SOS 2004 Preliminary Version

Toward the concept of backtracking computation

M. Kulaš

FernUniversität Hagen, FB Informatik, D-58084 Hagen, Germany [1]

Abstract

This article proposes a new mathematical definition of the execution of pure Prolog, in the form of axioms in a structural operational semantics. The main advantage of the model is its ease in representing backtracking, due to the functionality of the transition relation and its converse. Thus, *forward and backward* derivation steps are possible. A novel concept of *stages* is introduced, as a refinement of final states, which captures the evolution of a backtracking computation. An advantage over the traditional stack-of-stacks approaches is a *modularity* property. Finally, the model combines the intuition of the traditional 'Byrd box' metaphor with a compact representation of execution state, making it feasible to formulate and prove theorems about the model. In this paper we introduce the model and state some useful properties.

Key words: backtracking, Prolog, operational semantics

1 Motivation, aims and results

In this paper, we introduce S_1:PP, a new operational semantics for pure Prolog, and establish some useful properties, aiming toward an algebraic definition of the concept of Prolog computation. On the way, we obtain some new concepts useful for characterizing backtracking, but possibly also useful for objects which evolve over time. Such an object is in logic programming *the goal*, in its dynamic sense ('this unique, run-time invocation of a Prolog procedure'), as opposed to its static or syntactic sense ('this goal formula').

The goal is a basic concept of logic programming, but nevertheless one which proved hard to grasp in a formal way, even in the case of pure Prolog. The problem is the possible evolution of 'the' goal through slightly different identities, in case of a non-deterministic procedure.

[1] Email: marija.kulas@fernuni-hagen.de

This is a preliminary version. The final version will be published in Electronic Notes in Theoretical Computer Science
URL: www.elsevier.nl/locate/entcs

For example, assume we pose the following query to a Prolog program: $\mathsf{p(X,Y)}, \mathsf{q(Y)}, \mathsf{fail}$. Assume two answers for $\mathsf{p(X,Y)}$, say $\mathsf{p(a,Z)}$ and $\mathsf{p(b,3)}$. In the static sense, we have only one goal $\mathsf{q(Y)}$, but dynamically we can have two different goals: first $\mathsf{q(Z)}$, and if $\mathsf{q(Z)}, \mathsf{fail}$ terminates, then $\mathsf{q(3)}$. Both goals have their own creation, lifetime of forward and backward execution, and possible expiration. It is vital for an operational semantics of backtracking to differentiate between these objects. Moreover, assume $\mathsf{q}/1$ is recursive. During the computation of e. g. the goal $\mathsf{q(Z)}$, we need to pay attention to all the recursive invocations of $\mathsf{q}/1$ as well, in order to know exactly where to go after a failure. Are they all to be considered distinct objects as well, of the same kind as the above two, $\mathsf{q(Z)}$ and $\mathsf{q(3)}$, and how to manage them anyway?

In the rest of this paper we proceed as follows. First a canonical form of predicates is defined, into which the original pure Prolog program shall be transformed. Then, in Section 3, a novel operational semantics of pure Prolog is defined, in a structural operational manner. Throughout the Section 4 – Section 7 we develop formal tools (concepts and theorems) suitable for characterizing Prolog computation. This obviously includes defining in some way or other the (dynamic) concept of the goal as well. We solved this problem by means of *stages*, through which an initial event (representing the creation of a goal) passes in the course of computation. Stages can be seen as a generalization of the normal form idea, in the sense that stages are independent on the context of computation, as shown in Section 7, but organized by macro transitions. Starting from individual transitions as given in the model, simple derivations (forward and backward) are built, which are the basis of simple passes, and simple passes aggregate into composed passes. Finally, we show in Section 8 how composed passes model Prolog computations.

Our approach can be seen as a formalization of the original Byrd model. But there is an important detail: we extend the notion of a port, initially conceived by Byrd for selected atoms, to *general goal formulas*. The shifting of attention from atoms to general goal formulas proved to be a key idea and made a very simple model possible. The model in its first version, called S:PP, was proposed in [18]. But the handling of variables turned out to be difficult. The new model S_1:PP improves on that.

2 Preliminaries

Before it can be interpreted in our model, the original Prolog program has to be transformed into a canonical form, the common single-clause representation. This representation is arguably 'near enough' to the original program, the only differences concern the head-unification (which is now delegated to the body) and the choices (which are now uniformly expressed as disjunction).

Definition 2.1 (full canonical form) *We say that a predicate P/n is in* full canonical form*, if its definition in the given program Π consists of a single clause $P(X_1, ..., X_n) :- Bs$. Here $X_1, ..., X_n$ are distinct variables, and*

Bs is a disjunction of branches *(possibly empty, corresponding to* fail*). Each branch is of the form* $X_1{=}T_1, \ldots, X_n{=}T_n, Gs$*, where* Gs *is a conjunction of* goal formulas *(possibly empty, corresponding to* true*). No* X_i *may appear in any* $T_1, ..., T_n, Gs$*.*

Here, a *goal formula* may be of six kinds: user-defined atom $P(T_1, ..., T_n)$, special terms true and fail, equality $T_1{=}T_2$, conjunction and disjunction of goal formulas.

Transformation into full canonical form may proceed as follows: First rename the clauses for P/n apart, obtaining clauses $P(T_1, ..., T_n) :- G$. (For facts, set G to true.) Pick distinct variables $X_1, ..., X_n$ not appearing in any $T_1, ..., T_n, G$. For each body G, assemble a new body $X_1{=}T_1, \ldots, X_n{=}T_n, G$. Finally, make a disjunction of the new bodies, preserving the order of their appearance in Π, giving the body of the canonical form. The head shall be $P(X_1, ..., X_n)$. As an example, the following program

```
p(a,Z).
p(b,3).
q(0).
q(s(N)) :- q(N).
```

would be canonically represented as

```
p(X,Y) :- X=a, Y=Z, true; X=b, Y=3, true.
q(Z) :- Z=0, true; Z=s(N), q(N).
```

and simplified to

```
p(X,Y) :- X=a; X=b, Y=3.
q(Z) :- Z=0; Z=s(N), q(N).
```

For the purposes of this paper, a weaker notion is sufficient:

Definition 2.2 (canonical form) *We say that a predicate* P/n *is in* canonical form*, if its definition in the given program* Π *consists of a single clause* $P(X_1, ..., X_n) :- G$*. Here* $X_1, ..., X_n$ *are distinct variables, while* G *is an arbitrary goal formula.*

3 The semantics S_1:PP

The model S_1:PP we are proposing fits naturally into a structural operational format. For easy reference, the model is defined in two figures, Subsection 3.3 (syntax) and Subsection 3.4 (rules). Notice that there are no premisses to the transition rules, so the calculus consists solely of axioms.

Definition 3.1 (event) *An* event *is a quadruple* $(Port, Goal, A\text{-}stack, B\text{-}stack)$*, as given by the grammar in Subsection 3.3.*

Intuitively, an event is a state of Prolog computation, and it is determined in our model by four parameters:

- *goal:* the current focus or 'goal' of computation (a general goal formula)
- *port:* marks evolution of the current goal (*call*, *exit*, *fail* or *redo*)
- *A-stack:* the history of the current goal (stack of ancestors)
- *B-stack:* the current environment (stack of bets)

Definition 3.2 (transition) *Let Π be a pure Prolog program in canonical form, as defined by Subsection 3.3 and Definition 2.2.* The transition relation $\rightarrowtriangle_\Pi$ *is defined in Subsection 3.4. The converse relation shall be denoted by* $\leftarrowtriangle_\Pi$. *If* $E_1 \rightarrowtriangle_\Pi E$, *we say that* E_1 leads to E. *Alternatively, we say that* E_1 *is a* predecessor *to* E, *and* E *is a* successor *to* E_1. *An event* E *can be* entered, *if some event leads to it. An event* E *can be* left, *if it leads to some event.*

The left-hand sides of the transition rules are mutually disjoint, i. e. there are no critical pairs, so we have

Lemma 3.3 (transitions are deterministic) $\rightarrowtriangle_\Pi$ *is functional, i. e. for each event* E *there can be at most one event* E_1 *such that* $E \rightarrowtriangle_\Pi E_1$.

