Rooted Branching Bisimulation as a Congruence*

Wan Fokkink
University of Wales Swansea
Department of Computer Science
Singleton Park, Swansea SA2 8PP, Wales
w.j.fokkink@swan.ac.uk

Abstract

This article presents a congruence format, in structural operational semantics, for rooted branching bisimulation equivalence. The format imposes additional requirements of Groote's ntyft format. It extends an earlier format by Bloom with standard notions such as recursion, iteration, predicates, and negative premises.

1 Introduction

Structural operational semantics [29] and denotational semantics [6] have evolved as the two standard methodologies to provide specification languages, programming languages, and process algebras with a semantics. In structural operational semantics, transitions between states are derived from inductive proof rules, called transition rules. Intuitively, validity of the positive premises and invalidity of the negative premises of a transition rule, under a certain substitution, implies validity of the conclusion of this rule under the same substitution. In this article, states are the closed terms generated by a single-sorted first-order signature, and transitions are supplied with action labels. Many of the semantic definitions in Plotkin style that have been defined over the years are within this scope.

Przymusinski [30] introduced three-valued stable models in order to give meaning to transition rules with negative premises. Such a model partitions the set of transitions into three disjoint sets: transitions are true, false, or in limbo. Przymusinski showed that each collection of transition rules has a three-valued stable model in which the set of transitions in limbo is maximal, and that this so-called least three-valued stable model coincides with the well-founded semantics of van Gelder, Ross, and Schlipf [14]. A collection of transition rules is complete [21] (or positive after reduction [10]) if all transitions in its least three-valued stable model are either true or false.

Labelled transition systems can be distinguished by a wide range of behavioural equivalences [17, 18]. In the field of process algebra, four so-called weak equivalences have been developed to abstract away from internal machine behaviour, represented by a silent step τ: observation [27], branching [22], delay [26], and η [3]. This article focuses on branching bisimulation equivalence, in which one can abstract away from

*Sponsored in part by a grant from The Nuffield Foundation

an action τ if its execution does not implicate the loss of possible behaviour. In recent years, this equivalence has been used in a sizeable number of verifications in process algebra. See [20] for a lucid exposition on the motivations behind the definition of branching bisimulation equivalence.

In general, the behavioural equivalence induced by a collection of transition rules is not a congruence; that is, the equivalence class of a term $f(p_1, ..., p_n)$ need not be determined by the equivalence classes of its arguments $p_1, ..., p_n$. Congruence is an important property to fit the behavioural equivalence into an axiomatic framework. Congruence formats in structural operational semantics have been developed for a number of behavioural equivalences, to avoid repetitive congruence proofs, and to explore the boundaries for transition rules that constitute sensible semantic definitions. For strong bisimulation equivalence [28], Groote [23] defined the ntyft format (extending the earlier GSOS [9] and tyft [24] formats), which incorporates negative premises. Bol and Groote [10] proved that if a collection of transition rules is complete and in the ntyft format, then the strong bisimulation relation induced by its least three-valued stable model is a congruence. (They needed a well-foundedness requirement, which was later shown to be redundant [12].) Baeten and Verhoef [4, 34] extended the tyft and ntyft formats with predicates, to obtain the path and panth formats, respectively.

Bloom [8] and Ulidowski and Phillips [32] introduced congruence formats for branching bisimulation equivalence. However, the transition rules for several standard operators in process algebra (most notably alternative composition) are outside the scope of such formats, because they do not satisfy the congruence property with respect to weak equivalences. Milner [27] showed that this imperfection can often be remedied by the introduction of a rootedness condition. Bloom [8] presented a congruence format for rooted branching bisimulation equivalence, called RBB cool, which imposes additional requirements on the GSOS format. It recognizes so-called patience rules for arguments i of functions symbols f, which imply that a term $f(p_1, ..., p_n)$ inherits the τ-transitions of its argument p_i. The RBB cool format does not allow negative premises; it is stated that "negative rules seem incompatible with weak process equivalences" [8, p. 32].

This paper presents a new congruence format for rooted branching bisimulation equivalence, called RBB safe, which imposes additional requirements on the panth format. It incorporates negative premises and relaxes several syntactic restrictions of the RBB cool format. Following Bloom [8], we require the presence of patience rules. Furthermore, certain arguments of function symbols in right-hand sides of conclusions of transition rules are labelled 'wild'; this labelling is used to restrict occurrences of variables in the right-hand sides of conclusions and in the left-hand sides of premises of transition rules. We prove that if a collection of transition rules is complete and satisfies the syntactic restrictions of the RBB safe format, then the rooted branching bisimulation relation induced by its least three-valued stable model is a congruence.

Section 3.1 presents the syntactic restrictions of the RBB safe format, and Section 3.2 formulates an efficient algorithm to compute a wild labelling of arguments of function symbols, against which these restrictions can be checked. A detailed comparison between RBB cool and RBB safe is given in Section 3.3. Our generalizations of the RBB cool format provide answers to three of the open question that were posed by Bloom in the conclusion of his paper. We give several examples of operators from the

literature that are RBB safe but not RBB cool: recursion [16], iteration [25], empty process [35], and a weaker version of the priority operator [1]. Section 3.4 contains counter-examples to show that the syntactic requirements of the RBB safe format are all essential. Finally, Section 3.5 presents the proof of the congruence theorem.

Fokkink [11] presented a congruence format for language equivalence, which labels arguments of function symbols wild in a similar fashion as in the RBB safe format. Van Glabbeek [19] sketched congruence formats for ready simulation, ready trace, failure trace, and trace equivalence. The expressivity of the RBB safe format is incomparable with each of those formats.

2 Preliminaries

2.1 Terms

Definition 2.1 *A* signature Σ *consists of*

- *an infinite set of variables* $x, y, z, ...$;
- *a set* $\mathcal{F}$ *of function symbols* $f, g, h, ...$, *where each function symbol* f *has an arity* $ar(f)$.

A function symbol of arity zero is called a constant.

Definition 2.2 *Let* Σ *be a signature. The collection* $\mathbb{T}(\Sigma)$ *of* (open) terms $p, q, r, s, t, u, v, w, ...$ *over* Σ *is defined as the least set satisfying:*

- *each variable is in* $\mathbb{T}(\Sigma)$,
- *if* $t_1, ..., t_{ar(f)} \in \mathbb{T}(\Sigma)$, *then* $f(t_1, ..., t_{ar(f)}) \in \mathbb{T}(\Sigma)$.

A term is closed *if it does not contain variables. The set of closed terms is denoted by* $\mathcal{T}(\Sigma)$.

Definition 2.3 *A* substitution *is a mapping* $\sigma : \mathcal{V} \to \mathcal{T}(\Sigma)$. *A substitution extends to a mapping from open terms to closed terms as usual; the term* $\sigma(t)$ *is obtained by replacing occurrences of variables* x *in* t *by* $\sigma(x)$.

2.2 Transition Rules

We introduce the basic notions of structural operational semantics. We assume a signature Σ, a set of transition labels $a, b, c, ...$, and a set $\mathcal{P}$ of predicates $P, Q, ...$.

Definition 2.4 *Let* $t, t' \in \mathbb{T}(\Sigma)$, *a a transition label, and* P *a predicate symbol.*

- *Expressions* $t \xrightarrow{a} t'$ *and* tP *are* positive transitions.
- *Expressions* $t \not\xrightarrow{a}$ *and* $t\neg P$ *are* negative transitions.

A transition is closed *if its terms are.*

Closed transitions are derived by means of transition rules.

Definition 2.5 *A* transition rule *is an expression of the form H/π, where H is a collection of positive and negative transitions, called the* premises*, and π is a positive transition, called the* conclusion*. The left-hand side and the right-hand side (if present) of the conclusion are called the* source *and the* target*, respectively.*

A transition system specification *(TSS) is a collection of transition rules.*

Definition 2.6 *A* proof *from a TSS T of a closed transition rule H/π consists of an upwardly branching tree in which all upward paths are finite, where the nodes of the tree are labelled by positive and negative transitions such that:*

- *the root has label π,*
- *if some node has label ℓ, and K is the set of labels of nodes directly above this node, then*
  1. *either $K = \emptyset$ and $\ell \in H$,*
  2. *or K/ℓ is a closed substitution instance of a transition rule in T.*

2.3 Three-Valued Stable Models

We use the *least three-valued stable model*, introduced by Baeten, Bergstra, Klop, and Weijland [2] in term rewriting and by Przymusinski [30] in logic programming, to give a semantics to TSSs with negative premises. A three-valued stable model partitions the collection of closed positive transitions into three disjoint sets: the set $\mathtt{C}$ of closed transitions that are *certainly* true, the set $\mathtt{F}$ of closed transitions that are *false*, and the set of remaining closed transitions for which it is *unknown* whether or not they are true. Such a partitioning (which is determined by $\langle \mathtt{C}, \mathtt{F} \rangle$) constitutes a three-valued stable model for TSS T if:

- a closed positive transition π is in $\mathtt{C}$ if and only if there exist a transition rule ρ in T and a substitution σ such that the conclusion of $\sigma(\rho)$ is π, and
  - for each positive premise $s \xrightarrow{a} s'$ of ρ, $\sigma(s) \xrightarrow{a} \sigma(s') \in \mathtt{C}$;
  - for each positive premise tP of ρ, $\sigma(t)P \in \mathtt{C}$;
  - for each negative premise $s \not\xrightarrow{a}$ of ρ and each $s' \in \mathcal{T}(\Sigma)$, $\sigma(s) \xrightarrow{a} s' \in \mathtt{F}$;
  - for each negative premise $t\neg P$ of ρ, $\sigma(t)P \in \mathtt{F}$;
- a closed positive transition π is *not* in $\mathtt{F}$ if and only if there exist a transition rule ρ in T and a substitution σ such that the conclusion of $\sigma(\rho)$ is π, and
  - for each positive premise $s \xrightarrow{a} s'$ of ρ, $\sigma(s) \xrightarrow{a} \sigma(s') \notin \mathtt{F}$;
  - for each positive premise tP of ρ, $\sigma(t)P \notin \mathtt{F}$;
  - for each negative premise $s \not\xrightarrow{a}$ of ρ and each $s' \in \mathcal{T}(\Sigma)$, $\sigma(s) \xrightarrow{a} s' \notin \mathtt{C}$;
  - for each negative premise $t\neg P$ of ρ, $\sigma(t)P \notin \mathtt{C}$.

