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 on 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] has evolved as a standard methodology to provide specification languages, programming languages, and process algebras with a semantics. In structural operational semantics, transitions with action labels between algebraic terms are derived from inductive proof rules, called transition rules, which together make up a transition system specification (TSS). 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. This article focuses on a single-sorted first-order signature for terms.

Przymusinski [32] introduced three-valued stable models in order to give meaning to TSSs that incorporate 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 TSS has a three-valued stable model in which the set of transitions in limbo is maximal; this so-called least three-valued stable model coincides with the well-founded semantics of van Gelder, Ross, and Schlipf [14]. A TSS is complete [21] (or positive after reduction [10]) if all transitions in its least three-valued stable model are either true or false. Although in general it is not effectively decidable whether a TSS is complete, the notion of a stratification [23, 31] provides a convenient method to decide for many TSSs in the literature that they are complete.

Labeled transition systems can be distinguished by a wide range of behavioral equivalences [17, 18]. In the field of process algebra, four so-called weak equivalences

*Supported by a grant from The Nuffield Foundation

have been developed to abstract away from internal machine behavior, 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 an action τ if its execution does not implicate the loss of possible behavior. In recent years, this equivalence has been used in a sizable number of verifications in process algebra. See [20] for a lucid exposition on the motivations behind the definition of branching bisimulation equivalence.

In general, a behavioral equivalence relation induced by a TSS is not a congruence; that is, the equivalence class of a term $f(t_1, \ldots, t_n)$ need not be determined by the equivalence classes of its arguments $t_1, \ldots, t_n$. Congruence is an important property to fit such a behavioral equivalence into an axiomatic framework. Congruence formats in structural operational semantics have been developed for a number of behavioral 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 TSS 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, 38] extended the tyft and ntyft formats with predicates, to obtain the path and panth formats, respectively.

Bloom [8] and Ulidowski [34, 35] 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 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 positive GSOS format. It recognizes so-called patience rules for arguments i of functions symbols f, which imply that a term $f(p_1, \ldots, p_n)$ inherits the τ-transitions of its argument p_i. Furthermore, function symbols in the right-hand sides of conclusions of transition rules are labeled wild, and this labeling is used to restrict occurrences of variables in the left-hand sides of premises of transition rules. 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 more liberal congruence format for rooted branching bisimulation equivalence, called RBB safe, which imposes additional requirements on the panth format. The RBB safe format incorporates negative premises and relaxes several syntactic restrictions of the RBB cool format. Notably, only certain arguments of function symbols in right-hand sides of conclusions of transition rules are labeled 'wild'. If a TSS 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 labeling of arguments of function symbols, against which these restrictions can be checked. A detailed com-

parison between RBB cool and RBB safe is given in Section 3.3. The generalizations of RBB safe with respect to RBB cool provide answers to three open questions 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 [39], 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.

In [11] a precongruence format was presented for language preorder, which uses a wild labeling of arguments of function symbols in a similar fashion as in the RBB safe format. Vaandrager [37] introduced precongruence formats for failures, external trace, external failure, and must preorders. van Glabbeek [19] presented congruence formats for ready simulation, ready trace, and failure equivalence. Bloom [7] formulated a congruence format for 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* $x, y, z, \ldots$;
- *a set* $\mathcal{F}$ *of function symbols* $f, g, h, \ldots$, *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 $p, q, r, s, t, u, v, w, \ldots$ *over* Σ *is defined as the least set satisfying:*

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

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 Transitions

This section presents the basic notions of structural operational semantics. It assumes a signature Σ, a set of transition labels $a, b, c, \ldots$, and a set $\mathcal{P}$ of predicates $P, Q, \ldots$.

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

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

- *Expressions $t \xrightarrow{a}\!\!\!\!\!\!\rightarrow$ and $t\neg P$ are* negative transitions.

Definition 2.5 *A collection of negative transitions N* holds *for a set of positive transitions $\mathtt{P}$, denoted by $\mathtt{P} \models N$, if*

- *for each $t \xrightarrow{a}\!\!\!\!\!\!\rightarrow \in N$, $t \xrightarrow{a} t \notin \mathtt{P}$ for all $t \in \mathcal{T}(\Sigma)$;*

- *for each $t\neg P \in N$, $tP \notin \mathtt{P}$.*

Definition 2.6 *A* transition rule *is an expression of the form H/π, where H is a collection of expressions $t \xrightarrow{a} t'$, tP, $t \xrightarrow{a}\!\!\!\!\!\!\rightarrow$, and $t\neg P$ with $t, t' \in \mathbb{T}(\Sigma)$, called the* premises, *and π is an expression $t \xrightarrow{a} t'$ or tP with $t, t' \in \mathbb{T}(\Sigma)$, 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.*

A transition rule is closed *if it contains only closed terms.*

Definition 2.7 *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 labeled 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 substitution instance of a transition rule in T.*

2.3 Three-Valued Stable Models

We use the *least three-valued stable model*, introduced by Przymusinski [32] in logic programming, to give a semantics to TSSs with negative premises. A three-valued stable model partitions the collection of positive transitions into three disjoint sets: the set $\mathtt{C}$ of positive transitions that are *certainly true*, the set $\mathtt{U}$ of positive transitions for which it is *unknown* whether or not they are true, and the set of remaining positive transitions that are *false*. Such a partitioning (which is determined by $\langle \mathtt{C}, \mathtt{U} \rangle$) constitutes a three-valued stable model for TSS T if:

- a positive transition π is in $\mathtt{C}$ if and only if T proves a closed transition rule N/π where N contains only negative transitions and $\mathtt{C} \cup \mathtt{U} \models N$;

- a positive transition π is in $\mathtt{C} \cup \mathtt{U}$ if and only if T proves a closed transition rule N/π where N contains only negative transitions and $\mathtt{C} \models N$.

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

Definition 2.8 *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 transition rules with negative premises is always complete; see [21].