Remark 3.4 (converse relation) *The converse of the transition relation is not functional, since there may be more than one event leading to the same event. For example, call* $T_1{=}T_2\,\langle\frac{nil}{nil}\rangle \rightarrowtriangle_\Pi$ *fail* $T_1{=}T_2\,\langle\frac{nil}{nil}\rangle$, *as well as redo* $T_1{=}T_2\,\langle\frac{\sigma\bullet nil}{nil}\rangle \rightarrowtriangle_\Pi$ *fail* $T_1{=}T_2\,\langle\frac{nil}{nil}\rangle$. *Further down it will be shown that, for events that are* legal, *the converse relation is functional. In our example, redo* $T_1{=}T_2\,\langle\frac{\sigma\bullet nil}{nil}\rangle$ *is not a legal event.*

Definition 3.5 (derivation) *Let* E_0, E *be events. A* Π-derivation of E from E_0 in k steps, *written as* $E_0 \rightarrowtriangle_\Pi{}^k E$, *is a path of length* k *from* E_0 *to* E *in the graph of* $\rightarrowtriangle_\Pi$. *We say that* E *can be* reached *from* E_0. *Derivation of a nonzero length is denoted by* $E_0 \rightarrowtriangle_\Pi^+ E$, *and derivation of any length by* $E_0 \rightarrowtriangle_\Pi^* E$.

Definition 3.6 (initial event) *An* initial event *is call* $Q\,\langle\frac{nil}{nil}\rangle$ *for any* Q.

Intuitively, the goal Q of an initial event corresponds in Prolog to a *top-level goal* (*query*).

Definition 3.7 (legal derivation, legal event) *If there is a goal* Q *such that call* $Q\,\langle\frac{nil}{nil}\rangle \rightarrowtriangle_\Pi^* E_0 \rightarrowtriangle_\Pi^* E$, *then we say that* $E_0 \rightarrowtriangle_\Pi^* E$ *is a* legal Π-derivation, *and* E *is a* legal Π-event.

Definition 3.8 (final event) *A legal* Π*-event* E *is a* final Π-event, *if there is no transition* $E \rightarrowtriangle_\Pi E_1$.

Notation 1 (impossible event) *As a notational convenience, all the events which are not final and do not lead to any further events by means of* $\rightarrowtriangle_\Pi$ *are depicted as leading to the* impossible event, *written as* $\bot$. *Analogously for events that are not initial and cannot be entered.*

In particular, *redo* $\mathsf{fail} \rightarrowtriangle_\Pi \bot$ and *exit* $\mathsf{fail} \leftarrowtriangle_\Pi \bot$ for any Π. Some more examples: *call* $G\,\langle\frac{\sigma\bullet nil}{nil}\rangle \leftarrowtriangle_\Pi \bot$, *redo* $G\,\langle\frac{\Sigma}{nil}\rangle \leftarrowtriangle_\Pi \bot$ (cannot be entered,

non-initial), and *redo* $\mathsf{p}\,\langle\frac{nil}{U}\rangle \rightarrowtriangle_{\Pi} \bot$ (cannot be left, but not legal, p being an atomic goal). The last example is perhaps less obvious, and follows from Lemma 4.1 and (S_1:atom:2).

3.1 Remarks on the calculus

(i) In S_1:PP, the word *goal* is used in both its usual senses: as a syntax domain, meaning 'goal formula' (Subsection 3.3), and as one of the four components of an execution state, meaning 'current goal' (Subsection 3.4).

(ii) SLD-resolution is operating on the *selected atom*, but S_1:PP is operating on the whole *current goal*.

(iii) For the syntax domains that have not been defined in Subsection 3.3, we refer to [14] (substitutions), [21] (logic programming) and [12] (Prolog).

(iv) The most general unifiers σ shall be chosen to be idempotent, namely $\sigma(\sigma(T)) = \sigma(T)$. This is always possible.

(v) Resolution is modeled by (S_1:atom:1), and it is the only rule actually depending on Π. If the predicate of the atomic goal has a definition in Π, the resolution will succeed, because of canonical form.

(vi) Note the requirement $\sigma(G_A) = G_A$ in (S_1:atom:1). Since the clauses are in canonical form, unifying the head of a clause with a goal could do no more than rename the goal. We prefer the mgu to operate only on the clause.

(vii) A *fresh variable*, at a certain point of a derivation, represented by an event, is a variable not appearing in the previous course of the derivation, represented by the goal and the A-stack of the event.

3.2 Auxiliary notation

In addition to the language given in Subsection 3.3, here is some auxiliary notation that shall be used in theorems:

(i) *Anonymous meta-variable*: If some parts of an event are of no interest in a topic at hand, they shall be abstracted away by the underscore "$\_$".

(ii) To navigate an ancestor X, two functions are used: $\lfloor X \rfloor$, which is the selected half of X, as defined in Subsection 3.3, and $\lceil X \rceil$, which is X without tags: $\lceil N/A, B\rceil := A, B$, $\lceil N/A; B\rceil := A; B$, $\lceil G_A\rceil := G_A$.

(iii) *Stack addition and subtraction*: Concatenation to the right of a stack we denote by $+$, and if $U + V = W$, then $W - V := U$. Concatenating to both stacks of an event we denote by $\oplus$, with $\Gamma\ G\,\langle\frac{\Sigma}{U}\rangle \oplus \langle\frac{\Delta}{V}\rangle := \Gamma\ G\,\langle\frac{\Sigma+\Delta}{U+V}\rangle$, and $\Gamma\ G\,\langle\frac{\Sigma+\Delta}{U+V}\rangle \ominus \langle\frac{\Delta}{V}\rangle := \Gamma\ G\,\langle\frac{\Sigma}{U}\rangle$.

(iv) *Stack order*: If there is W such that $U = W + V$, then we say that $U \succeq V$.

(v) $\sigma\,|_T$ means substitution σ restricted to the variables of the term T.

(vi) *Macro transitions* $\overset{\triangleright}{\rightarrowtriangle}_{\Pi}, \overset{\triangleleft}{\rightarrowtriangle}_{\Pi}, \longrightarrow_{\Pi}, \Longrightarrow_{\Pi}$ will be defined in Section 5 and

event	::=	port goal $\left\langle \frac{\text{B-stack}}{\text{A-stack}} \right\rangle$
program	::=	$\{\text{definition.}\}^+$
definition	::=	atom :− goal
port	::=	push | pop
push	::=	*call* | *redo*
pop	::=	*exit* | *fail*
goal	::=	true | fail | atom | term = term | goal; goal | goal, goal
ancestor	::=	atom | tag/goal; goal | tag/goal, goal
tag	::=	*1* | *2*
memo	::=	BY(goal) | OR(tag)
bet	::=	mgu | memo
A-stack	::=	*nil* | ancestor • A-stack
B-stack	::=	*nil* | bet • B-stack

Meta-variables

E	:	event						
Π	:	program						
Γ	:	port,	$Push$	:	push,	Pop	:	pop
U, V	:	A-stack,	a, b, $\widehat{G}$	:	ancestor,	N	:	tag
Σ, Θ, Ψ, Ω, Δ	:	B-stack,	α	:	bet			
σ	:	substitution						
A, B, C, G, H	:	goal						
G_A	:	atom						
T	:	term						

Meta-functions

$\lfloor 1/A, B \rfloor := A$, $\lfloor 2/A, B \rfloor := B$, and analogously for disjunction
$\sigma(T)$ = application of σ upon T
$mgu(T_1, T_2)$ = mgu of T_1 and T_2
$subst(\Sigma)$ = current substitution = composition of all mgus from Σ;
$subst(\Sigma)(T)$ shall be abbreviated to $\Sigma(T)$, and is defined as follows:

$$nil(T) := T$$

$$\alpha \bullet \Sigma(T) := \begin{cases} \alpha(\Sigma(T)), & \text{if } \alpha \text{ is an mgu} \\ \Sigma(T), & \text{if } \alpha \text{ is a memo} \end{cases}$$

Syntax domains taken in their usual sense:

term (taken in the Prolog sense, as a superset of goal);
atom (user-defined predication in logic programming);
substitution, renaming, mgu.