Clearly $\mathtt{C}$ and $\mathtt{F}$ are disjoint.

Each TSS T allows a least three-valued stable model $\langle \mathtt{C}, \mathtt{F} \rangle$, in the sense that the sets $\mathtt{C}$ and $\mathtt{F}$ are minimal. Gelfond and Lifschitz [15] studied *two-valued* stable models, which are three-valued stable models for which the set $\mathtt{U}$ of unknown closed positive transitions is empty. Van Glabbeek [21] introduced the notion of a *complete* TSS.

Definition 2.7 *A TSS is* complete *if its least three-valued stable model is a two-valued stable model.*

If a TSS is complete, then it allows only one three-valued stable model. A TSS that does not contain rules with negative premises is always complete; see [21].

2.4 Rooted Branching Bisimulation

In the sequel we assume that the set of transition labels contains a special element τ. The reflexive-transitive closure of the relation $\xrightarrow{\tau}$ is denoted by $\xrightarrow{\varepsilon}$.

Assuming a collection of closed positive transitions $\mathtt{C}$, we define the notion of a branching bisimulation equivalence [22].

Definition 2.8 *A binary relation B over $\mathcal{T}(\Sigma)$ is a* branching bisimulation *with respect to $\mathtt{C}$ if it is symmetric and, whenever $s\,B\,t$,*

if $s \xrightarrow{a} s' \in \mathtt{C}$, then either

1. *$a = \tau$ and $s'\,B\,t$, or*
2. *$t \xrightarrow{\varepsilon} t' \xrightarrow{a} t'' \in \mathtt{C}$ for some t' and t'' such that $s\,B\,t'$ and $s'\,B\,t''$.*

if $sP \in \mathtt{C}$, then $t \xrightarrow{\varepsilon} t'P \in \mathtt{C}$ for some t' such that $s\,B\,t'$.

$s, t \in \mathcal{T}(\Sigma)$ are branching bisimilar *with respect to $\mathtt{C}$, denoted by $s \leftrightarrows_b t$, if there exists a branching bisimulation relation B such that $s\,B\,t$.*

Branching bisimulation is an equivalence relation; see [7].

Definition 2.9 *An equivalence relation R on $\mathcal{T}(\Sigma)$ is called a* congruence *if $s_i R t_i$ for $i = 1, ..., ar(f)$ implies $f(s_1, ..., s_{ar(f)}) R f(t_1, ..., t_{ar(f)})$.*

Branching bisimulation equivalence is not a congruence with respect to standard process algebras. Therefore, we introduce a rootedness condition.

Definition 2.10 *A binary relation R over $\mathcal{T}(\Sigma)$ is a* rooted branching bisimulation *with respect to $\mathtt{C}$ if it is symmetric and, whenever $s\,R\,t$,*

1. *if $s \xrightarrow{a} s' \in \mathtt{C}$, then $t \xrightarrow{a} t' \in \mathtt{C}$ for some t' such that $s' \leftrightarrows_b t'$;*
2. *if $sP \in \mathtt{C}$, then $tP \in \mathtt{C}$.*

$s, t \in \mathcal{T}(\Sigma)$ are rooted branching bisimilar *with respect to $\mathtt{C}$, denoted by $s \leftrightarrows_{rb} t$, if there exists a rooted branching bisimulation relation R such that $s\,R\,t$.*

Since branching bisimulation is an equivalence relation, it is easy to see that rooted branching bisimulation is also an equivalence relation.

2.5 Panth Rules

We present the panth format [34], whereby it is required that the source is not a single variable (i.e., we only consider the ntyft component of the panth format).

Definition 2.11 *A transition rule is a* panth *rule if it is of the form*

$$\frac{\{s_j \xrightarrow{a_j} y_j \,|\, j \in J\} \cup \{t_k P_k \,|\, k \in K\} \cup \{p_\ell \not\xrightarrow{b_\ell} \,|\, \ell \in L\} \cup \{q_m \neg Q_m \,|\, m \in M\}}{f(x_1, ..., x_{ar(f)}) \xrightarrow{a} r}$$

or

$$\frac{\{s_j \xrightarrow{a_j} y_j \,|\, j \in J\} \cup \{t_k P_k \,|\, k \in K\} \cup \{p_\ell \not\xrightarrow{b_\ell} \,|\, \ell \in L\} \cup \{q_m \neg Q_m \,|\, m \in M\}}{f(x_1, ..., x_{ar(f)})P}$$

where the $x_1, ..., x_{ar(f)}$ and the y_j for $j \in J$ are all distinct variables.

A panth rule without negative premises is called a path *rule.*

3 Rooted Branching Bisimulation as a Congruence

3.1 The RBB Safe Format

We assume a signature Σ, and use $C[]$ to denote a context, being a term with one occurrence of the context symbol $[]$.

In the sequel we assume that each arguments of each function symbol is labelled either *tame* or *wild*. A context is said to be w-nested if the context symbol occurs inside a nested string of wild arguments.

Definition 3.1 *The collection of* w-nested *contexts is defined inductively by:*

1. *$[]$ is w-nested;*
2. *if $C[]$ is w-nested, and argument i of function symbol f is wild, then*

$$f(t_1, ..., t_{i-1}, C[], t_{i+1}, ..., t_{ar(f)})$$

is w-nested.

Definition 3.2 *A* patience *rule for the i-th argument of a function symbol f is a path rule of the form*

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

Definition 3.3 *A TSS T is called* RBB safe*, with respect to a tame/wild labelling of arguments of function symbols, if each of its transition rules is*

1. *either a patience rule for a wild argument of a function symbol,*
2. *or a panth rule, with source $f(x_1, ..., x_{ar(f)})$ and right-hand sides of premises $\{y_j \,|\, j \in J\}$, such that the following requirements are fulfilled.*

- *Variables y_j for $j \in J$ do not occur in left-hand sides of premises.*
- *If argument i of f is wild, and does not have a patience rule in T, then x_i does not occur in left-hand sides of premises.*
- *If argument i of f is wild, and has a patience rule in T, then x_i occurs no more than once in the left-hand side of a premise, whereby this premise*
  - *is positive,*
  - *does not contain the relation $\xrightarrow{\tau}$, and*
  - *has left-hand side x_i.*
- *Variables y_j for $j \in J$ and variables x_i for i a wild argument of f only occur at w-nested positions in the target.*

Theorem 3.4 *If a complete TSS is RBB safe, then the rooted branching bisimulation equivalence that it induces is a congruence.*

A formal proof of Theorem 3.4 is presented in Section 3.5.

3.2 Construction of Tame/Wild Labels

Assuming that a complete TSS T consists of a finite number of transition rules, which each have finitely many premises, it is easy to verify whether the rules in T are panth. Moreover, given a tame/wild labelling of arguments of function symbols, it is easy to check whether each rule satisfies the restrictions as imposed by the RBB safe format in Definition 3.3. The crux in determining whether T is RBB safe is to find a suitable tame/wild labelling of arguments of function symbols. Assuming that the collection $\mathcal{F}$ of function symbols is finite, there exists an efficient procedure to compute a tame/wild labelling Λ such that (T, Λ) is RBB safe if and only if there exists a labelling Λ' such that (T, Λ') is RBB safe.

Procedure "Compute Wild Labels for $(\mathcal{F}, ar)$ and T":

The red/green directed graph G consists of vertices $\langle f, i\rangle$ for $f \in \mathcal{F}$ and $1 \leq i \leq ar(f)$. There is an edge from $\langle f, i\rangle$ to $\langle g, j\rangle$ in G if and only if there is a transition rule in T with its conclusion of the form

$$f(x_1, ..., x_{ar(f)}) \xrightarrow{a} C[g(t_1, ..., t_{j-1}, C'[x_i], t_{j+1}, ..., t_{ar(g)})].$$

A vertex $\langle g, j\rangle$ is red if and only if there is a transition rule in T with its target of the form

$$C[g(t_1, ..., t_{j-1}, C'[y], t_{j+1}, ..., t_{ar(g)})]$$

whereby y is the right-hand side of a premise of this rule. All other vertices in G are coloured green.