Stratifications In general it is not effectively decidable whether a finite TSS is complete. Van de Pol [30, Ex. 22] presented a striking example, in the realm of term rewriting with priorities [2], that it may take more than ω steps to compute the least three-valued stable model for a finite priority rewrite system.

In practice, a useful tool for showing that a TSS is complete is the notion of a *stratification* [23, 31]. Basically, a TSS is stratified if there exists a weight function on transitions such that for each substitution instance of each transition rule, the substitution instances of positive premises are smaller than or equal to the substitution instance of the conclusion, and the substitution instances of negative premises are strictly smaller than the substitution instance of the conclusion.

Definition 2.9 *A* stratification *for a TSS is a weight function ϕ which maps transitions to ordinals, such that for each transition rule ρ with conclusion π and for each substitution σ:*

1. *for positive premises $t \xrightarrow{a} t'$ and tP of ρ, $\phi(\sigma(t) \xrightarrow{a} \sigma(t')) \leq \phi(\sigma(\pi))$ and $\phi(\sigma(t)P) \leq \phi(\sigma(\pi))$, respectively;*
2. *for negative premises $t \stackrel{a}{\nrightarrow}$ and $t\neg P$ of ρ, $\phi(\sigma(t) \xrightarrow{a} t') < \phi(\sigma(\pi))$ for all closed terms t' and $\phi(\sigma(t)P) < \phi(\sigma(\pi))$, respectively.*

The following result stems from [10].

Theorem 2.10 *If a TSS allows a stratification, then it is complete.*

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

Definition 2.11 *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 [6].

Definition 2.12 *An equivalence relation* R *on* $\mathcal{T}(\Sigma)$ *is called a* congruence *if* $s_i \, R \, t_i$ *for* $i = 1, \ldots, ar(f)$ *implies* $f(s_1, \ldots, s_{ar(f)}) \, R \, f(t_1, \ldots, t_{ar(f)})$.

Branching bisimulation equivalence is not a congruence with respect to most process algebras from the literature. A rootedness condition has been introduced to remedy this imperfection [27].

Definition 2.13 *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

This section presents the panth format [38], with the additional requirement that the source is not a single variable (i.e., we only consider the *ntyft* component of the panth format).

Definition 2.14 *A transition rule is a* panth *rule if it is of the form*

$$\frac{\{s_j \xrightarrow{a_j} y_j \mid j \in J\} \cup \{t_k P_k \mid k \in K\} \cup \{p_\ell \overset{b_\ell}{\nrightarrow} \mid \ell \in L\} \cup \{q_m \neg Q_m \mid m \in M\}}{f(x_1, \ldots, x_{ar(f)}) \xrightarrow{a} r}$$

or

$$\frac{\{s_j \xrightarrow{a_j} y_j \mid j \in J\} \cup \{t_k P_k \mid k \in K\} \cup \{p_\ell \overset{b_\ell}{\nrightarrow} \mid \ell \in L\} \cup \{q_m \neg Q_m \mid m \in M\}}{f(x_1, \ldots, x_{ar(f)})P}$$

where the $x_1, \ldots, x_{ar(f)}$ *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 labeled 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*

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

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*

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

Definition 3.3 *A TSS* T *is called* RBB safe*, with respect to a tame/wild labeling 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* ρ *with source* $f(x_1, \ldots, x_{ar(f)})$ *and right-hand sides of premises* $\{y_j \mid j \in J\}$*, such that the following requirements are fulfilled.*
   - *Variables* y_j *for* $j \in J$ *do not occur in left-hand sides of premises of* ρ*.*
   - *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 of* ρ*.*
   - *If argument* i *of* f *is wild and has a patience rule in* T*, then* x_i *occurs in the left-hand side of no more than one premise of* ρ*, where 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 of* ρ*.*

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

Assume a TSS T that consists of a finite number of transition rules, which each have finitely many premises. Suppose we want to verify that rooted branching bisimulation equivalence as induced by T is a congruence.

- It can be attempted to find a suitable stratification (see Definition 2.9) to show that T is complete.
- It is easy to verify whether the transition rules in T are panth.
- Given a tame/wild labeling of arguments of function symbols, it is easy to check whether each transition rule satisfies the restrictions as imposed by the RBB safe format; see Definition 3.3.

The crux in determining whether the TSS T is RBB safe is to find a suitable tame/wild labeling 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 labeling Λ such that (T, Λ) is RBB safe if and only if there exists a labeling 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
>
> $$f(x_1, \ldots, x_{ar(f)}) \xrightarrow{a} C[g(t_1, \ldots, t_{j-1}, C'[x_i], t_{j+1}, \ldots, t_{ar(g)})].$$
>
> 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
>
> $$C[g(t_1, \ldots, t_{j-1}, C'[y], t_{j+1}, \ldots, t_{ar(g)})]$$
>
> where y is the right-hand side of a premise of this transition rule. All other vertices in G are colored green.
>
> The procedure colors 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 colored red.
>
> The procedure terminates if none of the green vertices can be colored 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 cool format is more restrictive than the RBB safe format in the following respects, in increasing order of importance.

1. It does not incorporate predicates.

2. It requires that the left-hand sides of the premises in a transition rule are distinct variables.

3. It only allows the relation $\xrightarrow{\tau}$ to occur in premises of patience rules.

4. It requires *all* arguments of function symbols that occur in the target of a transition rule to be wild (so that occurrences of variables in targets are by default w-nested).

   This results in a need for more patience rules, and in more severe restrictions on occurrences of variables in left-hand sides of positive premises.

5. It does not incorporate negative premises.

The RBB safe format is strictly more liberal than Bloom's *simply* RBB cool format [8]. 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, 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.

The fact that the RBB safe format is based on the panth format and incorporates negative premises, provides affirmative answers to the first two open questions in the conclusion of [8]. The next section presents several TSSs from process algebra that are RBB safe but not RBB cool, due to the distinctions between these formats as discussed above. These 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 TSS in this section shows that it is useful that the RBB safe format incorporates premises.

