. 14 is strengthened: right-hand sides of premises can only occur $\aleph\cap\Lambda$-liquid in the target. That this strengthening is essential is illustrated by the fact that BPA$_{\delta\tau}$ with action refinement fails to be compositional for rooted η-bisimilarity. For example, the law $a\cdot(\tau\cdot x + y) = a\cdot(\tau\cdot x + y) + a\cdot x$ is sound modulo rooted η-bisimilarity, while after refining a to $b\cdot c$, the resulting law $(b\cdot c)\cdot(\tau\cdot x + y) = (b\cdot c)(\tau\cdot x+y)+(b\cdot c)\cdot x$ is *not* sound modulo rooted η-bisimilarity; see [20, Sect. 7]. Note that the second rule for action refinement above violates condition $1'$ of Def. 17, because the occurrence of y in the target is at an $\aleph$-frozen position. And if we try to resolve this by making the second argument of sequential composition $\aleph$-liquid, then the TSS is not $\aleph\cap\Lambda$-patient (and thus not in rooted branching bisimulation format), due to the absence of a patience rule for the $\aleph\cap\Lambda$-liquid second argument of sequential composition.

6.7 Λ-liquid but $\aleph$-frozen

In most of the above applications, it suffices to take $\Lambda \subseteq \aleph$. The only exception so far is action refinement, but this operator does not fall in the rooted η-bisimulation format. The TSS of the following example does fall in the η-bisimulation format (Λ is universal); yet it is not possible to take $\Lambda \subseteq \aleph$.

Example 3. Let f be a binary operator that interleaves actions $\ell \in A \cup \{\tau\}$ from its arguments, until its first argument produces an action *crash*. Then f performs the actions *alert* and *prevent meltdown*, without any τ-steps in between, and subsequently continues as its second argument.

$$\frac{x \xrightarrow{\ell} y}{f(x_1,x_2) \xrightarrow{\ell} f(y,x_2)} \qquad \frac{x_2 \xrightarrow{\ell} y}{f(x_1,x_2) \xrightarrow{\ell} f(x_1,y)} \qquad \frac{x \xrightarrow{crash} x'}{f(x,y) \xrightarrow{alert} pm.y} \qquad pm.y \xrightarrow{\substack{prevent\\ meltdown}} y$$

Here pm is a CCS action-prefixing operator (cf, Example 2). For this TSS to be in (rooted) η-bisimulation format, it is essential that the argument of pm is marked as $\aleph$-frozen (and hence not accompanied by a patience rule) but Λ-liquid, for it harbours a process that has already started but is not currently running.

6.8 Counterexamples

We now present a series of counterexamples of complete TSSs in ntyft/ntyxt format, to show that none of the syntactic restrictions of our congruence formats can be omitted. (Of course it remains possible that certain restrictions can be refined.) In [22] a series of counterexamples can be found showing that the syntactic restrictions of the ntyft/ntyxt format are essential as well. Furthermore, in [8] a counterexample is given to show that completeness (there called positive after reduction) is essential.

It is well-known that branching and η-bisimilarity are not a congruence for $\mathrm{BPA}_{\varepsilon\delta\tau}$. For instance, $a \leftrightarrow_b \tau{\cdot}a$, but $a + b \not\leftrightarrow_\eta \tau{\cdot}a + b$. The TSS for $\mathrm{BPA}_{\varepsilon\delta\tau}$ (see Sect. 6.1) is in rooted η-bisimulation format, whereby it is essential that the arguments of alternative composition and the second argument of sequential composition are made Λ-frozen. This shows that universality of the predicate Λ cannot be omitted from the branching and η-bisimulation format.

The following counterexamples will focus on the rooted branching bisimulation format. They feature terms t_1, t_2 and a unary function symbol f with $t_1 \leftrightarrow_{rb} t_2$ and $f(t_1) \not\leftrightarrow_\eta f(t_2)$. This shows that none of the four semantics of this paper is a congruence, since $\leftrightarrow_{rb}$ and $\leftrightarrow_\eta$ are the strictest and most relaxed semantics, respectively, in this paper. The need for the strengthened restriction $1'$ of Def. 17 (compared to condition 1 of Def. 14) was already shown at the end of Sect. 6.6.

The examples in this section assume the TSS for $\mathrm{BPA}_{\varepsilon\delta\tau}$ with $Act = \{a, b, c\}$. The arguments of alternative composition and the second argument of sequential composition are Λ-frozen and $\aleph$-liquid, while the first argument of sequential composition is Λ- and $\aleph$-liquid.

Example 4. We extend $\mathrm{BPA}_{\varepsilon\delta\tau}$ with the following rule:

$$\frac{x \xrightarrow{a} y \quad y \xrightarrow{a} z}{f(x) \xrightarrow{c} \delta}$$

Clearly, $a{\cdot}a \leftrightarrow_{rb} a{\cdot}(\tau{\cdot}a)$. However, $f(a{\cdot}a) \not\leftrightarrow_\eta f(a{\cdot}(\tau{\cdot}a))$, since $f(a{\cdot}a) \xrightarrow{c} \delta$, while $f(a{\cdot}(\tau{\cdot}a))$ cannot perform any transition.

We make the argument of f Λ-frozen and $\aleph$-liquid. The rule above is not in rooted branching bisimulation format because it contains lookahead.

Example 5. We extend $\mathrm{BPA}_{\varepsilon\delta\tau}$ with the following rule:

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

$f(a{\cdot}a) \not\leftrightarrow_\eta f(a{\cdot}(\tau{\cdot}a))$, since $f(a{\cdot}a) \xrightarrow{a} f(\varepsilon{\cdot}a) \xrightarrow{a} f(\varepsilon)$, while $f(a{\cdot}(\tau{\cdot}a))$ can do an a-transition only to $f(\varepsilon{\cdot}(\tau{\cdot}a))$, which cannot perform any transition.

If the argument of f is Λ-frozen, then y occurs as the right-hand side of a premise and Λ-frozen in the target, violating condition 1 of Def. 14. And if the argument of f is $\aleph$-frozen, then x occurs $\aleph$-frozen in the source and $\aleph$-liquid in the premise, violating condition 3 of Def. 14. Finally, if the argument of f is $\aleph\cap\Lambda$-liquid, then it violates the restriction in Def. 15 that there should be an $\aleph\cap\Lambda$-patience rule for this argument.

Example 6. We extend BPA$_{\varepsilon\delta\tau}$ with the following rules:

$$\frac{x \xrightarrow{\tau} y}{f(x) \xrightarrow{\tau} f(y)} \qquad \frac{x \xrightarrow{a} y}{f(x) \xrightarrow{a} f(y)} \qquad \frac{x \stackrel{a}{\not\longrightarrow}}{f(x) \xrightarrow{c} \delta}$$

$f(a{\cdot}a) \not\leftrightarrow_\eta f(a{\cdot}(\tau{\cdot}a))$, since $f(a{\cdot}(\tau{\cdot}a)) \xrightarrow{a} f(\varepsilon{\cdot}(\tau{\cdot}a)) \xrightarrow{c} \delta$, while $f(a{\cdot}a)$ can do an a-transition only to $f(\varepsilon{\cdot}a)$, and $f(\varepsilon{\cdot}a) \stackrel{\alpha}{\not\longrightarrow}$ for $\alpha \in \{c, \tau\}$.

As in the previous example, it can be argued that the argument of f must be $\aleph\cap\Lambda$-liquid. Note that this time there is an $\aleph\cap\Lambda$-patience rule for the argument of f. However, the third rule is not rooted branching bisimulation safe, because the occurrence of x in the source is $\aleph\cap\Lambda$-liquid, and x occurs $\aleph$-liquid in the negative premise, violating condition 4 of Def. 14.