The procedure colours green vertices in G red as follows. If a vertex $\langle f, i\rangle$ is red, and there exists an edge in G from $\langle f, i\rangle$ to a green vertex $\langle g, j\rangle$, then $\langle g, j\rangle$ is coloured red.

The procedure terminates if none of the green vertices can be coloured red anymore, at which it outputs the red/green directed graph.

Λ labels an argument i of a function symbol f 'wild' if and only if the vertex $\langle f, i\rangle$ in the output graph of the procedure above is red.

3.3 Applications

The RBB safe format is strictly more liberal than Bloom's *simply* RBB cool format [8]. The RBB cool format is more restrictive than the RBB safe format in the following respects.

1. It requires that transition rules are GSOS [9], so that the left-hand sides of the premises in a transition rule must be distinct variables.
2. It does not incorporate predicates.
3. It requires that *all* arguments of function symbols that occur in the target of a transition rule are wild (so that occurrences of variables in targets are by default w-nested). This results in more severe syntactic restrictions on left-hand sides of positive premises.
4. It does not incorporate negative premises.
5. It only allows the relation $\xrightarrow{\tau}$ to occur in premises of patience rules.

The incorporation of predicates and negative premises in the RBB safe format provides affirmative answers to the first two open questions in the conclusion of [8].

Remark. The RBB cool format relaxes the simply RBB cool format, by allowing bifurcation rules [8, Def. 5.1] instead of patience rules. The definition of bifurcation rules is deplorably complicated, Bloom does not present examples of transition rules that benefit from this generalization, and we do not know of any examples from the literature that are RBB cool but not simply RBB cool. Therefore we refrain from this generalization here.

We present several TSSs from process algebra that are RBB safe, but not RBB cool, due to the distinctions between these formats as discussed above. The examples provide a negative answer to the fourth open question in the conclusion of [8].

3.3.1 Basic Process Algebra with Silent Step and Empty Process

The signature of basic process algebra (BPA) [5] consists of a collection A of constants, called atomic actions, together with two binary function symbols: the alternative composition $s + t$ executes either s or t, and the sequential composition $s \cdot t$ executes first s and then t. Furthermore, we add two special constants to the syntax: τ together with the empty process ϵ [35]. The latter constant terminates successfully, which is expresses by the predicate $\downarrow$. The set of transition labels consists of $A \cup \{\tau\}$. The intuitions above are made precise in the operational semantics of BPA, which is presented in Table 1, whereby the a ranges over $A \cup \{\tau\}$.

The TSS in Table 1 is in the path format. The procedure in Section 3.2 calculates the following tame/wild labelling: the first argument of sequential composition is wild (because of the target $y \cdot x_2$ in the third rule for sequential composition), and both arguments of alternative composition and the second argument of sequential composition are tame. The TSS in Table 1 is RBB safe with respect to this tame/wild labelling:

$$a \xrightarrow{a} \epsilon \qquad \epsilon \downarrow$$

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

$$\frac{x_1 \downarrow \quad x_2 \downarrow}{x_1 \cdot x_2 \downarrow} \qquad \frac{x_1 \downarrow \quad x_2 \xrightarrow{a} y}{x_1 \cdot x_2 \xrightarrow{a} y} \qquad \frac{x_1 \xrightarrow{a} y}{x_1 \cdot x_2 \xrightarrow{a} y \cdot x_2}$$

Table 1: Transition Rules for $\mathrm{BPA}_{\epsilon\tau}$

- the third rule for sequential composition with $a = \tau$ constitutes a patience rule for the first argument of sequential composition;
- in the first to rules for sequential composition, and in the third rule with $a \neq \tau$, the variable x_1 in the wild argument of the source occurs as the left-hand side of one positive premise which does not contain the relation $\xrightarrow{\tau}$;
- in the third rule for sequential composition, the variable y in the right-hand side of the premise occurs in a wild argument of the target.

Corollary 3.5 *Rooted branching bisimulation is a congruence with respect to $BPA_{\epsilon\tau}$.*

The transition rules in Table 2 are not RBB cool, due to the fact that some of them involve the predicate $\downarrow$. However, we observe that the RBB cool format can be extended with predicates in a trivial manner. The following three sections present more serious examples of RBB safe TSSs that violate Bloom's RBB cool format.

3.3.2 Recursion

Given a signature Σ, a recursive specification E is a finite set of equations $\{X_i = t_i \mid i = 1, ..., n\}$, 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 path rules for recursion in Table 2, which originate from [16]. The expression E in these transition 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 expressions $\langle X|E\rangle$ to be constants. It is easy to see that the two transition rules in Table 2 are RBB safe.

$$\frac{\langle t|E\rangle \downarrow}{\langle X|E\rangle \downarrow} \qquad \frac{\langle t|E\rangle \xrightarrow{a} y}{\langle X|E\rangle \xrightarrow{a} y}$$

Table 2: Transition Rules for Recursion

Corollary 3.6 *Rooted branching bisimulation is a congruence with respect to $BPA_{\epsilon\tau}$ with recursion.*

The transition rules in Table 2 are not RBB cool, due to the fact that the left-hand sides of their premises are not single variables (and because the rules involve the predicate $\downarrow$).

3.3.3 Iteration

Iteration t^*, dating back to Kleene [25], either terminates successfully or executes $t \cdot t^*$. Two transition rules for iteration are presented in Table 3, which are added to the transition rules for $\mathrm{BPA}_{\epsilon\tau}$ in Table 1. In view of the procedure in Section 3.2 we take the argument of iteration to be tame. Furthermore, as before, the first argument of sequential composition is wild, and the arguments of alternative composition and the second argument of sequential composition are tame. The path rules in Table 3 are RBB safe with respect to this tame/wild labelling: in the second rule, the right-hand side y of the premise occurs in a wild argument of the target.

$$x^* \downarrow \qquad \frac{x \xrightarrow{a} y}{x^* \xrightarrow{a} y \cdot x^*}$$

Table 3: Transition Rules for Iteration

Corollary 3.7 *Rooted branching bisimulation is a congruence with respect to $BPA_{\epsilon\tau}$ with iteration.*

The transition rules in Table 3 are not RBB cool. Namely, in the second transition rule for iteration, the iteration operator occurs in the target $y \cdot x^*$, and the argument of the source x^* occurs is the left-hand side of the premise, so the RBB cool format requires that there exists a patience rule for the argument of iteration. However, such a patience rule is not present in Table 3 nor in Table 1.

3.3.4 Weak Priority

Weak priority is a unary function symbol that assumes an ordering on labels. The term $\theta(t)$ executes the transitions of t, with the exception that an *initial* transition of t is blocked in $\theta(t)$ if there exists another initial transition of t with a greater label. This intuition is captured by the second transition rule in Table 4. The TSS that consists of the path rules in Table 1 together with the panth rules in Table 4 is complete (see Definition 2.7). The procedure in Section 3.2 labels the argument of weak priority tame. The panth rules in Table 4 are RBB safe with respect to this tame/wild labelling: in the second rule, the left-hand side x of the premises occurs in a tame argument of the source.

Corollary 3.8 *Rooted branching bisimulation is a congruence with respect to $BPA_{\epsilon\tau}$ with weak priority.*

$$\frac{x\downarrow}{\theta(x)\downarrow} \qquad \frac{x \xrightarrow{a} y \quad x \overset{b}{\not\rightarrow} \text{ for } a < b}{\theta(x) \xrightarrow{a} y}$$

Table 4: Transition Rules for Weak Priority

The second transition rule in Table 4 is not RBB cool, due to the fact that it contains negative premises.

Remark. We derived weak priority from the priority operator Θ, introduced by Baeten, Bergstra, and Klop [1]; in $\Theta(t)$ all transitions of t (so not only the initial ones) are blocked in $\Theta(t)$ by simultaneous transitions of t with a greater label. This is expressed by the rule

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

This transition rule is not RBB safe: in view of the target $\Theta(y)$, the procedure in Section 3.2 labels the argument of Θ wild; so the left-hand side x of the negative premises occurs in a wild argument of the source. In general, the priority operator Θ does not preserve rooted branching bisimulation equivalence (c.f. [33, pp. 130–132]).

3.4 Counter-Examples

We give a string of examples of TSSs in the panth format, to show that the additional syntactic restrictions of the RBB safe format are essential. The first example shows that the restriction to complete TSSs cannot be relaxed to TSSs that have a unique (not necessarily least) two-valued stable model. (The example is derived from Example 8.12 in [10].)

Example 3.9 *Suppose that a and b are constants, f is a unary function symbol with a tame argument, and P, Q_1, and Q_2 are predicates. The panth rules*

$$aP \qquad bP \qquad \frac{xP \quad f(x)\neg Q_1 \quad f(a)\neg Q_2}{f(x)Q_2} \qquad \frac{xP \quad f(x)\neg Q_2 \quad f(b)\neg Q_1}{f(x)Q_1}$$

satisfy the syntactic criteria of the RBB safe format. They induce a unique two-valued stable model, in which $\{aP, bP, f(a)Q_1, f(b)Q_2\}$ constitutes the set of closed positive transitions that are certainly true. Their least three-valued stable model, however, has a non-empty set of unknown transitions: $\{f(a)Q_1, f(a)Q_2, f(b)Q_1, f(b)Q_2\}$.