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 [39]. 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, where the a ranges over $A\cup\{\tau\}$.

The TSS in Table 1 is in path format, and since it does not involve negative premises it is complete. The procedure in Section 3.2 calculates the following tame/wild labeling: the first argument of sequential composition is wild (because of the target $y\cdot x_2$ in the third transition 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 labeling:

- the third transition rule for sequential composition with $a=\tau$ constitutes a patience rule for the first argument of sequential composition;

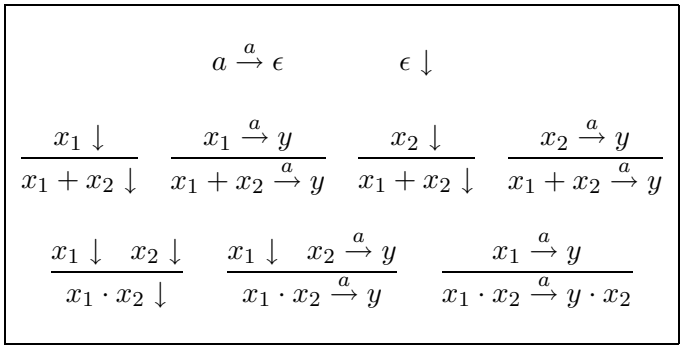

Table 1: Transition Rules for $\mathrm{BPA}_{\epsilon\tau}$

- in the first two transition rules for sequential composition, and in the third transition 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 transition 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 TSS in Table 1 is not RBB cool, due to the fact that some of its transition rules involve the predicate $\downarrow$. However, predicates can be encoded as binary transition relations (see [24]), and moreover the RBB cool format could be extended with predicates in a trivial manner. The following three sections, which focus on extensions of $\mathrm{BPA}_{\epsilon\tau}$, present more serious examples of RBB safe TSSs that violate Bloom's RBB cool format.

3.3.2 Recursion

The TSS in this section shows that it is useful that the RBB safe format allows left-hand sides of premises to be nonvariable.

Given a signature Σ, a recursive specification E is a finite set of equations $\{X_i = t_i \mid i = 1, \ldots, n\}$, where the X_i are recursion variables, and the t_i are open terms over Σ, with possible occurrences of recursion variables. Intuitively, the syntactic construct $\langle X|E\rangle$ denotes a solution of X with respect to E. The precise meaning of this construct is given by the transition rules for recursion 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 the expressions $\langle X|E\rangle$ as constants. The TSS for $\mathrm{BPA}_{\epsilon\tau}$ with recursion in Tables 1 and 2 is in path format, and since it does not involve negative premises it is complete. Furthermore, it is easy to see that this TSS is RBB safe with respect to the tame/wild labeling from Section 3.3.1.

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

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

Table 2: Transition Rules for Recursion

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 they involve the predicate $\downarrow$).

3.3.3 Iteration

The TSS in this section shows that it is useful that the RBB safe format allows certain arguments of function symbols in targets to be tame.

The iteration operator [25], denoted by t^*, 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. The resulting TSS for $\mathrm{BPA}_{\epsilon\tau}$ is in path format, and since it does not involve negative premises it is complete.

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

Table 3: Transition Rules for Iteration

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. It is not hard to see that the TSS for $\mathrm{BPA}_{\epsilon\tau}$ with iteration is RBB safe with respect to this tame/wild labeling. Note that in the second transition rule in Table 3, the right-hand side y of the premise occurs in a wild argument of the target.

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 Initial Priority

The TSS in this section shows that it is useful that the RBB safe format incorporates negative premises, and that it allows the relation symbol $\xrightarrow{\tau}$ to occur in premises of nonpatience rules.

Initial 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.

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

Table 4: Transition Rules for Initial Priority

The TSS for $\mathrm{BPA}_{\epsilon\tau}$ with initial priority consists of the path rules in Table 1 together with the panth rules in Table 4. This TSS is complete, which can be seen by giving a suitable weight function on transitions: the weight of a transition $t \xrightarrow{a} t'$ or tP is the number of occurrences of the initial priority operator in t. It is not hard to see that this weight function is a stratification (see Definition 2.9) for the TSS for $\mathrm{BPA}_{\epsilon\tau}$ with initial priority; i.e., for each substitution instance of a transition rule in this TSS, the positive premises are smaller or equal than the conclusion, and the negative premises are strictly smaller than the conclusion. So according to Theorem 2.10, the TSS is complete.

The procedure in Section 3.2 labels the argument of initial priority 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. It is not hard to see that the TSS for $\mathrm{BPA}_{\epsilon\tau}$ with initial priority is RBB safe with respect to this tame/wild labeling. Note that in the second transition rule in Table 4, the left-hand side x of the negative premises occurs in the tame argument of the source.

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

The second transition rule in Table 4 is not RBB cool, due to the fact that it contains negative premises. Moreover, the second transition rule in Table 4 for $a = t$ contains the relation symbol $\xrightarrow{\tau}$ in its premise, but it is not a patience rule.

Initial priority is derived 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 transition rule

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

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 (cf. [36, pp. 130–132]).

3.4 Counter-Examples

This section presents a string of examples of TSSs in panth format, to show that the additional syntactic restrictions of the RBB safe format are essential. The first counter-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 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. This TSS induces a unique two-valued stable model, in which $\{aP, bP, f(a)Q_1, f(b)Q_2\}$ constitutes the set of positive transitions that are certainly true. However, the TSS is not complete, because its least three-valued stable model has a nonempty 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 the TSS of $\mathrm{BPA}_{\epsilon\tau}$, where the set A of atomic actions consists of two elements a and b. Furthermore, we assume a unary function symbol f and a predicate P. The TSSs in the forthcoming examples are all in the panth format.

