Available online at www.sciencedirect.com
ScienceDirect
Electronic Notes in Theoretical Computer Science
ELSEVIER
Electronic Notes in Theoretical Computer Science 229 (2009) 85–106
www.elsevier.com/locate/entcs

Structural Operational Semantics with First-Order Logic

Muck van Weerdenburg[1] Michel A. Reniers[2]

Computer Science
Eindhoven University of Technology (TU/e)
Eindhoven, The Netherlands

Abstract

We define a formalism for Structural Operational Semantics (SOS) with first-order-logic formulas as premises. It is shown that in most uses (including all practical uses) this formalism has the same expressivity as SOS without first-order logic. Furthermore, we give a congruence format for (strong) bisimilarity. The latter is shown to be strongly related to the ntyft/ntyxt format.

Keywords: SOS, First-Order Logic, Expressiveness, Congruence.

1 Introduction

The use of Structural Operational Semantics (SOS) [17] to define semantics is widespread. In particular their effectiveness to define Transition-System Specifications (TSSs) [9] has played a role in this. Various flavours of SOS have been defined, including meta-theories on these specific flavours. The latter have given us formats that, if adhered to, guarantee well-definedness or congruence of equivalences, for example. In [1,14] overviews are given of the existing meta-theory involving SOS.

Recently, in [13], an attempt was made to extend TSSs with quantification over the variables that appear in a deduction rule and to define a congruence format for bisimilarity of such TSSs. The authors' motivation is mainly theoretical although they also refer to some specific examples in the literature [2,3,15,18] that make use of such TSSs. One of the two examples that is explicitly considered in [13] is the *weak termination predicate* of [2], which is defined by the following rules. (The other example is but a minimal variation on this.)

[1] Email: M.J.van.Weerdenburg@tue.nl
[2] Email: M.A.Reniers@tue.nl

1571-0661/$ – see front matter
doi:10.1016/j.entcs.2009.07.075

$$p\not\checkmark \Leftrightarrow \begin{cases} \text{(i)} & p \xrightarrow{\tau}\!\!\!\!\!\!\diagup\ \text{ and } p\checkmark\text{, or} \\ \text{(ii)} & p \xrightarrow{\tau} \text{ and, for each } q,\ p \xrightarrow{\tau} q \text{ implies } q\not\checkmark\text{.} \end{cases}$$

In traditional uses of SOS "predicates" of the form $x \xrightarrow{\tau}$ and $x \xrightarrow{\tau}\!\!\!\!\!\!\diagup$ in the premises of rules are typically treated as abbreviations. The predicate $x \xrightarrow{\tau}$ represents the statement "there is a y such that $x \xrightarrow{\tau} y$" and can be represented in traditional SOS by means of $x \xrightarrow{\tau} y$ with y a fresh variable w.r.t. the rule under consideration. The predicate $x \xrightarrow{\tau}\!\!\!\!\!\!\diagup$ represents the statement "for all y, it is not the case that $x \xrightarrow{\tau} y$". The later statement can not be expressed in a concise way. Definitions of such predicates in traditional SOS are often rather ad hoc. The presence of either one of these forms might already be sufficient reason to desire quantifications.

There are, however, some peculiarities in the approach of [13]. One of the most significant is that their rules are of a very restricted form. Every rule must adhere to the following format.

$$\exists_{\boldsymbol{z_0}}\ \forall_{\boldsymbol{z_1}}\ \exists_{\boldsymbol{z_2}}\ \frac{\varphi}{t \xrightarrow{l} u}$$

Here $\boldsymbol{z_0}$, $\boldsymbol{z_1}$ and $\boldsymbol{z_2}$ are sets of variables and φ is a formula built using the following syntax (where t is a (process) term, l a label and I a possibly infinite set of indices).

$$\varphi ::= t \xrightarrow{l} t \mid t \xrightarrow{l}\!\!\!\!\!\!\diagup\ t \mid \bigwedge_{i\in I} \varphi_i \mid \bigvee_{i\in I} \varphi_i$$

The limited number of alternations of quantifications is motivated by the fact that this is sufficient for the currently available applications, which is to some extent reasonable. However, the authors of [13] do note that "the ultimate goal would be to have a general [first-order] language and allow for all sorts of nested quantifiers." More problematic is that the rules as given in the example do not have a very direct translation to this formalism. One has to write their rules in this restricted – and somewhat unnatural – form. The above example would have to be written as follows.

$$\forall_y\ \frac{x \xrightarrow{\tau}\!\!\!\!\!\!\diagup\ y \wedge x\checkmark}{x\not\checkmark} \qquad \exists_{y'}\ \forall_y\ \frac{x \xrightarrow{\tau} y' \wedge (x \xrightarrow{\tau}\!\!\!\!\!\!\diagup\ y \vee y\not\checkmark)}{x\not\checkmark}$$

Note that the quantifications have been "pushed" outward and that the implication has been translated to an *or* (and that the *and* and *or* are sugar). We know of no standard result that states that in general a formula can be translated to this restricted format. This is mainly due to the possibility of infinite index sets in conjunctions and disjunctions, implying an infinitary logic (for which there typically is no prenex-like normal form).

Also w.r.t. traditional TSSs, the notation of [13] is somewhat deviant in the fact that unbound variables may only occur in the source of the conclusion of a rule. It should be noted that in examples of traditional SOS, where the sets of premises usually only consists of a small number of transitions, the set notations are usually omitted. We will not do so, for the sake of comparison. Where one would previously

write [3]

$$\frac{\{x \xrightarrow{a} x'\}}{x + y \xrightarrow{a} x'} \qquad \text{one must now write} \qquad \exists_{x'} \frac{x \xrightarrow{a} x'}{x + y \xrightarrow{a} x'}$$

to bind the occurrence of x' explicitly. There is an implicit binding of the variables from the source of the conclusion. As we will show in this paper, this needlessly increases the gap between traditional TSSs and those of [13]. In the syntax of the type of SOS specifications as introduced in this paper, named FOL-SOS (SOS with First-Order Logic), w.r.t. deduction rules without negative premises, the only difference with the traditional rule is that the set of premises are combined into a single formula by connecting them by means of conjunction. So the traditional rules [4]

$$\frac{\{x \xrightarrow{a} x'\}}{x + y \xrightarrow{a} x'} \qquad \frac{\{x \xrightarrow{a} x',\ y \xrightarrow{b} y'\}}{x \parallel y \xrightarrow{\gamma(a,b)} x' \parallel y'}$$

are represented as follows in the notations introduced in this paper

$$\frac{\bigwedge\{x \xrightarrow{a} x'\}}{x + y \xrightarrow{a} x'} \qquad \frac{\bigwedge\{x \xrightarrow{a} x', y \xrightarrow{b} y'\}}{x \parallel y \xrightarrow{\gamma(a,b)} x' \parallel y'}\ .$$

Another significant peculiarity is the confusing non-standard use of quantifiers. Take, for example, the rule for alternative composition given above. Most people would read the traditional rule as this "for all x, y and x', if $x \xrightarrow{a} x'$, then also $x + y \xrightarrow{a} x'$." It is then rather confusing to find an existential quantification for the variable x' in the syntax of [13]. The origin of this confusion is clear from the introduction of [13] where the authors say: "The semantics of a TSS [...] comes with an implicit *existential quantification* of valuations of variables used in the rule: if *there exists a* substitution on variables (appearing in the rule) such that the premises of the rule are satisfied, then the conclusion (with the same substitution applied to it) follows." Although the latter part of the sentence is correct – if a substitution satisfies the premise of a rule then the substitution also satisfies the conclusion – it does not imply an existential quantification. The reason for this is that *for every* substitution it holds that if it satisfies the premise it also satisfies the conclusion. So there is actually an *universal* quantification.

Next, the use of the previously mentioned abbreviations $t \xrightarrow{a}$ and $t \xrightarrow{a}\!\!\!\!\!\!\rightarrow$ in traditional TSS are not easily represented in the syntax of [13]. The reason is that the quantifiers need to be grouped in front of the deduction rule (and in limited alternations). As a consequence the rules are less readable. On the other hand, in the syntax of FOL-SOS these abbreviations are captured in a very intuitive way: $x \xrightarrow{a}$ can be replaced syntactically by $\exists_y\ (t \xrightarrow{a} y)$ (for y free w.r.t. t) and $t \xrightarrow{a}\!\!\!\!\!\!\rightarrow$ is syntactically replaced by $\forall_y\ (\neg\, t \xrightarrow{a} y)$(again, for y free w.r.t. t).

[3] The function symbol + represents alternative composition or nondeterministic choice as it occurs in many process algebras (e.g., [4]).

[4] The function symbol ∥ represents parallel composition as it occurs in many process algebras (e.g., [4]).

Finally, the format of [13] allows "universally" quantified variables (i.e. universally quantified in the sense of [13]) to occur in the targets of conclusions of rules. For example, the following rule is allowed.

$$\exists_x \ \forall_y \ \frac{x \xrightarrow{a} x}{x \xrightarrow{b} y}$$

The meaning of such rules is in no sense intuitive. One might expect, taking into account the meaning of quantifiers in the setting of [13], that the rule means something like "for all x such that $x \xrightarrow{a} x$ there is a y such that $x \xrightarrow{b} y$." The actual meaning is "for all x such that $x \xrightarrow{a} x$ we have that $x \xrightarrow{b} y$ for all y."

Besides the fact that there appears to be no intuition behind these semantics, we believe that existentially quantified variables in targets are of little (if any) use. One might be able to express that there is some term t such that $P(t)$, but in specifying a transition relation (which is the purpose of a TSS) this does not help one bit (since it is not clear which term t makes $P(t)$ true).

In this paper we give a formalism that allows arbitrary first-order logic in the premise (and only in the premise) and implicitly quantifies all free variables of both the premise and conclusion (as is standard). To illustrate, the weak termination predicate of [2] can be straightforwardly described in our formalism as follows (using the previously described abbreviations):

$$\frac{x \xrightarrow{\tau} \wedge\, x\checkmark}{x\checkmark\!\!\checkmark} \qquad \frac{x \xrightarrow{\tau} \wedge \forall_y \ (x \xrightarrow{\tau} y \Rightarrow y\checkmark\!\!\checkmark)}{x\checkmark\!\!\checkmark}$$

In Section 3.3 we give elegant rules, in FOL-SOS, that capture the previously mentioned examples of the use of quantification in SOS (i.e. those from [2,3,15,18]).

We show that allowing for the use of first-order logic formulas in premises does not increase expressiveness for well-defined TSSs (i.e. it only increases conciseness). Furthermore, we give a format for establishing congruence of (strong) bisimilarity [16,12] in the line of the ntyft/ntyxt format of [6].

2 Preliminaries

2.1 Terms, Predicates and Formulas

A *signature* Σ is a tuple of a set of *function symbols* $\mathbb{F}$ and a set of *predicate symbols* $\mathbb{P}$. *Terms* of a signature $\Sigma = \langle \mathbb{F}, \mathbb{P} \rangle$ and a set of variables $\mathbb{V}$, denoted by $T(\Sigma, \mathbb{V})$ are variables from $\mathbb{V}$ or functions symbols from $\mathbb{F}$ applied to zero or more terms. We write $T_c(\Sigma)$ (or just T_c if it is clear which Σ is used) for the set of *closed* terms (i.e. terms without variables) and $\mathrm{var}(t)$ for the set of variables occurring in term t. *Predicates* of Σ, denoted by $P(\Sigma, \mathbb{V})$, are predicate symbols from $\mathbb{P}$ applied to terms from $T(\Sigma, \mathbb{V})$. Mostly, variables will not be explicitly mentioned any further in our notations. We also refer to predicates as *atoms*. The set of variables that occur syntactically in atom a is denoted $\mathrm{var}(a)$.

