Divide and Congruence Applied to η-Bisimulation

Wan Fokkink [1]

Vrije Universiteit Amsterdam, Department of Computer Science, Amsterdam
CWI, Department of Software Engineering, Amsterdam

Rob van Glabbeek [2]

National ICT Australia, Sydney
Univ. of New South Wales, School of Computer Science and Engineering, Sydney

Paulien de Wind [3]

Vrije Universiteit Amsterdam, Department of Computer Science, Amsterdam

Abstract

We present congruence formats for η- and rooted η-bisimulation equivalence. These formats are derived using a method for decomposing modal formulas in process algebra. 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.

Key words: Structural operational semantics, modal logic, decomposition, congruence, η-bisimulation

1 Introduction

Structural operational semantics [16] 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.

[1] Email: `wanf@cs.vu.nl`

[2] Email: `rvg@cs.stanford.edu`

[3] Email: `pdwind@cs.vu.nl`

To appear in *Electronic Notes in Theoretical Computer Science*: Proceedings of the 2nd Workshop on *Structural Operational Semantics*, SOS 2005.

Labelled transition systems can be distinguished from each other by a wide range of semantic equivalences, based on e.g. branching structure or decorated versions of execution sequences. Van Glabbeek [8] classified equivalences for processes that take into account the internal action τ. Here we focus on one such equivalence, called η-bisimulation [1].

In general a semantic equivalence induced by a transition system specification is not a congruence, i.e. the equivalence class of a term $f(p_1, \ldots, p_n)$ need not be determined by 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 semantic equivalences, to ensure that such an equivalence is a congruence. These formats help to avoid repetitive congruence proofs. Several congruence formats were introduced for bisimulation, such as the De Simone format [17], the GSOS format [4], the tyft/tyxt format [13], and the ntyft/ntyxt format [12]. Bloom [2] introduced congruence formats for weak and branching bisimulation and for rooted weak and branching bisimulation. 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. Recently, van Glabbeek [11] significantly simplified the formats from [2], and presented similar formats for delay and η-bisimulation and for rooted delay and η-bisimulation.

Van Glabbeek [9,8] gave characterisations of equivalences 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 consists of a class C of modal formulas such that two processes are equivalent if and only if they make true the same formulas in C. For instance, Hennessy-Milner logic [14] is the modal characterisation of bisimulation.

Larsen and Liu [15] introduced a method for decomposing formulas from Hennessy-Milner logic for concrete processes, with respect to terms from a process algebra with a structural operational semantics in 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 [3] to ntyft/ntyxt format without lookahead, and by Fokkink, van Glabbeek & de Wind to tyft/tyxt format in the full version of [7]. In [3], the decomposition method was applied to obtain congruence formats for behavioural equivalences from [9]. 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.

In an unpublished manuscript, we extend the work of [3] to processes with τ-transitions. We present a method for decomposing formulas from modal logic for processes with τ-transitions, and use this decomposition method to obtain congruence formats for branching and rooted branching bisimulation.

In the current paper, we use this decomposition method to obtain congruence formats for η- and rooted η-bisimulation. Thus we drive home the point that, 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 semantic equivalences. Our formats use two predicates on arguments of function symbols, to mark both running processes and processes that may have started running.

2 Preliminaries

2.1 Equivalences on labelled transition systems

A *labelled transition system (LTS)* is a pair $(\mathbb{P}, \rightarrow)$ with $\mathbb{P}$ a set of *processes* and $\rightarrow \subseteq \mathbb{P} \times (A \cup \{\tau\}) \times \mathbb{P}$ where τ is an *internal action* and A a set of *actions* not containing τ. We use 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 $p \overset{\alpha}{\not\rightarrow}$ for $\neg\exists q \in \mathbb{P} : p \xrightarrow{\alpha} q$, and $\overset{\epsilon}{\Longrightarrow}$ for the transitive-reflexive closure of $\xrightarrow{\tau}$.

Definition 2.1 [1] A symmetric relation $B \subseteq \mathbb{P} \times \mathbb{P}$ is an *η-bisimulation* if pBq and $p \xrightarrow{\alpha} p'$ implies that either $\alpha = \tau$ and $p' \, B \, q$, or $q \overset{\epsilon}{\Longrightarrow} q' \xrightarrow{\alpha} q'' \overset{\epsilon}{\Longrightarrow} q'''$ for some q', q'', q''' with pBq' and $p'Bq'''$. Processes p, q are *η-bisimilar*, denoted by $p \leftrightarroweq_\eta q$, if there exists an η-bisimulation B with pBq.

η-bisimulation is not a congruence with respect to most process algebras from the literature, meaning that the equivalence class of a term $f(p_1, \ldots, p_n)$ is not always determined by the equivalence classes of its arguments $p_1, \ldots, p_n$. A rootedness condition remedies this imperfection.

Definition 2.2 [1] A symmetric relation $R \subseteq \mathbb{P} \times \mathbb{P}$ is a *rooted η-bisimulation* if pRq and $p \xrightarrow{\alpha} p'$ implies that $q \xrightarrow{\alpha} q' \overset{\epsilon}{\Longrightarrow} q''$ for some q', q'' with $p' \leftrightarroweq_\eta q''$. Processes p, q are *rooted η-bisimilar*, denoted by $p \leftrightarroweq_{r\eta} q$, if there exists a rooted η-bisimulation R with pRq.

2.2 Modal logic

Modal logic aims to formulate properties of processes in an LTS. Following [8], we extend Hennessy-Milner logic [14] with the modal connective $\langle\epsilon\rangle$.

Definition 2.3 The class $\mathbb{O}$ of *modal formulas* is defined as follows, where I ranges over all index sets:

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

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

Definition 2.4 The subclasses $\mathbb{O}_\eta$ and $\mathbb{O}_{r\eta}$ of $\mathbb{O}$ are defined as follows:

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

The classes $\mathbb{O}_\eta^\equiv$ and $\mathbb{O}_{r\eta}^\equiv$ are the closures of $\mathbb{O}_\eta$, respectively $\mathbb{O}_{r\eta}$, under $\equiv$.