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. Note that the TSS in this example is complete, because it does not involve negative premises.

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*

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

The resulting TSS is complete, which can be seen by giving a suitable weight function on transitions: the weight of a transition is the number of occurrences of the function symbol f in its left-hand side. It is not hard to see that this weight function is a stratification for the TSS in this example, so according to Theorem 2.10 the TSS is complete.

The first transition rule is a patience rule for the argument of f. Note that, in the second transition 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. Note that the TSS in this example is complete, because it does not involve negative premises.

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 transition rule is a patience rule for the argument of f. Note that, in the second transition rule, the wild argument x of the source is the left-hand side of the positive premise with relation symbol $\xrightarrow{\tau}$.

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. Note that the TSS in this example is complete, because it does not involve negative premises.

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 transition 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 transition 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 with respect to some tame/wild labeling of arguments of function symbols. The least three-valued stable model for T induces a branching equivalence $\leftrightarrows_b$ (Definition 2.11), and thus a rooted branching equivalence $\leftrightarrows_{rb}$ (Definition 2.13). Let R be the least binary relation over $\mathcal{T}(\Sigma)$ that satisfies:

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

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

- $s\,B\,t$ if $s \leftrightarrows_b t$;
- $f(s_1,\ldots,s_{ar(f)})\,B\,f(t_1,\ldots,t_{ar(f)})$ 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 $\langle \mathtt{C}_\alpha, \mathtt{U}_\alpha \rangle$ for ordinals α, using ordinal induction, and show that these pairs converge to a suitable three-valued stable model for T.

- $\mathtt{C}_0 = \emptyset$ and $\mathtt{U}_0$ contains all positive transitions.

- For ordinals α, $\langle \mathtt{C}_{\alpha+1}, \mathtt{U}_{\alpha+1}\rangle$ is constructed from $\langle \mathtt{C}_\alpha, \mathtt{U}_\alpha\rangle$ as follows.

  A positive transition π is in $\mathtt{C}_{\alpha+1}$ if and only if T proves a closed transition rule N/π where N contains only negative transitions and $\mathtt{C}_\alpha \cup \mathtt{U}_\alpha \models N$.

  A positive transition π is in $\mathtt{C}_{\alpha+1} \cup \mathtt{U}_{\alpha+1}$ if and only if T proves a closed transition rule N/π where N contains only negative transitions and $\mathtt{C}_\alpha \models N$.

- For limit ordinals α we define $\mathtt{C}_\alpha = \cup_{\beta<\alpha}\mathtt{C}_\beta$ and $\mathtt{U}_\alpha = \cap_{\beta<\alpha}\mathtt{U}_\beta$.

The following two inclusions can be derived for ordinals α and β with $\beta \leq \alpha$, by ordinal induction (cf. [13]):

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

Owing to these two inclusions, the Knaster-Tarski theorem [33] yields that there exists an ordinal λ such that $\mathtt{C}_\lambda = \mathtt{C}_{\lambda+1}$ and $\mathtt{U}_\lambda = \mathtt{U}_{\lambda+1}$. It is easy to see that $\langle \mathtt{C}_\lambda, \mathtt{U}_\lambda\rangle$ is a three-valued stable model for T (owing to the definitions of $\pi \in \mathtt{C}_{\lambda+1}$ and $\pi \in \mathtt{C}_{\lambda+1} \cup \mathtt{U}_{\lambda+1}$). Furthermore, if $\langle \mathtt{C}, \mathtt{U}\rangle$ is some three-valued stable model for T, then it follows by ordinal induction that $\mathtt{U} \subseteq \mathtt{U}_\alpha$ for all ordinals α, so in particular $\mathtt{U} \subseteq \mathtt{U}_\lambda$. Hence, $\langle \mathtt{C}_\lambda, \mathtt{U}_\lambda\rangle$ is the least three-valued stable model for T. In particular, since T is complete, $U_\lambda = \emptyset$.

We prove the following four statements in parallel, using ordinal induction with respect to α. (Statements I_α and II_α, and their proofs, are similar to the cases 1 and 2 in [10, Lem. 8.9], which form the basis of a congruence proof for the ntyft/ntyxt format modulo strong bisimulation.)

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

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

II_α. If $s\,R\,t$ 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 $s'\,B\,t'$.

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

We assume that I_β-I'_β and II_β-II'_β have already been proved for $\beta < \alpha$. In the proofs to come, we repeatedly use without mention the facts that $\mathtt{C}_{\lambda+1} = \mathtt{C}_\lambda$ and $\mathtt{U}_\lambda = \emptyset$.

Proof of I_α. I_0 follows from the fact that $\mathtt{C}_0 \cup \mathtt{U}_0$ contains all positive transitions, and if α is a limit ordinal then I_α follows by ordinal induction, owing to the fact that $\mathtt{C}_\alpha = \cup_{\beta<\alpha}\mathtt{C}_\beta$ and $\mathtt{U}_\alpha = \cap_{\beta<\alpha}\mathtt{U}_\beta$. We focus on the case where $\alpha - 1$ is well-defined.

If $s \leftrightarrows_{rb} t$, then I_α follows immediately from the definition of $\leftrightarrows_{rb}$ together with $\mathtt{C}_\lambda \subseteq \mathtt{C}_\alpha \cup \mathtt{U}_\alpha$. We focus on the case where $s = f(s_1, \ldots, s_{ar(f)})$ and $t = f(t_1, \ldots, t_{ar(f)})$, with $s_i\,R\,t_i$ for $i = 1, \ldots, ar(f)$.

Since $f(s_1, \ldots, s_{ar(f)}) \xrightarrow{a} s' \in \mathtt{C}_\lambda$, there is a proof from T for a closed transition rule $N/f(s_1, \ldots, s_{ar(f)}) \xrightarrow{a} s'$, where N contains only negative transitions

and $\mathtt{C}_\lambda \models N$. We apply induction with respect to the length γ of the proof for $N/f(s_1,\ldots,s_{ar(f)}) \xrightarrow{a} s'$ from T.