Formulas $\varphi, \psi, \ldots$ over a signature $\Sigma = \langle \mathbb{F}, \mathbb{P} \rangle$ and variables $\mathbb{V}$ are of the form (with $a \in P(\Sigma, \mathbb{V})$ and $x \in \mathbb{V}$):

$$\varphi ::= a \mid \neg\, \varphi \mid \bigwedge\{\varphi, \ldots\} \mid \forall_x\, \varphi$$

For a formula φ, $\mathrm{FV}(\varphi)$ denotes the set of unbound (or free) variables that occur in φ. This function is defined inductively as follows: $\mathrm{FV}(a) = \mathrm{var}(a)$, $\mathrm{FV}(\neg\, \varphi) = \mathrm{FV}(\varphi)$, $\mathrm{FV}(\bigwedge \Phi) = \bigcup_{\varphi \in \Phi} \mathrm{FV}(\varphi)$, and $\mathrm{FV}(\forall_x\, \varphi) = \mathrm{FV}(\varphi) \setminus \{x\}$.

We assume the other standard logic operators are sugar in the usual way. That is, $\mathrm{true} = \bigwedge \varnothing$, $\mathrm{false} = \neg\, \mathrm{true}$, $\varphi \wedge \psi = \bigwedge\{\varphi, \psi\}$, $\varphi \vee \psi = \neg\, (\neg\, \varphi \wedge \neg\, \psi)$, $\varphi \Rightarrow \psi = \neg\, \varphi \vee \psi$, $\bigvee \Phi = \neg\, \bigwedge\{\neg\, \varphi \;:\; \varphi \in \Phi\}$ and $\exists_x\, \varphi = \neg\, \forall_x\, \neg\, \varphi$. We often write $\bigwedge_{s \in S} \varphi(s)$ for $\bigwedge\{\varphi(s) \;:\; s \in S\}$.

We write $a, b, c, \ldots$ for atoms and $l, m, n, \ldots$ for literals. Literals are atoms or negations of atoms. That is, for each literal there is an atom a such that $l = a$ or $l = \neg\, a$. We call the former *positive* literals and the latter *negative* literals. The set of all literals is denoted by $\mathbb{L}$. For sets L of literals, we write L_+ for the set of positive literals in L and L_- for the set of negative literals in L.

We use formulas as premises for SOS rules, but we still need to consider the set of literals used in such a formula. Therefore we introduce the following function. Note that when introducing a function f on formulas, we often also introduce an auxiliary function $\overline{f}$ for functions in a negative context.

Definition 2.1 We define the set of literals from a formula φ, notation $\mathrm{Lit}(\varphi)$, as follows.

$$\begin{array}{lll}
\mathrm{Lit}(a) & = \{a\} & \overline{\mathrm{Lit}}(a) = \{\neg\, a\} \\
\mathrm{Lit}(\neg\, \varphi) & = \overline{\mathrm{Lit}}(\varphi) & \overline{\mathrm{Lit}}(\neg\, \varphi) = \mathrm{Lit}(\varphi) \\
\mathrm{Lit}(\bigwedge \Phi) & = \bigcup_{\varphi \in \Phi} \mathrm{Lit}(\varphi) & \overline{\mathrm{Lit}}(\bigwedge \Phi) = \bigcup_{\varphi \in \Phi} \overline{\mathrm{Lit}}(\varphi) \\
\mathrm{Lit}(\forall_x\, \varphi) & = \bigcup_{t \in T_c(\Sigma)} \mathrm{Lit}(\varphi[t/x]) & \overline{\mathrm{Lit}}(\forall_x\, \varphi) = \bigcup_{t \in T_c(\Sigma)} \overline{\mathrm{Lit}}(\varphi[t/x])
\end{array}$$

For simplicity we often refer to the positive and negative literals of $\mathrm{Lit}(\varphi)$ as the positive, respectively negative literals of φ.

2.2 Transition-System Specifications

In traditional TSSs [8] typically one only considers transition relations and unary predicates. In general, there is no reason to limit the use of TSSs to only unary and binary predicates. However, the equivalences and congruence formats defined on TSSs typically only deal with these specific predicates. Note that we actually do not explicitly consider unary predicates here either. These can be easily coded as binary predicates (see [5]).

In traditional TSSs positive and negative transitions are used in the premises. Positive transitions are the same as our atoms. Negative transitions are of the form $t \overset{\alpha}{\not\rightarrow}$, and have the intuitive meaning of $\forall_x\, (\neg\, t \overset{\alpha}{\rightarrow} x)$ (for x free w.r.t. t). We write

$t, u, v, \ldots$ for transitions. For sets T of transitions, we write T_+ for the set of positive transitions in T and T_- for the set of negative transitions in T.

A *traditional rule (for signature Σ)* is of the form $\frac{P}{a}$ where P is a set of transitions and a is an atom. A *traditional Transition-System Specification* (traditional TSS) is a tuple $\langle \Sigma, R \rangle$ where Σ is a signature and R is a set of traditional rules. We typically assume a signature and only refer to a TSS by the set of rules R. We say a rule is closed when it contains no variables. Similarly we say that a TSS is closed when all its rules are closed. A TSS R induces a closed TSS R' where R' consists of all rules of R instantiated with all closing substitutions.

We adopt the least three-valued stable model as the semantics of a traditional TSS (see [6] for definitions), because it is possible to write TSSs that contradict themselves in two-valued models (e.g. $\frac{a \not\to}{a \to a}$). In a three-valued model, closed transitions are partitioned into those that certainly hold, those that may hold, and those that certainly don't hold. Usually, however, we are only interested in TSS that have a well-defined two-valued model (i.e., for each closed transition it is clear that it either holds or does not hold). Such TSSs are called well-defined or complete [8,6]

The extension of traditional TSSs of [13] is described in some detail in the introduction. For the rather complex semantics we refer to that paper.

3 TSS with First-Order Logic

In this section, we introduce *First-Order Logic Transition-System Specifications* (FOL-TSSs). A *FOL-rule* is of the form $\frac{\varphi}{a}$ where φ is a formula and a is an atom. A FOL-TSS is a tuple $\langle \Sigma, R \rangle$ where Σ is a signature and R a set of FOL-rules. Again, we usually refer to such a FOL-TSS just by R. The semantics of FOL-TSSs is given in two parts. First we consider the semantics of the formulas and then the semantics of FOL-TSSs. We say a rule is closed when it contains no unbound variables. Similarly we say that a FOL-TSS is closed when all its rules are closed. A FOL-TSS R induces a closed FOL-TSS R' where R' consists of all rules of R instantiated with all closing substitutions.

The use of unary predicates in FOL-SOS poses no additional problems to the use of transition relations. In fact, unary predicates can easily be encoded as transitions, in a way similar to that of [5], as follows: (1) occurrences of $P(y)$ in premises of deduction rules can be replaced with the formula $\exists_x\ t \ \to_P\ x$ for $x \notin \mathrm{var}(t)$, and (2) occurrences of $P(t)$ as a conclusion of a deduction rule can be replaced with the formula $t \ \to_P\ x$ where x is fresh w.r.t. the original deduction rule. In this coding we assumed that $\to_P$ is a fresh transition relation.

3.1 Infinitary First-Order Kleene Logic

To be able to establish whether or not the premise of a rule is satisfied, we must first define when a formula φ is true (or false). As we define the semantics of FOL-TSSs by means of least three-valued stable models (similar to the definition of traditional TSSs in Section 2), we also need a three-valued logic. We base our logic on Kleene's (strong) three-valued logic [10]. Here there is an additional value unknown besides

the classical true and false. This definition is very similar to that of [11].

Definition 3.1 Let L be a set of closed literals. We define when a closed formula φ is true for L, notation $L \vDash \varphi$, or not true (i.e. false) for L, notation $L \nvDash \varphi$.

$$
\begin{array}{ll}
L \vDash a & \text{iff } a \in L \\
L \vDash \neg\, \varphi & \text{iff } L \nvDash \varphi \\
L \vDash \bigwedge \Psi & \text{iff } L \vDash \psi \text{ for all } \psi \in \Psi \\
L \vDash \forall_x\, \psi & \text{iff } L \vDash \psi[t/x] \text{ for all } t \in T_c
\end{array}
\qquad
\begin{array}{ll}
L \nvDash a & \text{iff } \neg\, a \in L \\
L \nvDash \neg\, \varphi & \text{iff } L \vDash \varphi \\
L \nvDash \bigwedge \Psi & \text{iff } L \nvDash \psi \text{ for some } \psi \in \Psi \\
L \nvDash \forall_x\, \psi & \text{iff } L \nvDash \psi[t/x] \text{ for some } t \in T_c
\end{array}
$$

3.2 Semantics of Transition-System Specifications with First-Order Logic

First the semantics of FOL-TSSs is defined. The definitions resemble those for traditional TSSs.

Definition 3.2 Let R be a closed FOL-TSS, L be a set of closed literals and a a closed atom. We can derive a from the set L in R, notation $L \vdash_R a$, when there exists a well-founded upwardly branching tree with literals as nodes and of which

- the root is labelled by a;
- if a node is labelled by literal l and the nodes above it form the set L', then one of the following two cases hold:
  - $l \in L$ and $L' = \varnothing$;
  - l is a positive literal and there is a rule $\frac{\varphi}{l} \in R$ such that $L' \vDash \varphi$ and such that L' is minimal w.r.t. φ [5].

We say L is minimal for φ if $L \vDash \varphi$ and there is no $L' \subset L$ with $L' \vDash \varphi$. Similarly, we say L is minimal if there is no $L' \subset L$ such that $L' \vdash_R t$.

Definition 3.3 A pair $\langle C, U \rangle$ of sets of *closed atoms* (where C stands for *Certainly true* and U for *Unknown*; the third set, *certainly false* is determined by the atoms not in $C \cup U$) is called a three-valued stable model for a FOL-TSS R when $C \cap U = \varnothing$ and for all atoms a the following holds.

- $a \in C$ if, and only if, there is a set of closed negative literals N such that $N \vdash_R a$ and $C \cup U \vDash N$.
- $a \in C \cup U$ if, and only if, there is a set of closed negative literals N such that $N \vdash_R a$ and $C \vDash N$.

Definition 3.4 The semantics of a FOL-TSS R is given by the information-least three-valued stable model $\langle C, U \rangle$ of the closed FOL-TSS induced by R.

In general, we are only interested in complete models; we want that every atom is certainly true or false.

[5] Note that minimality is not a necessary requirement. It only facilitates the proofs.

Definition 3.5 A FOL-TSS is called *complete* if, and only if, its information-least three-valued stable model is $\langle C, \varnothing \rangle$ for some set of closed atoms C.

To establish whether the least three-valued stable model of a (FOL-)TSS is complete one can calculate this model and check that the set of unknowns is empty. This can, however, be quite some task with complicated TSSs. As in [6], we can easily define a notion of stratification for FOL-TSSs by considering the literals of the premise of a rule that implies that the least three-valued stable model is complete.