Example 7. We extend BPA$_{\varepsilon\delta\tau}$ with the following rules:

$$\frac{x \xrightarrow{\tau} y}{f(x) \xrightarrow{\tau} f(y)} \qquad \frac{x \xrightarrow{a} y}{f(x) \xrightarrow{a} f(y)} \qquad \frac{x \xrightarrow{\tau} y}{f(x) \xrightarrow{c} \delta}$$

$f(a{\cdot}a) \not\leftrightarrow_\eta f(a{\cdot}(\tau{\cdot}a))$, since $f(a{\cdot}(\tau{\cdot}a)) \xrightarrow{a} f(\varepsilon{\cdot}(\tau{\cdot}a)) \xrightarrow{c} \delta$, while $f(a{\cdot}a)$ can do an a-transition only to $f(\varepsilon{\cdot}a)$, and $f(\varepsilon{\cdot}a) \stackrel{\alpha}{\not\longrightarrow}$ for $\alpha \in \{c, \tau\}$.

As in the previous examples, the argument of f must be $\aleph\cap\Lambda$-liquid. However, then the third rule is not rooted branching bisimulation safe, because the occurrence of x in the source is $\aleph\cap\Lambda$-liquid, and x occurs $\aleph$-liquid in the positive premise with the label τ, while this rule is not $\aleph\cap\Lambda$-patient, violating condition 4 of Def. 14.

Example 8. We extend BPA$_{\varepsilon\delta\tau}$ with the following rules:

$$\kappa \xrightarrow{\tau} \zeta \qquad \zeta \xrightarrow{\tau} \kappa \qquad \kappa \xrightarrow{a} \delta \qquad \zeta \xrightarrow{b} \delta \qquad \nu \xrightarrow{a} \delta \qquad \nu \xrightarrow{b} \delta$$

$$\frac{x \xrightarrow{\tau} y}{f(x) \xrightarrow{\tau} f(y)} \qquad f(x) \xrightarrow{\tau} g(x) \qquad \frac{x \xrightarrow{a} y \quad x \xrightarrow{b} z}{g(x) \xrightarrow{c} \delta}$$

Clearly, $\tau{\cdot}\nu \leftrightarrow_{rb} \tau{\cdot}\kappa$. However, $f(\tau{\cdot}\nu) \not\leftrightarrow_\eta f(\tau{\cdot}\kappa)$, since $f(\tau{\cdot}\nu) \xrightarrow{\tau} f(\varepsilon{\cdot}\nu) \xrightarrow{\tau} g(\varepsilon{\cdot}\nu) \xrightarrow{c} \delta$, while $f(\tau{\cdot}\kappa)$ only exhibits an infinite sequence of τ-transitions: $f(\tau{\cdot}\kappa) \xrightarrow{\tau} f(\varepsilon{\cdot}\kappa) \xrightarrow{\tau} f(\zeta) \xrightarrow{\tau} f(\kappa) \xrightarrow{\tau} \cdots$, with side transitions $f(\tau{\cdot}\kappa) \xrightarrow{\tau} g(\tau{\cdot}\kappa)$, $f(\zeta) \xrightarrow{\tau} g(\zeta)$, etc., to states from which no further actions are possible.

Again, the argument of f must be $\aleph\cap\Lambda$-liquid. Considering the last rule, the argument of g must be $\aleph$-liquid, in view of condition 3 of Def. 14. If the argument of g is Λ-liquid, then in the last rule the occurrence of x in the source is $\aleph\cap\Lambda$-liquid, and x has two $\aleph$-liquid occurrences in the premises, violating condition 4 of Def. 14. And if the argument of g is Λ-frozen, then in the last but one rule, x occurs Λ-liquid in the source and Λ-frozen in the target, violating condition 2 of Def. 14.

Example 9. In the TSS from Ex. 8, we replace the last two rules with the following rules:

$$\frac{x \xrightarrow{a} y \quad x \xrightarrow{b} z}{g(x) \xrightarrow{c} \delta} \qquad \frac{g(x) \xrightarrow{c} y}{f(x) \xrightarrow{c} \delta}$$

$f(\tau{\cdot}\nu) \not\leftrightarrow_\eta f(\tau{\cdot}\kappa)$, since $f(\tau{\cdot}\nu) \xrightarrow{\tau} f(\varepsilon{\cdot}\nu) \xrightarrow{c} \delta$, while $f(\tau{\cdot}\kappa)$ can only perform τ-transitions.

Again, the argument of f must be $\aleph\cap\Lambda$-liquid. Considering the first rule, the argument of g must be $\aleph$-liquid, in view of condition 3 of Def. 14, and Λ-frozen, in view of condition 4 of Def. 14. However, then the second rule is not rooted branching bisimulation safe, because x occurs Λ-liquid in the source and Λ-frozen in the left-hand side of the premise, violating condition 2 of Def. 14.

7 Abbreviation Expansion

Inspired by the congruence formats of [5], in [19] a two-tiered approach to structural operational semantics was introduced. It divides function symbols into two classes: principal operators and abbreviations. An abbreviation can be obtained by grouping together (and permuting) the arguments of a principal operator. For example, $f(x, y)$ could be an abbreviation of $g(x, y, x)$, for a ternary principal operator g. In the two-tiered approach the abbreviations do not have to obey the syntactic restrictions of a congruence format, as long as they abbreviate principal operators that do. Here we generalise this two-tiered approach from TSSs in GSOS format [7] to standard TSSs in decent ntyft format (using a different presentation than in [19]). This generalisation will help us to give a fair comparison between the congruence formats in [5,19] and the ones in the current paper; see Sect. 8.

The following *copying operator* occurs in [7, page 257]; there it plays an essential role in analysing the expressive power of the ready simulation format for processes without internal actions.

$$\frac{x \xrightarrow{a} y}{\mathrm{cp}(x) \xrightarrow{a} \mathrm{cp}(y)} \qquad \frac{x \xrightarrow{l} y_1 \quad x \xrightarrow{r} y_2}{\mathrm{cp}(x) \xrightarrow{s} \mathrm{cp}(y_1)\|\mathrm{cp}(y_2)}$$

The underlying alphabet of actions is $\{b, c, d, l, r, s\}$, with a ranging over the *normal* actions b, c, d, and l, r, s being the left and right *forking* actions and *split* action. The copying operator passes normal actions through, but when its argument process offers a choice between actions l and r, the copying process chooses *both* branches, thereby turning into a parallel composition of two copies of itself.

One way of extending this idea with the internal action τ is to add a patience rule for the argument of cp and replace the second rule by something like

$$\frac{x \Longrightarrow\xrightarrow{l} y_1 \quad x \Longrightarrow\xrightarrow{r} y_2}{\mathrm{cp}(x) \Longrightarrow\xrightarrow{s} \mathrm{cp}(y_1)\|\mathrm{cp}(y_2)}$$

where $\Longrightarrow$ stands for a sequence of internal transitions. This way we have

$$\mathrm{cp}(\tau{\cdot}l{\cdot}p + \tau{\cdot}r{\cdot}q) \Longrightarrow\xrightarrow{s} \mathrm{cp}(p)\|\mathrm{cp}(q)\ .$$

We also have $\mathrm{cp}(\tau{\cdot}l{\cdot}p + \tau{\cdot}r{\cdot}q) \xrightarrow{\tau} \mathrm{cp}(l.p)$, where the target process deadlocks.