Since there is a proof for $N/f(s_1,\ldots,s_{ar(f)}) \xrightarrow{a} s'$ from T, there exists a panth rule ρ in T of the form

$$\frac{\{u_j \xrightarrow{a_j} y_j \,|\, j \in J\} \cup \{v_k P_k \,|\, k \in K\} \cup \{p_\ell \overset{b_\ell}{\nrightarrow} \,|\, \ell \in L\} \cup \{q_m \neg Q_m \,|\, m \in M\}}{f(x_1,\ldots,x_{ar(f)}) \xrightarrow{a} r}$$

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

A. for each $j \in J$, there is a proof from T, shorter than γ, for a closed transition rule $N_j/\sigma(u_j) \xrightarrow{a_j} \sigma(y_j)$ with $N_j \subseteq N$;

B. for each $k \in K$, there is a proof from T, shorter than γ, for a closed transition rule $N'_k/\sigma(v_k)P_k$ with $N'_k \subseteq N$;

C. for each $\ell \in L$, $\sigma(p_\ell) \overset{b_\ell}{\nrightarrow} \in N$;

D. for each $m \in M$, $\sigma(q_m)\neg Q_m \in N$.

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

1. $\sigma'(z) = \sigma(z)$ for $z \notin \{x_i \,|\, i = 1,\ldots,ar(f)\} \cup \{y_j \,|\, j \in J\}$.

2. $\sigma'(x_i) = t_i$ for $i = 1,\ldots,ar(f)$.

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 $\{y_j \,|\, j \in J\}$, so $\sigma'(u_{j_0})$ is well-defined. Since $s_i \,R\, t_i$ for $i = 1,\ldots,ar(f)$, Lemma 3.14 implies $\sigma(u_{j_0}) \,R\, \sigma'(u_{j_0})$. Furthermore, by property A there is a proof from T for $N_j/\sigma(u_j) \xrightarrow{a_j} \sigma(y_j)$ with $\mathtt{C}_\lambda \models N \supseteq N_j$, so $\sigma(u_{j_0}) \xrightarrow{a_{j_0}} \sigma(y_{j_0}) \in \mathtt{C}_\lambda$. Since moreover this proof from T is shorter than γ, by induction I_α yields $\sigma'(u_{j_0}) \xrightarrow{a_{j_0}} w \in \mathtt{C}_\alpha \cup \mathtt{U}_\alpha$ for some $w \in \mathcal{T}(\Sigma)$. We define $\sigma'(y_{j_0}) = w$, so that
$$\sigma'(u_{j_0}) \xrightarrow{a_{j_0}} \sigma'(y_{j_0}) \in \mathtt{C}_\alpha \cup \mathtt{U}_\alpha. \tag{1}$$

$\sigma(z) \,R\, \sigma'(z)$ for $z \notin \{y_j \,|\, j \in J\}$, 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 $k \in K$.

  By property B there is a proof from T for $N'_k/\sigma(v_k)P_k$ with $\mathtt{C}_\lambda \models N \supseteq N'_k$, so $\sigma(v_k)P_k \in \mathtt{C}_\lambda$. Since moreover this proof from T is shorter than γ, by induction I'_α yields
  $$\sigma'(v_k)P_k \in \mathtt{C}_\alpha \cup \mathtt{U}_\alpha. \tag{2}$$

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

  Property C says $\sigma(p_\ell) \stackrel{b_\ell}{\not\to} \in N$, and $\mathtt{C}_\lambda \models N$, so $\sigma(p_\ell) \stackrel{b_\ell}{\to} p' \notin \mathtt{C}_\lambda$ for all $p' \in \mathcal{T}(\Sigma)$. Hence, $\mathrm{II}_{\alpha-1}$ implies

$$\sigma'(p_\ell) \stackrel{b_\ell}{\to} p' \notin \mathtt{C}_{\alpha-1} \text{ for } p' \in \mathcal{T}(\Sigma). \tag{3}$$

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

  Property D says $\sigma(q_m)\neg Q_m \in N$, and $\mathtt{C}_\lambda \models N$, so $\sigma(q_m)Q_m \notin \mathtt{C}_\lambda$. Hence, $\mathrm{II}'_{\alpha-1}$ implies

$$\sigma'(q_m)Q_m \notin \mathtt{C}_{\alpha-1}. \tag{4}$$

(1)–(4) together imply that transition rule ρ together with substitution σ' form the basis of a proof from T for a transition rule $N'/f(t_1,\ldots,t_{ar(f)}) \stackrel{a}{\to} \sigma'(r)$ with $\mathtt{C}_{\alpha-1} \models N'$. Hence,

$$f(t_1,\ldots,t_{ar(f)}) \stackrel{a}{\to} \sigma'(r) \in \mathtt{C}_\alpha \cup \mathtt{U}_\alpha. \qquad \Box$$

Proof of I'_α. Similar to the proof of I_α.

Proof of II_α. II_0 follows from the fact that $\mathtt{C}_0 = \emptyset$, and if α is a limit ordinal then II_α follows by ordinal induction, owing to the fact that $\mathtt{C}_\alpha = \cup_{\beta<\alpha}\mathtt{C}_\beta$. We focus on the case where $\alpha - 1$ is well-defined.

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 $s = f(s_1,\ldots,s_{ar(f)})$ and $t = f(t_1,\ldots,t_{ar(f)})$, with $s_i\, R\, t_i$ for $i = 1,\ldots,ar(f)$.

Since $f(s_1,\ldots,s_{ar(f)}) \stackrel{a}{\to} s' \in \mathtt{C}_\alpha$, there is a proof from T for a closed transition rule $N/f(s_1,\ldots,s_{ar(f)}) \stackrel{a}{\to} s'$, where N contains only negative transitions and $\mathtt{C}_{\alpha-1} \cup \mathtt{U}_{\alpha-1} \models N$. We apply induction with respect to the length γ of the proof for $N/f(s_1,\ldots,s_{ar(f)}) \stackrel{a}{\to} s'$ from T.