Conjunction

$$\begin{aligned}
\mathit{call}\ A, B\ \langle \tfrac{\Sigma}{U} \rangle &\rightarrow_\Pi \mathit{call}\ A\ \langle \tfrac{\Sigma}{1/A,B \bullet U} \rangle && (S_1\text{:conj:1})\\
\mathit{exit}\ A\ \langle \tfrac{\Sigma}{1/A,B \bullet U} \rangle &\rightarrow_\Pi \mathit{call}\ B\ \langle \tfrac{\Sigma}{2/A,B \bullet U} \rangle && (S_1\text{:conj:2})\\
\mathit{fail}\ A\ \langle \tfrac{\Sigma}{1/A,B \bullet U} \rangle &\rightarrow_\Pi \mathit{fail}\ A, B\ \langle \tfrac{\Sigma}{U} \rangle && (S_1\text{:conj:3})\\
\mathit{exit}\ B\ \langle \tfrac{\Sigma}{2/A,B \bullet U} \rangle &\rightarrow_\Pi \mathit{exit}\ A, B\ \langle \tfrac{\Sigma}{U} \rangle && (S_1\text{:conj:4})\\
\mathit{fail}\ B\ \langle \tfrac{\Sigma}{2/A,B \bullet U} \rangle &\rightarrow_\Pi \mathit{redo}\ A\ \langle \tfrac{\Sigma}{1/A,B \bullet U} \rangle && (S_1\text{:conj:5})\\
\mathit{redo}\ A, B\ \langle \tfrac{\Sigma}{U} \rangle &\rightarrow_\Pi \mathit{redo}\ B\ \langle \tfrac{\Sigma}{2/A,B \bullet U} \rangle && (S_1\text{:conj:6})
\end{aligned}$$

Disjunction

$$\begin{aligned}
\mathit{call}\ A; B\ \langle \tfrac{\Sigma}{U} \rangle &\rightarrow_\Pi \mathit{call}\ A\ \langle \tfrac{\Sigma}{1/A;B \bullet U} \rangle && (S_1\text{:disj:1})\\
\mathit{fail}\ A\ \langle \tfrac{\Sigma}{1/A;B \bullet U} \rangle &\rightarrow_\Pi \mathit{call}\ B\ \langle \tfrac{\Sigma}{2/A;B \bullet U} \rangle && (S_1\text{:disj:2})\\
\mathit{fail}\ B\ \langle \tfrac{\Sigma}{2/A;B \bullet U} \rangle &\rightarrow_\Pi \mathit{fail}\ A; B\ \langle \tfrac{\Sigma}{U} \rangle && (S_1\text{:disj:3})\\
\mathit{exit}\ C\ \langle \tfrac{\Sigma}{N/A;B \bullet U} \rangle &\rightarrow_\Pi \mathit{exit}\ A; B\ \langle \tfrac{OR(N) \bullet \Sigma}{U} \rangle, \text{ with } C\ldots^{2} && (S_1\text{:disj:4})\\
\mathit{redo}\ A; B\ \langle \tfrac{OR(N) \bullet \Sigma}{U} \rangle &\rightarrow_\Pi \mathit{redo}\ C\ \langle \tfrac{\Sigma}{N/A;B \bullet U} \rangle, \text{ with } C\ldots^{2} && (S_1\text{:disj:5})
\end{aligned}$$

True and Fail

$$\begin{aligned}
\mathit{call}\ \mathsf{true}\ \langle \tfrac{\Sigma}{U} \rangle &\rightarrow_\Pi \mathit{exit}\ \mathsf{true}\ \langle \tfrac{\Sigma}{U} \rangle && (S_1\text{:true:1})\\
\mathit{redo}\ \mathsf{true}\ \langle \tfrac{\Sigma}{U} \rangle &\rightarrow_\Pi \mathit{fail}\ \mathsf{true}\ \langle \tfrac{\Sigma}{U} \rangle && (S_1\text{:true:2})\\
\mathit{call}\ \mathsf{fail}\ \langle \tfrac{\Sigma}{U} \rangle &\rightarrow_\Pi \mathit{fail}\ \mathsf{fail}\ \langle \tfrac{\Sigma}{U} \rangle && (S_1\text{:fail})
\end{aligned}$$

Equality

$$\begin{aligned}
\mathit{call}\ T_1{=}T_2\ \langle \tfrac{\Sigma}{U} \rangle &\rightarrow_\Pi \begin{cases} \mathit{exit}\ T_1{=}T_2\ \langle \tfrac{\sigma \bullet \Sigma}{U} \rangle, & \text{if } \mathit{mgu}\ldots^{3} \\ \mathit{fail}\ T_1{=}T_2\ \langle \tfrac{\Sigma}{U} \rangle, & \text{otherwise} \end{cases} && (S_1\text{:unif:1})\\
\mathit{redo}\ T_1{=}T_2\ \langle \tfrac{\sigma \bullet \Sigma}{U} \rangle &\rightarrow_\Pi \mathit{fail}\ T_1{=}T_2\ \langle \tfrac{\Sigma}{U} \rangle && (S_1\text{:unif:2})
\end{aligned}$$

User-defined atom G_A

$$\begin{aligned}
\mathit{call}\ G_A\ \langle \tfrac{\Sigma}{U} \rangle &\rightarrow_\Pi \begin{cases} \mathit{call}\ \sigma(B)\ \langle \tfrac{\Sigma}{G_A \bullet U} \rangle, & \text{if } H{:-}B\ldots^{4} \\ \mathit{fail}\ G_A\ \langle \tfrac{\Sigma}{U} \rangle, & \text{otherwise} \end{cases} && (S_1\text{:atom:1})\\
\mathit{exit}\ B\ \langle \tfrac{\Sigma}{G_A \bullet U} \rangle &\rightarrow_\Pi \mathit{exit}\ G_A\ \langle \tfrac{BY(B) \bullet \Sigma}{U} \rangle && (S_1\text{:atom:2})\\
\mathit{fail}\ B\ \langle \tfrac{\Sigma}{G_A \bullet U} \rangle &\rightarrow_\Pi \mathit{fail}\ G_A\ \langle \tfrac{\Sigma}{U} \rangle && (S_1\text{:atom:3})\\
\mathit{redo}\ G_A\ \langle \tfrac{BY(B) \bullet \Sigma}{U} \rangle &\rightarrow_\Pi \mathit{redo}\ B\ \langle \tfrac{\Sigma}{G_A \bullet U} \rangle && (S_1\text{:atom:4})
\end{aligned}$$

---

[2] with $C = \lfloor N/A; B \rfloor$.

[3] if $mgu(\Sigma(T_1), \Sigma(T_2)) = \sigma$.

[4] if $H :- B$ is a fresh renaming of a clause in Π, and $\sigma = mgu(G'_A, H)$ with $G'_A := \Sigma(G_A)$ and $\sigma(G'_A) = G'_A$.

Section 6.

Notation 2 (distinguishing two levels) *Object-level terms (i. e. actual Prolog terms) are shown in sans serif, like* true*. Meta-level terms (i. e. anything else in the calculus) are shown in italics, like call,*α*.*

Notation 3 (dropping Π) *In the following we usually drop any reference to Π, since a program in pure Prolog cannot change during a derivation. However, a fixed program Π is always assumed. Observe that all the new relations in this paper, built upon the transition relation, also implicitly depend on Π.*

4 Uniqueness claim

First we state a useful property, which we call the pendant lemma. Observe that the formulation is *non-deterministic* in that we only claim the existence of a pendant event (with the identical B-stack), but it is not known whether it is the only one.