Clearly, $a \leftrightarrows_{rb} b$, but $f(a)$ and $f(b)$ are not *rooted branching bisimilar with respect to the two-valued stable model.*

In the remaining examples we assume the syntax and semantics of $\mathrm{BPA}_{\epsilon\tau}$, whereby the set A of atomic actions consists of a and b. Furthermore, we assume a unary function symbol f and a predicate P.

The next counter-example shows that the RBB safe format cannot allow the right-hand side of a premise to occur in the left-hand side of a premise.

Example 3.10 *Let the argument of f be tame. We extend the transition rules in Table 1 with*

$$\frac{x \xrightarrow{a} y \quad y \xrightarrow{b} z}{f(x)P}$$

Note that y is both the right-hand side of the first premise and the left-hand side of the second premise.

Clearly, $a \cdot b \leftrightarrows_{rb} a \cdot \tau \cdot b$. However, $f(a \cdot b)$ and $f(a \cdot \tau \cdot b)$ are not *rooted branching bisimilar, because $f(a \cdot b)P$ holds while $f(a \cdot \tau \cdot b)P$ does not hold.*

The next counter-example shows that the RBB safe format cannot allow a wild argument of the source to occur as the left-hand side of a negative premise, even if there exists a patience rule for this argument. (An example from the literature of a violation of this requirement is the operational semantics of the priority operator [1]; see Section 3.3.4.)

Example 3.11 *Let the argument of f be wild. We extend the transition rules in Table 1 with*

$$\frac{x \xrightarrow{\tau} y}{f(x) \xrightarrow{\tau} f(y)} \qquad \frac{x \not\xrightarrow{a}}{f(x)P}$$

The resulting TSS is complete. The first rule is a patience rule for the argument of f. Note that, in the second rule, the wild argument x of the source is the left-hand side of the negative premise.

Clearly, $\tau \cdot a \leftrightarrows_{rb} \tau \cdot \tau \cdot a$. However, $f(\tau \cdot a)$ and $f(\tau \cdot \tau \cdot a)$ are not *rooted branching bisimilar, because no execution sequence of $f(\tau \cdot a)$ matches $f(\tau \cdot \tau \cdot a) \xrightarrow{\tau} f(\tau \cdot a)P$.*

The next counter-example shows that the RBB safe format cannot allow a wild argument of the source to occur as the left-hand side of a positive premise with relation symbol $\xrightarrow{\tau}$, even if there exists a patience rule for this argument.

Example 3.12 *Let the argument of f be wild. We extend the transition rules in Table 1 with*

$$\frac{x \xrightarrow{\tau} y}{f(x) \xrightarrow{\tau} f(y)} \qquad \frac{x \xrightarrow{\tau} y}{f(x)P}$$

The first rule is a patience rule for the argument of f. Note that, in the second rule, the wild argument x of the source is the left-hand side of the negative premise.

Clearly, $\tau \cdot a \leftrightarrows_{rb} \tau \cdot \tau \cdot a$. However, $f(\tau \cdot a)$ and $f(\tau \cdot \tau \cdot a)$ are not *rooted branching bisimilar, because no execution sequence of $f(\tau \cdot a)$ matches $f(\tau \cdot \tau \cdot a) \xrightarrow{\tau} f(\tau \cdot a)P$.*

The next counter-example shows that the RBB safe format can only allow a wild argument of the source to occur as the left-hand side of a positive premise if there exists a patience rule for this argument.

Example 3.13 *Let the argument of f be wild. We extend the transition rules in Table 1 with*

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

Note that there is no patience rule for the argument of f, and that in the second rule the wild argument x of the source is the left-hand side of the premise.

Clearly, $a \cdot b \leftrightarrows_{rb} a \cdot \tau \cdot b$. However, $f(a \cdot b)$ and $f(a \cdot \tau \cdot b)$ are not *rooted branching bisimilar, because no execution sequence of $f(a \cdot \tau \cdot b)$ matches $f(a \cdot b) \xrightarrow{a} f(b)P$.*

Finally, if in Examples 3.11–3.13 the argument of f were defined to be tame, then the examples would violate the RBB safe format due to the fact that, in the first rule of each example, the right-hand side y of the premise does not occur at a w-nested position in the target. This shows that the RBB safe format can only allow right-hand sides of premises to occur at w-nested positions in the target. A similar counter-example can be given to show that wild arguments of the source may only occur at w-nested positions in the target.

3.5 Proof of the Congruence Theorem

Proof of Theorem 3.4. Let the complete TSS T be RBB safe. The least three-valued stable model for T induces a branching equivalence $\leftrightarrows_b$ (Definition 2.8), and thus a rooted branching equivalence $\leftrightarrows_{rb}$ (Definition 2.10). Let R be the least binary relation over $\mathcal{T}(\Sigma)$ that satisfies:

- $s \, R \, t$ if $s \leftrightarrows_{rb} t$;
- $f(s_1, ..., s_{ar(f)}) \, R \, f(t_1, ..., t_{ar(f)})$ if $s_i \, R \, t_i$ for $i = 1, ..., ar(f)$.

Furthermore, let B be the least binary relation over $\mathcal{T}(\Sigma)$ that satisfies:

- $s \, B \, t$ if $s \leftrightarrows_b t$;
- $f(s_1, ..., s_{ar(f)}) \, B \, f(t_1, ..., t_{ar(f)})$ if
  - $s_i \, R \, t_i$ for tame arguments i of f, and
  - $s_i \, B \, t_i$ for wild arguments i of f.

Since $\leftrightarrows_{rb}$ and $\leftrightarrows_b$ are symmetric, the same holds for R and B. We show that R is a rooted branching bisimulation relation and that B is a branching bisimulation relation. The next two lemmas follow by structural induction with respect to $t \in \mathbb{T}(\Sigma)$, using the definitions of R and B.

Lemma 3.14 *If $\sigma(x) \, R \, \sigma'(x)$ for all variables x in t, then $\sigma(t) \, R \, \sigma'(t)$.*

Lemma 3.15 *If for all variables x in t,*

- *either $\sigma(x) \, R \, \sigma'(x)$,*
- *or $\sigma(x) \, B \, \sigma'(x)$ and x only occurs at w-nested positions in t,*

then $\sigma(t) \, B \, \sigma'(t)$.

We construct pairs of disjoint sets of positive transitions $\langle \mathtt{C}_\alpha, \mathtt{F}_\alpha \rangle$ for ordinals α, using ordinal induction, such that these pairs converge to the least three-valued stable model for T. $\langle \mathtt{C}_\alpha, \mathtt{F}_\alpha \rangle$ is constructed from $\langle \mathtt{C}_\beta, \mathtt{F}_\beta \rangle$ for $\beta < \alpha$. Intuitively, $\mathtt{C}_\alpha$ and $\mathtt{F}_\alpha$ consist of the closed positive transitions that can be shown to be certainly true and false, respectively, in α steps.

- $\pi \in \mathtt{C}_\alpha$ if and only if there exist a transition rule ρ in T and a substitution σ such that the conclusion of $\sigma(\rho)$ is π, and
  - for each positive premise $s \xrightarrow{a} s'$ of ρ, $\sigma(s) \xrightarrow{a} \sigma(s') \in \mathtt{C}_\beta$ for some $\beta < \alpha$;
  - for each positive premise tP of ρ, $\sigma(t)P \in \mathtt{C}_\beta$ for some $\beta < \alpha$;
  - for each negative premise $s \xrightarrow{a}\!\!\!\!\!/\;$ and each $s' \in \mathcal{T}(\Sigma)$, $\sigma(s) \xrightarrow{a} s' \in \mathtt{F}_\beta$ for some $\beta < \alpha$;
  - for each negative premise $t\neg P$ of ρ, $\sigma(t)P \in \mathtt{F}_\beta$ for some $\beta < \alpha$.
- $\pi \notin \mathtt{F}_\alpha$ if and only if there exist a transition rule ρ in T and a substitution σ such that the conclusion of $\sigma(\rho)$ is π, and
  - for each positive premise $s \xrightarrow{a} s'$ of ρ, $\sigma(s) \xrightarrow{a} \sigma(s') \notin \mathtt{F}_\beta$ for all $\beta < \alpha$;
  - for each positive premise tP of ρ, $\sigma(t)P \notin \mathtt{F}_\beta$ for all $\beta < \alpha$;
  - for each negative premise $s \xrightarrow{a}\!\!\!\!\!/\;$ of ρ and each $s' \in \mathcal{T}(\Sigma)$, $\sigma(s) \xrightarrow{a} s' \notin \mathtt{C}_\beta$ for all $\beta < \alpha$;
  - for each negative premise $t\neg P$ of ρ, $\sigma(t)P \notin \mathtt{C}_\beta$ for all $\beta < \alpha$.