3.3 Examples

In this section we present rules for the examples found in literature (see [2,3,15,18]) mentioned in the introduction. As is customary in giving SOS for concrete examples we sometimes use the standard abbreviations: $t \xrightarrow{a}$ represents $\exists_x\ (t \xrightarrow{a} x)$ and $t \not\xrightarrow{a}$ represents $\forall_x\ (\neg\, t \xrightarrow{a} x)$ (where x is free w.r.t. t).

Example 3.6 In [3] a definition of an "is in a deadlock" predicate is given. Roughly translated (from Dutch):

> "We say that a process is in a deadlock [...] if it cannot do any action. That is, if p is such a process, we have that $p \not\xrightarrow{a} q$ and $p \not\xrightarrow{a} \checkmark$ for every a, q; [...]."

Let us write $\delta(p)$ for "p is in a deadlock." Then the definition of δ in FOL-TSS is as follows.

$$\frac{\bigwedge_{a \in A} (\forall_{x'}\ (\neg\, x \xrightarrow{a} x' \wedge \neg\, x \xrightarrow{a} \checkmark))}{\delta(x)}$$

Note that universal quantification (and existential quantification for that matter) are only used for variables over process terms. The universal quantification over the actions is therefore represented by means of an infinitary conjunction.

Example 3.7 In [2], the following definition of the *weak termination predicate* $\checkmark\!\!\!\checkmark$ is given.

$$p\checkmark\!\!\!\checkmark \Leftrightarrow \begin{cases} \text{(i)} & p \not\xrightarrow{\tau} \text{ and } p\checkmark, \text{ or} \\ \text{(ii)} & p \xrightarrow{\tau} \text{ and, for each } q,\ p \xrightarrow{\tau} q \text{ implies } q\checkmark\!\!\!\checkmark. \end{cases}$$

This is trivially translated to the following FOL-TSS.

$$\frac{\forall_y\ (\neg\, x \xrightarrow{\tau} y) \wedge x\checkmark}{x\checkmark\!\!\!\checkmark} \qquad \frac{\exists_y\ (x \xrightarrow{\tau} y) \wedge \forall_y\ (x \xrightarrow{\tau} y \Rightarrow y\checkmark\!\!\!\checkmark)}{x\checkmark\!\!\!\checkmark}$$

Note that due to the previously mentioned implicit existential quantification of unbound variables that are introduced in a premise, the existential quantification in the above rule can be omitted. Also using the abbreviations mentioned above, this gives the following alternative for the rules:

$$\frac{x \not\xrightarrow{\tau} \wedge\, x\checkmark}{x\checkmark\!\!\!\checkmark} \qquad \frac{x \xrightarrow{\tau} \wedge \forall_y\ (x \xrightarrow{\tau} y \Rightarrow y\checkmark\!\!\!\checkmark)}{x\checkmark\!\!\!\checkmark}$$

Note the close correspondence with the original informal statement of the weak termination predicate.

Example 3.8 In [2], also a definition of *semantical convergence* $\Downarrow$ is given by means of the following.

$$p \downarrow \text{ and (for each } q,\ p \xrightarrow{\tau} q \text{ implies } q \Downarrow) \text{ imply } p \Downarrow$$

Its translation to a FOL-TSS results in the following rule.

$$\frac{x \downarrow \wedge \forall_y\ (x \xrightarrow{\tau} y \Rightarrow y \Downarrow)}{x \Downarrow}$$

Example 3.9 In [18] traditional TSSs were extended with universal quantification of negative transitions. That is, there negative literals are written as $\forall \overrightarrow{x}.\ t \nrightarrow u$ where the variables from $\overrightarrow{x}$ are only allowed to occur in u. This can straightforwardly be transformed to our setting as $\forall_{x_0} \ldots \forall_{x_n}\ (\neg\, t \rightarrow u)$, with $\overrightarrow{x} = x_0, \ldots, x_n$ for some n. The semantics given in [18] corresponds to our semantics.

4 Expressiveness

First we discuss the expressiveness of FOL-TSS w.r.t. the TSS of [13]. Both types of TSS are capable of expressing every three-valued stable model. Therefore these frameworks are equally expressive. It remains to be seen if a translation between the settings can be given that preserves at least some of the structure of the TSSs.

Next, we discuss expressiveness of FOL-TSSs w.r.t. traditional TSSs. We show that traditional TSSs can straightforwardly be expressed as FOL-TSSs. We basically replace each negative transition $t \overset{\alpha}{\nrightarrow}$ by a formula $\forall_x\ (\neg\, t \xrightarrow{\alpha} x)$ (with x fresh w.r.t. t) and put a $\bigwedge$ around the premise. Note that this requires that there is a large enough supply of variables.

Definition 4.1 Let $\langle \Sigma, R \rangle$ be a traditional TSS. The translation of $\langle \Sigma, R \rangle$ to a FOL-TSS is $\langle \Sigma, R' \rangle$, where

$$R' = \left\{ \frac{\bigwedge P_+ \cup \{\forall_{x(t)}\ (\neg\, t \xrightarrow{\alpha} x(t))\ :\ t \overset{\alpha}{\nrightarrow} \in P_-\}}{a}\ :\ \frac{P}{a} \in R \right\},$$

where x is a mapping that associates with a term $t \in T(\Sigma)$ a variable that does not occur free in t, i.e., $x(t) \notin \mathrm{var}(t)$.

Note that using the abbreviations this amounts to putting an infinitary conjunction around the set of premises from the traditional TSS.

Theorem 4.2 *The three-valued stable models of a traditional TSS and its translation to a FOL-TSS are identical.*

Proof. A proof of this theorem is given in Appendix A. □

As this means that both TSSs have the same three-valued stable models it follows trivially that they have the same least three-valued stable model.

Corollary 4.3 *The least three-valued stable model of a traditional TSS and the least three-valued stable model of its translation to a FOL-TSS are identical.*

The above theorem states that for every traditional TSS there is an equivalent FOL-TSS. The converse is also true for complete FOL-TSSs.

Theorem 4.4 *For every complete FOL-TSS there is a traditional TSS with the same least three-valued stable model.*

Proof. Let R be a complete FOL-TSS and let $\langle C, U\rangle$ be the least three-valued stable model of R. As R is complete we have that $U = \varnothing$. Then the traditional TSS with rules $\left\{\frac{}{a} \;:\; a \in C\right\}$ is trivially equivalent to R. □

For FOL-TSSs that are not complete it is not the case that there is always an equivalent traditional TSS (under the assumption that the signature has to remain untouched). This is demonstrated by the following example. Since any FOL-TSS without negative literals is complete the example necessarily uses negative literals.

Example 4.5 Let $\Sigma = \langle \varnothing, \{a, b\}, \{\rightarrow\}\rangle$ and let R consist of the following rules:

$$\frac{}{a \rightarrow a} \qquad \frac{}{b \rightarrow b} \qquad \frac{\neg\, b \rightarrow a}{a \rightarrow b} \qquad \frac{\neg\, a \rightarrow b}{b \rightarrow a}$$

The least three-valued stable model of FOL-TSS $\langle \Sigma, R\rangle$ is $\langle \{a \rightarrow a, b \rightarrow b\}, \{a \rightarrow b, b \rightarrow a\}\rangle$. For (traditional) TSSs with least three-valued stable model $\langle C, U\rangle$, the only way to get an atom c in the set of unknowns U is to have a derivation $\frac{N}{c}$ where N is not empty and $C \vDash N$ but not $C \cup U \vDash N$. As N can only contain negative transitions, it must contain $a \not\rightarrow$ or $b \not\rightarrow$. However, we have that neither $C \vDash a \not\rightarrow$ or $C \vDash b \not\rightarrow$ are true. Thus there cannot be a traditional TSS with model $\langle C, U\rangle$.

As we are usually only interested in complete TSSs, we can reasonably say that FOL-TSSs are effectively as expressive as traditional TSSs.

5 Congruence Format

We define a congruence format for (strong) bisimilarity on the syntax of TSSs. Most of the requirements for this format are straightforward extensions of the ntyft/ntyxt format (see [6] for the definitions and the congruence theorem).

However, as we have formulas instead of sets of literals, these requirements appear a bit more complex. The following definitions give the functions that are to be used in the definition of the congruence format. For each definition we also give some (hopefully enlightening) examples of its use. Note that in these examples we use the sugar defined in Section 2 and implicitly work through it.

In traditional TSS, the ntyft/ntyxt format restricts the premises such that the right-hand sides of all positive transitions among the premises (in traditional TSS only positive transitions have a right-hand side) are variables and need to be different. Here we need a similar restriction, though, due to the presence of the full generality of first-order logic formulas as premises, establishing this is more involved.

First, we define a function dv (for distinct variables) to determine whether or not all right-hand sides of literals are variables and whether the unbound ones are unique or not. The function ubrhs (for unbound in right-hand side), also defined below, returns the unbound (or free) variables that occur in right-hand sides of literals of a given formula.

Definition 5.1 The functions dv and ubrhs are defined as follows.

$$\begin{aligned}
\mathrm{dv}(t \xrightarrow{\alpha} u) &= u \in \mathbb{V} \\
\mathrm{dv}(\neg\, \psi) &= \mathrm{dv}(\psi) \\
\mathrm{dv}(\textstyle\bigwedge \Psi) &= \forall_{\psi\in\Psi}\ \mathrm{dv}(\psi) \wedge \forall_{\psi,\psi'\in\Psi}\ (\psi \neq \psi' \Rightarrow \mathrm{ubrhs}(\psi) \cap \mathrm{ubrhs}(\psi') = \varnothing) \\
\mathrm{dv}(\forall_x\ \psi) &= \mathrm{dv}(\psi)
\end{aligned}$$

$$\begin{aligned}
\mathrm{ubrhs}(t \xrightarrow{\alpha} u) &= \mathrm{var}(u) & \mathrm{ubrhs}(\textstyle\bigwedge \Psi) &= \textstyle\bigcup_{\psi\in\Psi} \mathrm{ubrhs}(\psi) \\
\mathrm{ubrhs}(\neg\, \psi) &= \mathrm{ubrhs}(\psi) & \mathrm{ubrhs}(\forall_x\ \psi) &= \mathrm{ubrhs}(\psi) \setminus \{x\}
\end{aligned}$$

Example 5.2 We consider the formula $\bigwedge\{\forall_x\ (f(x) \to y), \neg\, c \to y\}$:

$$\begin{aligned}
&\mathrm{dv}(\textstyle\bigwedge\{\forall_x\ (f(x) \to y), \neg\, c \to y\}) \\
=\ &\mathrm{dv}(\forall_x\ (f(x) \to y)) \wedge \mathrm{dv}(\neg\, c \to y) \\
&\wedge (\mathrm{ubrhs}(\forall_x\ (f(x) \to y)) \cap \mathrm{ubrhs}(\neg\, c \to y) = \varnothing) \\
=\ &\mathrm{dv}(f(x) \to y) \wedge \mathrm{dv}(c \to y) \wedge ((\mathrm{ubrhs}(f(x) \to y) \setminus \{x\}) \cap \mathrm{ubrhs}(c \to y) = \varnothing) \\
=\ &\mathrm{true} \wedge \mathrm{true} \wedge ((\{y\}) \setminus \{x\}) \cap \{y\} = \varnothing) \\
=\ &\mathrm{false}
\end{aligned}$$

In [13], for the purpose of the congruence theorem, the authors require that the right-hand sides of positive literals are existentially bound variables and those of negative literals are universally bound variables and show that this restriction is necessary (in the sense that omitting the restriction ruins the congruence result). As we are in a comparable setting here, we also need such a restriction. The function ext, defined below, checks only the former, but by supplying the negation of a formula – thus interchanging existential and universal quantifications and positive and negative literals – the latter can also be established. Note that positive (negative) literals that occur in the left-hand side of an implication, as in Example 3.7 and Example 3.8, are considered negative (positive) literals and thus need to be universally (existentially) bound.