Lemma 4.1 (pendant) *If call $G\,\langle\frac{nil}{nil}\rangle \rightarrowtriangle^* fail\ H\,\langle\frac{\Theta}{W}\rangle$, then*

$$call\ G\,\langle\tfrac{nil}{nil}\rangle \rightarrowtriangle^* call\ H\,\langle\tfrac{\Theta}{W}\rangle \rightarrowtriangle^* fail\ H\,\langle\tfrac{\Theta}{W}\rangle.$$

If $call\ G\,\langle\frac{nil}{nil}\rangle \rightarrowtriangle^* redo\ H\,\langle\frac{\Theta}{W}\rangle$, then

$$call\ G\,\langle\tfrac{nil}{nil}\rangle \rightarrowtriangle^* exit\ H\,\langle\tfrac{\Theta}{W}\rangle \rightarrowtriangle^* redo\ H\,\langle\tfrac{\Theta}{W}\rangle.$$

The pendant lemma enables us to prove a vital property of our calculus: there can be only one successor to a given event, and moreover, for a legal event there can be only one predecessor.

Theorem 4.2 (legal transitions are unique) *If E is a legal event, then E can have only one legal predecessor, and only one successor. In case E is non-initial, there is exactly one legal predecessor. In case E is non-final, there is exactly one successor.*

Having established functionality of the transition relation and its converse, we may unfold a legal derivation from each of its endpoints. First let us see how far we can go from an initial event by means of transitions.

Lemma 4.3 (ancestor) *If $\Gamma\ G\,\langle\frac{\Sigma}{a\bullet V}\rangle$ is a legal event, then there are Push and Σ' such that $Push\,\lceil a\rceil\,\langle\frac{\Sigma'}{V}\rangle \rightarrowtriangle^+ \Gamma\ G\,\langle\frac{\Sigma}{a\bullet V}\rangle$ is a legal derivation.*

Lemma 4.4 (tagged parent) *If $\Gamma\ G\,\langle\frac{\Sigma}{a\bullet V}\rangle$ is a legal event, and $a = N/A, B$ or $a = N/A; B$, then $G = \lfloor a\rfloor$.*

Lemma 4.5 (final event) *If E is a legal pop event with a non-empty A-stack, then there is always a transition $E \rightarrowtriangle E_1$.*

Lemma 4.6 *If $call\ G\,\langle\frac{nil}{nil}\rangle \rightarrowtriangle^+ \Gamma\ H\,\langle\frac{\Theta}{W}\rangle$ and $\Gamma\ H\,\langle\frac{\Theta}{W}\rangle \neq Pop\ \_\,\langle\frac{-}{nil}\rangle$, then $W = V + \widehat{G}\bullet nil$ for some V and an ancestor $\widehat{G}$ such that $\lceil\widehat{G}\rceil = G$.*

Lemma 4.7 *If $call\ G\,\langle\frac{nil}{nil}\rangle \rightarrowtriangle^* Pop\ H\,\langle\frac{\Sigma}{nil}\rangle$, then $H = G$.*

The above lemmas suggest events of the form $Pop\ \_\langle\frac{\div}{nil}\rangle$ as natural endpoints of derivation. For this reason we develop a concept of derivation around such events. We start with a concept of simple derivation.

5 Simple derivation and subevent

In this section, we set about defining some new, 'macro' transition relations, by collapsing whole sequences of transition steps into one big step. Arguably, illegal derivations do not make much sense in such a context, therefore we exclude them:

Notation 4 (only legal derivations) *In the transition relations that we shall define from now on, namely* $\overset{\triangleright}{\rightarrow}, \overset{\triangleleft}{\rightarrow}, \longrightarrow, \Longrightarrow$, *it is always assumed that the derivations are legal.*

Definition 5.1 (forward or backward simple derivation) *Consider a legal derivation* $Push\ G\ \langle\frac{\Sigma}{U}\rangle \rightarrow^{+} E$ *such that there is no* $Pop\ \_\langle\frac{\div}{U}\rangle$ *within this derivation, i. e.* $Push\ G\ \langle\frac{\Sigma}{U}\rangle \rightarrow^{+} Pop\ \_\langle\frac{\div}{U}\rangle \rightarrow^{+} E$ *is not allowed. Such a derivation we call a* forward derivation relative to U*, and denote by* $Push\ G\ \langle\frac{\Sigma}{U}\rangle \overset{\triangleright}{\rightarrow} E$*. Analogously, a legal derivation* $Pop\ G\ \langle\frac{\Sigma}{U}\rangle \rightarrow^{+} E$ *such that there is no* $Push\ \_\langle\frac{\div}{U}\rangle$ *within this derivation, is a* backward derivation relative to U*, denoted by* $Pop\ G\ \langle\frac{\Sigma}{U}\rangle \overset{\triangleleft}{\rightarrow} E$*. A forward or a backward derivation relative to* U *is a* simple derivation relative to U*.*

Forward derivation gives rise to *subevents*:

Definition 5.2 (subevent) *If* $Push\ G\ \langle\frac{\Sigma}{U}\rangle \overset{\triangleright}{\rightarrow} \Gamma\ H\ \langle\frac{\Theta}{W}\rangle$ *then* $\Gamma\ H\ \langle\frac{\Theta}{W}\rangle$ *is a* subevent *of* $Push\ G\ \langle\frac{\Sigma}{U}\rangle$*.*

Lemma 4.6 and Lemma 4.7 can be generalized to provide for the case of an arbitrary push event and an arbitrary A-stack, and proven in the same manner as the original lemmas:

Theorem 5.3 (the subevent property) *If* $Push\ G\ \langle\frac{\Sigma}{U}\rangle \overset{\triangleright}{\rightarrow} \Gamma\ H\ \langle\frac{\Theta}{W}\rangle$ *and* $\Gamma\ H\ \langle\frac{\Theta}{W}\rangle \neq Pop\ \_\langle\frac{\div}{U}\rangle$*, then* $W = V + \widehat{G} \bullet U$ *for some* V *and an ancestor* $\widehat{G}$ *such that* $\lceil\widehat{G}\rceil = G$*.*

Lemma 5.4 (endpoint) *If* $Push\ G\ \langle\frac{\Sigma}{U}\rangle \overset{\triangleright}{\rightarrow} Pop\ H\ \langle\frac{\Theta}{U}\rangle$ *then* $H = G$*.*

Obviously, each push transition is a forward derivation. Some forward derivations can be composed as well. This follows from the subevent property.

Corollary 5.5 *If* $Push_1\ G\ \langle\frac{\Sigma}{U}\rangle \overset{\triangleright}{\rightarrow} Push\ H\ \langle\frac{\Theta}{a \bullet U}\rangle \overset{\triangleright}{\rightarrow} E$*, then* $Push_1\ G\ \langle\frac{\Sigma}{U}\rangle \overset{\triangleright}{\rightarrow} E$*. Also, if* $Push\ G\ \langle\frac{\Sigma}{U}\rangle \overset{\triangleright}{\rightarrow} E \rightarrow E_1$ *with* $E \neq Pop\ \_\langle\frac{\div}{U}\rangle$*, then* $Push\ G\ \langle\frac{\Sigma}{U}\rangle \overset{\triangleright}{\rightarrow} E_1$*.*

Lemma 5.6 (forward pass) *If* $Pop\ G\ \langle\frac{\Sigma}{U}\rangle$ *is a legal event, then* $Pop\ G\ \langle\frac{\Sigma}{U}\rangle \overset{\triangleleft}{\leftarrow} Push\ G\ \langle\frac{\Sigma^{\circ}}{U}\rangle$ *for some* $Push$ *and* Σ°*.*

The next statement follows from the subevent property. Analogous claim with pop and push swapping places, holds due to the simple pass lemma.

Lemma 5.7 (no pop no push) *Let* $Push_0\ G\ \langle \frac{\div}{U} \rangle \twoheadrightarrow^* E$ *be a legal derivation, such that there is no* $Pop\ \_\ \langle \frac{\div}{U} \rangle$ *within the derivation. Then there is no* $Push\ \_\ \langle \frac{\div}{U} \rangle$ *within the derivation as well. The same holds for derivations of the form* $E \twoheadrightarrow^* Push_0\ G\ \langle \frac{\div}{U} \rangle$.

Taking into account composition of forward derivations, we can prove a stronger version of the ancestor lemma. The new version has the advantage of determinacy, i. e. there is only one place in a derivation where the parent event can be: the most recent past event of the form $Push\ \lceil a \rceil\ \langle \frac{\div}{V} \rangle$.