Of course, the suggested TSS does not fit in the ready simulation format, due to the presence of lookahead, but the following TSS in ready simulation format implements the same idea.

$$\frac{x \xrightarrow{a} y}{\mathrm{cp}(x) \xrightarrow{a} \mathrm{cp}(y)} \qquad \frac{x \xrightarrow{l} y_1 \quad x \xrightarrow{r} y_2}{\mathrm{cp}(x) \xrightarrow{s} \mathrm{cp}(y_1)\|\mathrm{cp}(y_2)}$$

$$\frac{x \xrightarrow{\tau} y}{\mathrm{cp}(x) \xrightarrow{\tau} \mathrm{bn}(y,x)} \qquad \frac{x \xrightarrow{\tau} y}{\mathrm{cp}(x) \xrightarrow{\tau} \mathrm{bn}(x,y)}$$

$$\frac{x_1 \xrightarrow{\tau} y}{\mathrm{bn}(x_1,x_2) \xrightarrow{\tau} \mathrm{bn}(y,x_2)} \qquad \frac{x_2 \xrightarrow{\tau} y}{\mathrm{bn}(x_1,x_2) \xrightarrow{\tau} \mathrm{bn}(x_1,y)}$$

$$\frac{x_1 \xrightarrow{a} y}{\mathrm{bn}(x_1,x_2) \xrightarrow{a} \mathrm{cp}(y)} \qquad \frac{x_2 \xrightarrow{a} y}{\mathrm{bn}(x_1,x_2) \xrightarrow{a} \mathrm{cp}(y)}$$

$$\frac{x_1 \xrightarrow{l} y_1 \quad x_2 \xrightarrow{r} y_2}{\mathrm{bn}(x_1,x_2) \xrightarrow{s} \mathrm{cp}(y_1)\|\mathrm{cp}(y_2)}$$

Here the auxiliary operator bn is a binary version of the copying operator, that can probe two branches of τ-actions at the same time. If these two branches offer a pair of forking actions, the desired split occurs, whereas a normal action encountered on any of the branches collapses bn back to its unary counterpart.

In view of the first, fifth and sixth rule, the arguments of both cp and bn need to be $\aleph{\cap}\Lambda$-liquid. Therefore the second, third and fourth rule violate condition 4 of the rooted branching bisimulation format; moreover, the patience rule for $\mathrm{cp}(x)$ is missing. In spite of this, $\leftrightarrow_b$, $\leftrightarrow_{rb}$, $\leftrightarrow_\eta$ and $\leftrightarrow_{r\eta}$ are congruences for cp. This can be seen by thinking of $\mathrm{cp}(x)$ as an abbreviation for $\mathrm{bn}(x,x)$. This way, all rules for cp can be construed as special cases of the rules for bn, and the resulting five rules for bn fit the η-bisimulation format.

In general, we will translate one TSS P into another TSS P^*, by finding a principal operator (like bn) for each abbreviation (like cp). From the fact that a weak semantics $\sim$ is a congruence for P^*, we will then conclude that $\sim$ is a congruence for P.

Definition 19. A *translation* from a TSS $P=(\Sigma,R)$ to a TSS $P^*=(\Sigma^*,R^*)$ is a function $^*:\Sigma\to\Sigma^*$, together with, for each $f\in\Sigma$, an *argument matching* $f:\{1,\ldots,ar(f^*)\}\to\{1,\ldots,ar(f)\}$.

Here we overload the symbol f; it is always clear from the context whether we mean the function symbol in Σ or its corresponding argument matching. Translations naturally extend to functions $^*:\mathbb{T}(\Sigma)\to\mathbb{T}(\Sigma^*)$ by $x^*=x$ for $x\in V$ and $f(t_1,\ldots,t_{ar(f)})^* = f^*(t^*_{f(1)},\ldots,t^*_{f(ar(f^*))})$. They further extend trivially to (sets of) literals. Moreover, they extend to substitutions $\sigma:V\to\mathbb{T}(\Sigma)$ by $\sigma^*(x):=\sigma(x)^*$, so that $\sigma(t)^*=\sigma^*(t^*)$ for any term t.

A translation from P to P^* is lifted to a conversion of rules of P to rules of P^* in two steps. First the sources are “starred”, producing an intermediate TSS P'. Next the premises are starred, yielding P^*. There is a degree of freedom in converting P to P'; however, both P and P^* are completely determined by P'.

Definition 20. A standard TSS $P=(\Sigma,R)$ in decent ntyft format is an *alternative representation* of a standard TSS $P^*=(\Sigma^*,R^*)$ in decent ntyft format if there exists a translation $^*:\Sigma\to\Sigma^*$, and an intermediate standard TSS $P'=(\Sigma\cup\Sigma^*,R')$ in decent ntyft format, whose rules have sources in $\mathbb{T}(\Sigma^*)$ but premises and targets in $\mathbb{T}(\Sigma)$, such that

$$R^*=\left\{\frac{H^*}{t\xrightarrow{\alpha}u^*}\;\middle|\;\frac{H}{t\xrightarrow{\alpha}u}\in R'\right\}$$

and

$$R=\left\{\frac{\sigma_f(H)}{f(x_1,\ldots,x_{ar(f)})\xrightarrow{\alpha}\sigma_f(u)}\;\middle|\;\frac{H}{f^*(z_1,\ldots,z_{ar(f^*)})\xrightarrow{\alpha}u}\in R'\right\}$$

where the substitution $\sigma_f:\{z_1,\ldots,z_{ar(f^*)}\}\to\{x_1,\ldots,x_{ar(f)}\}$ is given by $\sigma_f(z_i)=x_{f(i)}$. (By α-conversion we can ensure that all sources $f(\ldots)$ in R have arguments $x_1,\ldots,x_{ar(f)}$, and all sources $f^*(\ldots)$ in R^* have arguments $z_1,\ldots,z_{ar(f^*)}$.)

In our running example, the translation is given by cp*=bn with cp(1)=cp(2)=1 and bn*=bn with bn(1)=1 and bn(2)=2. The intermediate TSS consists of just the five rules for bn in the original TSS. Note that the first rule of P is linked to the third (and fourth) rule of P', the second rule of P is linked to the fifth rule of P', and the third and fourth rule of P are linked to the first and second rule of P'. The TSS P^* is obtained by replacing terms cp(z) in targets in P' by bn(z,z).

$$\frac{x_1\xrightarrow{\tau}y}{\mathrm{bn}(x_1,x_2)\xrightarrow{\tau}\mathrm{bn}(y,x_2)}\qquad\frac{x_2\xrightarrow{\tau}y}{\mathrm{bn}(x_1,x_2)\xrightarrow{\tau}\mathrm{bn}(x_1,y)}$$

$$\frac{x_1\xrightarrow{a}y}{\mathrm{bn}(x_1,x_2)\xrightarrow{a}\mathrm{bn}(y,y)}\qquad\frac{x_2\xrightarrow{a}y}{\mathrm{bn}(x_1,x_2)\xrightarrow{a}\mathrm{bn}(y,y)}$$

$$\frac{x_1\xrightarrow{l}y_1\quad x_2\xrightarrow{r}y_2}{\mathrm{bn}(x_1,x_2)\xrightarrow{s}\mathrm{bn}(y_1,y_1)\|\mathrm{bn}(y_2,y_2)}$$

This TSS is in η-bisimulation format, with both arguments of bn $\aleph\cap\Lambda$-liquid. Thus $\leftrightarrow_b$, $\leftrightarrow_{rb}$, $\leftrightarrow_\eta$ and $\leftrightarrow_{r\eta}$ are congruences for P^*.

In all applications of this approach we have encountered so far, including the example above, we have (possibly up to renaming of function symbols) $\Sigma^* \subseteq \Sigma$ and $R' \subseteq R$. In [19] this was even required by definition. It is in this case that we call P a *two-tiered TSS*, with the operators in Σ^* *principal operators* and the ones in $\Sigma \setminus \Sigma^*$ *abbreviations.*

We will now prove that, if P is an alternative representation of P^*, then for any weak semantics $\sim$ including strong bisimilarity, any congruence result for P^* carries over to P.