Since there is a proof for $N/f(s_1,\ldots,s_{ar(f)}) \stackrel{a}{\to} s'$ from T, there exists a panth rule ρ in T of the form

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

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

A. for each $j \in J$, there is a proof from T, shorter than γ, for a closed transition rule $N_j/\sigma(u_j) \stackrel{a_j}{\to} \sigma(y_j)$ with $N_j \subseteq N$;

B. for each $k \in K$, there is a proof from T, shorter than γ, for a closed transition rule $N'_k/\sigma(v_k)P_k$ with $N'_k \subseteq N$;

C. for each $\ell \in L$, $\sigma(p_\ell) \stackrel{b_\ell}{\not\to} \in N$;

D. for each $m \in M$, $\sigma(q_m)\neg Q_m \in N$.

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

1. $\sigma'(z) = \sigma(z)$ for $z \notin \{x_i \mid i = 1,\ldots,ar(f)\} \cup \{y_j \mid j \in J\}$.
2. $\sigma'(x_i) = t_i$ for $i = 1,\ldots,ar(f)$.
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 $i = 1,\ldots,ar(f)$, Lemma 3.14 implies $\sigma(u_{j_0})\,R\,\sigma'(u_{j_0})$. Furthermore, by property A there is a proof from T for $N_j/\sigma(u_j) \xrightarrow{a_j} \sigma(y_j)$ with $\mathtt{C}_{\alpha-1}\cup\mathtt{U}_{\alpha-1} \models N \supseteq N_j$, so $\sigma(u_{j_0}) \xrightarrow{a_{j_0}} \sigma(y_{j_0}) \in \mathtt{C}_\alpha$. Since moreover this proof from T is shorter than γ, by induction II_α 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 $k \in K$.

  By property B there is a proof from T for $N'_k/\sigma(v_k)P_k$ with $\mathtt{C}_{\alpha-1} \cup \mathtt{U}_{\alpha-1} \models N \supseteq N'_k$, so $\sigma(v_k)P_k \in \mathtt{C}_\alpha$. Since moreover this proof from T is shorter than γ, by induction II'_α yields
  $$\sigma'(v_k)P_k \in \mathtt{C}_\lambda. \tag{7}$$

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

  Property C says $\sigma(p_\ell) \xrightarrow{b_\ell} \in N$, and $\mathtt{C}_{\alpha-1} \cup \mathtt{U}_{\alpha-1} \models N$, so $\sigma(p_\ell) \xrightarrow{b_\ell} p' \notin \mathtt{C}_{\alpha-1} \cup \mathtt{U}_{\alpha-1}$ for all $p' \in \mathcal{T}(\Sigma)$. Hence, $\mathrm{I}_{\alpha-1}$ implies

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

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

  Property D says $\sigma(q_m)\neg Q_m \in N$, and $\mathtt{C}_{\alpha-1} \cup \mathtt{U}_{\alpha-1} \models N$, so $\sigma(q_m)Q_m \notin \mathtt{C}_{\alpha-1} \cup \mathtt{U}_{\alpha-1}$. Hence, $\mathrm{I}'_{\alpha-1}$ implies

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

(6)–(9) together imply that transition rule ρ together with substitution σ' form the basis of a proof from T for a transition rule $N'/f(t_1,\ldots,t_{ar(f)}) \xrightarrow{a} \sigma'(r)$ with $\mathtt{C}_\lambda \models N'$. Hence,

$$f(t_1,\ldots,t_{ar(f)}) \xrightarrow{a} \sigma'(r) \in \mathtt{C}_\lambda.$$

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

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_β-III'_β have already been proved for ordinals $\beta < \alpha$.

Proof of III_α. III_0 follows from the fact that $\mathtt{C}_0 = \emptyset$, and if α is a limit ordinal then I_α follows by ordinal induction, owing to the fact that $\mathtt{C}_\alpha = \cup_{\beta<\alpha}\mathtt{C}_\beta$ We focus on the case where $\alpha - 1$ is well-defined.

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 $s = f(s_1, \ldots, s_{ar(f)})$ and $t = f(t_1, \ldots, t_{ar(f)})$, with $s_i\,R\,t_i$ for tame arguments i of f and $s_i\,B\,t_i$ for wild arguments i of f.

Since $f(s_1, \ldots, s_{ar(f)}) \xrightarrow{a} s' \in \mathtt{C}_\alpha$, there is a proof from T for a closed transition rule $N/f(s_1, \ldots, s_{ar(f)}) \xrightarrow{a} s'$, where N contains only negative transitions and $\mathtt{C}_{\alpha-1} \cup \mathtt{U}_{\alpha-1} \models N$. We apply induction with respect to the length γ of the proof for $N/f(s_1, \ldots, s_{ar(f)}) \xrightarrow{a} s'$ from T. Since there is a proof for $N/f(s_1, \ldots, s_{ar(f)}) \xrightarrow{a} s'$ from T, 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 of the form
$$\frac{x_{i_0} \xrightarrow{\tau} y}{f(x_1, \ldots, x_{i_0}, \ldots, x_{ar(f)}) \xrightarrow{\tau} f(x_1, \ldots, y, \ldots, x_{ar(f)})}$$
and a substitution σ with $\sigma(x_i) = s_i$ for $i = 1, \ldots, ar(f)$ and $f(s_1, \ldots, \sigma(y), \ldots, s_{ar(f)}) = s'$, such that:

  A. there is a proof from T, shorter than γ, for the closed transition rule $N/s_{i_0} \xrightarrow{\tau} \sigma(y)$.

  By property A there is a proof from T for $N/s_{i_0} \xrightarrow{\tau} \sigma(y)$ with $\mathtt{C}_{\alpha-1} \cup \mathtt{U}_{\alpha-1} \models N$, so $s_{i_0} \xrightarrow{\tau} \sigma(y) \in \mathtt{C}_\alpha$. Since moreover this proof from T is shorter than γ, and $s_{i_0}\,B\,t_{i_0}$, by induction III_α offers two possibilities.