Lemma 5.8 (ancestor, stronger) *If* $\Gamma\ G\ \langle \frac{\Sigma}{a \bullet V} \rangle$ *is a legal event, then there are Push and* Σ' *such that* $Push\ \lceil a \rceil\ \langle \frac{\Sigma'}{V} \rangle \overset{\triangleright}{\twoheadrightarrow} \Gamma\ G\ \langle \frac{\Sigma}{a \bullet V} \rangle$ *is a legal derivation.*

Lemma 5.9 (subevent, converse) *If for a legal event* $\Gamma\ H\ \langle \frac{\Theta}{W} \rangle$ *holds that* $W = V + a \bullet U$, *then* $\Gamma\ H\ \langle \frac{\Theta}{W} \rangle$ *is a subevent of* $Push\ \lceil a \rceil\ \langle \frac{\Sigma}{U} \rangle$ *for some Push and some* Σ.

As we have seen, forward or backward derivations between events with identical A-stack and identical goal play a special role in the calculus. We abstract such derivations to a new concept:

Definition 5.10 (forward or backward simple pass) *A forward derivation* $Push\ G\ \langle \frac{\div}{U} \rangle \overset{\triangleright}{\twoheadrightarrow} Pop\ G\ \langle \frac{\div}{U} \rangle$ *we call a* forward pass relative to G and U. *Analogously, a backward derivation* $Pop\ G\ \langle \frac{\div}{U} \rangle \overset{\triangleleft}{\twoheadrightarrow} Push\ G\ \langle \frac{\div}{U} \rangle$ *we call a* backward pass relative to G and U. *A forward or a backward pass from* E_1 *to* E_2 *is a* simple pass, *denoted by* $E_1 \longrightarrow E_2$. *The events* E_1, E_2 *are called* stages. *To denote stages we may use the fixed parts (the goal and the A-stack) as superscripts, like this:* $E^{G,U}$.

6 Composed derivation

Fortified with the useful results like the pendant and forward pass lemma, we are now in a position to prove stronger results. As with the stronger version of the ancestor lemma, the advantage is in the deterministic specification of the correlated events from the past. For example, for a fail event we now know that its pendant call (with the identical B-stack) is the most recent call event bearing the same goal and the same A-stack.

Theorem 6.1 (pendant, stronger) *Let* $fail\ H\ \langle \frac{\Theta}{W} \rangle$ *be a a legal event. Then* $fail\ H\ \langle \frac{\Theta}{W} \rangle \twoheadleftarrow^* call\ H\ \langle \frac{\Theta}{W} \rangle$, *where* $call\ H\ \langle \frac{\div}{W} \rangle$ *does not appear within the derivation. Furthermore, if* $redo\ H\ \langle \frac{\Theta}{W} \rangle$ *is a legal event, then* $redo\ H\ \langle \frac{\Theta}{W} \rangle \longleftarrow exit\ H\ \langle \frac{\Theta}{W} \rangle$.

Lemma 6.2 (B-stack) *If* $call\ G\ \langle \frac{\Sigma}{U} \rangle \longrightarrow \Gamma\ H\ \langle \frac{\Sigma'}{U} \rangle$, *then* $\Sigma' \succeq \Sigma$.

Lemma 6.3 (no call no fail) *If* $E \twoheadrightarrow^* Pop\ G\ \langle \frac{\div}{U} \rangle$ *is a legal derivation, and* $call\ G\ \langle \frac{\div}{U} \rangle$ *is not within this derivation, then* $fail\ G\ \langle \frac{\div}{U} \rangle$ *is also not within this*

derivation.

From Lemma 5.6 and Theorem 6.1 we know that a legal pop event can run through a series of past stages, like $\mathit{exit}\ \_\langle\frac{\div}{U}\rangle \longleftarrow \mathit{redo}\ \_\langle\frac{\div}{U}\rangle \longleftarrow \mathit{exit}\ \_\langle\frac{\div}{U}\rangle \longleftarrow \ldots$ Due to the finiteness of a converse derivation and Lemma 5.6, a $\mathit{call}\ \_\langle\frac{\div}{U}\rangle$ is bound to appear. So we must ultimately reconstruct a derivation $\mathit{exit}\ \_\langle\frac{\div}{U}\rangle \mathrel{\leftarrow\!\!*} \mathit{call}\ \_\langle\frac{\div}{U}\rangle$, where no events of the form $\mathit{call}\ \_\langle\frac{\div}{U}\rangle$ intervene. Similarly for a fail event.

Definition 6.4 (composed pass) *Consider a sequence of simple passes $E_1^{G,U} \longrightarrow^+ E_2^{G,U}$ such that there is no call $G\,\langle\frac{\div}{U}\rangle$ within this sequence, i. e. $E_1^{G,U} \longrightarrow^+ \mathit{call}\ G\,\langle\frac{\div}{U}\rangle \longrightarrow^+ E_2^{G,U}$ is not allowed. Such a sequence is called* composable, *or a* composed pass relative to G and U, *and denoted by $E_1^{G,U} \Longrightarrow E_2^{G,U}$.*

It can be seen that $\mathit{call}\ G\,\langle\frac{\div}{U}\rangle$ cannot appear within a composed pass relative to G and U even if regarded as a derivation, i. e. neither among the stages of the simple passes nor somewhere in between. One important question remains: How do the B-stacks of the particular stages relate to each other? This is the main concern of our next claim, companion to Lemma 5.6.

Theorem 6.5 (composed pass) *The following two relationships hold:*

$$\mathit{If\ fail}\ G\,\langle\tfrac{\Sigma}{U}\rangle \text{ is legal, then } \mathit{fail}\ G\,\langle\tfrac{\Sigma}{U}\rangle \Longleftarrow \mathit{call}\ G\,\langle\tfrac{\Sigma}{U}\rangle. \tag{1}$$

$$\mathit{If\ exit}\ G\,\langle\tfrac{\Sigma}{U}\rangle \text{ is legal, then } \mathit{exit}\ G\,\langle\tfrac{\Sigma}{U}\rangle \Longleftarrow \mathit{call}\ G\,\langle\tfrac{\Sigma^\circ}{U}\rangle, \text{ with } \Sigma \succeq \Sigma^\circ \tag{2}$$

For (2) further holds: If G is a disjunction, then $\Sigma \succ \Sigma^\circ$, starting with $OR(N)$ for some N, and analogously for a unification or an atomic goal.

We already know that for a legal redo event holds $\mathit{redo}\ G\,\langle\frac{\Sigma}{U}\rangle \longleftarrow \mathit{exit}\ G\,\langle\frac{\Sigma}{U}\rangle$, which adds some more relationships like

$$\text{If } \mathit{fail}\ G\,\langle\tfrac{\Sigma}{U}\rangle \Longleftarrow \mathit{redo}\ G\,\langle\tfrac{\Sigma'}{U}\rangle, \text{ then } \Sigma' \succeq \Sigma \tag{3}$$

$$\text{If } \mathit{redo}\ G\,\langle\tfrac{\Sigma}{U}\rangle \text{ is legal, then } \mathit{redo}\ G\,\langle\tfrac{\Sigma}{U}\rangle \Longleftarrow \mathit{call}\ G\,\langle\tfrac{\Sigma^\circ}{U}\rangle, \text{ with } \Sigma \succeq \Sigma^\circ \tag{4}$$

As a by-product, the following supplement to Lemma 4.5 can be obtained, leading to a conclusion that the only final events are legal pop events with an empty A-stack:

Lemma 6.6 (non-final event) *If E is a legal push event, then there is always a transition $E \rightarrowtriangle E_1$.*

7 Independence claim and modularity

Remark 7.1 (up-to-date calls) *Bearing in mind the canonical form of the clauses, as well as idempotency of the mgus in our model, it can be seen from the rule (S_1:atom:1) that for any legal event $\mathit{call}\,\sigma(B)\,\langle\frac{\Sigma}{G_A \bullet U}\rangle$ holds: $\Sigma(\sigma(B)) = \sigma(B)$. In other words, the goal part of such an event is up-to-date with respect to the B-stack.*