The function ext determines whether the right-hand sides of the positive literals (of the formula it is applied to) are existentially bound. It is parameterised by a set of variables that keeps track of those variables that are existentially bound in the current scope.

Definition 5.3 The function ext_S is defined as follows.

$$\begin{aligned}
\mathrm{ext}_S(t \xrightarrow{\alpha} u) &= u \in S & \overline{\mathrm{ext}}_S(t \xrightarrow{\alpha} u) &= \mathrm{true}\\
\mathrm{ext}_S(\neg\ \psi) &= \overline{\mathrm{ext}}_S(\psi) & \overline{\mathrm{ext}}_S(\neg\ \psi) &= \mathrm{ext}_S(\psi)\\
\mathrm{ext}_S(\bigwedge \Psi) &= \forall_{\psi\in\Psi}\ \mathrm{ext}_S(\psi) & \overline{\mathrm{ext}}_S(\bigwedge \Psi) &= \forall_{\psi\in\Psi}\ \overline{\mathrm{ext}}_S(\psi)\\
\mathrm{ext}_S(\forall_x\ \psi) &= \mathrm{ext}_{S\setminus\{x\}}(\psi) & \overline{\mathrm{ext}}_S(\forall_x\ \psi) &= \overline{\mathrm{ext}}_{S\cup\{x\}}(\psi)
\end{aligned}$$

Example 5.4 We consider the formula $\exists_x\ (t \to x \wedge \forall_x\ (\neg\ u \to x \wedge v \to x))$. First we check whether all positive literals have an existentially bound right-hand side.

$$\begin{aligned}
&\mathrm{ext}_\varnothing(\exists_x\ (t \to x \wedge \forall_x\ (\neg\ u \to x \wedge v \to x)))\\
=\ &\mathrm{ext}_{\{x\}}(t \to x \wedge \forall_x\ (\neg\ u \to x \wedge v \to x))\\
=\ &\mathrm{ext}_{\{x\}}(t \to x) \wedge \mathrm{ext}_{\{x\}}(\forall_x\ (\neg\ u \to x \wedge v \to x))\\
=\ &\mathrm{true} \wedge \mathrm{ext}_\varnothing(\neg\ u \to x \wedge v \to x)\\
=\ &\mathrm{ext}_\varnothing(\neg\ u \to x) \wedge \mathrm{ext}_\varnothing(v \to x)\\
=\ &\overline{\mathrm{ext}}_\varnothing(u \to x) \wedge \mathrm{false} = \mathrm{true} \wedge \mathrm{false}\\
=\ &\mathrm{false}
\end{aligned}$$

Next we check that all negative literals have a universally bound right-hand side.

$$\begin{aligned}
&\mathrm{ext}_\varnothing(\neg\ \exists_x\ (t \to x \wedge \forall_x\ (\neg\ u \to x \wedge v \to x)))\\
=\ &\overline{\mathrm{ext}}_\varnothing(\exists_x\ (t \to x \wedge \forall_x\ (\neg\ u \to x \wedge v \to x)))\\
=\ &\overline{\mathrm{ext}}_\varnothing(t \to x \wedge \forall_x\ (\neg\ u \to x \wedge v \to x))\\
=\ &\overline{\mathrm{ext}}_\varnothing(t \to x) \wedge \overline{\mathrm{ext}}_\varnothing(\forall_x\ (\neg\ u \to x \wedge v \to x))\\
=\ &\mathrm{true} \wedge \overline{\mathrm{ext}}_{\{x\}}(\neg\ u \to x \wedge v \to x)\\
=\ &\overline{\mathrm{ext}}_{\{x\}}(\neg\ u \to x) \wedge \overline{\mathrm{ext}}_{\{x\}}(v \to x)\\
=\ &\mathrm{ext}_{\{x\}}(u \to x) \wedge \mathrm{true}\\
=\ &\mathrm{true}
\end{aligned}$$

Similar to the format of [13] we cannot allow existentially bound variables x to depend on universally bound variables or variables that depend on universally bound variables that are bound within the scope of x. The function that specifies

this looks rather complex, but it basically checks for each existential quantification binding a variable x that there is no variable y on which x depends that is bound by a later universal quantification (or existential quantification following a later universal quantification).

Thereto the function h collects successive existentially quantified variables and, when it encounters a universally quantified variable, checks – with function k – whether these existentially bound variables do not depend on variables bound by other quantifications in their scope. The superscript S of h, a set of variables, keeps track of successive existentially quantified variables and subscript T of k, also a set of variables, keeps track of the quantified variables following those in superscript S.

Definition 5.5 The functions h^S, $\overline{h^S}$, and k^S_T are defined as follows.

$$\begin{aligned} h^S(t \xrightarrow{\alpha} u) &= \text{true} & \overline{h}^S(t \xrightarrow{\alpha} u) &= \text{true} \\ h^S(\neg\, \varphi) &= \overline{h}^S(\varphi) & \overline{h}^S(\neg\, \varphi) &= h^S(\varphi) \\ h^S(\bigwedge \Phi) &= \forall_{\varphi\in\Phi}\ h^S(\varphi) & \overline{h}^S(\bigwedge \Phi) &= \forall_{\varphi\in\Phi}\ \overline{h}^S(\varphi) \\ h^S(\forall_x\ \varphi) &= h^{\varnothing}(\varphi) \wedge k^S_{\{x\}}(\varphi) & \overline{h}^S(\forall_x\ \varphi) &= \overline{h}^{S\cup\{x\}}(\varphi) \end{aligned}$$

$$\begin{aligned} k^S_T(t \xrightarrow{\alpha} u) &= u \in S \setminus T \Rightarrow \mathrm{var}(t) \cap T = \varnothing \\ k^S_T(\neg\, \varphi) &= k^S_T(\varphi) \\ k^S_T(\bigwedge \Phi) &= \forall_{\varphi\in\Phi}\ k^S_T(\varphi) \\ k^S_T(\forall_x\ \varphi) &= k^S_{T\cup\{x\}}(\varphi) \end{aligned}$$

Example 5.6 We consider the formula $\exists_x\ (t \to x \wedge \exists_y\ (\forall_x\ (x \to y)))$.

$$\begin{aligned} & h^{\varnothing}(\exists_x\ (t \to x \wedge \exists_y\ (\forall_x\ (x \to y)))) \\ = {} & h^{\{x\}}(t \to x \wedge \exists_y\ (\forall_x\ (x \to y))) \\ = {} & h^{\{x\}}(t \to u) \wedge h^{\{x\}}(\exists_y\ (\forall_x\ (x \to y))) \\ = {} & \text{true} \wedge h^{\{x,y\}}(\forall_x\ (x \to y)) \\ = {} & h^{\varnothing}(x \to y) \wedge k^{\{x,y\}}_{\{x\}}(x \to y) \\ = {} & \text{true} \wedge ((y \in \{x,y\} \setminus \{x\}) \Rightarrow (\{x\} \cap \{x\} = \varnothing)) \\ = {} & \text{true} \Rightarrow \text{false} \\ = {} & \text{false} \end{aligned}$$

Using the previously defined functions we arrive at the following definition of the congruence format.

Definition 5.7 Let V be a set of variables. A formula φ is in the *FOL-ty format w.r.t. V* if the following holds.

(i) The right-hand sides of literals are distinct variables different from V; that is, $\mathrm{dv}(\varphi)$ and $\mathrm{ubrhs}(\varphi) \cap V = \varnothing$.

(ii) The right-hand side of a positive literal is existentially bound; that is, $\mathrm{ext}_{\mathrm{FV}(\varphi)\setminus V}(\varphi)$.

(iii) The right-hand side of a negative literal is universally bound; that is, $\mathrm{ext}_{\varnothing}(\neg\,\varphi)$.

(iv) The variable in the right-hand side of a positive literal is bound inside the scope of the variables in the left-hand side of that literal; that is, $h^{\mathrm{FV}(\varphi)\setminus V}(\varphi)$.

A FOL-rule r is in the *FOL-tyft format* if it is of the form

$$\frac{\varphi}{f(x_1,\ldots,x_n) \xrightarrow{\alpha} t},$$

for distinct variables $x_1,\ldots,x_n$, and φ is in the FOL-ty format w.r.t. $\{x_1,\ldots,x_n\}$. A FOL-rule r is in the *FOL-tyxt format* if it is of the form

$$\frac{\varphi}{x \xrightarrow{\alpha} t}$$

and φ is in the FOL-ty format w.r.t. $\{x\}$.

A FOL-TSS is in the FOL-tyft format if all its rules are in the FOL-tyft format. A FOL-TSS is in the FOL-tyxt format if all its rules are in the FOL-tyxt format. A FOL-TSS is in the FOL-tyft/tyxt format if all its rules are either in the FOL-tyft format or the FOL-tyxt format.

The rules of Example 3.6, Example 3.7 and Example 3.8 all satisfy the FOL-tyft/tyxt format. (The rules of Example 3.9 do not, but there the notion of bisimilarity is not relevant, as this is essentially meant to be a term rewriting system.)

Definition 5.8 Let T be a complete TSS with least three-valued stable model $\langle C, U\rangle$. A symmetric relation R is a *bisimulation relation* if for all (closed) terms p and q such that $p\,R\,q$ we have that

if $p \xrightarrow{\alpha} p' \in C$ for some $\xrightarrow{\alpha}$ and (closed) term p', then there is a (closed) term q' such that $q \xrightarrow{\alpha} q' \in C$ and $p'\,R\,q'$.

Two terms p and q are *bisimilar*, notation $\leftrightarrows$, if there is a bisimulation relation R with $p\,R\,q$.

As is custom in proofs for congruence formats, we assume that rules do not contain circular variable dependencies. A variable x depends on a variable y if there is a literal $t \xrightarrow{\alpha} u$ in φ such that x occurs in u and y occurs in t. For ntyft/ntyxt it has been shown in [7] that this requirement is not necessary. This is possibly also the case for our format.

Definition 5.9 A formula φ is well-founded if there is an α-equivalent φ' such that the variable-dependency graph of the right-hand sides of positive literals in φ' is well-founded. That is, $\{(x,y) \;:\; t \xrightarrow{\alpha} u \in \mathrm{Lit}(\varphi')_+ \wedge x \in \mathrm{var}(t) \wedge y \in \mathrm{var}(u)\}$ is a well-founded order.

A FOL-rule is well-founded if its premise is well-founded. A TSS is well-founded if all of its rules are well-founded.

Theorem 5.10 *Let T be a complete and well-founded FOL-TSS that is in the FOL-tyft/tyxt format. Bisimilarity on T is a congruence for all operators of T.*

Proof. A proof of this theorem is given in Appendix B. □

With respect to the use of unary predicates in deduction rules it should be noted that with the encoding by means of transitions as described before, the requirements are precisely those of the use of predicates in ntyft/ntyxt.