  - Case 1.1: $\sigma(y)\,B\,t_{i_0}$.

Since i_0 is a wild argument of f, $s_i \mathrel{R} t_i$ for tame arguments i of f, and $s_i \mathrel{B} t_i$ for wild arguments i of f, the definition of B yields

$$f(s_1, \ldots, \sigma(y), \ldots, s_{ar(f)}) \mathrel{B} f(t_1, \ldots, t_{i_0}, \ldots, t_{ar(f)}),$$

or in other words, $s' \mathrel{B} t$.

- Case 1.2: $t_{i_0} \xrightarrow{\varepsilon} t' \xrightarrow{\tau} t'' \in \mathtt{C}_\lambda$ with $s_{i_0} \mathrel{B} t'$ and $\sigma(y) \mathrel{B} t''$.
  Since $t_{i_0} \xrightarrow{\varepsilon} t' \xrightarrow{\tau} t'' \in \mathtt{C}_\lambda$, and $t = f(t_1, \ldots, t_{i_0}, \ldots, t_{ar(f)})$, the patience rule for argument i_0 of f yields

  $$t \xrightarrow{\varepsilon} f(t_1, \ldots, t', \ldots, t_{ar(f)}) \xrightarrow{\tau} f(t_1, \ldots, t'', \ldots, t_{ar(f)}) \in \mathtt{C}_\lambda.$$

  Since $s_{i_0} \mathrel{B} t'$, $\sigma(y) \mathrel{B} t''$, i_0 is a wild argument of f, $s_i \mathrel{R} t_i$ for tame arguments i of f, and $s_i \mathrel{B} t_i$ for wild arguments i of f, the definition of B yields

  $$f(s_1, \ldots, s_{i_0}, \ldots, s_{ar(f)}) \mathrel{B} f(t_1, \ldots, t', \ldots, t_{ar(f)})$$
  $$f(s_1, \ldots, \sigma(y), \ldots, s_{ar(f)}) \mathrel{B} f(t_1, \ldots, t'', \ldots, t_{ar(f)}),$$

  or in other words, $s \mathrel{B} f(t_1, \ldots, t', \ldots, t_{ar(f)})$ and $s' \mathrel{B} f(t_1, \ldots, t'', \ldots, t_{ar(f)})$.

- Case 2: There exists a panth rule ρ in T of the form

$$\frac{\{u_j \xrightarrow{a_j} y_j \mid j \in J\} \cup \{v_k P_k \mid k \in K\} \cup \{p_\ell \overset{b_\ell}{\nrightarrow} \mid \ell \in L\} \cup \{q_m \neg Q_m \mid m \in M\}}{f(x_1, \ldots, x_{ar(f)}) \xrightarrow{a} r}$$

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

A. for each $j \in J$, there is a proof from T, shorter than γ, for a closed transition rule $N_j / \sigma(u_j) \xrightarrow{a_j} \sigma(y_j)$ with $N_j \subseteq N$;
B. for each $k \in K$, there is a proof from T, shorter than γ, for a closed transition rule $N'_k / \sigma(v_k) P_k$ with $N'_k \subseteq N$;
C. for each $\ell \in L$ and $p' \in \mathcal{T}(\Sigma)$, $\sigma(p_\ell) \xrightarrow{b_\ell} p' \notin \mathtt{C}_{\alpha-1} \cup \mathtt{U}_{\alpha-1}$;
D. for each $m \in M$, $\sigma(q_m) Q_m \notin \mathtt{C}_{\alpha-1} \cup \mathtt{U}_{\alpha-1}$.

We define a substitution σ', such that the patience rules for wild arguments of f together with ρ and σ' prove

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

where $f(s_1, \ldots, s_{ar(f)}) \mathrel{B} \sigma'(f(x_1, \ldots, x_{ar(f)}))$ and $\sigma(r) \mathrel{B} \sigma'(r)$.

1. $\sigma'(z) = \sigma(z)$ for $z \notin \{x_i \mid i = 1, \ldots, ar(f)\} \cup \{y_j \mid j \in J\}$.
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 u_{j_0} does not contain variables from $\{x_i \mid i \text{ a wild argument of } f\}$, 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, by property A there is a proof from T for $N_{j_0}/\sigma(u_{j_0}) \stackrel{a_{j_0}}{\to} \sigma(y_{j_0})$ with $\mathtt{C}_{\alpha-1} \cup \mathtt{U}_{\alpha-1} \models N \supseteq N_{j_0}$, so $\sigma(u_{j_0}) \stackrel{a_{j_0}}{\to} \sigma(y_{j_0}) \in \mathtt{C}_\alpha$. Then II_α yields $\sigma'(u_{j_0}) \stackrel{a_{j_0}}{\to} w \in \mathtt{C}_\lambda$ for some $w \in \mathcal{T}(\Sigma)$ with $\sigma(y_{j_0}) \, B \, w$. We define $\sigma'(y_{j_0}) = w$, so that

$$\sigma(y_{j_0}) \, B \, \sigma'(y_{j_0}) \tag{10}$$

$$\sigma'(u_{j_0}) \stackrel{a_{j_0}}{\to} \sigma'(y_{j_0}) \in \mathtt{C}_\lambda. \tag{11}$$

4. Suppose v_{k_0} does not contain variables from $\{x_i \mid i \text{ a wild argument of } f\}$, 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, by property B there is a proof from T for $N'_{k_0}/\sigma(v_{k_0})P_{k_0}$ with $\mathtt{C}_{\alpha-1} \cup \mathtt{U}_{\alpha-1} \models N \supseteq N'_{k_0}$, so $\sigma(v_{k_0})P_{k_0} \in \mathtt{C}_\alpha$. Then II'_α yields