The last clause in the definition of $\mathbb{O}_{r\eta}$ guarantees that $\mathbb{O}_\eta \subseteq \mathbb{O}_{r\eta}$, which we need in the proof of Prop. 3.7. Also without this clause it would follow that $\mathbb{O}_\eta^\equiv \subseteq \mathbb{O}_{r\eta}^\equiv$, using structural induction and $\langle\epsilon\rangle\varphi \equiv \varphi \vee \langle\tau\rangle\langle\epsilon\rangle\varphi$.

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

Theorem 2.5 *$p \leftrightarroweq_\eta q \Leftrightarrow p \sim_{\mathbb{O}_\eta} q$ and $p \leftrightarroweq_{r\eta} q \Leftrightarrow p \sim_{\mathbb{O}_{r\eta}} q$, for all $p, q \in \mathbb{P}$.*

A proof of this theorem is presented in the appendix.

2.3 Structural operational semantics

Let V be an infinite set of variables, with typical elements x, y, z. A syntactic object is *closed* if it does not contain any variables. A *signature* is a set Σ of function symbols f with arity $ar(f)$. We always take $|\Sigma|, |A| \leq |V|$. The set $\mathbb{T}(\Sigma)$ of terms over Σ and V is defined as usual. t, u denote terms and p, q closed terms. $var(t)$ is the set of variables that occur in t. A substitution is a partial function from V to $\mathbb{T}(\Sigma)$. A closed substitution σ is a total function from V to closed terms.

Definition 2.6 A (*positive* or *negative*) *literal* is an expression $t \xrightarrow{\alpha} t'$ or $t \overset{\alpha}{\not\longrightarrow}$. A *(transition) rule* is of the form $\frac{H}{t \xrightarrow{\alpha} t'}$ with H a set of literals called the *premises*. $t \xrightarrow{\alpha} t'$ is the *conclusion* and t the *source* of the rule. A rule $\frac{\emptyset}{t \xrightarrow{\alpha} t'}$ is also written $t \xrightarrow{\alpha} t'$. A *transition system specification (TSS)* is a set of transition rules.

Definition 2.7 Let $P = (\Sigma, R)$ be a TSS. An *irredundant proof* from P of a rule $\frac{H}{t \xrightarrow{\alpha} t'}$ is a well-founded tree with the nodes labelled by literals and some of the leaves marked "hypothesis", such that the root has label $t \xrightarrow{\alpha} t'$, 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 μ is positive and $\frac{K}{\mu}$ is a substitution instance of a rule in R.

The proof of $\frac{H}{t \xrightarrow{\alpha} t'}$ is called irredundant because H must equal (instead of include) the set of labels of the hypotheses. This irredundancy will be crucial for the preservation of our congruence formats in Sect. 3.1 (see Prop. 3.4).

A TSS is meant to specify an LTS in which the transitions are closed positive literals. A 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 [10] we adopt the notion of a well-supported proof of a closed literal. Literals $t \xrightarrow{\alpha} t'$ and $t \overset{\alpha}{\not\longrightarrow}$ are said to *deny* each other.

Definition 2.8 Let $P = (\Sigma, R)$ be a TSS. A *well-supported proof* from P of a closed literal μ is a well-founded tree with the nodes labelled by closed literals, such that the root is labelled by μ, and if ν is the label of a node and K is the set of labels of the children of this node, then:

(i) either ν is positive and $\frac{K}{\nu}$ is a closed substitution instance of a rule in R;

(ii) or ν is negative and for each set N of closed negative literals with $\frac{N}{\kappa}$ irredundantly provable from P and κ a closed positive literal denying ν, a literal in K denies one in N.

$P \vdash_{ws} \mu$ denotes that a well-supported proof from P of μ exists. P is *complete* if for each p and α, either $P \vdash_{ws} p \overset{\alpha}{\not\longrightarrow}$ or $P \vdash_{ws} p \xrightarrow{\alpha} p'$ for some p'.

A complete TSS specifies an LTS, consisting of the *ws*-provable closed positive literals.

2.4 Notions regarding transition rules

In this section we present terminology for syntactic restrictions on rules, originating from [3,12,13].

Definition 2.9 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.

Definition 2.10 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.

The ntyft/ntyxt and ready simulation formats [12,3] were originally introduced to guarantee congruence for bisimulation and ready simulation.

Definition 2.11 A TSS is in *ntyft/ntyxt format* if it consists of ntyft and ntyxt rules, and in *ready simulation format* if moreover its rules do not have lookahead.

A predicate $\aleph$ marks arguments of function symbols that contain running processes (cf. [3]). Typically, in process algebra, $\aleph$ holds for the arguments of the merge $\|$, but not for the arguments of alternative composition $+$.

Definition 2.12 Let $\aleph$ be a unary predicate on $\{(f, i) \mid 1 \leq i \leq ar(f),\ f \in \Sigma\}$. If $\aleph(f, i)$, then argument i of f is *liquid*; otherwise it is *frozen*. An occurrence of x in t is *at an $\aleph$-liquid position* if either $t = x$, or $t = f(t_1, \ldots, t_{ar(f)})$ and the occurrence is at an $\aleph$-liquid position in t_i for a liquid argument i of f.

A patience rule for an argument i of a function symbol f expresses that term $f(p_1, \ldots, p_n)$ inherits the τ-transitions of argument p_i (cf. [2,5]). We will require the presence of patience rules for liquid arguments.

Definition 2.13 An ntyft rule is a *patience rule* for (f, i), with $1 \leq i \leq ar(f)$, if it is of the form

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

It is an $\aleph$-*patience rule* if $\aleph(f, i)$.

An ntytt rule is *patient* with respect to $\aleph$ if it is irredundantly provable from the $\aleph$-patience rules. Such rules have the form $\frac{t \xrightarrow{\tau} y}{C[t] \xrightarrow{\tau} C[y]}$ with $C[]$ an $\aleph$-liquid context, meaning that the context symbol $[]$ occurs at an $\aleph$-liquid position.

Definition 2.14 A TSS is $\aleph$-*patient* if it contains all $\aleph$-patience rules. It is *abstraction-free* if only $\aleph$-patience rules have a conclusion of the form $t \xrightarrow{\tau} u$.