We compare our format with ntyft/ntyxt. We have already shown that traditional TSSs can easily be translated to our setting (Section 4).

Theorem 5.11 *Let $\langle \Sigma, R \rangle$ be a traditional TSS and let $\langle \Sigma, R' \rangle$ be the translation of $\langle \Sigma, R \rangle$ to a FOL-TSS (as given by Definition 4.1). If $\langle \Sigma, R \rangle$ is in the ntyft/ntyxt format, then $\langle \Sigma, R' \rangle$ is in the FOL-tyft/tyxt format.*

Proof. A proof of this theorem is given in Appendix C. □

6 Conclusion

We have introduced SOS with the full power of first-order logic in the premises and given an intuitive semantics that is strongly related to traditional semantics. Furthermore we have given a conservative extension of the ntyft/ntyxt congruence format and the format for congruence in the setting of [13]. We show that our format is strictly more expressive than the ntyft/ntyxt format. In order to also give a (partial) comparison of our format with the UNTyft/UNTyxt format of [13], we first need to have a (partial) translation from the TSSs of [13] to FOL-TSSs that preserves at least some of the logical structure of the deduction rules.

Acknowledgement

The authors would like to thank MohammadReza Mousavi for his valuable input.

References

[1] Aceto, L., W. J. Fokkink and C. Verhoef, *Structural operational semantics*, in: J. A. Bergstra, A. Ponse and S. A. Smolka, editors, *Handbook of Process Algebra, Chapter 3* (2001), pp. 197–292.

[2] Aceto, L. and M. Hennessy, *Termination, deadlock, and divergence*, Journal of the ACM **39** (1992), pp. 147–187.

[3] Baeten, J. and J. Bergstra, *Processen en procesexpressies*, Informatie **30** (1988), pp. 214–222.

[4] Baeten, J. and W. Weijland, "Process Algebra," Cambridge Tracts in Theoretical Computer Science **18**, Cambridge University Press, 1990.

[5] Baeten, J. C. M. and C. Verhoef, *A congruence theorem for structured operational semantics with predicates*, in: E. Best, editor, *Proceedings of the 4th International Conference on Concurrency Theory (CONCUR '93)*, Lecture Notes in Computer Science **715** (1993), pp. 477–492.

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

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

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

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

[10] Kleene, S. C., *On notation for ordinal numbers*, Journal of Symbolic Logic **3** (1938), pp. 150–155.

[11] Langholm, T., "Partiality, Truth and Persistence," CSLI Publications, 1988.

[12] Milner, R., "A Calculus of Communicating Systems," Lecture Notes in Computer Science **92**, Springer Verlag, 1980.

[13] Mousavi, M. R. and M. A. Reniers, *A congruence rule format with universal quantification*, Electronic Notes in Theoretical Computer Science **192** (2007), pp. 109–124.

[14] Mousavi, M. R., M. A. Reniers and J. F. Groote, *SOS formats and meta-theory: 20 years after*, Theoretical Computer Science **373** (2007), pp. 238–272.

[15] Mousavi, M. R., M. Sirjani and F. Arbab, *Formal semantics and analysis of component connectors in reo*, Electronic Notes in Theoretical Computer Science **154** (2006), pp. 83–99.

[16] Park, D., *Concurrency and automata on infinite sequences*, in: P. Deussen, editor, *Proceedings 5th GI Conference*, Lecture Notes in Computer Science **104** (1981), pp. 167–183.

[17] Plotkin, G. D., *The origins of structural operational semantics*, Journal of Logic and Algebraic Programming **60–61** (2004), pp. 3–15.

[18] van de Pol, J., *Operational semantics of rewriting with priorities*, Theoretical Computer Science **200** (1998), pp. 289–312.

A Proof of Theorem 4.2

First we relate the derivation trees of traditional TSSs and FOL-TSSs.

Lemma A.1 *Let $\langle \Sigma, R\rangle$ be a closed traditional TSS and let $\langle \Sigma, R'\rangle$ be its translation to a closed FOL-TSS. For all sets P of transitions and atoms a we have that $\vdash_R \frac{P}{a}$ if, and only if, $P_+ \cup \{\neg\, t \xrightarrow{\alpha} u \;:\; t \overset{\alpha}{\nrightarrow} \in P_- \wedge u \in T_c(\Sigma)\} \vdash_{R'} a$.*

Proof. Let P be a set of transitions. Assume that we have $\vdash_R \frac{P}{a}$ and that tree Υ is a witness of this. We show that Υ is a FOL-tree witnessing $P_+ \cup \{\neg\, t \xrightarrow{\alpha} u \;:\; t \overset{\alpha}{\nrightarrow} \in P_- \wedge u \in T_c(\Sigma)\} \vdash_{R'} a$ by induction on the depth of Υ.

- Υ consists of only one node, viz. transition u; by definition we have that $u = a$ and either $a \in P$ or $\frac{\varnothing}{a} \in R$. If $a \in P$, then trivially $a \in P_+$ and therefore also $P_+ \cup \{\neg\, t \xrightarrow{\alpha} u \;:\; t \overset{\alpha}{\nrightarrow} \in P_- \wedge u \in T_c(\Sigma)\} \vdash_{R'} a$. If $\frac{\varnothing}{a} \in R$, then we also have $\frac{\text{true}}{a} \in R'$. As $\varnothing \vDash \text{true}$ trivially holds (and $\varnothing$ is obviously minimal w.r.t. true), we have that Υ is a FOL-tree witnessing $P_+ \cup \{\neg\, t \xrightarrow{\alpha} u \;:\; t \overset{\alpha}{\nrightarrow} \in P_- \wedge u \in T_c(\Sigma)\} \vdash_{R'} a$.
- Υ consists of more than one node; by definition we have that the root of Υ is a. Let K be the set of (direct) subtrees of the root. By definition we have that every tree $k \in K$ is a witness for $\vdash_R \frac{P}{b}$, where b is the root of k. Then by induction we have that such a k is also a witness for $P_+ \cup \{\neg\, t \xrightarrow{\alpha} u \;:\; t \overset{\alpha}{\nrightarrow} \in P_- \wedge u \in T_c(\Sigma)\} \vdash_{R'} b$. From this is follows that to show that Υ is a tree witnessing $P_+ \cup \{\neg\, t \xrightarrow{\alpha} u \;:\; t \overset{\alpha}{\nrightarrow} \in P_- \wedge u \in T_c(\Sigma)\} \vdash_{R'} a$, we only need to show that the node-condition (for FOL-trees) holds for the root of Υ. Let P' be the set of transitions that are a root of a tree in K. We know that either $a \in P$ and $P' = \varnothing$ or there is a rule $\frac{P'}{a} \in R$. In the first case we are done. The second case means that we have $\frac{\bigwedge P'_+ \cup \{\forall_x\ (\neg\, t' \xrightarrow{\alpha} x) : t' \overset{\alpha}{\nrightarrow} \in P'_-\}}{a} \in R'$. But then also we trivially have that $P'_+ \cup \{\neg\, t' \xrightarrow{\alpha} u \;:\; t' \overset{\alpha}{\nrightarrow} \in P'_- \wedge u \in T_c(\Sigma)\} \vDash \bigwedge P'_+ \cup \{\forall_x\ (\neg\, t' \xrightarrow{\alpha} x) \;:\; t' \overset{\alpha}{\nrightarrow} \in P'_-\}$ and that $P'_+ \cup \{\neg\, t' \xrightarrow{\alpha} u \;:\; t' \overset{\alpha}{\nrightarrow} \in P'_- \wedge u \in T_c(\Sigma)\}$ is minimal w.r.t. $\bigwedge P'_+ \cup \{\forall_x\ (\neg\, t' \xrightarrow{\alpha} x) \;:\; t' \overset{\alpha}{\nrightarrow} \in P'_-\}$.

Assume that we have a set P of transitions such that $P_+ \cup \{\neg\, t \xrightarrow{\alpha} u \;:\; t \overset{\alpha}{\not\to} \in P_-\} \vdash_{R'} a$ and that FOL-tree Υ is a witness of this. The proof that Υ is also a tree witnessing $\vdash_R \frac{P}{a}$ can easily be given in a similar way as the converse. □

Lemma A.2 *Let $\langle \Sigma, R\rangle$ be a closed traditional TSS and let $\langle \Sigma, R'\rangle$ be its translation to a closed FOL-TSS. If $P \vdash_{R'} a$ for some minimal set P of literals and atom a, then there is a set P' of positive and negative transitions such that $P = P'_+ \cup \{\neg\, t \xrightarrow{\alpha} u \;:\; t \overset{\alpha}{\not\to} \in P'_-\}$.*

Proof. We prove this by induction on the tree witnessing $P \vdash_{R'} a$. In the base case we have that root node a is the only node in the tree. We either have that $a \in P$ or that there is a rule $\frac{\varphi}{a} \in R'$ such that $\varnothing \vDash \varphi$. If $a \in P$ then, due to minimality, $P = \{a\}$, and hence we take $P' = P$. Otherwise we have that $P = \varnothing$ as P is minimal and thus we take $P' = \varnothing$ as well.

Now let root node a have a non-empty set of children Q. For every $b \in Q_+$ we trivially have that $P \vdash_R b$ and thus that there is a minimal $P_b \subseteq P$ with $P_b \vdash_R b$. Note that due to the minimality we have that $P = \bigcup_{b\in Q_+} P_b \cup Q_-$. By induction we have P'_b such that $P_b = P'_{b+} \cup \{\neg\, t \xrightarrow{\alpha} u \;:\; t \overset{\alpha}{\not\to} \in P'_{b-}\}$.

For every $b = \neg\, t \xrightarrow{\beta} u \in Q_-$ we know that this b is necessary support for φ. Also, as $\langle \Sigma, R'\rangle$ is the translation of a traditional TSS, we have that $\varphi = \bigwedge S_+ \cup \{\forall_x\, (\neg\, t' \xrightarrow{\alpha} x) \;:\; t' \overset{\alpha}{\not\to} \in S_-\}$ for some set S of transitions. Clearly, negative literal b is not required for the elements in S_+. Thus there must be a $t' \overset{\beta}{\not\to} \in S_-$ such that b supports $\forall_x\, (\neg\, t' \xrightarrow{\beta} x)$. The we have that $t' = t$ and, for all u', that $\neg\, t \xrightarrow{\beta} u'$ must also be in Q_-. Therefore, if Q' is the set of negative transitions supported by a $b \in Q_-$, then we have that $Q_- = \{\neg\, t \xrightarrow{\alpha} u \;:\; t \overset{\alpha}{\not\to} \in Q'\}$.

We now have that $P = \bigcup_{b\in Q_+} P_b \cup Q_- = \bigcup_{b\in Q_+} (P'_{b+} \cup \{\neg\, t \xrightarrow{\alpha} u \;:\; t \overset{\alpha}{\not\to} \in P'_{b-}\}) \cup \{\neg\, t \xrightarrow{\alpha} u \;:\; t \overset{\alpha}{\not\to} \in Q'\} = \bigcup_{b\in Q_+} P'_{b+} \cup \{\neg\, t \xrightarrow{\alpha} u \;:\; t \overset{\alpha}{\not\to} \in \bigcup_{b\in Q_+} P'_{b-} \cup Q'\}$ and we therefore take $P' = \bigcup_{b\in Q_+} P'_{b+} \cup \bigcup_{b\in Q_+} P'_{b-} \cup Q' = \bigcup_{b\in Q_+} P'_b \cup Q'$. □

We show that every three-valued stable model of $\langle \Sigma, R\rangle$ is also a three-valued stable model of $\langle \Sigma, R'\rangle$ and vice versa.