Theorem 7.2 (adding stacks) *Let $\mathit{call}\ G\,\langle\frac{nil}{nil}\rangle \rightarrowtriangle E_1 \rightarrowtriangle \ldots \rightarrowtriangle E_n \rightarrowtriangle$*

Pop $G\,\langle\frac{\Omega}{nil}\rangle$ *be a legal derivation. Then for every* G°, U *and* Σ *such that call* $G^\circ\,\langle\frac{\Sigma}{U}\rangle$ *is a legal event and* $\Sigma(G^\circ) = G$, *holds:*

$$call\ G^\circ\,\langle\tfrac{\Sigma}{U}\rangle \rightarrowtriangle E_1^\circ \oplus \langle\tfrac{\Sigma}{U}\rangle \rightarrowtriangle \ldots \rightarrowtriangle E_n^\circ \oplus \langle\tfrac{\Sigma}{U}\rangle \rightarrowtriangle Pop\ G^\circ\,\langle\tfrac{\Omega+\Sigma}{U}\rangle$$

is also a legal derivation. Here if $E_i = \Gamma\ H\,\langle\frac{\Theta}{V}\rangle$, *then* $E_i^\circ := \Gamma\ H^\circ\,\langle\frac{\Theta}{V^\circ}\rangle$, *where* $\Sigma(H^\circ) = H$ *and* $\Sigma(V^\circ) = V$.

Theorem 7.3 (subtracting stacks) *Let call* $G\,\langle\frac{\Sigma}{U}\rangle \rightarrowtriangle E_1 \rightarrowtriangle \ldots \rightarrowtriangle E_n \rightarrowtriangle Pop\ G\,\langle\frac{\Omega}{U}\rangle$ *be a legal derivation, and* $E_i \neq Pop\ \_\,\langle\frac{-}{U}\rangle$ *for every* i. *Then for* $G' := \Sigma(G)$ *holds:*

$$call\ G'\,\langle\tfrac{nil}{nil}\rangle \rightarrowtriangle E_1' \ominus \langle\tfrac{\Sigma}{U}\rangle \rightarrowtriangle \ldots \rightarrowtriangle E_n' \ominus \langle\tfrac{\Sigma}{U}\rangle \rightarrowtriangle Pop\ G'\,\langle\tfrac{\Omega-\Sigma}{nil}\rangle$$

is also a legal derivation. Here if $E_i = \Gamma\ H\,\langle\frac{\Theta}{V}\rangle$, *then* $E_i' := \Gamma\ H'\,\langle\frac{\Theta}{V'}\rangle$, *where* $\Sigma(H) = H'$ *and* $\Sigma(V) = V'$.

The previous two claims show that a simple pass, starting with a call event, is independent on the starting contents of the stacks. The stacks represent the *context* of the computation for the goal at hand.

It would be interesting to see whether similar properties hold for a simple pass starting with a redo event. First note that if *redo* $G\,\langle\frac{\Sigma}{U}\rangle$ is a legal event, then, according to Theorem 6.1, *redo* $G\,\langle\frac{\Sigma}{U}\rangle \longleftarrow$ *exit* $G\,\langle\frac{\Sigma}{U}\rangle$. Using Theorem 6.5, we further obtain that *redo* $G\,\langle\frac{\Sigma}{U}\rangle \Longleftarrow$ *call* $G\,\langle\frac{\Sigma^\circ}{U}\rangle$, where $\Sigma \succeq \Sigma^\circ$. This gives us a hint on how much we may cut off of the stacks. Observe that, as opposed to a call event, we do not necessarily arrive at a legal redo event by means of adding or subtracting stacks. (For example, a redo event with an empty A-stack cannot be reached.) But if we do, then we may know its next stage, due to the following claim:

Lemma 7.4 (starting with redo) *Let redo* $G\,\langle\frac{\Sigma}{U}\rangle \rightarrowtriangle E_1 \rightarrowtriangle \ldots \rightarrowtriangle E_n \rightarrowtriangle Pop\ G\,\langle\frac{\Omega}{U}\rangle$ *be a legal derivation, and* $E_i \neq Pop\ \_\,\langle\frac{-}{U}\rangle$ *for every* i. *Further let redo* $G\,\langle\frac{\Sigma}{U}\rangle \Longleftarrow$ *call* $G\,\langle\frac{\Sigma^\circ}{U}\rangle$. *Then for* $G' := \Sigma^\circ(G)$ *and* E_i' *analogous as in Theorem 7.3 holds:*

$$redo\ G'\,\langle\tfrac{\Sigma-\Sigma^\circ}{nil}\rangle \rightarrowtriangle E_1' \ominus \langle\tfrac{\Sigma^\circ}{U}\rangle \rightarrowtriangle \ldots \rightarrowtriangle E_n' \ominus \langle\tfrac{\Sigma^\circ}{U}\rangle \rightarrowtriangle Pop\ G'\,\langle\tfrac{\Omega-\Sigma^\circ}{nil}\rangle.$$

Moreover, for $\Theta(G^\circ) = G$ *and* E_i° *analogous as in Theorem 7.2 holds:*

$$redo\ G^\circ\,\langle\tfrac{\Sigma+\Theta}{U+V}\rangle \rightarrowtriangle E_1^\circ \oplus \langle\tfrac{\Theta}{V}\rangle \rightarrowtriangle \ldots \rightarrowtriangle E_n^\circ \oplus \langle\tfrac{\Theta}{V}\rangle \rightarrowtriangle Pop\ G^\circ\,\langle\tfrac{\Omega+\Theta}{U+V}\rangle.$$

Our next aim is to prove: even upon backtracking, the goal shall pass always the same stages, independent on the starting contents of the stacks.

Lemma 7.5 (backward independence) *Consider the sequences*

$$call\ G\,\langle\tfrac{\Sigma}{U}\rangle \longrightarrow exit\ G\,\langle\tfrac{\Sigma'}{U}\rangle \longrightarrow redo\ G\,\langle\tfrac{\Sigma'}{U}\rangle \longrightarrow exit\ G\,\langle\tfrac{\Omega}{U}\rangle$$
$$call\ H\,\langle\tfrac{\Theta}{V}\rangle \longrightarrow exit\ H\,\langle\tfrac{\Theta'}{V}\rangle \longrightarrow redo\ H\,\langle\tfrac{\Theta'}{V}\rangle \longrightarrow exit\ H\,\langle\tfrac{\Psi}{V}\rangle$$

where $\Theta(H) = \Sigma(G)$. *Then* $\Theta' = \Sigma' - \Sigma + \Theta$ *and* $\Psi = \Omega - \Sigma + \Theta$.

In other words, the second derivation starts from a different context, but from the same goal, and it passes the same stages as in the first derivation. Notice the presence of backtracking.

This result can be generalized for arbitrary composed passes. Bearing in mind that there can be only one appearance of a fail stage in a composed pass, due to Lemma 6.3, we arrive at a general claim of independence upon the context of derivation.

Theorem 7.6 (independence) *Let the following sequences be composable, where $m, k \geq 1$, and let $\Theta(H) = \Sigma(G)$.*

$$\begin{aligned} &\mathit{call}\ G\ \langle \tfrac{\Sigma}{U} \rangle \longrightarrow E_1^{G,U} \longrightarrow \ldots \longrightarrow E_m^{G,U} \\ &\mathit{call}\ H\ \langle \tfrac{\Theta}{V} \rangle \longrightarrow E_1^{H,V} \longrightarrow \ldots \longrightarrow E_k^{H,V} \end{aligned}$$

Then for any i in common ($i \leq m, k$) holds:

$$\mathit{If}\ E_i^{G,U} := \Gamma\ G\ \langle \tfrac{\Sigma'}{U} \rangle \quad \mathit{then} \quad E_i^{H,V} = \Gamma\ H\ \langle \tfrac{\Theta'}{V} \rangle, \quad \mathit{such\ that}\ \Theta' = \Sigma' - \Sigma + \Theta.$$

Independence on the context of derivation, together with compositionality with respect to conjunction and disjunction, and aggregation of transitions into bigger steps, establishes a kind of *modularity* in S_1:PP. There is an illustration at the end of Section 8.