2.5 Decomposition of modal formulas

To decompose modal formulas, we use a result from [3], where for any TSS P in ready simulation format a collection of decent nxytt rules, called P-*ruloids*, is constructed. We explain this construction on a rather superficial level; the precise transformation can be found in [3].

First P is converted to a TSS in decent ntyft format. In this conversion from [13], free variables in a rule are replaced by closed terms, and if the source is of the form x then this variable is replaced by a term $f(x_1, \ldots, x_n)$ for each $f \in \Sigma$. Next, using a construction from [6], left-hand sides of positive premises are reduced to variables. Roughly the idea is, given a premise $f(t_1, \ldots, t_n) \xrightarrow{\alpha} y$ in a rule r, and a 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 left-hand side have disappeared. In the final transformation step, rules with a *negative* conclusion $t \stackrel{\alpha}{\not\longrightarrow}$ are introduced. The motivation is that instead of the notion of well-founded provability in Def. 2.8, we want a more constructive notion like Def. 2.7, by making it possible that a negative premise is matched with a negative conclusion. A rule r with a conclusion $f(x_1, \ldots, x_n) \stackrel{\alpha}{\not\longrightarrow}$ is obtained by picking one premise from each rule with a conclusion $f(x_1, \ldots, x_n) \xrightarrow{\alpha} t$, and including the denial of each of the selected premises as a premise of r. For this last transformation it is essential that rules do not have lookahead.

The resulting TSS, which is in decent ntyft format, is denoted by P^+. In [3] it is established that $P^+ \vdash \mu \Leftrightarrow P \vdash_{ws} \mu$ for all closed literals μ. The notion of irredundant provability is adapted in a straightforward fashion to accommodate rules with a negative conclusion. P-ruloids are the decent nxytt rules that are irredundantly provable from P^+. The following correspondence result from [3] 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 $p \xrightarrow{a} q$, with p a closed substitution instance of a term t, if and only if there is a proof of this transition that uses at the root a P-ruloid with source t.

Proposition 2.15 *Let P be a TSS in ready simulation format. Then $P \vdash_{ws} \sigma(t) \xrightarrow{\alpha} p$ if and only if there are a P-ruloid $\frac{H}{t \xrightarrow{\alpha} u}$ and a σ' with $P \vdash_{ws} \sigma'(\mu)$ for $\mu \in H$, $\sigma'(t) = \sigma(t)$ and $\sigma'(u) = p$.*

We now 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 $\sigma(t) \models \varphi$ if $\sigma(x) \models \psi(x)$ for $x \in \mathit{var}(t)$. Vice versa, whenever $\sigma(t) \models \varphi$, there is a decomposition mapping $\psi \in t^{-1}(\varphi)$ with $\sigma(x) \models \psi(x)$ for $x \in \mathit{var}(t)$. This is formalised in Thm. 2.17.

In order minimise the complexity inherent in the combination of modal decomposition and the internal action τ, we apply the decomposition method to abstraction-free TSSs, and extend our derived congruence results to the general case using the well-known compositionality of the abstraction operator.

Definition 2.16 Let P be an $\aleph$-patient, abstraction-free TSS in ready simulation format. We define $\cdot^{-1} : \mathbb{T}(\Sigma) \times \mathbb{O} \to \mathcal{P}(V \to \mathbb{O})$ as follows. Let t denote a univariate term, i.e. without multiple occurrences of the same variable.

(i) $\psi \in t^{-1}(\bigwedge_{i\in I} \varphi_i)$ iff for $x \in V$
$$\psi(x) = \bigwedge_{i \in I} \psi_i(x)$$
where $\psi_i \in t^{-1}(\varphi_i)$ for $i \in I$.

(ii) $\psi \in t^{-1}(\langle\alpha\rangle\varphi)$ iff there is a P-ruloid $\frac{H}{t \xrightarrow{\alpha} u}$ and a $\chi \in u^{-1}(\varphi)$ with
$$\psi(x) = \begin{cases} \chi(x) \ \wedge \ \bigwedge\limits_{x \xrightarrow{\beta} y \in H} \langle\beta\rangle\chi(y) \ \wedge \ \bigwedge\limits_{x \stackrel{\gamma}{\not\rightarrow} \in H} \neg\langle\gamma\rangle\top & \text{if } x \in \mathit{var}(t) \\ \top & \text{if } x \notin \mathit{var}(t) \end{cases}$$

(iii) $\psi \in t^{-1}(\neg\varphi)$ iff there is a function $h : t^{-1}(\varphi) \to \mathit{var}(t)$ with
$$\psi(x) = \bigwedge_{\chi \in h^{-1}(x)} \neg\chi(x) \qquad \text{for } x \in V$$

(iv) $\psi \in t^{-1}(\langle\epsilon\rangle\varphi)$ iff there is a $\chi \in t^{-1}(\varphi)$ with
$$\psi(x) = \begin{cases} \langle\epsilon\rangle\chi(x) & \text{if } x \text{ occurs } \aleph\text{-liquid in } t \\ \chi(x) & \text{otherwise} \end{cases}$$

(v) $\psi \in \rho(t)^{-1}(\varphi)$ for $\rho : \mathit{var}(t) \to V$ not injective iff there is a $\chi \in t^{-1}(\varphi)$ with
$$\psi(x) = \bigwedge_{y \in \rho^{-1}(x)} \chi(y) \qquad \text{for } x \in V$$

It is not hard to see that if $\psi \in t^{-1}(\varphi)$, then $\psi(x) \equiv \top$ for $x \notin var(t)$.