Note that $\mathtt{C}_0 = \emptyset$, $\mathtt{F}_0 = \emptyset$, and $\mathtt{C}_\alpha \cap \mathtt{F}_\alpha = \emptyset$. The following two inclusions can be derived for ordinals α and β with $\beta \leq \alpha$, by ordinal induction (c.f. [13]):

1. $\mathtt{C}_\beta \subseteq \mathtt{C}_\alpha$;
2. $\mathtt{F}_\beta \subseteq \mathtt{F}_\alpha$.

Owing to these two inclusions, the Knaster-Tarski theorem [31] yields that there exists an ordinal λ such that $\mathtt{C}_\lambda = \mathtt{C}_{\lambda+1}$ and $\mathtt{F}_\lambda = \mathtt{F}_{\lambda+1}$. It is easy to see that $\langle \mathtt{C}_\lambda, \mathtt{F}_\lambda \rangle$ is a three-valued stable model for T (owing to the definitions of $\pi \in \mathtt{C}_{\lambda+1}$ and $\pi \notin \mathtt{F}_{\lambda+1}$). Furthermore, if $\langle \mathtt{C}, \mathtt{F} \rangle$ is some three-valued stable model for T, then it follows by ordinal induction that $\mathtt{C}_\alpha \subseteq \mathtt{C}$ and $\mathtt{F}_\alpha \subseteq \mathtt{F}$ for all ordinals α, so in particular $\mathtt{C}_\lambda \subseteq \mathtt{C}$ and $\mathtt{F}_\lambda \subseteq \mathtt{F}$. Hence, $\langle \mathtt{C}_\lambda, \mathtt{F}_\lambda \rangle$ is the least three-valued stable model for T.

We prove the following four statements in parallel, using ordinal induction with respect to α. (The statements $\mathrm{I}_{\alpha,\beta}$ and II_α, and their proofs, are similar to the cases 1 and 2 in the proof of Lemma 8.9 in [10], which forms the basis of a congruence proof for complete TSSs in the ntyft/ntyxt format modulo strong bisimulation).

$\mathrm{I}_{\alpha,\beta}$. If $s \mathrel{R} t$ and $s \xrightarrow{a} s' \in \mathtt{C}_\beta$, then $t \xrightarrow{a} t' \notin \mathtt{F}_\alpha$ for some $t' \in \mathcal{T}(\Sigma)$.

$\mathrm{I}'_{\alpha,\beta}$. If $s \mathrel{R} t$ and $sP \in \mathtt{C}_\beta$, then $tP \notin \mathtt{F}_\alpha$.

II_α. If $s \mathrel{R} t$ and $s \xrightarrow{a} s' \in \mathtt{C}_\alpha$, then $t \xrightarrow{a} t' \in \mathtt{C}_\lambda$ for some $t' \in \mathcal{T}(\Sigma)$ with $s' \mathrel{B} t'$.

II'_α. If $s \mathrel{R} t$ and $sP \in \mathtt{C}_\alpha$, then $tP \in \mathtt{C}_\lambda$.

We assume that $\mathrm{I}_{\gamma,\beta}$-$\mathrm{I}'_{\gamma,\beta}$ and II_γ-II'_γ have already been proved for $\gamma < \alpha$. First we prove $\mathrm{I}_{\gamma,\beta}$-$\mathrm{I}'_{\gamma,\beta}$ in parallel, using ordinal induction with respect to β.

Proof of $\mathrm{I}_{\alpha,\beta}$. We assume that $\mathrm{I}_{\alpha,\gamma}$ and $\mathrm{I}'_{\alpha,\gamma}$ have already been proved for $\gamma < \beta$.

If $s \leftrightarrows_{rb} t$, then $\mathrm{I}_{\alpha,\beta}$ follows immediately from the definition of $\leftrightarrows_{rb}$ together with the facts that $\mathtt{C}_\beta \subseteq \mathtt{C}_\lambda$ and $\mathtt{F}_\alpha \cap \mathtt{C}_\lambda = \emptyset$. We focus on the case where $s = f(s_1, ..., s_{ar(f)})$ and $t = f(t_1, ..., t_{ar(f)})$, with $s_i \, R \, t_i$ for $i = 1, ..., ar(f)$. If $f(s_1, ..., s_{ar(f)}) \xrightarrow{a} s' \in \mathtt{C}_\beta$ for some $s' \in \mathcal{T}(\Sigma)$, then there exists a panth rule ρ in T of the form

$$\frac{\{u_j \xrightarrow{a_j} y_j \mid j \in J\} \cup \{v_k P_k \mid k \in K\} \cup \{p_\ell \xrightarrow{b_\ell} \not\rightarrow \mid \ell \in L\} \cup \{q_m \neg Q_m \mid m \in M\}}{f(x_1, ..., x_{ar(f)}) \xrightarrow{a} r}$$

and a substitution σ with $\sigma(x_i) = s_i$ for $i = 1, ..., ar(f)$ and $\sigma(r) = s'$, such that:

- for each $j \in J$, $\sigma(u_j) \xrightarrow{a_j} \sigma(y_j) \in \mathtt{C}_\gamma$ for some $\gamma < \beta$;
- for each $k \in K$, $\sigma(v_k)P_k \in \mathtt{C}_\gamma$ for some $\gamma < \beta$;
- for each $\ell \in L$ and $p' \in \mathcal{T}(\Sigma)$, $\sigma(p_\ell) \xrightarrow{b_\ell} p' \in \mathtt{F}_\gamma$ for some $\gamma < \beta$;
- for each $m \in M$, $\sigma(q_m)Q_m \in \mathtt{F}_\gamma$ for some $\gamma < \beta$.

We define a substitution σ', such that together with ρ it proves $f(t_1, ..., t_{ar(f)}) \xrightarrow{a} \sigma'(r) \notin \mathtt{F}_\alpha$.

1. $\sigma'(z) = \sigma(z)$ for $z \notin \{x_i \mid i = 1, ..., ar(f)\} \cup \{y_j \mid j \in J\}$.
2. $\sigma'(x_i) = t_i$ for $i = 1, ..., ar(f)$.
3. For $j_0 \in J$, $\sigma'(y_{j_0})$ is defined as follows. The RBB safe format enforces that u_{j_0} does not contain variables from $\{y_j \mid j \in J\}$, so $\sigma'(u_{j_0})$ is well-defined. Since $s_i \, R \, t_i$ for $i = 1, ..., ar(f)$, Lemma 3.14 implies $\sigma(u_{j_0}) \, R \, \sigma'(u_{j_0})$. Furthermore, $\sigma(u_{j_0}) \xrightarrow{a_{j_0}} \sigma(y_{j_0}) \in \mathtt{C}_\gamma$ for some $\gamma < \beta$, so $\mathrm{I}_{\alpha,\gamma}$ yields $\sigma'(u_{j_0}) \xrightarrow{a_{j_0}} w \notin \mathtt{F}_\alpha$ for some $w \in \mathcal{T}(\Sigma)$. We define $\sigma'(y_{j_0}) = w$, so that

$$\sigma'(u_{j_0}) \xrightarrow{a_{j_0}} \sigma'(y_{j_0}) \notin \mathtt{F}_\alpha. \tag{1}$$

$\sigma(z) \, R \, \sigma'(z)$ for $z \notin \{y_j \mid j \in J\}$, and the RBB safe format enforces that variables y_j for $j \in J$ do not occur in left-hand sides of premises of ρ, so by Lemma 3.14 we can draw the following three conclusions.

- $\sigma(v_k) \, R \, \sigma'(v_k)$ for $k \in K$.

  $\sigma(v_k)P_k \in \mathtt{C}_\gamma$ for some $\gamma < \beta$, so $\mathrm{I}'_{\alpha,\gamma}$ implies

$$\sigma'(v_k)P_k \notin \mathtt{F}_\alpha. \tag{2}$$

- $\sigma(p_\ell) \, R \, \sigma'(p_\ell)$ for $\ell \in L$.

  For each $p' \in \mathcal{T}(\Sigma)$ there is a $\gamma < \beta$ such that $\sigma(p_\ell) \xrightarrow{b_\ell} p' \in \mathtt{F}_\gamma$, and $\mathtt{F}_\gamma \cap \mathtt{C}_\lambda = \emptyset$, so II_δ for $\delta < \alpha$ implies

$$\sigma'(p_\ell) \xrightarrow{b_\ell} p' \notin \mathtt{C}_\delta \text{ for } p' \in \mathcal{T}(\Sigma). \tag{3}$$

- $\sigma(q_m)\, R\, \sigma'(q_m)$ for $m \in M$.

  There is a $\gamma < \beta$ such that $\sigma(q_m)Q_m \in \mathtt{F}_\gamma$, and $\mathtt{F}_\gamma \cap \mathtt{C}_\lambda = \emptyset$, so II'_δ for $\delta < \alpha$ implies

$$\sigma'(q_m)Q_m \notin \mathtt{C}_\delta. \tag{4}$$

(1)–(4) together imply that transition rule ρ together with substitution σ' prove $f(t_1, ..., t_{ar(f)}) \xrightarrow{a} \sigma'(r) \notin \mathtt{F}_\alpha$. $\Box$

Proof of $\mathbf{I}'_{\alpha,\beta}$. Similar to the proof of $\mathrm{I}_{\alpha,\beta}$.