Let $\langle C, U\rangle$ be a three-valued stable model of $\langle \Sigma, R\rangle$. Let $a \in C$. Then we have that $\vdash_R \frac{N}{a}$ for some set N of negative transitions such that $C \cup U \vDash N$. By Lemma A.1 we then also have that $\{\neg\, t \xrightarrow{\alpha} u \;:\; t \overset{\alpha}{\not\to} \in N \wedge u \in T_c(\Sigma)\} \vdash_{R'} a$. As $C \cup U \vDash N$ means that for all $t \overset{\alpha}{\not\to} \in N$ and closed terms u it holds that $t \xrightarrow{\alpha} u \notin C \cup U$. This trivially means that $\{\neg\, t \xrightarrow{\alpha} u \;:\; t \overset{\alpha}{\not\to} \in N \wedge u \in T_c(\Sigma)\} \vdash_{R'} a$. A similar reasoning can be given for all $a \in U$. Thus $\langle C, U\rangle$ is a three-valued stable model for $\langle \Sigma, R'\rangle$.

Let $\langle C, U\rangle$ be a three-valued stable model of $\langle \Sigma, R'\rangle$. Let $a \in C$. Then we have that $N \vdash_{R'} a$ for some set N of negative literals such that $C \cup U \vDash N$. By Lemma A.2 we know that $N = N'_+ \cup \{\neg\, t \xrightarrow{\alpha} u \;:\; t \overset{\alpha}{\not\to} \in N'_- \wedge u \in T_c(\Sigma)\}$ for some set N' of positive and negative transitions. Then, by Lemma A.1 and the fact that $N'_+ = \varnothing$ as N only contains negative literals, we also have that $\vdash_R \frac{N}{a}$. We also trivially have that from $C \cup U \vDash N$ it follows that $C \cup U \vDash N'$. Again, a similar reasoning can be given for all $a \in U$. Thus $\langle C, U\rangle$ is a three-valued stable model for $\langle \Sigma, R\rangle$. This concludes the proof of Theorem 4.2.

B Proof of Theorem 5.10

In the proof of Theorem 5.10 we need to show that, given that $\varphi\sigma$ holds for some formula φ and substitution σ, there is a specific substitution τ such that $\varphi\tau$ also holds. Essential in showing this are the values that are chosen for the bound variables in φ. To be able to talk about these bound variables we make the structure of Definition 3.1 more explicit by introducing (dis)proof trees.

As one bound variable may be assigned different values depending on the values of other variables (e.g. $\forall_x\, (\exists_y\, (x = y))$), we also use labels to differentiate between such assignments. Each node of a tree is labelled with a sequence indicating the choices that have been made on the path from the root to this node. For instance, in the proof tree for $\bigwedge\{\varphi, \psi\}$ with $\psi = \forall_x\, (\bigwedge\{x \to y, \neg\, t \to x\})$, the subtree for $t \to x$ where x is "assigned" the value u will have label $\psi.u.(\neg\, t \to x)$. Also, in that subtree x will be replaced with $x^{\psi.u}$, which makes it easy to determine where x is bound and which value x is supposed to have.

Definition B.1 We define what a proof tree and disproof tree are for a formula φ. A proof tree with label λ for φ is built inductively on φ as follows:

- $t \xrightarrow{\alpha} u$: leaf $t \xrightarrow{\alpha} u$.
- $\neg\, \psi$: let T be the disproof tree with label λ for ψ; node $\neg$ with subtree T.
- $\bigwedge \Psi$: let T_ψ be the proof trees with label $\lambda.\psi$ for $\psi \in \Psi$; node $\bigwedge$ with subtrees T_ψ for all $\psi \in \Psi$.
- $\forall_x\, \psi$: let T_t be the proof tree with label $\lambda.t$ for $\psi[x^{\lambda.t}/x]$ for all $t \in T(\Sigma)$; node $\forall_x$ with subtrees T_t for all $t \in T(\Sigma)$.

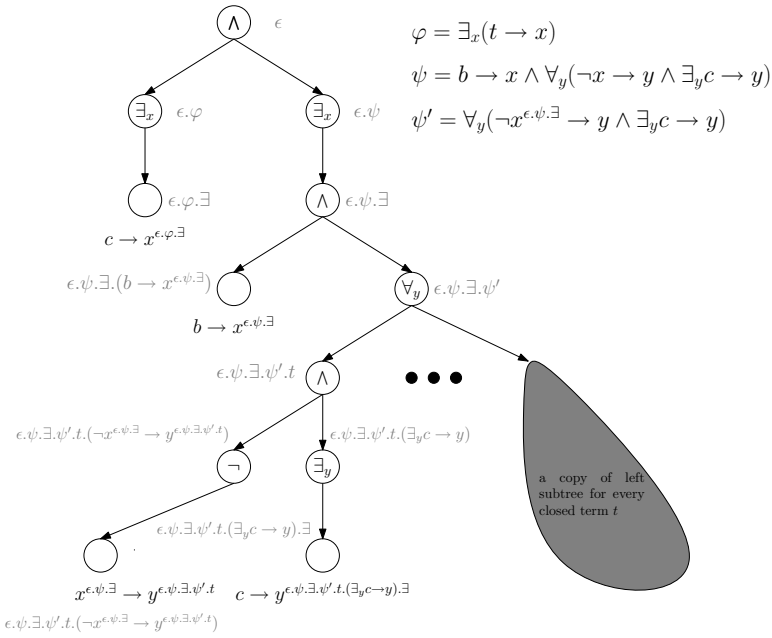

Fig. B.1. Proof tree

A disproof tree with label λ for φ is built inductively on φ as follows:

- $t \xrightarrow{\alpha} u$: leaf $t \xrightarrow{\alpha} u$.
- $\neg\ \psi$: let T be the proof tree with label λ for ψ; node $\neg$ with subtree T.
- $\bigwedge \Psi$: let T_ψ be the disproof trees with label $\lambda.\psi$ for $\psi \in \Psi$; node $\bigwedge$ with subtree T_ψ for some $\psi \in \Psi$.
- $\forall_x\ \psi$: let T be the disproof tree with label $\lambda.\exists$ for $\psi[x^{\lambda.\exists}/x]$; node $\forall_x$ with subtree T.

We often interpret the leaves of a (dis)proof tree as literals (i.e. a proof-tree leaf is a positive literal and a disproof-tree leaf is a negative literal).

Proposition B.2 *If Υ is a proof tree for φ and σ is a substitution such that for all universally quantified variables x in Υ, $t \in T(\Sigma)$ and labels λ, $\sigma(x^{\lambda.t}) = t$, then, with L the leaves of Υ, it holds that $L\sigma \vDash \varphi$.*

Example B.3 The proof tree for the formula $\exists_x\ (a \to x) \wedge \exists_x\ (b \to x \wedge \forall_y\ (\neg\ x \to y \wedge \exists_y\ c \to y))$ is given in Figure B.1.

Lemma B.4 *If φ is in the FOL-ty format w.r.t. V and Υ is a proof tree for φ, then we have the following properties on the leaves L of Υ.*

(i) *Let $u \xrightarrow{\alpha} v$ be a positive literal from L. Then v is an existentially bound variable x^λ for some $x \notin V$ and some label λ and there is no u' with $u' \neq u$ such that $u' \xrightarrow{\alpha} v \in L$.*

(ii) *Let $\neg\ u \xrightarrow{\alpha} v$ be a negative literal from L. Then v is an universally bound variable $x \notin V$.*

(iii) *If φ is well-founded, then so is L.*

Proof.

(i) This follows from $\mathrm{dv}(\varphi)$, $\mathrm{ubrhs}(\varphi) \cap V = \varnothing$, $\mathrm{ext}_{\mathrm{FV}(\varphi)\setminus V}(\varphi)$ and $h^{\mathrm{FV}(\varphi)\setminus V}(\varphi)$. The latter makes sure that there is no problem in duplicating positive literals (which occur only once in φ) in the proof tree. This is the case because for each positive literal $u \xrightarrow{\alpha} x^\lambda \in L$ we have that the variables in u are bound outside the scope of x and are thus uniquely labelled with labels that are a prefix of λ.

(ii) This follows from $\mathrm{dv}(\varphi)$, $\mathrm{ubrhs}(\varphi) \cap V = \varnothing$ and $\mathrm{ext}_\varnothing(\neg\ \varphi)$.

(iii) We transform φ to φ' by subscripting all variables with the set of labels they have gotten in the proof tree Υ. That is, for a universally bound variable x in φ we have the following. The $\forall_x$ occurs at several places as root of a subtree in the proof tree. Each subtree has a label λ and in those subtrees the variable x is "instantiated" with $x^{\lambda.u}$ for all closed terms u. We replace this variable x within its scope in φ by x_Λ, where $\Lambda = \{\lambda.u \ : \$ "there is a subtree of Υ with label λ and $u \in T_c(\Sigma)$"$\}$.

For existentially quantified variables x in φ we do the same. That is, we replace it with x_Λ, where $\Lambda = \{\lambda.\exists \ : \$ "there is a subtree of Υ with label λ"$\}$.

It is straightforward to prove that φ' is α-equivalent to φ.

As φ is well-founded, we get a well-founded order $<$ on the variables of φ'. Let $<_L$ be the dependency order on the variables of L. We have that for each x^λ and y^μ such that $x^\lambda <_L y^\mu$ there is a literal in L with atom $u \xrightarrow{\alpha} y^\mu$ and $x^\lambda \in \mathrm{var}(u)$. But this means that there is a similar literal $u' \xrightarrow{\alpha} y_M$ in φ' such that $\mu \in M$ and there is a $x_\Lambda \in \mathrm{var}(u)$ with $\lambda \in \Lambda$. But then also $x_\Lambda < y_M$.

Now, assume that $<_L$ is not well-founded. That means that there is a x^λ with $x^\lambda < y_1^{\mu_1} < \ldots < y_n^{\mu_n} < x^\lambda$. By the previous considerations we then also have Λ and M_i for $1 \leq i \leq n$ such that $x^\Lambda < y_1^{M_1} < \ldots < y_n^{M_n} < x^\Lambda$. However, this contradicts that $<$ is well-founded and therefore we may conclude that $<_L$ is also well-founded.

□

We must show that $\leftrightarrows$ is a congruence relation. That is, for all operators f and assuming $p_1, \ldots, p_n$ and $q_1, \ldots, q_n$ (with n the arity of f) such that $p_i \leftrightarrows q_i$ for all $1 \leq i \leq n$, we must show that $f(p_1, \ldots, p_n) \leftrightarrows f(q_1, \ldots, q_n)$.

Let R be the smallest congruence containing $\leftrightarrows$. That is, let R be the smallest relation such that:

- $p \ R \ q$, if $p \leftrightarrows q$;
- $f(p_1, \ldots, p_n) \ R \ f(q_1, \ldots, q_n)$ for all function symbols f (with arity n), if $p_i \ R \ q_i$ for all i with $1 \leq i \leq n$.