To explain the idea behind Def. 2.16, we expand on two of its cases. Consider $t^{-1}(\langle\alpha\rangle\varphi)$, and let σ be any closed substitution. The question is under which conditions $\psi(x) \in \mathbb{O}$ on $\sigma(x)$, for $x \in var(t)$, there is a transition $\sigma(t) \xrightarrow{\alpha} q$ with $q \models \varphi$. According to Prop. 2.15, there is such a transition if and only if there is a closed substitution σ' with $\sigma'(t) = \sigma(t)$ and a P-ruloid $\frac{H}{t \xrightarrow{\alpha} u}$ such that (1) the premises in $\sigma'(H)$ are satisfied and (2) $\sigma'(u) \models \varphi$. The first condition is covered if for $x \in var(t)$, $\psi(x)$ contains conjuncts $\langle\beta\rangle\top$ for $x \xrightarrow{\beta} y \in H$ and conjuncts $\neg\langle\gamma\rangle\top$ for $x \stackrel{\gamma}{\nrightarrow} \in H$. 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.

Consider $t^{-1}(\neg\varphi)$, and let σ be any closed substitution. We have $\sigma(t) \not\models \varphi$ if and only if there is no $\chi \in t^{-1}(\varphi)$ such that $\sigma(x) \models \chi(x)$ 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 $x \in var(t)$.

The following theorem, whose proof is omitted here, will be the key to the forthcoming congruence results.

Theorem 2.17 *Given a complete, $\aleph$-patient, abstraction-free TSS in ready simulation format. For any term t, closed substitution σ and $\varphi \in \mathbb{O}$:*

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

3 η-Bisimulation as a Congruence

We proceed to apply the decomposition method from the previous section to derive congruence formats for η- and rooted η-bisimulation equivalence. The idea is that the η-bisimulation format must guarantee that a formula from $\mathbb{O}_\eta$ is always decomposed into formulas from $\mathbb{O}_\eta^\equiv$ (see Prop. 3.6). Likewise, the rooted η-bisimulation format must guarantee that a formula from $\mathbb{O}_{r\eta}$ is always decomposed into formulas from $\mathbb{O}_{r\eta}^\equiv$ (see Prop. 3.7). This implies the desired congruence results (see Thm. 3.9 and Thm. 3.11). In deriving the congruence formats, we will circumvent the restriction to abstraction-free TSSs, using compositionality of the abstraction operator.

3.1 Congruence formats

We assume a second predicate Λ on arguments of function symbols, to denote that the processes they contain may have started running, but might currently be resting, in which case no patience rule is needed for these arguments. Always $\aleph \subseteq \Lambda$. Typically, in process algebra, $\aleph$ holds for the first argument of sequential composition while only Λ holds for the second argument, and Λ does not hold for the arguments of alternative composition.

Definition 3.1 Let $\aleph \subseteq \Lambda$. An ntytt rule $\frac{H}{t \xrightarrow{\alpha} u}$ is *rooted η-bisimulation safe* with respect to $\aleph$ and Λ if:

(i) it has no lookahead,

(ii) right-hand sides of premises occur only at $\aleph$-liquid positions in u, and

(iii) if x occurs exactly once[4] in t, at a Λ-liquid position, then:
  (a) all occurrences of x in the rule are at Λ-liquid positions,
  (b) x has no $\aleph$-liquid occurrences in left-hand sides of negative premises,
  (c) x has at most one $\aleph$-liquid occurrence in the left-hand side of one positive premise, and this premise has a label from A, and
  (d) if x occurs at an $\aleph$-frozen position in t, then x does not occur at $\aleph$-liquid positions in left-hand sides of premises.

In case Λ is the universal predicate, we say that the rule is *η-bisimulation safe* with respect to $\aleph$.

Definition 3.2 A TSS in ready simulation format is in *rooted η-bisimulation format* if, for some $\aleph \subseteq \Lambda$, it consists of the $\aleph$-patience rules and rules that are rooted η-bisimulation safe with respect to $\aleph$ and Λ.

A TSS in ready simulation format is in *η-bisimulation format* if, for some $\aleph$, it consists of the $\aleph$-patience rules and rules that are η-bisimulation safe with respect to $\aleph$.

The operators *initial priority* (with frozen argument) and *binary Kleene star* (with both arguments frozen) of [3] fit the rooted η-bisimulation format. In these applications, as well as for capturing the operators of CCS and similar languages, it suffices to take $\Lambda = \aleph$. In the following example this is not possible.

Example 3.3 Let f be a binary operator that interleaves actions $\alpha \in A \cup \{\tau\}$ 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.

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

pm is a CCS action-prefixing operator. 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.

In the definition of modal decomposition, we did not use the rules from the original TSS P, but the P-ruloids. Therefore we must verify that if P is in (rooted) η-bisimulation format, then so are the P-ruloids.

[4] Only the requirements for rules in which t is univariate matter. The current formulation of Def. 3.1 for general terms t paves the way for Prop. 3.4.

Proposition 3.4 *If a TSS P is in (rooted) η-bisimulation format with respect to some $\aleph$, then each P-ruloid is either patient or (rooted) η-bisimulation safe with respect to $\aleph$.*

The proof of Prop. 3.4 is omitted here. The key part of the proof is to show that the decent (rooted) η-bisimulation format is preserved under irredundant provability. (The adjective irredundant is essential here.)

3.2 Preservation of modal characterisations

From now on we will mention patient and (rooted) branching safe rules without reference to the accompanying predicates $\aleph$ and Λ.

Lemma 3.5 *Given a TSS in ready simulation format. For any term t, $\varphi \in \mathbb{O}$ and variable x that occurs only $\aleph$-liquid in t, $\psi \in t^{-1}(\langle\epsilon\rangle\varphi) \Rightarrow \psi(x) \equiv \langle\epsilon\rangle\psi(x)$.*

Proof. Let $\psi \in t^{-1}(\langle\epsilon\rangle\varphi)$ and x occur only $\aleph$-liquid in t. Then by Def. 2.16.iv and 2.16.v, $\psi(x)$ is of the form $\bigwedge_{i\in I}\langle\epsilon\rangle\varphi_i$. So $\psi(x) \equiv \langle\epsilon\rangle\psi(x)$. □

Proposition 3.6 *Let P be an abstraction-free TSS in rooted η-bisimulation format. 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. We apply structural induction on φ. Let $\varphi \in \mathbb{O}_\eta$. Let $t \in \mathbb{T}(\Sigma)$ and $\psi \in t^{-1}(\varphi)$, and let x occur only Λ-liquid in t. First we treat the case where t is univariate. If $x \notin var(t)$, then $\psi(x) \equiv \top \in \mathbb{O}_\eta^{\equiv}$. Suppose x occurs once in t.