Proof of $\mathbf{II}_\alpha$. If $s \leftrightarrows_{rb} t$, then II_α follows immediately from the definition of $\leftrightarrows_{rb}$ together with $\mathtt{C}_\alpha \subseteq \mathtt{C}_\lambda$ and $\leftrightarrows_b \subseteq B$. We focus on the case where $s = f(s_1, ..., s_{ar(f)})$ and $t = f(t_1, ..., t_{ar(f)})$, with $s_i\, R\, t_i$ for $i = 1, ..., ar(f)$. If $f(s_1, ..., s_{ar(f)}) \xrightarrow{a} s' \in \mathtt{C}_\alpha$, then there exists a panth rule ρ in T of the form

$$\frac{\{u_j \xrightarrow{a_j} y_j \mid j \in J\} \cup \{v_k P_k \mid k \in K\} \cup \{p_\ell \xrightarrow{b_\ell} \not\rightarrow \mid \ell \in L\} \cup \{q_m \neg Q_m \mid m \in M\}}{f(x_1, ..., x_{ar(f)}) \xrightarrow{a} r}$$

and a substitution σ with $\sigma(x_i) = s_i$ for $i = 1, ..., ar(f)$ and $\sigma(r) = s'$, such that:

- for each $j \in J$, $\sigma(u_j) \xrightarrow{a_j} \sigma(y_j) \in \mathtt{C}_\gamma$ for some $\gamma < \alpha$;
- for each $k \in K$, $\sigma(v_k)P_k \in \mathtt{C}_\gamma$ for some $\gamma < \alpha$;
- for each $\ell \in L$ and $p' \in \mathcal{T}(\Sigma)$, $\sigma(p_\ell) \xrightarrow{b_\ell} p' \in \mathtt{F}_\gamma$ for some $\gamma < \alpha$;
- for each $m \in M$, $\sigma(q_m)Q_m \in \mathtt{F}_\gamma$ for some $\gamma < \alpha$.

We define a substitution σ', such that together with ρ it proves $f(t_1, ..., t_{ar(f)}) \xrightarrow{a} \sigma'(r) \in \mathtt{C}_\lambda$ and $\sigma(r)\, B\, \sigma'(r)$.

1. $\sigma'(z) = \sigma(z)$ for $z \notin \{x_i \mid i = 1, ..., ar(f)\} \cup \{y_j \mid j \in J\}$.
2. $\sigma'(x_i) = t_i$ for $i = 1, ..., ar(f)$.
3. For $j_0 \in J$, $\sigma'(y_{j_0})$ is defined as follows. The RBB safe format enforces that u_{j_0} does not contain variables from $\{y_j \mid j \in J\}$, so $\sigma'(u_{j_0})$ is well-defined. Since $s_i\, R\, t_i$ for $i = 1, ..., ar(f)$, Lemma 3.14 implies $\sigma(u_{j_0})\, R\, \sigma'(u_{j_0})$. Furthermore, $\sigma(u_{j_0}) \xrightarrow{a_{j_0}} \sigma(y_{j_0}) \in \mathtt{C}_\gamma$ for some $\gamma < \alpha$, so II_γ yields $\sigma'(u_{j_0}) \xrightarrow{a_{j_0}} w \in \mathtt{C}_\lambda$ for some $w \in \mathcal{T}(\Sigma)$ with $\sigma(y_{j_0})\, B\, w$. We define $\sigma'(y_{j_0}) = w$, so that

$$\sigma(y_{j_0})\, B\, \sigma'(y_{j_0}) \tag{5}$$

$$\sigma'(u_{j_0}) \xrightarrow{a_{j_0}} \sigma'(y_{j_0}) \in \mathtt{C}_\lambda. \tag{6}$$

$\sigma(z)\, R\, \sigma'(z)$ for $z \notin \{y_j \mid j \in J\}$, and the RBB safe format enforces that variables y_j for $j \in J$ do not occur in left-hand sides of premises of ρ, so by Lemma 3.14 we can draw the following three conclusions.

- $\sigma(v_k)\, R\, \sigma'(v_k)$ for $k \in K$.

  $\sigma(v_k)P_k \in \mathtt{C}_\gamma$ for some $\gamma < \alpha$, so II'_γ implies

$$\sigma'(v_k)P_k \in \mathtt{C}_\lambda. \tag{7}$$

- $\sigma(p_\ell)\, R\, \sigma'(p_\ell)$ for $\ell \in L$.

  For each $p' \in \mathcal{T}(\Sigma)$ there is a $\gamma < \alpha$ such that $\sigma(p_\ell) \xrightarrow{b_\ell} p' \in \mathtt{F}_\gamma$, so $\mathrm{I}_{\gamma,\lambda}$ implies $\sigma'(p_\ell) \xrightarrow{b_\ell} p' \notin \mathtt{C}_\lambda$ for $p' \in \mathcal{T}(\Sigma)$, or in other words,

$$\sigma'(p_\ell) \xrightarrow{b_\ell} p' \in \mathtt{F}_\lambda \text{ for } p' \in \mathcal{T}(\Sigma). \tag{8}$$

- $\sigma(q_m)\, R\, \sigma'(q_m)$ for $m \in M$.

  There is a $\gamma < \alpha$ such that $\sigma(q_m)Q_m \in \mathtt{F}_\gamma$, so $\mathrm{I}'_{\gamma,\lambda}$ implies $\sigma'(q_m)Q_m \notin \mathtt{C}_\lambda$, or in other words,

$$\sigma'(q_m)Q_m \in \mathtt{F}_\lambda. \tag{9}$$

(6)–(9) together imply that transition rule ρ together with substitution σ' prove $f(t_1, ..., t_{ar(f)}) \xrightarrow{a} \sigma'(r) \in \mathtt{C}_{\lambda+1} = \mathtt{C}_\lambda$. By (5) we have $\sigma(y_j)\, B\, \sigma'(y_j)$ for $j \in J$, and the RBB safe format enforces that the variables y_j for $j \in J$ only occur at w-nested positions in r. Furthermore, $\sigma(z)\, R\, \sigma'(z)$ for $z \notin \{y_j \mid j \in J\}$, so Lemma 3.15 implies $\sigma(r)\, B\, \sigma'(r)$. □

Proof of II'_α. Similar to the proof of II_α.

We prove two more statements in parallel, using ordinal induction with respect to α.

III_α. If $s\, B\, t$ and $s \xrightarrow{a} s' \in \mathtt{C}_\alpha$, then either

1. $a = \tau$ and $s'\, B\, t$, or
2. $t \xrightarrow{\varepsilon} t' \xrightarrow{a} t'' \in \mathtt{C}_\lambda$ with $s\, B\, t'$ and $s'\, B\, t''$.

III'_α. If $s\, B\, t$ and $sP \in \mathtt{C}_\alpha$, then $t \xrightarrow{\varepsilon} t'P \in \mathtt{C}_\lambda$ with $s\, B\, t'$.

We assume that III_γ-III'_γ have already been proved for ordinals $\gamma < \alpha$.

Proof of III_α. If $s \leftrightarrows_b t$, then III_α follows immediately from the definition of $\leftrightarrows_b$ together with $\mathtt{C}_\alpha \subseteq \mathtt{C}_\lambda$ and $\leftrightarrows_b \subseteq B$. We focus on the case where $s = f(s_1, ..., s_{ar(f)})$ and $t = f(t_1, ..., t_{ar(f)})$, with $s_i\, R\, t_i$ for tame arguments i of f and $s_i\, B\, t_i$ for wild arguments i of f. If $f(s_1, ..., s_{ar(f)}) \xrightarrow{a} s' \in \mathtt{C}_\alpha$ for some $s' \in \mathcal{T}(\Sigma)$, then we can distinguish the following two cases.

- Case 1: $a = \tau$, and there exists a patience rule for a wild argument i_0 of f,

$$\frac{x_{i_0} \xrightarrow{\tau} y}{f(x_1, ..., x_{i_0}, ..., x_{ar(f)}) \xrightarrow{\tau} f(x_1, ..., y, ..., x_{ar(f)})}$$

  and a substitution σ with $\sigma(x_i) = s_i$ for $i = 1, ..., ar(f)$, $f(s_1, ..., \sigma(y), ..., s_{ar(f)}) = s'$, and $s_{i_0} \xrightarrow{\tau} \sigma(y) \in \mathtt{C}_\gamma$ for some $\gamma < \alpha$.

  Since $s_{i_0}\, B\, t_{i_0}$ and $s_{i_0} \xrightarrow{\tau} \sigma(y) \in \mathtt{C}_\gamma$, III_γ offers two possibilities.