By showing that R is a bisimulation relation we trivially get that $\leftrightarrows$ is a congruence. That R satisfies the transfer conditions for the pairs $\langle p, q \rangle$ with $p \leftrightarrows q$ is trivial. We therefore only consider terms $p = f(p_1, \ldots, p_n)$ and $q = f(q_1, \ldots, q_n)$ with $p_i \ R \ q_i$ for all i with $1 \leq i \leq n$.

We use induction on the "length" of the derivation $p \xrightarrow{\gamma} p' \in C$. This is firstly done by induction on approximations of the least three-valued stable model of T and within this induction on the depth of the tree witnessing $p \xrightarrow{\gamma} p'$.

We define approximations $\langle C_\alpha, U_\alpha \rangle$, for ordinals α, of the least three-valued stable model $\langle C, U \rangle$ of T as follows.

- $C_\alpha = \{a \ : \ \exists_{\beta<\alpha} \ (\exists_N \ (N \vdash_T a \wedge C_\beta \cup U_\beta \vDash N))\}$
- $U_\alpha = \{a \ : \ \forall_{\beta<\alpha} \ (\exists_N \ (N \vdash_T a \wedge C_\beta \vDash N))\} \setminus C_\alpha$

We clearly have that $C_0 = \varnothing$ and $U_0 = \mathbb{L}$ and $C_\alpha \subseteq C_\beta$ and $U_\alpha \supseteq U_\beta$. The latter, with the Knaster-Tarski theorem (and the identity function), means that there are ordinals λ_1 and λ_2 such that $C_{\lambda_1} = \bigcup_\alpha C_\alpha$ and $U_{\lambda_2} = \bigcap_\alpha U_\alpha$. Thus for all ordinals μ_1 and μ_2 with $\lambda_1 \leq \mu_1$, $\lambda_2 \leq \mu_1$ and $\mu_1 \leq \mu_2$ we trivially have that $C_{\mu_1} = C_{\mu_2}$ and $U_{\mu_1} = U_{\mu_2}$. Specifically, we have that $C = C_{\lambda_1}$ and $U = U_{\lambda_2}$.

We now (simultaneously) prove the following statements with induction on α. If pRq, then

(i) if $p \xrightarrow{\gamma} p' \in C_\alpha$ for some p', then there exists a q' such that $q \xrightarrow{\gamma} q' \in C$ and $p' \ R \ q'$;

(ii) if $p \xrightarrow{\gamma} p' \in C$ for some p', then there exists a q' such that $q \xrightarrow{\gamma} q' \in C_\alpha \cup U_\alpha$ and $p' \ R \ q'$.

The first statement is the logical adaption of the transfer condition of bisimilarity and the second is needed in the proof of the first (and vice versa).

Let $p = f(p_1, \ldots, p_n)$ and $q = f(q_1, \ldots, q_n)$ with $p_i \ R \ q_i$ for all i with $1 \leq i \leq n$. We prove the first statement. The proof of the second statement is very similar to this proof (and therefore omitted).

Assume $p \xrightarrow{\gamma} p' \in C_\alpha$ for some closed process p'. Then by definition we have that there is a set N of negative literals such that $N \vdash_T p \xrightarrow{\gamma} p'$ and $C_\beta \cup U_\beta \vDash N$ for some $\beta < \alpha$. Let χ be the tree witnessing $N \vdash_T p \xrightarrow{\gamma} p'$ and L the set of predecessors of the root $p \xrightarrow{\gamma} p'$ of χ. By definition we then have a rule $\rho = \dfrac{\varphi}{f(x_1,\ldots,x_n) \xrightarrow{\gamma} u}$ and substitution σ such that $\sigma(x_i) = p_i$ for all i with $1 \leq i \leq n$, $u\sigma = p'$ and $L \vDash \varphi\sigma$.

We show that there is a q' such that $q \xrightarrow{\gamma} q' \in C$ and $p' \ R \ q'$ by induction on the length of χ. We observe that the subtrees of χ are witnesses for the statements $N \vdash_T l$ for all $l \in L$. As these subtrees are obviously smaller than χ itself, we can apply the induction hypothesis to the elements of L.

We construct a substitution τ such that we can derive that there is an N' with $N' \vdash_T f(q_1, \ldots, q_n) \xrightarrow{\gamma} u\tau$. To do this we want to use the proof for $\varphi\sigma$ to prove $\varphi\tau$. For this we need to be able to mimic every element l of L. If l is a positive literal, this will follow from the bisimulation relation. If it is a negative literal, then

there are two cases: either we can mimic l or we can not. If we can, we are satisfied. Otherwise we will show that there is another proof that does not depend on this negative literal.

Let Υ be a proof tree of φ with leaves L' and substitution σ' such that $\varphi\sigma' = \varphi\sigma$, $(f(x_1,\ldots,x_n) \xrightarrow{\gamma} u)\sigma' = (f(x_1,\ldots,x_n) \xrightarrow{\gamma} u)\sigma$ and $L = L'\sigma'$. That is, $\Upsilon\sigma'$ is a witness for $L \vDash \varphi\sigma$. We say that a negative literal $\neg\, t \xrightarrow{\delta} y$ is not required (w.r.t. Υ) if there is a process r such that $\sigma'(y)Rr$ and $t\sigma' \xrightarrow{\delta} r \in C_\beta \cup U_\beta$. That is, a negative literal $\neg\, t \xrightarrow{\delta} y$ is not required if there are (one or more) terms that are related to $\sigma'(y)$ (the value that y has in the actual proof of $\varphi\sigma$) by R that are "reachable" from t (i.e. $t \xrightarrow{\delta} r$). Note that this requirement comes from the negation of the transfer condition of the bisimulation relation; we can only deduce that a term u related to t (i.e. uRt) cannot reach $\sigma'(y)$ if t cannot reach any term of the equivalence class of $\sigma'(y)$ (induced by R). We remove all not-required negative literals from the proof as these literals (and only these) might not be possible to mimic. We do this from the outermost $\forall$s of φ inwards.

We use a recursive procedure to eliminate all not-required negative literals from Υ. The result will be a tree Υ'. In every step of the procedure we remove all negative literals from the outermost relevant $\forall$ (i.e. a $\forall$ that binds a variable x that occurs as the right-hand side of a not-required negative literal). While doing this, we derive a σ'' from σ' such that for all positive literals $l \in L'$ we have that $C_\alpha \vDash l\sigma''$, for all negative literals $l \in L'$ we have that $C_\beta \cup U_\beta \vDash l\sigma''$, and for all universally quantified variables x from φ we have that $\sigma''(x^{\lambda.t})Rt$ for all t. We start with $\sigma'' = \sigma'$.

Now, let $\neg\, t \xrightarrow{\delta} y \in L'$ be a literal that is not required such that there is no literal $\neg\, s \xrightarrow{\delta'} z \in L'$ where y is bound within the scope of z and such that this negative literal is also not required. By definition we then have a process r such that $\sigma''(y)Rr$ and $t\sigma'' \xrightarrow{\delta} r \in C_\beta \cup U_\beta$. As y is universally bound, we know that there is a $\forall_x$ in Υ' such that $y = x^{\lambda.\sigma''(y)}$, for some label λ, and that $\neg\, t \xrightarrow{\delta} y$ is a leaf of the subtree corresponding to $\sigma''(y)$ of this $\forall_x$. We replace this subtree by the subtree corresponding to r where each variable $z^{\lambda.r.\mu}$ is substituted by $z^{\lambda.\sigma''(y).\mu}$ (such that the resulting tree is correctly labelled). Finally we change σ'' such that $\sigma''(z^{\lambda.\sigma''(y).\mu})$ becomes $\sigma''(z^{\lambda.r.\mu})$ for all variables z and labels μ. That is, we also make sure that the values assigned to the variables in the new subtree get the values form the subtree corresponding to r. It is trivial to see that σ'' satisfies the requirements for this new tree.

We now construct a substitution τ that, together with proof tree Υ' for φ, gives a q' such that $q \to q' \in C$. We define $\tau(x_i) = q_i$ for all $1 \leq i \leq n$ and take $\tau(x^{\lambda.t}) = t$ for all variables x, terms t and labels λ (as is required for τ for be a witness for $\varphi\tau$). Also, for all variables x^λ bound existentially in Υ' that do not occur in the right-hand sides of leaves of Υ' we set $\tau(x^\lambda) = \sigma''(x^\lambda)$. Note that we have that $\tau(x)R\sigma''(x)$ for all variables x for which we have defined τ. When defining the rest of τ we will maintain this property.

Let S be the set of (existential) variables that occur in the right-hand sides of leaves of Υ' (for which τ is not yet defined). Also, let $<$ be the well-founded order on S. With induction on the size of S we complete the definition of τ. If S is empty, then we are obviously finished. Let x be a smallest element of S and $t \xrightarrow{\delta} x \in L''$ where L'' consists of the leaves of Υ'. By definition we have that $t\tau$ is well-defined and thus that $t\sigma' \xrightarrow{\delta} \sigma'(x)Rt\tau$. As we have that $t\sigma'' \in C_\alpha$, we get by induction that there is a r such that $t\tau \xrightarrow{\delta} r \in C$ and $\sigma''(x)Rr$. We take $\tau(x) = r$.

This completes the definition of τ. This means we have an application of ρ that gives a q' such that $q \xrightarrow{\gamma} q'$. To show that this $q \xrightarrow{\gamma} q'$ is in C, we must show that there are proofs for the positive literals in L and that the negative literals form L are not in C. The former follows trivially form the induction. For negative literals $\neg\, t \xrightarrow{\delta} y \in L''$ we have the following. By construction of Υ' we have that there is no r such that $\sigma'(y)Rr$ and $t\sigma' \xrightarrow{\delta} r \in C_\beta \cup U_\beta$. By induction we then have that there is also no r such that $t\tau \xrightarrow{\delta} r \in C$. Thus $C \vDash \neg\, t\tau \xrightarrow{\delta} \tau(y)$.

This concludes the proof.

C Proof of Theorem 5.11

Let $\langle \Sigma, R\rangle$ be a traditional TSS and let $\langle \Sigma, R'\rangle$ be the translation of $\langle \Sigma, R\rangle$ to a FOL-TSS. Assume that $\langle \Sigma, R\rangle$ is in the ntyft/ntyxt format. We show that each deduction rule of $\langle \Sigma, R'\rangle$ is either in the FOL-tyft format or in the FOL-tyxt format. Each deduction rule from R' originates from a deduction rule from R. Consider such a deduction rule. Since $\langle \Sigma, R\rangle$ is in the ntyft/ntyxt format, the deduction rule it is of the form

$$\frac{\{t_i \xrightarrow{a_i} y_i \mid i \in I\} \cup \{t'_j \overset{a'_j}{\nrightarrow} \mid j \in J\}}{f(x_1,\cdots,x_n) \xrightarrow{a} t'} \quad \text{or} \quad \frac{\{t_i \xrightarrow{a_i} y_i \mid i \in I\} \cup \{t'_j \overset{a'_j}{\nrightarrow} \mid j \in J\}}{x \xrightarrow{a} t'}$$

where the variables in the right-hand sides of the premises and the source of the conclusion are all different.