- $\varphi = \bigwedge_{i\in I}\varphi_i$ with $\varphi_i \in \mathbb{O}_\eta$ for $i \in I$. By Def. 2.16.i, $\psi(x) = \bigwedge_{i\in I}\psi_i(x)$ with $\psi_i \in t^{-1}(\varphi_i)$ for $i \in I$. By induction, $\psi_i(x) \in \mathbb{O}_\eta^{\equiv}$ for $i \in I$, so $\psi(x) \in \mathbb{O}_\eta^{\equiv}$.
- $\varphi = \neg\varphi'$ with $\varphi' \in \mathbb{O}_\eta$. By Def. 2.16.iii, 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, $\chi(x) \in \mathbb{O}_\eta^{\equiv}$ for $\chi \in h^{-1}(x)$, so $\psi(x) \in \mathbb{O}_\eta^{\equiv}$.
- $\varphi = \langle\epsilon\rangle\varphi'$ with $\varphi' \in \mathbb{O}_\eta$. By Def. 2.16.iv, either $\psi(x) = \langle\epsilon\rangle\chi(x)$ or $\psi(x) = \chi(x)$ for some $\chi \in t^{-1}(\varphi')$. By induction on formula size, $\chi(x) \in \mathbb{O}_\eta^{\equiv}$. So $\psi(x) \in \mathbb{O}_\eta^{\equiv}$.
- $\varphi = \langle\epsilon\rangle(\varphi_1\langle a\rangle\langle\epsilon\rangle\varphi_2)$ with $\varphi_1, \varphi_2 \in \mathbb{O}_\eta$. By Def. 2.16.iv, either $\psi(x) = \langle\epsilon\rangle\chi(x)$ or $\psi(x) = \chi(x)$ for some $\chi \in t^{-1}(\varphi_1\langle a\rangle\langle\epsilon\rangle\varphi_2)$. By Def. 2.16.i, $\chi(x) = \chi_1(x) \wedge \chi_2(x)$ with $\chi_1 \in t^{-1}(\varphi_1)$ and $\chi_2 \in t^{-1}(\langle a\rangle\langle\epsilon\rangle\varphi_2)$. By induction on formula size, $\chi_1(x) \in \mathbb{O}_\eta^{\equiv}$. By Def. 2.16.ii,

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

for some $\xi \in u^{-1}(\langle\epsilon\rangle\varphi_2)$ and some P-ruloid $\frac{H}{t \xrightarrow{a} u}$. Since $a \neq \tau$, by Prop. 3.4, $\frac{H}{t \xrightarrow{a} u}$ is rooted η-bisimulation safe. Since the occurrence of x in t is Λ-liquid, x occurs only Λ-liquid in u. Moreover, variables in right-hand sides of premises in H occur only $\aleph$-liquid (hence Λ-liquid) in u. So in the previous case we proved that $\xi(x) \in \mathbb{O}_\eta^\equiv$ and $\xi(y) \in \mathbb{O}_\eta^\equiv$ for $x \xrightarrow{\beta} y \in H$. We distinguish two cases.

Case 1: The occurrence of x in t is $\aleph$-liquid. Then $\psi(x) = \langle\epsilon\rangle\chi(x)$. Since $\frac{H}{t \xrightarrow{a} u}$ is rooted η-bisimulation safe and an nxytt rule, x does not occur in left-hand sides of negative premises in H, and at most once in the left-hand side of one positive premise in H, which is of the form $x \xrightarrow{b} y$ with $b \in A$. The right-hand side, y, of such a premise, occurs only $\aleph$-liquid in u, so by Lem. 3.5, $\xi(y) \equiv \langle\epsilon\rangle\xi(y)$. Hence, either $\chi_2(x) = \xi(x)$ or $\chi_2(x) = \xi(x)\langle b\rangle\xi(y) \equiv \xi(x)\langle b\rangle\langle\epsilon\rangle\xi(y)$. Since $\psi(x) = \langle\epsilon\rangle(\chi_1(x) \wedge \chi_2(x))$, either $\psi(x) = \langle\epsilon\rangle(\chi_1(x)\wedge\xi(x)) \in \mathbb{O}_\eta^\equiv$ or $\psi(x) \equiv \langle\epsilon\rangle(\chi_1(x)\wedge\xi(x)\langle b\rangle\langle\epsilon\rangle\xi(y)) \in \mathbb{O}_\eta^\equiv$.

Case 2: The occurrence of x in t is $\aleph$-frozen. Then $\psi(x) = \chi(x)$. Since $\frac{H}{t \xrightarrow{a} u}$ is rooted η-bisimulation safe and an nxytt rule, x does not occur in left-hand sides of premises in H. So $\chi_2(x) = \xi(x)$, and thus $\psi(x) = \chi_1(x) \wedge \chi_2(x) = \chi_1(x) \wedge \xi(x) \in \mathbb{O}_\eta^\equiv$.

Finally, suppose t is not univariate. Then $t = \rho(u)$ for some univariate term u and $\rho : \mathit{var}(u) \to V$ not injective. By Def. 2.16.v, $\psi(x) = \bigwedge_{y\in\rho^{-1}(x)} \chi(y)$ for some $\chi \in u^{-1}(\varphi)$. Since u is univariate, and for each $y \in \rho^{-1}(x)$ the occurrence in u is Λ-liquid, $\chi(y) \in \mathbb{O}_\eta^\equiv$ for $y \in \rho^{-1}(x)$. Hence, $\psi(x) \in \mathbb{O}_\eta^\equiv$. □