Lemma 7. *Let $P = (\Sigma, R)$ be an alternative representation of $P^* = (\Sigma^*, R^*)$ and $p, q' \in \mathbb{T}(\Sigma)$ closed. Then $P^* \vdash_{ws} p^* \xrightarrow{\alpha} q'$ iff $\exists q : P \vdash_{ws} p \xrightarrow{\alpha} q \wedge q^* = q'$.*

Proof. "Only if": The proof of this direction will be delivered in three steps.

Claim 1: Let $\frac{K}{\mu}$ be a closed substitution instance of a rule of P. Then $\frac{K^*}{\mu^*}$ is a closed substitution instance of a rule of P^*.

Proof: Let $\frac{K}{p \xrightarrow{\alpha} q}$ be obtained by applying the closed substitution σ to the rule in R corresponding to $\frac{H}{f^*(z_1,\ldots,z_{ar(f^*)}) \xrightarrow{\alpha} u}$ in R'. Then $K = \sigma(\sigma_f(H))$, and thus $K^* = \sigma^*(\sigma_f^*(H^*)) = \sigma^*(\sigma_f(H^*))$ (because $\sigma_f^* = \sigma_f$). Likewise $q^* = \sigma^*(\sigma_f(u^*))$. Furthermore,

$$\begin{aligned} p^* &= \sigma(f(x_1, \ldots, x_{ar(f)}))^* \\ &= \sigma^*(f^*(x_{f(i)}, \ldots, x_{f(ar(f^*))})) \\ &= \sigma^*(\sigma_f(f^*(z_1, \ldots, z_{ar(f^*)}))) \,. \end{aligned}$$

Thus $\frac{K^*}{p^* \xrightarrow{\alpha} q^*}$ is a substitution instance of the rule $\frac{H^*}{f^*(z_1,\ldots,z_{ar(f^*)}) \xrightarrow{\alpha} u^*}$ in R^*, using the closed substitution $\sigma \circ \sigma_f$. ■

Claim 2: If a closed transition rule $\frac{N}{p^* \xrightarrow{\alpha} q'}$, where N contains only negative premises and $p \in \mathbb{T}(\Sigma)$, is irredundantly provable from P^*, then $N = M^*$ and $q' = q^*$, with $\frac{M}{p \xrightarrow{\alpha} q}$ irredundantly provable from P.

Proof: By induction on the structure of irredundant proofs. Let π be an irredundant proof of $\frac{N}{p^* \xrightarrow{\alpha} q'}$ from P^*, and let $r \in R^*$ be the rule and ρ the substitution used in the last step of π. Let $r = \frac{H^*}{t \xrightarrow{\alpha} u^*} \in R^*$ with $\frac{H}{t \xrightarrow{\alpha} u} \in R'$. Furthermore, let $p = f(p_1, \ldots, p_{ar(f)})$. Then $p^* = f^*(p^*_{f(1)}, \ldots, p_{f(ar(f^*))}) = \rho(t)$. Hence $t = f^*(z_1, \ldots, z_{ar(f^*)})$ and $\rho(z_i) = p^*_{f(i)}$ for $1 \leq i \leq ar(f^*)$. Let the closed substitution $\omega : \{z_1, \ldots, z_{ar(f^*)}\} \to \mathbb{T}(\Sigma^*)$ be given by $\omega(z_i) = p_{f(i)}$ for $1 \leq i \leq ar(f^*)$. Then $\rho(z_i) = \omega^*(z_i)$ for $1 \leq i \leq ar(f^*)$.

For each positive premise $v_y \xrightarrow{\alpha_y} y$ in H, a strict subproof of π irredundantly proves a rule $\frac{N_y}{\rho(v_y^*) \xrightarrow{\alpha_y} \rho(y)}$, with $N_y \subseteq N$. Since r is decent, $var(v_y^*) \subseteq var(t) = \{z_1, \ldots, z_{ar(f^*)}\}$. Hence $\rho(v_y^*) = \omega^*(v_y^*) = \omega(v_y)^*$. By induction we obtain $N_y = M_y^*$ and $\rho(y) = q_y^*$ with $\frac{M_y}{\omega(v_y) \xrightarrow{\alpha_y} q_y}$ irredundantly provable from P.

We have $r' = \frac{\sigma_f(H)}{f(x_1,\ldots,x_{ar(f)}) \xrightarrow{\alpha} \sigma_f(u)} \in R$ with $\sigma_f(z_i) = x_{f(i)}$ for $1 \leq i \leq ar(f^*)$. Let $\sigma : var(r') \to \mathbb{T}(\Sigma)$ be the closed substitution with $\sigma(x_i) = p_i$ for $1 \leq i \leq ar(f)$ and $\sigma(y) = q_y$ for $y \in rhs(\sigma_f(H)) = rhs(H)$. Then $\sigma \circ \sigma_f(z_i) = \sigma(x_{f(i)}) = p_{f(i)} = \omega(z_i)$ for $1 \leq i \leq ar(f^*)$. Therefore, for each $v_y \xrightarrow{\alpha_y} y \in H$ we have $\sigma \circ \sigma_f(v_y \xrightarrow{\alpha_y} y) = \omega(v_y) \xrightarrow{\alpha_y} q_y$, and for each $v \overset{\beta}{\not\longrightarrow} \in H$ we have $\sigma \circ \sigma_f(v \overset{\beta}{\not\longrightarrow}) = \omega(v) \overset{\beta}{\not\longrightarrow}$.

The rules $\frac{M_y}{\omega(v_y) \xrightarrow{\alpha_y} q_y}$ for $y \in rhs(H)$ are irredundantly provable from P, and applying σ to $r' \in R$ yields

$$\frac{\{\omega(v_y) \xrightarrow{\alpha_y} q_y \mid y \in rhs(H)\} \;\cup\; \{\omega(v) \overset{\beta}{\not\longrightarrow} \mid v \overset{\beta}{\not\longrightarrow} \in H\}}{f(p_1,\ldots,p_{ar(f)}) \xrightarrow{\alpha} \sigma(\sigma_f(u))} \,.$$

So $\frac{M}{p \xrightarrow{\alpha} \sigma(\sigma_f(u))}$ with $M := \bigcup_{y \in rhs(H)} M_y \cup \{\omega(v) \overset{\beta}{\not\longrightarrow} \mid v \overset{\beta}{\not\longrightarrow} \in H\}$ is irredundantly provable from P.

$$\begin{aligned} M^* &= \textstyle\bigcup_{y \in rhs(H)} M^*_y \;\cup\; \{\omega^*(v^*) \overset{\beta}{\not\longrightarrow} \mid v \overset{\beta}{\not\longrightarrow} \in H\} = \\ &= \textstyle\bigcup_{y \in rhs(H)} N_y \;\cup\; \{\rho(v^*) \overset{\beta}{\not\longrightarrow} \mid v \overset{\beta}{\not\longrightarrow} \in H\} \;=\; N \,. \end{aligned}$$

As $\frac{H}{t \xrightarrow{\alpha} u}$ is decent, u contains no other variables than z_i for $1 \leq i \leq ar(f^*)$ and y for $y \in rhs(H)$. We have $(\sigma \circ \sigma_f)^*(z_i) = \omega^*(z_i) = \rho(z_i)$ for $1 \leq i \leq ar(f^*)$ and $(\sigma \circ \sigma_f)^*(y) = \sigma(\sigma_f(y))^* = \sigma(y)^* = q^*_y = \rho(y)$ for $y \in rhs(H)$. Therefore $\sigma(\sigma_f(u))^* = \rho(u^*) = q'$. ∎

The following claim strengthens the direction "only if" of the lemma.