Let us consider the case that the deduction rule is of the first form. The deduction rule from R' that

corresponds to this traditional deduction rule is

$$\frac{\bigwedge\{t_i \xrightarrow{a_i} y_i \mid i \in I\} \cup \{\forall_{x(t'_j)}\ (\neg\ t'_j \xrightarrow{a'_j} x(t'_j)) \mid j \in J\}}{f(x_1, \cdots, x_n) \xrightarrow{a} t'}$$

where x is a mapping that associates with a term t a variable that does not occur free in t, i.e., $x(t) \notin \mathrm{var}(t)$.

Next, we will show that this deduction rule is in the FOL-tyft format. We use the following notations: $\phi = \bigwedge\{t_i \xrightarrow{a_i} y_i \mid i \in I\} \cup \{\forall_{x(t'_j)}\ (\neg\ t'_j \xrightarrow{a'_j} x(t'_j)) \mid j \in J\}$, $P_+ = \{t_i \xrightarrow{a_i} y_i \mid i \in I\}$, and $N = \{\forall_{x(t'_j)}\ (\neg\ t'_j \xrightarrow{a'_j} x(t'_j)) \mid j \in J\}$.

(i)
$$\begin{aligned}
\mathrm{dv}(\phi) &= \forall_{\psi\in P_+\cup N}\ \mathrm{dv}(\psi) \wedge \forall_{\psi,\psi'\in P_+\cup N}\ (\psi \neq \psi' \Rightarrow \mathrm{ubrhs}(\psi) \cap \mathrm{ubrhs}(\psi') = \varnothing)\\
&= \forall_{\psi\in P_+}\ \mathrm{dv}(\psi) \wedge \forall_{\psi\in N}\ \mathrm{dv}(\psi)\\
&\wedge\ \forall_{\psi,\psi'\in P_+\cup N}\ (\psi \neq \psi' \Rightarrow \mathrm{ubrhs}(\psi) \cap \mathrm{ubrhs}(\psi') = \varnothing)\\
&= \forall_{i\in I}\ \mathrm{dv}(t_i \xrightarrow{a_i} y_i) \wedge \forall_{j\in J}\ \mathrm{dv}(\forall_{x(t'_j)}\ (\neg\ t'_j \xrightarrow{a'_j} x(t'_j)))\\
&\wedge\ \forall_{\psi,\psi'\in P_+}\ (\psi \neq \psi' \Rightarrow \mathrm{ubrhs}(\psi) \cap \mathrm{ubrhs}(\psi') = \varnothing)\\
&\wedge\ \forall_{\psi\in P_+,\psi'\in\cup N}\ (\psi \neq \psi' \Rightarrow \mathrm{ubrhs}(\psi) \cap \mathrm{ubrhs}(\psi') = \varnothing)\\
&\wedge\ \forall_{\psi\in N,\psi'\in P_+}\ (\psi \neq \psi' \Rightarrow \mathrm{ubrhs}(\psi) \cap \mathrm{ubrhs}(\psi') = \varnothing)\\
&\wedge\ \forall_{\psi,\psi'\in N}\ (\psi \neq \psi' \Rightarrow \mathrm{ubrhs}(\psi) \cap \mathrm{ubrhs}(\psi') = \varnothing)\\
&= \mathrm{true} \wedge \forall_{j\in J}\ \mathrm{dv}(\neg\ t'_j \xrightarrow{a'_j} x(t'_j))\\
&\wedge\ \forall_{i,i'\in I}\ (t_i \xrightarrow{a_i} y_i \neq t_{i'} \xrightarrow{a_{i'}} y_{i'} \Rightarrow \mathrm{ubrhs}(t_i \xrightarrow{a_i} y_i) \cap \mathrm{ubrhs}(t_{i'} \xrightarrow{a_{i'}} y_{i'}) = \varnothing)\\
&\wedge\ \forall_{\psi\in P_+,\psi'\in\cup N}\ (\psi \neq \psi' \Rightarrow \mathrm{ubrhs}(\psi) \cap \mathrm{ubrhs}(\psi') = \varnothing)\\
&\wedge\ \forall_{\psi\in N,\psi'\in P_+}\ (\psi \neq \psi' \Rightarrow \mathrm{ubrhs}(\psi) \cap \mathrm{ubrhs}(\psi') = \varnothing)\\
&\wedge\ \forall_{\psi,\psi'\in N}\ (\psi \neq \psi' \Rightarrow \mathrm{ubrhs}(\psi) \cap \mathrm{ubrhs}(\psi') = \varnothing)\\
&= \forall_{i,i'\in I}\ (i \neq i' \Rightarrow \{y_i\} \cap \{y_{i'}\} = \varnothing)\\
&\wedge\ \forall_{\psi\in P_+,\psi'\in\cup N}\ (\psi \neq \psi' \Rightarrow \mathrm{ubrhs}(\psi) \cap \mathrm{ubrhs}(\psi') = \varnothing)\\
&\wedge\ \forall_{\psi\in N,\psi'\in P_+}\ (\psi \neq \psi' \Rightarrow \mathrm{ubrhs}(\psi) \cap \mathrm{ubrhs}(\psi') = \varnothing)\\
&\wedge\ \forall_{\psi,\psi'\in N}\ (\psi \neq \psi' \Rightarrow \mathrm{ubrhs}(\psi) \cap \mathrm{ubrhs}(\psi') = \varnothing)\\
&= \forall_{i,i'\in I}\ (i \neq i' \Rightarrow y_i \neq y_{i'})
\end{aligned}$$

Note that the last step is due to the fact that for any $n \in N$, $\mathrm{ubrhs}(n) = \varnothing$.

$$\begin{aligned}
\mathrm{ubrhs}(\phi) &= \mathrm{ubrhs}(\textstyle\bigwedge P_+ \cup N)\\
&= \textstyle\bigcup_{\psi\in P_+\cup N} \mathrm{ubrhs}(\psi)\\
&= \textstyle\bigcup_{\psi\in P_+} \mathrm{ubrhs}(\psi) \cup \bigcup_{\psi\in N} \mathrm{ubrhs}(\psi)\\
&= \textstyle\bigcup_{i\in I} \mathrm{ubrhs}(t_i \xrightarrow{a_i} y_i) \cup \bigcup_{j\in J} \mathrm{ubrhs}(\forall_{x(t'_j)}\ (\neg\ t'_j \xrightarrow{a'_j} x(t'_j)))\\
&= \textstyle\bigcup_{i\in I} \mathrm{var}(y_i) \cup \bigcup_{j\in J} (\mathrm{ubrhs}(\neg\ t'_j \xrightarrow{a'_j} x(t'_j)) \setminus \{x(t'_j)\})\\
&= \{y_i \mid i \in I\} \cup \textstyle\bigcup_{j\in J} (\mathrm{ubrhs}(t'_j \xrightarrow{a'_j} x(t'_j)) \setminus \{x(t'_j)\})\\
&= \{y_i \mid i \in I\} \cup \textstyle\bigcup_{j\in J} (\mathrm{var}(x(t'_j)) \setminus \{x(t'_j)\})\\
&= \{y_i \mid i \in I\}
\end{aligned}$$

The requirement $\mathrm{ubrhs}(\phi) \cap \{x_1, \cdots, x_n\} = \varnothing$ thus reduces to $\{y_i \mid i \in I\} \cap \{x_1, \cdots, x_n\} = \varnothing$.

Clearly these requirements are implied by the restrictions of the ntyft format on the original deduction rule that the right-hand sides of the premises and the variables in the source of the conclusion are all different.

(ii) Let $S = FV(\phi) \setminus \{x_1, \cdots, x_n\}$. Then,

$$\begin{aligned}
\mathrm{ext}_S(\phi) &= \mathrm{ext}_S(\textstyle\bigwedge P_+ \cup N)\\
&= \forall_{\psi\in P_+\cup N}\ \mathrm{ext}_S(\psi)\\
&= \forall_{\psi\in P_+}\ \mathrm{ext}_S(\psi) \wedge \forall_{\psi\in N}\ \mathrm{ext}_S(\psi)\\
&= \forall_{i\in I}\ \mathrm{ext}_S(t_i \xrightarrow{a_i} y_i) \wedge \forall_{j\in J}\ \mathrm{ext}_S(\forall_{x(t'_j)}\ (\neg\ t'_j \xrightarrow{a'_j} x(t'_j)))\\
&= \forall_{i\in I}\ y_i \in S \wedge \forall_{j\in J}\ \mathrm{ext}_{S\setminus\{x(t'_j)\}}(\neg\ t'_j \xrightarrow{a'_j} x(t'_j))\\
&= \forall_{j\in J}\ \overline{\mathrm{ext}}_{S\setminus\{x(t'_j)\}}(t'_j \xrightarrow{a'_j} x(t'_j))\\
&= \mathrm{true}
\end{aligned}$$

(iii) $\mathrm{ext}_\varnothing(\neg\,\phi) = \overline{\mathrm{ext}}_\varnothing(\phi) = \overline{\mathrm{ext}}_\varnothing(\bigwedge P_+ \cup N) = \forall_{\psi\in P_+}\ \overline{\mathrm{ext}}_\varnothing(\psi) \wedge \forall_{\psi\in N}\ \overline{\mathrm{ext}}_\varnothing(\psi) = \forall_{i\in I}\ \overline{\mathrm{ext}}_\varnothing(t_i \xrightarrow{a_i} y_i) \wedge \forall_{j\in J}\ \overline{\mathrm{ext}}_\varnothing(\forall_{x(t'_j)}\ (\neg\ t'_j \xrightarrow{a'_j} x(t'_j))) = \forall_{j\in J}\ \overline{\mathrm{ext}}_{\{x(t'_j)\}}(\neg\ t'_j \xrightarrow{a'_j} x(t'_j)) = \forall_{j\in J}\ \mathrm{ext}_{\{x(t'_j)\}}(t'_j \xrightarrow{a'_j} x(t'_j)) = \forall_{j\in J}\ x(t'_j) \in \{x(t'_j)\} = \mathrm{true}$

(iv) Let $S = FV(\phi) \setminus \{x_1, \cdots, x_n\}$. Then,

$$\begin{aligned}
h^S(\phi) &= h^S(\textstyle\bigwedge P_+ \cup N)\\
&= \forall_{i\in I}\ h^S(t_i \xrightarrow{a_i} y_i) \wedge \forall_{j\in J}\ h^S(\forall_{x(t'_j)}\ (\neg\ t'_j \xrightarrow{a'_j} x(t'_j)))\\
&= \forall_{j\in J}\ (h^\varnothing(\neg\ t'_j \xrightarrow{a'_j} x(t'_j)) \wedge k^S_{\{x(t'_j)\}}(\neg\ t'_j \xrightarrow{a'_j} x(t'_j)))\\
&= \forall_{j\in J}\ (\overline{h}^\varnothing(t'_j \xrightarrow{a'_j} x(t'_j)) \wedge k^S_{\{x(t'_j)\}}(t'_j \xrightarrow{a'_j} x(t'_j)))\\
&= \forall_{j\in J}\ (x(t'_j) \in S \setminus \{x(t'_j)\} \Rightarrow \mathrm{var}(t'_j) \cap \{x(t'_j)\} = \varnothing)\\
&= \mathrm{true}
\end{aligned}$$

The case where the deduction rule is in the ntyxt format is similar and therefore omitted.