Proposition 3.7 *Let P be an abstraction-free TSS in rooted η-bisimulation format. 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. We apply structural induction on φ. Let $\varphi \in \mathbb{O}_{r\eta}$, $t \in \mathbb{T}(\Sigma)$ and $\psi \in t^{-1}(\varphi)$. 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. 3.6. If $x \notin \mathit{var}(t)$, then $\psi(x) \equiv \top \in \mathbb{O}_{r\eta}^\equiv$. 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. 3.6.
- $\varphi = \langle\alpha\rangle\langle\epsilon\rangle\varphi'$ with $\varphi' \in \mathbb{O}_\eta$. By Def. 2.16.ii,

$$\psi(x) = \chi(x) \wedge \bigwedge_{x \xrightarrow{\beta} y \in H} \langle\beta\rangle\chi(y) \wedge \bigwedge_{x \overset{\gamma}{\nrightarrow} \in H} \neg\langle\gamma\rangle\top$$

  for some $\chi \in u^{-1}(\langle\epsilon\rangle\varphi')$ and some P-ruloid $\frac{H}{t \xrightarrow{\alpha} u}$. By induction, $\chi(x) \in \mathbb{O}_{r\eta}$. (Induction may be applied because $\langle\epsilon\rangle\varphi' \in \mathbb{O}_\eta \subseteq \mathbb{O}_{r\eta}$ and $\langle\epsilon\rangle\varphi'$ is a strict subformula of φ.) By Prop. 3.4, $\frac{H}{t \xrightarrow{\alpha} u}$ is either rooted η-bisimulation safe or $\aleph$-patient. Thus variables in right-hand sides of premises in H occur only $\aleph$-liquid in u. By Prop. 3.6, $\chi(y) \in \mathbb{O}_\eta$ for $x \xrightarrow{\beta} y \in H$. Moreover, by Lemma 3.5, $\chi(y) \equiv \langle\epsilon\rangle\chi(y)$. Hence $\langle\beta\rangle\chi(y) \equiv \langle\beta\rangle\langle\epsilon\rangle\chi(y) \in \mathbb{O}_{r\eta}^\equiv$. Also $\neg\langle\gamma\rangle\top \equiv \neg\langle\gamma\rangle\langle\epsilon\rangle\top \in \mathbb{O}_{r\eta}^\equiv$. Hence, $\psi(x) \in \mathbb{O}_{r\eta}$.
- $\varphi \in \mathbb{O}_\eta$. If the occurrence of x in t is Λ-liquid, then $\psi(x) \in \mathbb{O}_{r\eta}^\equiv$ follows from Prop. 3.6. So we can assume that this occurrence is Λ-frozen. The cases $\varphi = \bigwedge_{i\in I}\varphi_i$ and $\varphi = \neg\varphi'$ proceed as before. We focus on the other two cases.

* $\varphi = \langle\epsilon\rangle\varphi'$ with $\varphi' \in \mathbb{O}_\eta \subseteq \mathbb{O}_{r\eta}$. Since the occurrence of x in t is Λ-frozen, by Def. 2.16.iv, $\psi(x) = \chi(x)$ for some $\chi \in t^{-1}(\varphi')$. By induction on formula size, $\chi(x) \in \mathbb{O}_{r\eta}$. So $\psi(x) \in \mathbb{O}_{r\eta}$.
* $\varphi = \langle\epsilon\rangle(\varphi_1\langle a\rangle\langle\epsilon\rangle\varphi_2)$ with $\varphi_1, \varphi_2 \in \mathbb{O}_\eta \subseteq \mathbb{O}_{r\eta}$. Since the occurrence of x in t is Λ-frozen, by Def. 2.16.iv, $\psi(x) = \chi(x)$ for some $\chi \in t^{-1}(\varphi_1\langle a\rangle\langle\epsilon\rangle\varphi_2)$. By Def. 2.16.i, $\chi(x) = \chi_1(x) \wedge \chi_2(x)$ with $\chi_1 \in t^{-1}(\varphi_1)$ and $\chi_2 \in t^{-1}(\langle a\rangle\langle\epsilon\rangle\varphi_2)$. By induction on formula size, $\chi_1(x), \chi_2(x) \in \mathbb{O}_{r\eta}$. Hence, $\psi(x) \in \mathbb{O}_{r\eta}$. □

3.3 Congruence results

Finally we are in a position to prove the promised congruence results.

Lemma 3.8 *Given a complete abstraction-free TSS in η-bisimulation format. If $\sigma(x) \leftrightarroweq_\eta \sigma'(x)$ for $x \in var(t)$, then $\sigma(t) \leftrightarroweq_\eta \sigma'(t)$.*

Proof. By Thm. 2.5, $\sigma(x) \leftrightarroweq_\eta \sigma'(x)$ implies $\sigma(x) \sim_{\mathbb{O}_\eta^\equiv} \sigma'(x)$ for $x \in var(t)$. Let $\sigma(t) \models \varphi \in \mathbb{O}_\eta$. By Thm. 2.17 there is a $\psi \in t^{-1}(\varphi)$ with $\sigma(x) \models \psi(x)$ for $x \in var(t)$. Since Λ is universal, by Prop. 3.6, $\psi(x) \in \mathbb{O}_\eta^\equiv$ for $x \in var(t)$. Since $\sigma(x) \sim_{\mathbb{O}_\eta^\equiv} \sigma'(x)$, $\sigma'(x) \models \psi(x)$ for $x \in var(t)$. By Thm. 2.17, $\sigma'(t) \models \varphi$. Likewise, $\sigma'(t) \models \varphi \in \mathbb{O}_\eta$ implies $\sigma(t) \models \varphi$. So $\sigma(t) \sim_{\mathbb{O}_\eta} \sigma'(t)$. Hence, $\sigma(t) \leftrightarroweq_\eta \sigma'(t)$. □

Theorem 3.9 *Given a complete TSS $P = (\Sigma, R)$ in η-bisimulation format. If $\sigma(x) \leftrightarroweq_\eta \sigma'(x)$ for $x \in var(t)$, then $\sigma(t) \leftrightarroweq_\eta \sigma'(t)$.*

Proof. Let P' be obtained from P, by changing in all rules expect $\aleph$-patience rules a conclusion of the form $t \xrightarrow{\tau} u$ into $t \xrightarrow{i} u$, for a fresh action $i \notin A \cup \{\tau\}$. By construction, P' is abstraction-free and in η-bisimulation format. So by Lemma 3.8, $\leftrightarroweq_\eta$ is a congruence for all operators of P'.