In the remaining technical section we define a concept of computation intended to mimick SLD-resolution in the style of Prolog, i. e. SLD-resolution with leftmost selection and a sequential, left-to-right depth-first search. For brevity, thus restricted SLD-resolution shall be called *Prolog computation*.

8 Modeling pure Prolog computations

Based on the concept of stages, it is possible to express Prolog computation in a succinct way. Throughout this section, Π denotes again a pure Prolog program in canonical form, as defined in Subsection 3.3 and Definition 2.2.

Theorem 8.1 (existential termination, success, failure) *Prolog computation of a goal G relative to Π terminates existentially if there is a simple pass*

$$\mathit{call}\ G\ \langle \tfrac{nil}{nil} \rangle \longrightarrow_\Pi \mathit{Pop}\ G\ \langle \tfrac{\Delta}{nil} \rangle$$

In case $Pop = exit$, the Prolog computation of G is successful, otherwise it is failed.

Theorem 8.2 (the first computed answer) *If $\mathit{call}\ G\ \langle \frac{nil}{nil} \rangle \longrightarrow_\Pi \mathit{exit}\ G\ \langle \frac{\Delta}{nil} \rangle$, then $subst(\Delta)\mid_G$ is the first computed answer substitution for G relative to Π in Prolog.*

Theorem 8.3 (all computed answers, universal termination) *In a composable sequence*

$$\mathit{call}\ G\ \langle \tfrac{nil}{1/G, \mathsf{fail} \bullet nil} \rangle \longrightarrow_\Pi^{2k-1} \mathit{exit}\ G\ \langle \tfrac{\Delta}{1/G, \mathsf{fail} \bullet nil} \rangle$$

is $subst(\Delta)\ |_G$ *the kth computed answer substitution for* G *relative to* Π *in Prolog. Furthermore,* G *is universally terminating relative to* Π *if*

$$call\ G\ \langle \tfrac{nil}{1/G,\mathsf{fail}\bullet nil}\rangle \Longrightarrow_\Pi fail\ G\ \langle \tfrac{nil}{1/G,\mathsf{fail}\bullet nil}\rangle$$

Actually we can be a bit more precise, by considering a goal G as a subgoal of other goals. Recall that, if $call\ G\ \langle \frac{\Sigma}{U}\rangle$ is legal and $U = a_n \bullet ... \bullet a_1 \bullet nil$, then $call\ G\ \langle \frac{\Sigma}{U}\rangle \overset{\triangleleft}{\leftarrow}_\Pi Push\ \lceil a_1 \rceil\ \langle \frac{\dot{-}}{nil}\rangle$. In other words, $call\ G\ \langle \frac{\Sigma}{U}\rangle$ is a subevent of the most recent $Push\ \lceil a_1 \rceil\ \langle \frac{\dot{-}}{nil}\rangle$. Moreover, since a push event with an empty A-stack cannot be reached, we know that ours must be the oldest event of the derivation, of the form $call\ \lceil a_1 \rceil\ \langle \frac{nil}{nil}\rangle$. In analogy to subgoals in Prolog, let us define two new concepts:

Definition 8.4 (S_1-supergoal, S_1-subgoal) *Let* $call\ G\ \langle \frac{\Sigma}{U}\rangle$ *be legal. If* $U = a_n \bullet ... \bullet a_1 \bullet nil$*, we say that* $\lceil a_1 \rceil$ *is the* S_1-supergoal of G'*, and* G' *is a* S_1-subgoal *of* $\lceil a_1 \rceil$*, where* $G' := \Sigma(G)$*. In case* $U = nil$*, we define* G' *to be its own* S_1*-supergoal.*

Theorem 8.5 (supergoal) *Let* $call\ G\ \langle \frac{\Sigma^\circ}{U}\rangle$ *be a legal event. Consider the maximal composable sequence* $call\ G\ \langle \frac{\Sigma^\circ}{U}\rangle \longrightarrow_\Pi{}^{2k-1} exit\ G\ \langle \frac{\Sigma}{U}\rangle$*. Then holds:* $subst(\Sigma - \Sigma^\circ)\ |_{G'}$ *is the kth and last computed answer substitution for* G' *relative to* Π *in Prolog, where* $G' := \Sigma^\circ(G)$ *and the query of the Prolog computation was the* S_1*-supergoal of* G'*.*

In conclusion, we show an example of a modular derivation. Let $exit\ A, B\ \langle \frac{\Sigma}{U}\rangle$ be a legal event. How could it have been derived? Luckily, converse transitions are deterministic for legal events (uniqueness property), so we may unfold a legal derivation from whichever endpoint it seems more promising. The index Π will be omitted, since the following holds for any program.

$$\begin{aligned}
&exit\ A, B\ \langle \tfrac{\Sigma}{U}\rangle \\
&\leftarrow\ exit\ B\ \langle \tfrac{\Sigma}{2/A,B\bullet U}\rangle, \text{ by } (S_1\text{:conj:4}) \\
&\Longleftarrow\ call\ B\ \langle \tfrac{\Sigma^\circ}{2/A,B\bullet U}\rangle, \text{ by Theorem 6.5.(2), with } \Sigma \succeq \Sigma^\circ \\
&\leftarrow\ exit\ A\ \langle \tfrac{\Sigma^\circ}{1/A,B\bullet U}\rangle, \text{ by } (S_1\text{:conj:2}) \\
&\Longleftarrow\ call\ A\ \langle \tfrac{\Sigma^{\circ\circ}}{1/A,B\bullet U}\rangle, \text{ by Theorem 6.5.(2), with } \Sigma^\circ \succeq \Sigma^{\circ\circ} \\
&\leftarrow\ call\ A, B\ \langle \tfrac{\Sigma^{\circ\circ}}{U}\rangle, \text{ by } (S_1\text{:conj:1})
\end{aligned}$$

Additionally, it can be seen that the whole derivation is a composable sequence. So we obtain

Lemma 8.6 (conjunction) *Let* $exit\ A, B\ \langle \frac{\Sigma}{U}\rangle$ *be a legal event. Then* $\Sigma^\circ, \Sigma^{\circ\circ}$ *exist with* $\Sigma \succeq \Sigma^\circ \succeq \Sigma^{\circ\circ}$ *such that* $exit\ A, B\ \langle \frac{\Sigma}{U}\rangle \Longleftarrow call\ A, B\ \langle \frac{\Sigma^{\circ\circ}}{U}\rangle$*, and also* $exit\ B\ \langle \frac{\Sigma}{2/A,B\bullet U}\rangle \Longleftarrow call\ B\ \langle \frac{\Sigma^\circ}{2/A,B\bullet U}\rangle$ *and* $exit\ A\ \langle \frac{\Sigma^\circ}{1/A,B\bullet U}\rangle \Longleftarrow call\ A\ \langle \frac{\Sigma^{\circ\circ}}{1/A,B\bullet U}\rangle$*.*

In a similar way we can unfold a disjunction and an atomic goal, thus simulating the vanilla meta-interpreter in S_1:PP.