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

5. Suppose $u_{j_0} = x_{i_0}$ 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.

   By property A there is a proof from T for $N_{j_0}/s_{i_0} \stackrel{a_{j_0}}{\to} \sigma(y_{j_0})$ with $\mathtt{C}_{\alpha-1} \cup \mathtt{U}_{\alpha-1} \models N \supseteq N_{j_0}$, so $s_{i_0} \stackrel{a_{j_0}}{\to} \sigma(y_{j_0}) \in \mathtt{C}_\alpha$. Since moreover this proof from T is shorter than γ, $s_{i_0} \, B \, t_{i_0}$, and $a_{j_0} \neq \tau$, by induction the second option of III_α implies $t_{i_0} \stackrel{\varepsilon}{\to} t' \stackrel{a_{j_0}}{\to} t'' \in \mathtt{C}_\lambda$ 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

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

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

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

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

   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 positive premise in ρ.

6. Suppose $v_{k_0} = x_{i_0}$ 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.

   By property B there is a proof from T for $N_{k_0}/s_{i_0}P_{k_0}$ with $\mathtt{C}_{\alpha-1} \cup \mathtt{U}_{\alpha-1} \models N \supseteq N_{j_0}$, so $s_{i_0}P_{k_0} \in \mathtt{C}_\alpha$. Since moreover this proof from T is shorter than γ, and $s_{i_0} \, B \, t_{i_0}$, by induction III'_α implies $t_{i_0} \stackrel{\varepsilon}{\to} t'P_{k_0} \in \mathtt{C}_\lambda$ with $s_{i_0} \, B \, t'$. We define $\sigma'(x_{i_0}) = t'$, so that

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

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

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

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 positive premise in ρ.

7. Fix an $\ell_0 \in L$. The RBB safe format enforces that p_{ℓ_0} does not contain variables from $\{x_i \,|\, i \text{ a wild argument of } f\} \cup \{y_j \,|\, j \in J\}$. Since $\sigma(z)\, R\, \sigma'(z)$ for $z \notin \{x_i \,|\, i \text{ a wild argument of } f\} \cup \{y_j \,|\, j \in J\}$, Lemma 3.14 implies $\sigma(p_{\ell_0})\, R\, \sigma'(p_{\ell_0})$. Furthermore, by property C $\sigma(p_{\ell_0}) \stackrel{b_{\ell_0}}{\to} p' \notin \mathtt{C}_{\alpha-1} \cup \mathtt{U}_{\alpha-1}$ for all $p' \in \mathcal{T}(\Sigma)$, so $\mathrm{I}_{\alpha-1}$ yields

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

8. Fix an $m_0 \in M$. The RBB safe format enforces that q_{m_0} does not contain variables from $\{x_i \,|\, i \text{ a wild argument of } f\} \cup \{y_j \,|\, j \in J\}$. Since $\sigma(z)\, R\, \sigma'(z)$ for $z \notin \{x_i \,|\, i \text{ a wild argument of } f\} \cup \{y_j \,|\, j \in J\}$, Lemma 3.14 implies $\sigma(q_{m_0})\, R\, \sigma'(q_{m_0})$. Furthermore, by property D $\sigma(q_{m_0})Q_{m_0} \notin \mathtt{C}_{\alpha-1} \cup \mathtt{U}_{\alpha-1}$, so $\mathrm{I}'_{\alpha-1}$ yields

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

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

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

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

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

If i is a tame argument 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 positive premise of ρ then $\sigma'(x_i) = t_i$, so that $s_i\, B\, \sigma'(x_i)$. Finally, if i is a wild argument 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

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

$\sigma(x_i) = s_i$ for $i = 1, \ldots, ar(f)$, so $\sigma(x_i)\, R\, \sigma'(x_i)$ for tame arguments i of f and $\sigma(x_i)\, B\, \sigma'(x_i)$ for wild arguments i of f (see above). Furthermore, (10) and (16) together yield $\sigma(y_j)\, B\, \sigma'(y_j)$ for $j \in J$. Finally, $\sigma(z) = \sigma'(z)$ for $z \notin \{x_i \,|\, i = 1, \ldots, ar(f)\} \cup \{y_j \,|\, j \in J\}$. The RBB safe format enforces that variables x_i for wild arguments 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 agrees with $\leftrightarrows_{rb}$, 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*, 9(2):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] T. Basten. Branching bisimilarity is an equivalence indeed! *Information Processing Letters*, 58(3):141–147, 1996.

[7] B. Bloom. When is partial trace equivalence adequate? *Formal Aspects of Computing*, 6(3):317–338, 1994.

[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*, 146(1):24–54, 1998.

[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*, Seattle, 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] J.C. van de Pol. Operational semantics of rewriting with priorities. *Theoretical Computer Science*, 200(1/2):289–312, 1998.

[31] T.C. Przymusinski. On the declarative semantics of deductive databases and logic programs. In J. Minker, ed., *Foundations of Deductive Databases and Logic Programming*, Los Altos, pp. 193–216. Morgan Kaufmann, 1988.

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

[33] A. Tarski. A lattice-theoretical fixpoint theorem and its applications. *Pacific Journal of Mathematics*, 5:285–309, 1955.

[34] I. Ulidowski. Equivalences on observable processes. In *Proceedings 7th IEEE Symposium on Logic in Computer Science (LICS'92)*, Santa Cruz, California, pp. 148–159. IEEE Computer Society Press, 1992.

[35] 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.

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

[37] F.W. Vaandrager. On the relationship between process algebra and input/output automata. In *Proceedings 6th IEEE Symposium on Logic in Computer Science (LICS'91)*, Amsterdam, pp. 387–398. IEEE Computer Society Press, 1991.

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

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