Claim 3: If $P \vdash_{ws} \lambda$ for a closed literal λ, then $P^* \vdash_{ws} \lambda^*$.

Proof: By induction on the structure of well-supported proofs. Let π be a well-supported proof from P of a closed positive literal $\lambda = p \xrightarrow{\alpha} q$. Then there is a closed substitution instance $\frac{K}{\lambda}$ of a rule in R, and each literal $\mu \in K$ is provable by a strict subproof of π. By induction, $P^* \vdash_{ws} \mu^*$ for each such μ. By Claim 1 $\frac{K^*}{\mu^*}$ is a closed substitution instance of a rule of P^*. It follows that $P^* \vdash_{ws} \lambda^*$.

Now let π be a well-supported proof from P of a closed negative literal $\lambda = p \overset{\alpha}{\not\longrightarrow}$. In order to show that $P^* \vdash_{ws} \lambda^*$, let N be a set of closed negative literals such that, for some closed term q', the rule $\frac{N}{p^* \xrightarrow{\alpha} q'}$ is irredundantly provable from P^*. It suffices to show that $P^* \vdash_{ws} \nu$, for a literal ν denying a literal in N.

By Claim 2, $\frac{N}{p^* \xrightarrow{\alpha} q'} = \frac{M^*}{p^* \xrightarrow{\alpha} q^*}$, for some rule $\frac{M}{p \xrightarrow{\alpha} q}$ irredundantly provable from P. Since $P \vdash_{ws} p \overset{\alpha}{\not\longrightarrow}$, by Def. 7, a strict subproof of π proves a literal μ denying a literal in M. By induction, $P^* \vdash_{ws} \mu^*$. Moreover, μ^* denies a literal in $M^* = N$. ∎

The last claim strengthens the "if"-direction of the lemma. Its proof proceeds along the same lines as for the claims above.

Claim 4: Let $p \in \mathbb{T}(\Sigma)$. If $P^* \vdash_{ws} p^* \xrightarrow{\alpha} q'$, then $P \vdash_{ws} p \xrightarrow{\alpha} q$ with $q^* = q'$.

Likewise, if $P^* \vdash_{ws} p^* \overset{\alpha}{\not\longrightarrow}$, then $P \vdash_{ws} p \overset{\alpha}{\not\longrightarrow}$.

Proof: By induction on the structure of irredundant proofs. Let π be a well-supported proof from P^* of $p^* \xrightarrow{\alpha} q'$ and let $r \in R^*$ be the rule and ρ the substitution used in the last step of π. Let $r = \frac{H^*}{t \xrightarrow{\alpha} u^*} \in R^*$ with $\frac{H}{t \xrightarrow{\alpha} u} \in R'$. Furthermore, let $p = f(p_1, \ldots, p_{ar(f)})$. Then $p^* = f^*(p^*_{f(1)}, \ldots, p_{f(ar(f^*))}) = \rho(t)$. Hence $t = f^*(z_1, \ldots, z_{ar(f^*)})$ and $\rho(z_i) = p^*_{f(i)}$ for $1 \leq i \leq ar(f^*)$. Let the closed substitution $\omega : \{z_1, \ldots, z_{ar(f^*)}\} \to \mathbb{T}(\Sigma^*)$ be given by $\omega(z_i) = p_{f(i)}$ for $1 \leq i \leq ar(f^*)$. Then $\rho(z_i) = \omega^*(z_i)$ for $1 \leq i \leq ar(f^*)$.

For each positive premise $v_y \xrightarrow{\alpha_y} y$ in H, a strict subproof of π proves $\rho(v^*_y) \xrightarrow{\alpha_y} \rho(y)$. Since r is decent, $var(v^*_y) \subseteq var(t) = \{z_1, \ldots, z_{ar(f^*)}\}$. Hence $\rho(v^*_y) = \omega^*(v^*_y) = \omega(v_y)^*$. By induction $P \vdash_{ws} \omega(v_y) \xrightarrow{\alpha_y} q_y$ for some q_y with $q^*_y = \rho(y)$. Likewise, for each negative premise $v \overset{\beta}{\not\longrightarrow}$ in H, a strict subproof of π proves $\rho(v^*) \overset{\beta}{\not\longrightarrow}$. Again $\rho(v^*) = \omega^*(v^*) = \omega(v)^*$. By induction $P \vdash_{ws} \omega(v) \overset{\beta}{\not\longrightarrow}$.

We have $r' = \frac{\sigma_f(H)}{f(x_1,\ldots,x_{ar(f)}) \xrightarrow{\alpha} \sigma_f(u)} \in R$ with $\sigma_f(z_i) = x_{f(i)}$ for $1 \leq i \leq ar(f^*)$. Let $\sigma : var(r') \to \mathbb{T}(\Sigma)$ be the closed substitution with $\sigma(x_i) = p_i$ for $1 \leq i \leq ar(f)$ and $\sigma(y) = q_y$ for $y \in rhs(\sigma_f(H)) = rhs(H)$. Then $\sigma \circ \sigma_f(z_i) = \sigma(x_{f(i)}) = p_{f(i)} = \omega(z_i)$ for $1 \leq i \leq ar(f^*)$. Therefore, for each $v_y \xrightarrow{\alpha_y} y \in H$ we have $\sigma \circ \sigma_f(v_y \xrightarrow{\alpha_y} y) = \omega(v_y) \xrightarrow{\alpha_y} q_y$, and for each $v \overset{\beta}{\not\longrightarrow} \in H$ we have $\sigma \circ \sigma_f(v \overset{\beta}{\not\longrightarrow}) = \omega(v) \overset{\beta}{\not\longrightarrow}$.

Applying σ to $r' \in R$ yields

$$\frac{\{\omega(v_y) \xrightarrow{\alpha_y} q_y \mid y \in rhs(H)\} \ \cup \ \{\omega(v) \overset{\beta}{\not\longrightarrow} \mid v \overset{\beta}{\not\longrightarrow} \in H\}}{f(p_1, \ldots, p_{ar(f)}) \xrightarrow{\alpha} \sigma(\sigma_f(u))} .$$

So $P \vdash_{ws} p \xrightarrow{\alpha} \sigma(\sigma_f(u))$.

As $\frac{H}{t \xrightarrow{\alpha} u}$ is decent, u contains no other variables than z_i for $1 \leq i \leq ar(f^*)$ and y for $y \in rhs(H)$. We have $(\sigma \circ \sigma_f)^*(z_i) = \omega^*(z_i) = \rho(z_i)$ for $1 \leq i \leq ar(f^*)$ and $(\sigma \circ \sigma_f)^*(y) = \sigma(\sigma_f(y))^* = \sigma(y)^* = q^*_y = \rho(y)$ for $y \in rhs(H)$. Therefore $\sigma(\sigma_f(u))^* = \rho(u^*) = q'$.

Now let π be a well-supported proof from P^* of $p^* \overset{\alpha}{\not\longrightarrow}$. In order to show that $P \vdash_{ws} p \overset{\alpha}{\not\longrightarrow}$, let N be a set of closed negative literals such that, for some closed term q, the rule $\frac{N}{p \xrightarrow{\alpha} q}$ is irredundantly provable from P. It suffices to show that $P \vdash_{ws} \lambda$, for a literal λ denying a literal ν in N.

By Claim 1, and a trivial induction on the structure of irredundant proofs, the rule $\frac{N^*}{p^* \xrightarrow{\alpha} q^*}$ is irredundantly provable from P^*. Since $P^* \vdash_{ws} p^* \overset{\alpha}{\not\longrightarrow}$, by Def. 7, a strict subproof of π proves a literal μ denying a literal in N^*. The latter literal must have the form ν^* for some $\nu \in N$.