Let P'' be obtained from P' by adding a new operator τ_i with rules

$$\frac{x \xrightarrow{\alpha} y}{\tau_i(x) \xrightarrow{\alpha} \tau_i(y)}\ (\alpha \neq i) \qquad\qquad \frac{x \xrightarrow{i} y}{\tau_i(x) \xrightarrow{\tau} \tau_i(y)}$$

This operator turns all i-labels into τ-labels. It is well-known [1] and trivial to check that $\leftrightarroweq_\eta$ is a congruence for τ_i as well.

If follows trivially that for any operator $f \in \Sigma$ the behaviour of $\tau_i \circ f$ in P'' is the same as the behaviour of f in P. So as $\leftrightarroweq_\eta$ is a congruence for $\tau_i \circ f$ in P'', it must be a congruence for f in P. □

Lemma 3.10 *Given a complete abstraction-free TSS in rooted η-bisimulation format. If $\sigma(x) \leftrightarroweq_{r\eta} \sigma'(x)$ for $x \in var(t)$, then $\sigma(t) \leftrightarroweq_{r\eta} \sigma'(t)$.*

Theorem 3.11 *Given a complete TSS in rooted η-bisimulation format. If $\sigma(x) \leftrightarroweq_{r\eta} \sigma'(x)$ for $x \in var(t)$, then $\sigma(t) \leftrightarroweq_{r\eta} \sigma'(t)$.*

The proof of Lemma 3.10 is similar to the one of Lemma 3.8, except that Prop. 3.7 is applied instead of Prop. 3.6. Likewise, the proof of Thm. 3.11 is similar to the one of Thm. 3.9.

4 Related work

The only other formats for η- and rooted η-bisimulation that we are aware of appeared in [11]. Those formats are contained in the GSOS format [4]. The formats of [11] distinguish so-called "principal" operators and "abbreviations". The latter can be regarded as syntactic sugar, adding nothing that could not be expressed with principal operators. Our formats are incomparable with the ones of [11]. However, our formats generalise the result of simplifying the formats of [11] by requiring all operators to be principal.

For the η-bisimulation format our generalisation consists in allowing transition rules outside the GSOS format; the simplified format of [11] is exactly the intersection of our η-bisimulation format and the GSOS format. However, the intersection of our rooted η-bisimulation format and the GSOS format is still a proper generalisation of the simplified format for rooted η-bisimulation of [11]. The latter can be described as the intersection of our rooted η-bisimulation format and the GSOS format in which all arguments of all operators that occur in the right-hand sides of conclusions of transition rules, are required to be Λ-liquid.

The format of [11] for rooted η-bisimulation, following [2], distinguishes "tame" and "wild" function symbols. In terms of our approach, wild operators have only Λ-frozen arguments, and tame operators only Λ-liquid arguments. The idea to allow operators with both kinds of arguments stems from [5].

References

[1] J.C.M. Baeten & R.J. van Glabbeek (1987): *Another look at abstraction in process algebra.* In *Proc. ICALP'87*, LNCS 267, Springer, pp. 84–94.

[2] B. Bloom (1995): *Structural operational semantics for weak bisimulations. Theoretical Computer Science* 146(1/2), pp. 25–68.

[3] 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.

[4] B. Bloom, S. Istrail & A.R. Meyer (1995): *Bisimulation can't be traced. Journal of the ACM* 42(1), pp. 232–268.

[5] W.J. Fokkink (2000): *Rooted branching bisimulation as a congruence. Journal of Computer and System Sciences* 60(1), pp. 13–37.

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

[7] W.J. Fokkink, R.J. van Glabbeek & P. de Wind (2005): *Compositionality of Hennessy-Milner logic by structural operational semantics.* Available at http://theory.stanford.edu/~rvg/abstracts.html#61. To appear in *Theoretical Computer Science*. Extended abstract appeared in: *Proc. FCT'03*, LNCS 2751, Springer, 2003, pp. 412–422.

[8] 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, Springer, pp. 66–81.

[9] R.J. van Glabbeek (2001): *The linear time – branching time spectrum I: The semantics of concrete, sequential processes.* In J.A. Bergstra, A. Ponse & S.A. Smolka, editors: *Handbook of Process Algebra*, chapter 1, Elsevier, pp. 3–99.

[10] 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.

[11] R.J. van Glabbeek (2005): *On cool congruence formats for weak bisimulations.* Available at http://theory.stanford.edu/~rvg/abstracts.html#58. Extended abstract in *Proc. ICTAC'05*, LNCS 3722, Springer, pp. 331–346.

[12] J.F. Groote (1993): *Transition system specifications with negative premises.* *Theoretical Computer Science* 118(2), pp. 263–299.

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

[14] M. Hennessy & R. Milner (1985): *Algebraic laws for non-determinism and concurrency.* *Journal of the ACM* 32(1), pp. 137–161.

[15] K.G. Larsen & X. Liu (1991): *Compositionality through an operational semantics of contexts.* *Journal of Logic and Computation* 1(6), pp. 761–795.

[16] 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.

[17] R. de Simone (1985): *Higher-level synchronising devices in* Meije*–SCCS.* *Theoretical Computer Science* 37(3), pp. 245–267.

A Modal Characterisation of η-Bisimulation

We prove the first part of Thm. 2.5, which states that $\mathbb{O}_\eta$ is a modal characterisation of η-bisimulation equivalence. We need to prove, given an LTS $(\mathbb{P}, \longrightarrow)$, that $p \leftrightarroweq_\eta q \Leftrightarrow p \sim_{\mathbb{O}_\eta} q$ for all $p, q \in \mathbb{P}$.