- Case 1.1: $\sigma(y)\,B\,t_{i_0}$.
  Since i_0 is a wild argument of f, $s_i\,R\,t_i$ for tame arguments i of f, and $s_i\,B\,t_i$ for wild arguments i of f, the definition of B yields
  $$f(s_1,...,\sigma(y),...,s_{ar(f)})\,B\,f(t_1,...,t_{i_0},...,t_{ar(f)}),$$
  or in other words, $s'\,B\,t$.
- Case 1.2: $t_{i_0} \xrightarrow{\varepsilon} t' \xrightarrow{\tau} t'' \in \mathtt{C}_\lambda$ with $s_{i_0}\,B\,t'$ and $\sigma(y)\,B\,t''$.
  Since $t_{i_0} \xrightarrow{\varepsilon} t' \xrightarrow{\tau} t'' \in \mathtt{C}_\lambda$, the patience rule for argument i_0 of f yields
  $$f(t_1,...,t_{i_0},...,t_{ar(f)}) \xrightarrow{\varepsilon} f(t_1,...,t',...,t_{ar(f)}) \xrightarrow{\tau} f(t_1,...,t'',...,t_{ar(f)}) \in \mathtt{C}_\lambda.$$
  Since $s_{i_0}\,B\,t'$ and $\sigma(y)\,B\,t''$, and i_0 is a wild argument of f, the definition of B yields
  $$f(s_1,...,s_{i_0},...,s_{ar(f)})\,B\,f(t_1,...,t',...,t_{ar(f)})$$
  $$f(s_1,...,\sigma(y),...,s_{ar(f)})\,B\,f(t_1,...,t'',...,t_{ar(f)}).$$

- Case 2: There exists a panth rule ρ in T of the form
  $$\frac{\{u_j \xrightarrow{a_j} y_j \mid j \in J\} \cup \{v_k P_k \mid k \in K\} \cup \{p_\ell \overset{b_\ell}{\not\to} \mid \ell \in L\} \cup \{q_m \neg Q_m \mid m \in M\}}{f(x_1,...,x_{ar(f)}) \xrightarrow{a} r}$$
  and a substitution σ with $\sigma(x_i) = s_i$ for $i = 1,...,ar(f)$ and $\sigma(r) = s'$, such that:
  - for each $j \in J$, $\sigma(u_j) \xrightarrow{a_j} \sigma(y_j) \in \mathtt{C}_\gamma$ for some $\gamma < \alpha$;
  - for each $k \in K$, $\sigma(v_k)P_k \in \mathtt{C}_\gamma$ for some $\gamma < \alpha$;
  - for each $\ell \in L$ and $p' \in \mathcal{T}(\Sigma)$, $\sigma(p_\ell) \xrightarrow{b_\ell} p' \in \mathtt{F}_\gamma$ for some $\gamma < \alpha$;
  - for each $m \in M$, $\sigma(q_m)Q_m \in \mathtt{F}_\gamma$ for some $\gamma < \alpha$.

  We define a substitution σ', such that the patience rules for wild arguments of f together with ρ and σ' prove
  $$f(t_1,...,t_{ar(f)}) \xrightarrow{\varepsilon} \sigma'(f(x_1,...,x_{ar(f)})) \xrightarrow{a} \sigma'(r) \in \mathtt{C}_\lambda$$
  where $f(s_1,...,s_{ar(f)})\,B\,\sigma'(f(x_1,...,x_{ar(f)}))$ and $\sigma\,B\,\sigma'(r)$.
  1. $\sigma'(z) = \sigma(z)$ for $z \notin \{x_i \mid i = 1,...,ar(f)\} \cup \{y_j \mid j \in J\}$.
  2. $\sigma'(x_i) = t_i$ if i is a tame argument of f, or if x_i does not occur as the left-hand side of a premise of ρ.
  3. Suppose that $u_{j_0} \notin \{x_i \mid i \text{ a wild argument of } f\}$, for some $j_0 \in J$. The RBB safe format enforces that u_{j_0} does not contain variables from $\{y_j \mid j \in J\}$, so $\sigma'(u_{j_0})$ is well-defined. Since $s_i\,R\,t_i$ for tame arguments i of f, Lemma 3.14 implies $\sigma(u_{j_0})\,R\,\sigma'(u_{j_0})$. Furthermore, $\sigma(u_{j_0}) \xrightarrow{a_{j_0}} \sigma(y_{j_0}) \in \mathtt{C}_\gamma$ for some γ, so II_γ yields $\sigma'(u_{j_0}) \xrightarrow{a_{j_0}} w \in \mathtt{C}_\lambda$ for some $w \in \mathcal{T}(\Sigma)$ with $\sigma(y_{j_0})\,B\,w$. We define $\sigma'(y_{j_0}) = w$, so that
     $$\sigma(y_{j_0})\,B\,\sigma'(y_{j_0}) \tag{10}$$
     $$\sigma'(u_{j_0}) \xrightarrow{a_{j_0}} \sigma'(y_{j_0}) \in \mathtt{C}_\lambda. \tag{11}$$

4. Suppose that $v_{k_0} \notin \{x_i \mid i \text{ a wild argument of } f\}$, for some $k_0 \in K$. The RBB safe format enforces that v_{k_0} does not contain variables from $\{y_j \mid j \in J\}$, so $\sigma'(v_{k_0})$ is well-defined. Since $s_i \, R \, t_i$ for tame arguments i of f, Lemma 3.14 implies $\sigma(v_{k_0}) \, R \, \sigma'(v_{k_0})$. Furthermore, $\sigma(v_{k_0})P_{k_0} \in \mathtt{C}_\gamma$ for some $\gamma < \alpha$, so II'_γ yields

$$\sigma'(v_{k_0})P_{k_0} \in \mathtt{C}_\lambda. \tag{12}$$

5. Suppose that $u_{j_0} = x_{i_0}$ with i_0 a wild argument of f, for some $j_0 \in J$. The RBB safe format enforces that $a_{j_0} \neq \tau$, and that there is a patience rule for argument i_0 of f.

   $s_{i_0} \, B \, t_{i_0}$, $s_{i_0} \xrightarrow{a_{j_0}} \sigma(y_{j_0}) \in \mathtt{C}_\gamma$ for some $\gamma < \alpha$, and $a_{j_0} \neq \tau$, so the second option of III_γ implies $t_{i_0} \xrightarrow{\varepsilon} t' \xrightarrow{a_{j_0}} t'' \in \mathtt{C}_\lambda$ with $s_{i_0} \, B \, t'$ and $\sigma(y_{j_0}) \, B \, t''$. We define $\sigma'(x_{i_0}) = t'$ and $\sigma'(y_{j_0}) = t''$, so that

$$t_{i_0} \xrightarrow{\varepsilon} \sigma'(x_{i_0}) \in \mathtt{C}_\lambda \tag{13}$$

$$\sigma'(x_{i_0}) \xrightarrow{a_{j_0}} \sigma'(y_{j_0}) \in \mathtt{C}_\lambda \tag{14}$$

$$s_{i_0} \, B \, \sigma'(x_{i_0}) \tag{15}$$

$$\sigma(y_{j_0}) \, B \, \sigma'(y_{j_0}). \tag{16}$$

   Note that $\sigma'(x_{i_0})$ is uniquely defined, owing to the RBB safe restriction that x_{i_0} is the left-hand side of no more than one premise in ρ.

6. Suppose that $v_{k_0} = x_{i_0}$ with i_0 a wild argument of f, for some $k_0 \in K$. The RBB safe format enforces that there is a patience rule for argument i_0 of f.

   $s_{i_0} \, B \, t_{i_0}$ and $s_{i_0}P_{k_0} \in \mathtt{C}_\gamma$ for some $\gamma < \alpha$, so the second option of III'_γ implies $t_{i_0} \xrightarrow{\varepsilon} t'P_{k_0} \in \mathtt{C}_\lambda$ with $s_{i_0} \, B \, t'$. We define $\sigma'(x_{i_0}) = t'$, so that

$$t_{i_0} \xrightarrow{\varepsilon} \sigma'(x_{i_0})P_{k_0} \in \mathtt{C}_\lambda \tag{17}$$

$$\sigma'(x_{i_0})P_{k_0} \in \mathtt{C}_\lambda \tag{18}$$

$$s_{i_0} \, B \, \sigma'(x_{i_0}). \tag{19}$$

   Note that $\sigma'(x_{i_0})$ is uniquely defined, owing to the RBB safe restriction that x_{i_0} is the left-hand side of no more than one premise in ρ.