In case $\nu = (p_\nu \xrightarrow{\alpha_\nu} q_\nu)$, we have $\mu = (p^*_\nu \overset{\alpha_\nu}{\not\longrightarrow})$. By induction $P \vdash_{ws} p_\nu \overset{\alpha_\nu}{\not\longrightarrow}$, and this literal denies $\nu \in N$.

In case $\nu = (p_\nu \overset{\beta_\nu}{\not\longrightarrow})$, we have $\mu = (p^*_\nu \xrightarrow{\beta_\nu} q'_\nu)$ for some $p_\nu \in \mathbb{T}(\Sigma)$ and $q' \in \mathbb{T}(\Sigma^*)$. By induction $P \vdash_{ws} p_\nu \xrightarrow{\beta_\nu} q_\nu$ for some $q_\nu \in \mathbb{T}(\Sigma)$ with $q^*_\nu = q'_\nu$. Again this literal denies $\nu \in N$. ■ □

A symmetric relation $R \subseteq \mathbb{P} \times \mathbb{P}$ is a *strong bisimulation* if pRq and $p \xrightarrow{\alpha} p'$ implies that $q \xrightarrow{\alpha} q'$ for some q' with $p'Rq'$. Processes p, q are *strongly bisimilar*, denoted $p \leftrightarrow q$, if there exists a strong bisimulation R with pRq. Processes in different TSSs can be compared up to strong bisimilarity, namely by considering the disjoint union of the TSSs.

Corollary 7. *Let P be an alternative representation of P^*, and $\sim$ any equivalence relation on processes satisfying $p \leftrightarrow q \Rightarrow p \sim q$. Then $p^* \sim q^*$ iff $p \sim q$.*

Proof. We have $p \leftrightarrow p^*$ for all closed terms $p \in \mathbb{T}(\Sigma)$, because the relation $\{(p, p^*), (p^*, p) \mid p \in \mathbb{T}(\Sigma)\}$ is a strong bisimulation by Lemma 7. Hence if $p \sim q$ then $p^* \leftrightarrow p \sim q \leftrightarrow q^*$, implying $p^* \sim q^*$, and if $p^* \sim q^*$ then $p \leftrightarrow p^* \sim q^* \leftrightarrow q$, implying $p \sim q$. □

Corollary 8. *Let P be an alternative representation of P^* and $\sim$ be any equivalence relation on processes satisfying $p \leftrightarrow q \Rightarrow p \sim q$, such that $\sim$ is a congruence for P^*. Then $\sim$ is a congruence for P.*

Proof. Suppose $p_i \sim q_i$ for $i = 1, \ldots, ar(f)$. By Cor. 7, $p_i^* \sim q_i^*$ for $i = 1, \ldots, ar(f)$. By assumption, $f(p_1, \ldots, p_{ar(f)})^* \sim f(q_1, \ldots, q_{ar(f)})^*$. Thus, by Cor. 7, $f(p_1, \ldots, p_{ar(f)}) \sim f(q_1, \ldots, q_{ar(f)})$. □

As for our running example, it now follows that $\leftrightarrow_b$, $\leftrightarrow_{rb}$, $\leftrightarrow_\eta$ and $\leftrightarrow_{r\eta}$ are congruences for cp.

Definition 21. Let F be a congruence format on standard TSSs that lays within the decent ntyft format. A TSS is said to be in the *two-tiered F-format* [19], iff it is an alternative representation of a TSS in F-format.

Theorem 7. *Let $\sim$ be an equivalence relation on LTSs such that $p \leftrightarrow q \Rightarrow p \sim q$, and let F be a format on standard TSSs within the decent ntyft format so that for any TSS in F-format, $\sim$ is a congruence. Then $\sim$ is a congruence for any TSS in the two-tiered F-format.*

Proof. Apply Cor. 8. □

8 Related Work

The first congruence formats for branching and rooted branching bisimilarity were presented in [5], and reformulated in [19]. The latter paper also added the first formats for η- and rooted η-bisimilarity. Those four formats, which are contained in the GSOS format [7] (and thereby also in the intersection of the nxytt and decent ntyft formats), distinguish so-called "principal" function symbols and "abbreviations". The latter can be regarded as syntactic sugar, adding nothing that could not be expressed with principal function symbols. In [5,19] also simplified variants of these four formats were proposed, obtained by requiring all function symbols to be principal; as shown in [19], the general formats of [5,19] can be obtained as the two-tiered version of the corresponding simplified formats. Our main formats strictly generalise the simplified formats of [5,19]. Consequently, the two-tiered versions of our formats (or more precisely, of the intersection of our formats with the decent ntyft format) generalise the full formats of [5,19].

For the branching bisimulation format our generalisation consists of allowing transition rules outside the GSOS format; the simplified format of [5,19] is exactly the intersection of our branching bisimulation format and the GSOS format. (Our predicate $\aleph$ marks the arguments of function symbols that are called *active* in [19].) Likewise, the simplified η-bisimulation format of [19] is the intersection of our η-bisimulation format and the GSOS format. However, the intersections of our rooted formats and the GSOS format are still proper generalisations of the simplified rooted formats of [5,19]. The latter can be described as the intersections of our rooted formats and the GSOS format in which all arguments of all function symbols that occur in targets of rules are required to be Λ-liquid.

The applications of our formats presented in Sections 6.1 (sequential composition in basic process algebra), 6.4 (the binary Kleene star) and 6.6 (action refinement) fall inside the GSOS format but outside the formats of [5,19]. The replication operator of Section 6.3 is an example of an application of our formats that falls outside the GSOS format; recursion (Section 6.2) is another example. We have not repeated applications of our approach that fall already within the formats of [5,19]; these include the full process algebras CCS [28] and CSP [24].

The rooted formats of [5,19] distinguish "tame" and "wild" function symbols. In terms of our approach, wild function symbols have only Λ-frozen arguments, and tame function symbols only Λ-liquid arguments. The idea to allow function symbols with both kinds of arguments stems from [10], where a so-called RBB safe format for rooted branching bisimilarity was proposed, which generalises the simplified format of [5,19]. Given that it applies to TSSs with predicates, it is incomparable with our current rooted branching bisimulation format. However, predicates can easily be encoded in terms of transitions, and when disregarding predicates, our current format is more liberal than the format of [10]. Still, all applications of our work discussed in Sect. 6 fall within the format of [10]. The main point of the current framework is not that we have obtained more liberal congruence formats, but that their formulations are simpler and more elegant than existing congruence formats, and, most importantly, that they were obtained in a systematic way from the modal characterisations of (rooted) branching and η-bisimilarity.

9 Conclusions

We have extended the method from [6] for modal decomposition and the derivation of congruence formats to weak semantics. This paper gives a deeper insight into the link between modal logic and congruence formats, and provides a framework for the derivation of congruence formats for the spectrum of weak semantics from [17].

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 [18] (or actually [15] in logic programming); the ntyft/ntyxt format from [8]; the transformation to ruloids, which for the main part goes back to [11]; and the work on modal decomposition and congruence formats from [6].

In spite of these technicalities, we have arrived at a relatively simple framework for the derivation of congruence formats for weak semantics. Namely, 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. As shown in Sect. 5, steps (2) and (3) are very similar in structure for the different weak semantics based on branching and η-bisimulation. And as said, the end results are congruence formats that are more general and at the same time more elegant than existing congruence formats for these weak semantics in the literature.

We have been working on this paper over an extended period of time. Most of all it has taken a lot of effort to arrive at the proper predicates $\aleph$ and Λ, and the corresponding notion of patience rules. When in the end these notions were all in place, it turned out that the whole machinery works like clockwork.