Proof. ($\Rightarrow$) Suppose $p \leftrightarroweq_\eta q$, and $p \models \varphi$ for some $\varphi \in \mathbb{O}_\eta$. We prove $q \models \varphi$, by structural induction on φ. The reverse implication 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 $q \models \neg\varphi'$.
- $\varphi = \langle\epsilon\rangle\varphi$. Then for some n there are $p_0, \ldots, p_n \in \mathbb{P}$ with $p_n = p$, $p_{i+1} \xrightarrow{\tau} p_i$ for $i \in \{0, \ldots, n-1\}$, and $p_0 \models \varphi$. We apply induction on n.
  ***n* = 0** $p \models \varphi$, so by induction on formula size, $q \models \varphi$. Hence $q \models \langle\epsilon\rangle\varphi$.
  ***n* > 0** Since $p_n \xrightarrow{\tau} p_{n-1}$, according to Def. 2.1 there are two possibilities.
  (i) Either $p_{n-1} \leftrightarroweq_\eta q$. Since $p_{n-1} \models \langle\epsilon\rangle\varphi$, by induction on n, $q \models \langle\epsilon\rangle\varphi$.
  (ii) Or $q \overset{\epsilon}{\Longrightarrow} q' \xrightarrow{\tau} q'' \overset{\epsilon}{\Longrightarrow} q'''$ with $p_{n-1} \leftrightarroweq_\eta q'''$. Since $p_{n-1} \models \langle\epsilon\rangle\varphi$, by induction on n, $q''' \models \langle\epsilon\rangle\varphi$. Hence $q \models \langle\epsilon\rangle\varphi$.

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

  $\boldsymbol{n = 0}$ Then $p \models \varphi_1$, and there is a $p' \in \mathbb{P}$ with $p \xrightarrow{a} p'$ and $p' \models \langle\epsilon\rangle\varphi_2$. By Def. 2.1, $q \overset{\epsilon}{\Longrightarrow} q' \xrightarrow{a} q'' \overset{\epsilon}{\Longrightarrow} q'''$ with $p \leftrightarroweq_\eta q'$ and $p' \leftrightarroweq_\eta q'''$. By induction on formula size, $q' \models \varphi_1$ and $q''' \models \langle\epsilon\rangle\varphi_2$. Hence $q \models \langle\epsilon\rangle(\varphi_1\langle a\rangle\langle\epsilon\rangle\varphi_2)$.

  $\boldsymbol{n > 0}$ Since $p_n \xrightarrow{\tau} p_{n-1}$, according to Def. 2.1 there are two possibilities.

  (i) Either $p_{n-1} \leftrightarroweq_\eta q$. Since $p_{n-1} \models \langle\epsilon\rangle(\varphi_1\langle a\rangle\langle\epsilon\rangle\varphi_2)$, by induction on n, $q \models \langle\epsilon\rangle(\varphi_1\langle a\rangle\langle\epsilon\rangle\varphi_2)$.

  (ii) Or $q \overset{\epsilon}{\Longrightarrow} q' \xrightarrow{\tau} q'' \overset{\epsilon}{\Longrightarrow} q'''$ with $p_{n-1} \leftrightarroweq_\eta q'''$. By induction on n, $p_{n-1} \models \langle\epsilon\rangle(\varphi_1\langle a\rangle\langle\epsilon\rangle\varphi_2)$, so $q''' \models \langle\epsilon\rangle(\varphi_1\langle a\rangle\langle\epsilon\rangle\varphi_2)$. Hence $q \models \langle\epsilon\rangle(\varphi_1\langle a\rangle\langle\epsilon\rangle\varphi_2)$.

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

($\Leftarrow$) We prove that $\sim_{\mathbb{O}_\eta}$ is an η-bisimulation relation. The relation is clearly symmetric. Let $p \sim_{\mathbb{O}_\eta} q$. Suppose $p \xrightarrow{\alpha} p'$. If $\alpha = \tau$ and $p' \sim_{\mathbb{O}_\eta} q$, then the first condition of Def. 2.1 is fulfilled. So we can assume that either (i) $\alpha \neq \tau$ or (ii) $p' \not\sim_{\mathbb{O}_\eta} q$. We define

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

For each $q' \in Q'$, let $\varphi_{q'}$ be a formula in $\mathbb{O}_\eta$ such that $p \models \varphi_{q'}$ and $q' \not\models \varphi_{q'}$. (Such a formula always exists because $\mathbb{O}_\eta$ 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 $\mathbb{O}_\eta$ such that $p' \models \psi_{q'''}$ and $q''' \not\models \psi_{q'''}$. We define

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

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

(i) $\alpha \neq \tau$. Since $p \models \langle\epsilon\rangle(\varphi\langle\alpha\rangle\langle\epsilon\rangle\psi) \in \mathbb{O}_\eta$ and $p \sim_{\mathbb{O}_\eta} q$, also $q \models \langle\epsilon\rangle(\varphi\langle\alpha\rangle\langle\epsilon\rangle\psi)$. Hence, $q \overset{\epsilon}{\Longrightarrow} q' \xrightarrow{\alpha} q'' \overset{\epsilon}{\Longrightarrow} q'''$ with $q' \models \varphi$ and $q''' \models \psi$. By the definition of φ and ψ it follows that $p \sim_{\mathbb{O}_\eta} q'$ and $p' \sim_{\mathbb{O}_\eta} q'''$.

(ii) $\alpha = \tau$ and $p' \not\sim_{\mathbb{O}_\eta} q$. Let $\hat{\varphi} \in \mathbb{O}_\eta$ such that $p' \models \hat{\varphi}$ and $q \not\models \hat{\varphi}$. Since $p \models \langle\epsilon\rangle(\hat{\varphi} \wedge \psi) \in \mathbb{O}_\eta$ and $p \sim_{\mathbb{O}_\eta} q$, also $q \models \langle\epsilon\rangle(\hat{\varphi} \wedge \psi)$. Since $q \not\models \hat{\varphi}$, $q \xrightarrow{\tau} q'' \overset{\epsilon}{\Longrightarrow} q'''$ with $q''' \models \hat{\varphi} \wedge \psi$. By the definition of ψ it follows that $p' \sim_{\mathbb{O}_\eta} q'''$.

Both cases imply that the second condition of Def. 2.1 is fulfilled. We therefore conclude that $\sim_{\mathbb{O}_\eta}$ is an η-bisimulation relation. $\square$

Using the first part of Thm. 2.5, which was proved above, it is not hard to derive the second part of Thm. 2.5, i.e. that $\mathbb{O}_{r\eta}$ is a modal characterisation of rooted η-bisimulation equivalence.