7. Fix an $\ell_0 \in L$. The RBB safe format enforces that p_{ℓ_0} does not contain variables from $\{x_i \mid i \text{ a wild argument of } f\} \cup \{y_j \mid j \in J\}$. Since $s_i \, R \, t_i$ for tame arguments i of f, Lemma 3.14 implies $\sigma(p_{\ell_0}) \, R \, \sigma'(p_{\ell_0})$. Furthermore, for each $p' \in \mathcal{T}(\Sigma)$ there is a $\gamma < \alpha$ such that $\sigma(p_{\ell_0}) \xrightarrow{b_{\ell_0}} p' \in \mathtt{F}_\gamma$, so $\mathrm{I}_{\gamma,\lambda}$ yields $\sigma'(p_{\ell_0}) \xrightarrow{b_{\ell_0}} p' \notin \mathtt{C}_\lambda$ for $p' \in \mathcal{T}(\Sigma)$. In other words,

$$\sigma'(p_{\ell_0}) \xrightarrow{b_{\ell_0}} p' \in \mathtt{F}_\lambda \text{ for } p' \in \mathcal{T}(\Sigma). \tag{20}$$

8. Fix an $m_0 \in M$. The RBB safe format enforces that q_{m_0} does not contain variables from $\{x_i \mid i \text{ a wild argument of } f\} \cup \{y_j \mid j \in J\}$. Since $s_i \, R \, t_i$ for tame arguments i of f, Lemma 3.14 implies $\sigma(q_{m_0}) \, R \, \sigma'(q_{m_0})$. Furthermore, there is a $\gamma < \alpha$ such that $\sigma(q_{m_0})Q_{m_0} \in \mathtt{F}_\gamma$, so $\mathrm{I}'_{\gamma,\lambda}$ yields $\sigma'(q_{m_0})Q_{m_0} \notin \mathtt{C}_\lambda$. In other words,

$$\sigma'(q_{m_0})Q_{m_0} \in \mathtt{F}_\lambda. \tag{21}$$

By (13) and (17), the patience rules for wild arguments i of f for which x_i is the left-hand side of a premise in ρ yield

$$f(t_1, ..., t_{ar(f)}) \xrightarrow{\varepsilon} \sigma'(f(x_1, ..., x_{ar(f)})) \in \mathtt{C}_\lambda.$$

By (11), (14), (12), (18), (20), and (21), the panth rule ρ together with the substitution σ' yield

$$\sigma'(f(x_1, ..., x_{ar(f)})) \xrightarrow{a} \sigma'(r) \in \mathtt{C}_\lambda.$$

If i is a tame arguments of f then $\sigma'(x_i) = t_i$, so that $s_i \, R \, \sigma'(x_i)$. Furthermore, if i is a wild argument i of f and x_i does not occur as the left-hand side of a premise of ρ then $\sigma'(x_i) = t_i$, so that $s_i \, B \, \sigma'(x_i)$. Finally, if i is a wild argument i of f and x_i is the left-hand side of a premise of ρ, then (15) and (19) together yield $s_i \, B \, \sigma'(x_i)$. So according to the definition of B

$$f(s_1, ..., s_{ar(f)}) \, B \, \sigma'(f(x_1, ..., x_{ar(f)})).$$

$\sigma(x_i) = s_i$ for $i = 1, ..., ar(f)$, so $\sigma(x_i) \, B \, \sigma'(x_i)$ for wild arguments i of f (see above), and $\sigma(x_i) \, R \, \sigma'(x_i)$ for tame arguments i of f (because $\sigma'(x_i) = t_i$ for such i). Furthermore, (10) and (16) together yield $\sigma(y_j) \, B \, \sigma'(y_j)$ for $j \in J$. Finally, $\sigma(z) = \sigma'(z)$ for $z \notin \{x_i \mid i = 1, ..., ar(f)\} \cup \{y_j \mid j \in J\}$. The RBB safe format enforces that variables x_i for wild arguments of i of f and variables y_j for $j \in J$ only occur at w-nested positions in r, so Lemma 3.15 implies

$$\sigma(r) \, B \, \sigma'(r).$$

Proof of III'_α. Similar to the proof of III_α.

Since B is symmetric, III_λ-III'_λ together imply that B is a branching bisimulation relation. So since R is symmetric, II_λ-II'_λ together imply that R is a rooted branching bisimulation relation. Hence, $R = \leftrightarrows_{rb}$, i.e., rooted branching bisimulation equivalence is a congruence. □

References

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

[2] J.C.M. Baeten, J.A. Bergstra, J.W. Klop, and W.P. Weijland. Term-rewriting systems with rule priorities. *Theoretical Computer Science*, 67(2/3):283–301, 1989.

[3] J.C.M. Baeten and R.J. van Glabbeek. Another look at abstraction in process algebra. In T. Ottmann, ed., *Proceedings 14th Colloquium on Automata, Languages and Programming (ICALP'87)*, Karlsruhe, LNCS 267, pp. 84–94. Springer, 1987.

[4] J.C.M. Baeten and C. Verhoef. A congruence theorem for structured operational semantics with predicates. In E. Best, ed., *Proceedings 4th Conference on Concurrency Theory (CONCUR'93)*, Hildesheim, LNCS 715, pp. 477–492. Springer, 1993.

[5] J.C.M. Baeten and W.P. Weijland. *Process Algebra*. Cambridge Tracts in Theoretical Computer Science 18, Cambridge University Press, 1990.

[6] J.W. de Bakker and J.I. Zucker. Processes and the denotational semantics of concurrency. *Information and Control*, 54(1/2):70–120. 1982.

[7] T. Basten. Branching bisimilarity is an equivalence indeed! *Information Processing Letters*, 58(3):141–147, 1996.

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

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

[10] R.N. Bol and J.F. Groote. The meaning of negative premises in transition system specifications, *Journal of the ACM*, 43(5):863–914, 1996.

[11] W.J. Fokkink. Language preorder as a precongruence. *Theoretical Computer Science*, To appear.

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

[13] W.J. Fokkink and C. Verhoef. A conservative look at operational semantics with variable binding. *Information and Computation*, To appear.

[14] A. van Gelder, K. Ross, and J.S. Schlipf. The well-founded semantics for general logic programs, *Journal of the ACM*, 38(3):620–650, 1991.

[15] M. Gelfond and V. Lifschitz. The stable model semantics for logic programming. In *Proceedings 5th Conference on Logic Programming*, pp. 1070–1080. MIT Press, 1988.

[16] R.J. van Glabbeek. Bounded nondeterminism and the approximation induction principle in process algebra. In F.J. Brandeburg, G. Vidal-Naquet and M. Wirsing, eds., *Proceedings 4th Symposium on Theoretical Aspects of Computer Science (STACS'87)*, Passau, LNCS 247, pp. 336–347. Springer, 1987.

[17] R.J. van Glabbeek. The linear time – branching time spectrum. In J.C.M. Baeten and J.W. Klop, eds., *Proceedings 1st Conference on Concurrency Theory (CONCUR'90)*, Amsterdam, LNCS 458, pp. 278–297. Springer, 1990.

[18] R.J. van Glabbeek. The linear time – branching time spectrum II: the semantics of sequential systems with silent moves. In E. Best, ed., *Proceedings 4th Conference on Concurrency Theory (CONCUR'93)*, Hildesheim, LNCS 715, pp. 66–81. Springer, 1993.

[19] R.J. van Glabbeek. Full abstraction in structural operational semantics. In M. Nivat, C. Rattray, T. Rus and G. Scollo, eds., *Proceedings 3rd Conference on Algebraic Methodology and Software Technology (AMAST'93)*, Enschede, *Workshops in Computing*, pp. 77–84. Springer, 1993.

[20] R.J. van Glabbeek. What is branching time and why to use it? In M. Nielsen, ed., *The Concurrency Column*, *Bulletin of the EATCS*, 53:190-198, 1994.

[21] R.J. van Glabbeek. The meaning of negative premises in transition system specifications II. Extended abstract in F. Meyer auf der Heide and B. Monien, eds., *Proceedings 23rd Colloquium on Automata, Languages and Programming (ICALP'96)*, Paderborn, pp. 502–513, LNCS 1099. Springer, 1996.

[22] R.J. van Glabbeek and W.P. Weijland. Branching time and abstraction in bisimulation semantics. *Journal of the ACM*, 43(3):555–600, 1996.

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

[24] J.F. Groote and F.W. Vaandrager. Structured operational semantics and bisimulation as a congruence. *Information and Computation*, 100(2):202–260, 1992.

[25] S.C. Kleene. Representation of events in nerve nets and finite automata. In *Automata Studies*, pages 3–41. Princeton University Press, 1956.

[26] R. Milner. A modal characterisation of observable machine-behaviour. In E. Astesiano and C. Böhm, eds., *Proceedings 6th Colloquium on Trees in Algebra and Programming (CAAP'81)*, Genoa, LNCS 112, pp. 25–34. Springer, 1981.

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

[28] D.M.R. Park. Concurrency and automata on infinite sequences. In P. Deussen, ed., *Proceedings 5th GI Conference*, Karlsruhe, LNCS 104, pp. 167–183. Springer, 1981.

[29] G.D. Plotkin. A structural approach to operational semantics. Report DAIMI FN-19, Aarhus University, 1981.

[30] T.C. Przymusinski. The well-founded semantics coincides with the three-valued stable semantics. *Fundamenta Informaticae*, 13(4):445–463, 1990.

[31] A. Tarski. A lattice theoretical fixed point theorem and its applications. *Pacific Journal of Mathematics*, 5:285–309, 1955.

[32] I. Ulidowski and I. Phillips. Formats of ordered SOS rules with silent actions. In M. Bidoit and M. Dauchet, eds., *Proceedings 7th Conference on Theory and Practice of Software Development (TAPSOFT'97)*, Lille, LNCS 1214, pp. 297–308. Springer, 1997.

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

[34] C. Verhoef. A congruence theorem for structured operational semantics with predicates and negative premises. *Nordic Journal of Computing*, 2(2):274–302, 1995.

[35] J.L.M. Vrancken. The algebra of communicating processes with empty process. *Theoretical Computer Science*, 177(2):287-328, 1997.