The door is now open to derive congruence formats for a wide range of weak semantics. However, we have found that weak semantics like delay bisimilarity [27] and weak bisimilarity [28], for which the relation pBq' from Def. 1 is missing, require a non-trivial extension of the framework put forward in this paper. We intend to put forward this extended framework in a follow-up of the current paper. 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.

References

1. J.C.M. Baeten, J.A. Bergstra & J.W. Klop (1986): *Syntax and defining equations for an interrupt mechanism in process algebra. Fundamenta Informaticae* 9(2), pp. 127–167.
2. J.C.M. Baeten & R.J. van Glabbeek (1987): *Another look at abstraction in process algebra.* In *Proc. ICALP'87*, LNCS 267, pp. 84–94. Springer.
3. T. Basten (1996): *Branching bisimulation is an equivalence indeed!*, *Information Processing Letters* 58(3), pp. 141-147.
4. J.A. Bergstra & J.W. Klop (1984): *Process algebra for synchronous communication. Information and Control* 60(1/3), pp. 109–137.
5. B. Bloom (1995): *Structural operational semantics for weak bisimulations. Theoretical Computer Science* 146(1/2), pp. 25–68.

6. B. Bloom, W.J. Fokkink & R.J. van Glabbeek (2004): *Precongruence formats for decorated trace semantics. ACM Transactions on Computational Logic* 5(1), pp. 26–78.
7. B. Bloom, S. Istrail & A.R. Meyer (1995): *Bisimulation can't be traced. Journal of the ACM* 42(1), pp. 232–268.
8. R.N. Bol & J.F. Groote (1996): *The meaning of negative premises in transition system specifications. Journal of the ACM* 43(5), pp. 863–914.
9. R. De Nicola & F.W. Vaandrager (1995): *Three logics for branching bisimulation. Journal of the ACM* 42(2), pp. 458–487.
10. W.J. Fokkink (2000): *Rooted branching bisimulation as a congruence. Journal of Computer and System Sciences* 60(1), pp. 13–37.
11. W.J. Fokkink & R.J. van Glabbeek (1996): *Ntyft/ntyxt rules reduce to ntree rules. Information and Computation* 126(1), pp. 1–10.
12. W.J. Fokkink, R.J. van Glabbeek & P. de Wind (2006): *Compositionality of Hennessy-Milner logic by structural operational semantics. Theoretical Computer Science* 354(3), pp. 421–440.
13. W.J. Fokkink, R.J. van Glabbeek & P. de Wind (2006a): *Divide and congruence: From decomposition of modalities to preservation of branching bisimulation.* In *Proc. FMCO'05*, LNCS 4111, pp. 195–218. Springer.
14. W.J. Fokkink, R.J. van Glabbeek & P. de Wind (2006b): *Divide and congruence applied to η-bisimulation.* In *Proc. SOS'05*, ENTCS 156(1), pp. 97–113. Elsevier.
15. A. van Gelder, K. Ross & J.S. Schlipf (1991): *The well-founded semantics for general logic programs. Journal of the ACM* 38(3), pp. 620–650.
16. R.J. van Glabbeek (1987): *Bounded nondeterminism and the approximation induction principle in process algebra.* In *Proc. STACS'87*, LNCS 247, pp. 336–347. Springer.
17. R.J. van Glabbeek (1993): *The linear time-branching time spectrum II: The semantics of sequential systems with silent moves.* In *Proc. CONCUR'93*, LNCS 715, pp. 66–81. Springer.
18. R.J. van Glabbeek (2004): *The meaning of negative premises in transition system specifications II. Journal of Logic and Algebraic Programming* 60/61, pp. 229–258.
19. R.J. van Glabbeek (2011): *On cool congruence formats for weak bisimulations. Theoretical Computer Science*, 412(28):3283–3302, 2011.
20. R.J. van Glabbeek & W.P. Weijland (1996): *Branching time and abstraction in bisimulation semantics. Journal of the ACM* 43(3), pp. 555–600.
21. J.F. Groote (1993): *Transition system specifications with negative premises. Theoretical Computer Science* 118(2), pp. 263–299.
22. J.F. Groote & F.W. Vaandrager (1992): *Structured operational semantics and bisimulation as a congruence. Information and Computation* 100(2), pp. 202–260.
23. M. Hennessy & R. Milner (1985): *Algebraic laws for non-determinism and concurrency. Journal of the ACM* 32(1), pp. 137–161.
24. C.A.R. Hoare (1985): *Communicating Sequential Processes.* Prentice Hall.
25. S.C. Kleene (1956): *Representation of events in nerve nets and finite automata.* In (C. Shannon and J. McCarthy, eds.) *Automata Studies*, pp. 3–41. Princeton University Press.
26. K.G. Larsen & X. Liu (1991): *Compositionality through an operational semantics of contexts. Journal of Logic and Computation* 1(6), pp. 761–795.
27. R. Milner (1981): *A modal characterisation of observable machine-behaviour.* In *Proc. CAAP'81*, LNCS 112, pp. 25–34. Springer.

28. R. Milner (1989): *Communication and Concurrency.* Prentice-Hall.
29. R. Milner (1999): *Communicating and Mobile Systems: the π-Calculus.* Cambridge University Press.
30. G.D. Plotkin (2004): *A structural approach to operational semantics. Journal of Logic and Algebraic Programming* 60/61, pp. 17–139. Originally appeared in 1981.
31. R. de Simone (1985): *Higher-level synchronising devices in* Meije–*SCCS. Theoretical Computer Science* 37(3), pp. 245–267.
32. F.W. Vaandrager (1990): *Algebraic Techniques for Concurrency and their Application.* PhD thesis, University of Amsterdam.

A Modal Characterisations

We prove the first part of Thm. 1, which states that $\mathbb{O}_b$ is a modal characterisation of branching bisimilarity. The proof is based on [9]. We need to prove, given an LTS $(\mathbb{P}, \rightarrow)$, that $p \leftrightarroweq_b q \Leftrightarrow p \sim_{\mathbb{O}_b} q$ for all $p, q \in \mathbb{P}$.

Proof. ($\Rightarrow$) Suppose $p \leftrightarroweq_b q$, and $p \models \varphi$ for some $\varphi \in \mathbb{O}_b$. We prove $q \models \varphi$, by structural induction on φ. The reverse implication ($q \models \varphi$ implies $p \models \varphi$) follows by symmetry.

– $\varphi = \bigwedge_{i \in I} \varphi_i$. Then $p \models \varphi_i$ for $i \in I$. By induction $q \models \varphi_i$ for $i \in I$, so $q \models \bigwedge_{i \in I} \varphi_i$.
– $\varphi = \neg\varphi'$. Then $p \not\models \varphi'$. By induction $q \not\models \varphi'$, so $q \models \neg\varphi'$.
– $\varphi = \langle\varepsilon\rangle\varphi_1\langle\hat{\tau}\rangle\varphi_2$. Then for some n there are $p_0, \ldots, p_n \in \mathbb{P}$ with $p_0 = p$, $p_i \xrightarrow{\tau} p_{i+1}$ for $i \in \{0, \ldots, n-1\}$, and $p_n \models \varphi_1\langle\hat{\tau}\rangle\varphi_2$. We apply induction on n.

  ***n* = 0** Then $p \models \varphi_1$, so by induction on formula size, $q \models \varphi_1$. Furthermore, either (1) $p \models \varphi_2$ or (2) there is a $p' \in \mathbb{P}$ with $p \xrightarrow{\tau} p'$ and $p' \models \varphi_2$. In case (1), by induction on formula size, $q \models \varphi_2$, so $q \models \langle\varepsilon\rangle\varphi_1\langle\hat{\tau}\rangle\varphi_2$. In case (2), since $p \leftrightarroweq_b q$, by Def. 1 either (2.1) $p' \leftrightarroweq_b q$ or (2.2) $q \overset{\varepsilon}{\Longrightarrow} q' \xrightarrow{\tau} q''$ with $p \leftrightarroweq_b q'$ and $p' \leftrightarroweq_b q''$. In case (2.1), by induction on formula size, $q \models \varphi_2$. In case (2.2), by induction on formula size, $q' \models \varphi_1$ and $q'' \models \varphi_2$. In both cases, $q \models \langle\varepsilon\rangle\varphi_1\langle\hat{\tau}\rangle\varphi_2$.

  ***n* > 0** Since $p \xrightarrow{\tau} p_1$, and $p \leftrightarroweq_b q$, according to Def. 1 there are two possibilities.
  1. Either $p_1 \leftrightarroweq_b q$. Since $p_1 \models \langle\varepsilon\rangle\varphi_1\langle\hat{\tau}\rangle\varphi_2$, by induction on n, $q \models \langle\varepsilon\rangle\varphi_1\langle\hat{\tau}\rangle\varphi_2$.
  2. Or $q \overset{\varepsilon}{\Longrightarrow} q' \xrightarrow{\tau} q''$ with $p_1 \leftrightarroweq_b q''$. Since $p_1 \models \langle\varepsilon\rangle\varphi_1\langle\hat{\tau}\rangle\varphi_2$, by induction on n, $q'' \models \langle\varepsilon\rangle\varphi_1\langle\hat{\tau}\rangle\varphi_2$. Hence $q \models \langle\varepsilon\rangle\varphi_1\langle\hat{\tau}\rangle\varphi_2$.

– $\varphi = \langle\varepsilon\rangle\varphi_1\langle a\rangle\varphi_2$. Then for some n there are $p_0, \ldots, p_n \in \mathbb{P}$ with $p_0 = p$, $p_i \xrightarrow{\tau} p_{i+1}$ for $i \in \{0, \ldots, n-1\}$, and $p_n \models \varphi_1\langle a\rangle\varphi_2$. We apply induction on n.

  ***n* = 0** Then $p \models \varphi_1$, and there is a $p' \in \mathbb{P}$ with $p \xrightarrow{a} p'$ and $p' \models \varphi_2$. Since $p \leftrightarroweq_b q$, by Def. 1 $q \overset{\varepsilon}{\Longrightarrow} q' \xrightarrow{a} q''$ with $p \leftrightarroweq_b q'$ and $p' \leftrightarroweq_b q''$. By induction on formula size, $q' \models \varphi_1$ and $q'' \models \varphi_2$. Hence $q \models \langle\varepsilon\rangle\varphi_1\langle a\rangle\varphi_2$.

$n > 0$ Since $p \xrightarrow{\tau} p_1$, and $p \leftrightarroweq_b q$, according to Def. 1 there are two possibilities.

1. Either $p_1 \leftrightarroweq_b q$. Since $p_1 \models \langle\varepsilon\rangle\varphi_1\langle a\rangle\varphi_2$, by induction on n, $q \models \langle\varepsilon\rangle\varphi_1\langle a\rangle\varphi_2$.
2. Or $q \overset{\varepsilon}{\Longrightarrow} q' \xrightarrow{\tau} q''$ with $p_1 \leftrightarroweq_b q''$. Since $p_1 \models \langle\varepsilon\rangle\varphi_1\langle a\rangle\varphi_2$, by induction on n, $q'' \models \langle\varepsilon\rangle\varphi_1\langle a\rangle\varphi_2$. Hence $q \models \langle\varepsilon\rangle\varphi_1\langle a\rangle\varphi_2$.

We conclude that $p \sim_{\mathbb{O}_b} q$.

($\Leftarrow$) We prove that $\sim_{\mathbb{O}_b}$ is a branching bisimulation. The relation is clearly symmetric. Let $p \sim_{\mathbb{O}_b} q$. Suppose $p \xrightarrow{\alpha} p'$. If $\alpha = \tau$ and $p' \sim_{\mathbb{O}_b} q$, then the first condition of Def. 1 is fulfilled. So we can assume that either (i) $\alpha \neq \tau$ or (ii) $p' \not\sim_{\mathbb{O}_b} q$. We define two sets:

$$\begin{aligned} Q' &= \{q' \in \mathbb{P} \mid q \overset{\varepsilon}{\Longrightarrow} q' \wedge p \not\sim_{\mathbb{O}_b} q'\} \\ Q'' &= \{q'' \in \mathbb{P} \mid \exists q' \in \mathbb{P} : q \overset{\varepsilon}{\Longrightarrow} q' \xrightarrow{\alpha} q'' \wedge p' \not\sim_{\mathbb{O}_b} q''\} \end{aligned}$$

For each $q' \in Q'$, let $\varphi_{q'}$ be a formula in $\mathbb{O}_b$ such that $p \models \varphi_{q'}$ and $q' \not\models \varphi_{q'}$. (Such a formula always exists because $\mathbb{O}_b$ is closed under negation $\neg$.) We define

$$\varphi = \bigwedge_{q' \in Q'} \varphi_{q'}$$

Similarly, for each $q'' \in Q''$, let $\psi_{q''}$ be a formula in $\mathbb{O}_b$ such that $p' \models \psi_{q''}$ and $q'' \not\models \psi_{q''}$. We define

$$\psi = \bigwedge_{q'' \in Q''} \psi_{q''}$$

Clearly, $\varphi, \psi \in \mathbb{O}_b$, $p \models \varphi$ and $p' \models \psi$. We distinguish two cases.

1. $\alpha \neq \tau$. Since $p \models \langle\varepsilon\rangle\varphi\langle\alpha\rangle\psi \in \mathbb{O}_b$ and $p \sim_{\mathbb{O}_b} q$, also $q \models \langle\varepsilon\rangle\varphi\langle\alpha\rangle\psi$. Hence $q \overset{\varepsilon}{\Longrightarrow} q' \xrightarrow{\alpha} q''$ with $q' \models \varphi$ and $q'' \models \psi$. By the definition of φ and ψ it follows that $p \sim_{\mathbb{O}_b} q'$ and $p' \sim_{\mathbb{O}_b} q''$.
2. $\alpha = \tau$ and $p' \not\sim_{\mathbb{O}_b} q$. Let $\tilde{\varphi} \in \mathbb{O}_b$ such that $p' \models \tilde{\varphi}$ and $p, q \not\models \tilde{\varphi}$. Since $p \models \langle\varepsilon\rangle\varphi\langle\hat{\tau}\rangle(\tilde{\varphi} \wedge \psi) \in \mathbb{O}_b$ and $p \sim_{\mathbb{O}_b} q$, also $q \models \langle\varepsilon\rangle\varphi\langle\hat{\tau}\rangle(\tilde{\varphi} \wedge \psi)$. So $q \overset{\varepsilon}{\Longrightarrow} q'$ with $q' \models \varphi\langle\hat{\tau}\rangle(\tilde{\varphi} \wedge \psi)$. By definition of φ it follows that $p \sim_{\mathbb{O}_b} q'$. Thus $q' \not\models \tilde{\varphi}$, so $q' \xrightarrow{\tau} q''$ with $q'' \models \tilde{\varphi} \wedge \psi$. By the definition of ψ it follows that $p' \sim_{\mathbb{O}_b} q''$.

Both cases imply that the second condition of Def. 1 is fulfilled. We therefore conclude that $\sim_{\mathbb{O}_b}$ is a branching bisimulation. □

Using the first part of Thm. 1, which was proved above, it is not hard to derive the second part of Thm. 1, i.e. that $\mathbb{O}_{rb}$ is a modal characterisation of rooted branching bisimilarity.

The validity of the modal characterisation of η-bisimilarity can be proved in a similar fashion.