9 Related work

The earliest and still authoritative model of pure Prolog execution is *SLD-derivation* [3]. There is no explicit disjunctive information in this model, so backtracking is not formalized at all. This issue is addressed in another popular model, *SLD-tree* [3]. One drawback with both is lack of compositionality: it is not possible to express e. g. an SLD-tree of a conjunction via the SLD-trees of the conjuncts. Much further work was inspired by the traditional metaphor of a box with four ports, known as the Byrd model [9], designed to represent the evolution of a procedure call in Prolog. The Byrd model is a seminal work in control flow, but it proved hard to formalize, and variable handling was not tackled in the original work at all. Tobermann and Beckstein formalize in [24] the graph traversal idea of Byrd, defining the central notion of execution *trace* (of a given query with respect to a given program), as a path in a trace graph. The ports are quite lucidly defined as hierarchical nodes of such a graph. However, trace graphs are not very manageable. Cheng et al. [10] propose another graphical model called *SLD-contour*, a variant of SLD-tree, but whose traversal should corresponds better to the execution trace. The idea of a SLD-contour turned out to be extendable to conjunctive and disjunctive goals, giving the presumably first compositional model of pure Prolog execution. In essence, they extend the notion of the Byrd's box to general goal formulas, thus preceding our own research. Another formal approach in the spirit of the Byrd model is a continuation-based idea of Jahier, Ducassé and Ridoux [16]. There is also an attempt to define Byrd's box within our earlier work [19], but it suffers essentially the same problem as the other trace specifications, starting from the tracer break/1 of [9]: In these approaches, traces of Prolog execution are generated, but there is no support for reasoning about the structure of the execution, so the trace generation would have to be accompanied by trace analysis via some other device. There are various other models of pure (or even full) Prolog execution. Comparable to our work are stack-based approaches. Stärk gives in [23], as a side issue, a simple operational semantics of pure logic programming. A state of execution is a stack of frame stacks. The seminal paper of Jones and Mycroft [17] was the first to present a stack-of-stacks model of execution, applicable to pure Prolog with cut added. Stack-of-stacks idea reflects the usual memory management of Prolog implementations, and it is in general not possible to abstract parts of execution.

10 Summary

The motivation for this work was to be able to prove that a certain program transformation used in debugging does not affect the original program in any 'serious' way. It soon became obvious that even to formulate such a property is no easy task, and requires quite a powerful model of Prolog execution.

The model needed to be able to represent Prolog execution in sufficient detail, and on the other hand, to support formal reasoning about it (this implied some means of abstraction, like modularity). Apart from these two vital requirements, the model should lend itself to efficient implementation, for the sake of its own development and practical relevance (this implied purely symbolic representation and small overhead).

As a result, a rather simple calculus S_1:PP defining pure Prolog execution has been developed. Due to compositionality with respect to conjunction and disjunction, and aggregation of transitions into bigger steps, a kind of modularity is achieved. Forward and backward computation of Prolog are treated in a uniform manner, making the model suitable to represent backtracking. It remains to be seen how far this idea can be stretched towards defining full Standard Prolog. Also, the potential for specifying other kinds of reversible computation seems worth investigating.

Acknowledgement

Many thanks for helpful comments upon an earlier draft of this paper are due to C. Beierle and the anonymous referees.

References

[1] J. H. Andrews. *Logic Programming: Operational Semantics and Proof Theory.* Cambridge University Press, 1992.

[2] J. H. Andrews. The witness properties and the semantics of the Prolog cut. *Theory and Practice of Logic Programming*, 3(1):1–59, 2003.

[3] K. R. Apt and M. H. van Emden. Contributions to the theory of logic programming. *J. of the ACM*, 29:841–862, 1982.

[4] B. Arbab and D. M. Berry. Operational and denotational semantics of Prolog. *J. of Logic Programming*, 4(4):309–329, 1987.

[5] M. Baudinet. Proving termination properties of Prolog programs: A semantic approach. *J. of Logic Programming*, 14:1–29, 1992.

[6] M. Billaud. Simple operational and denotational semantics for Prolog with cut. *Theoretical Computer Science*, 71(2):193–208, 1990.

[7] M. Billaud. Axiomatizations of backtracking. In *Proc. of the STACS*, pages 71–82, 1992.

[8] E. Börger and D. Rosenzweig. A mathematical definition of full Prolog. *Science of Computer Programming*, 24(3):249–286, 1995.

[9] Lawrence Byrd. Understanding the control flow of Prolog programs. In S. A. Tärnlund, editor, *Proc. of the 1980 Logic Programming Workshop*, pages 127–138, Debrecen, Hungary, 1980. Also as D. A. I. Research Paper No. 151.

[10] M. H. M. Cheng, M. H. van Emden, R. N. Horspool, and M. Levy. Compositional operational semantics for Prolog programs. *J. New Generation Computing,*, 10(3):315–328, 1992.

[11] E. P. de Vink. Comparative semantics for Prolog with cut. *Science of Computer Programming*, 13(1):237–264, 1989.

[12] P. Deransart, A. Ed-Dbali, and L. Cervoni. *Prolog: The Standard (Reference Manual)*. Springer-Verlag, 1996.

[13] P. Deransart and G. Ferrand. An operational formal definition of Prolog: A specification method an its application. *J. New Generation Computing*, 10:121–171, 1992.

[14] E. Eder. Properties of substitutions and unifications. *J. Symbolic Computation*, 1:31–46, 1985.

[15] B. Elbl. A declarative semantics for depth-first logic programs. *J. of Logic Programming*, 41:27–66, 1999.

[16] E. Jahier, M. Ducassé, and O. Ridoux. Specifying Byrd's box model with a continuation semantics. In *Proc. of the WLPE'99, Las Cruces, NM*, volume 30 of *ENTCS*. Elsevier, 2000. http://www.elsevier.com/locate/entcs/.

[17] N. D. Jones and A. Mycroft. Stepwise development of operational and denotational semantics for Prolog. In *Proc. of the ISLP'84*, pages 281–288, Atlantic City, 1984.

[18] M. Kulaš. Pure Prolog execution in 21 rules. In *Proc. of the 5th Workshop on Rule-Based Constraint Reasoning and Programming (RCoRP'03), Kinsale*, September 2003.

[19] M. Kulaš and C. Beierle. Defining Standard Prolog in rewriting logic. In K. Futatsugi, editor, *Proc. of WRLA 2000, Kanazawa*, volume 36 of *ENTCS*. Elsevier, 2001. http://www.elsevier.com/locate/entcs.

[20] T. Lindgren. A continuation-passing style for Prolog. In *Proc. of the 11th Int. Symposium on Logic Programming (ILPS'94)*, pages 603–617, Ithaca, NY, 1994.

[21] J. W. Lloyd. *Foundations of Logic Programming*. Springer-Verlag, 1987.

[22] T. Nicholson and N. Foo. A denotational semantics for Prolog. *ACM Trans. on Prog. Lang. and Systems*, 11(4):650–665, 1989.

[23] Robert F. Stärk. The theoretical foundations of LPTP (a logic program theorem prover). *J. of Logic Programming*, 36(3):241–269, 1998. Source distribution http://www.inf.ethz.ch/~staerk/lptp.html.

[24] G. Tobermann and C. Beckstein. What's in a trace: The box model revisited. In *Proc. of the 1st Int. Workshop on Automated and Algorithmic Debugging (AADEBUG'93), Linköping*, volume 749 of *LNCS*. Springer-Verlag, 1993.

A An example proof

As an illustration of S_1:PP, here is a proof of the pendant lemma, and its application in proving uniqueness of legal transitions. Although lengthy, the proof of Lemma 4.1 is rather schematic.

Lemma 4.1 (pendant) *If* $call\ G\ \langle \frac{nil}{nil} \rangle \rightarrowtriangle^*\ fail\ H\ \langle \frac{\Theta}{W} \rangle$, *then*

$$call\ G\ \langle \tfrac{nil}{nil} \rangle \rightarrowtriangle^*\ call\ H\ \langle \tfrac{\Theta}{W} \rangle \rightarrowtriangle^*\ fail\ H\ \langle \tfrac{\Theta}{W} \rangle. \tag{5}$$

If $call\ G\ \langle \frac{nil}{nil} \rangle \rightarrowtriangle^*\ redo\ H\ \langle \frac{\Theta}{W} \rangle$, *then*

$$call\ G\ \langle \tfrac{nil}{nil} \rangle \rightarrowtriangle^*\ exit\ H\ \langle \tfrac{\Theta}{W} \rangle \rightarrowtriangle^*\ redo\ H\ \langle \tfrac{\Theta}{W} \rangle. \tag{6}$$

Proof. We shall use induction on the length n of derivation to prove the two parts of the lemma simultaneously. The inductive assumption for each of the parts shall be denoted as IND(FAIL) and IND(REDO) respectively. First let us construct base cases. With Lemma 4.1(5) we are lucky, since derivations of length $n = 1$ can be failed. There are three such cases, and they directly satisfy Lemma 4.1(5):

$$call\ \mathsf{fail}\ \langle \tfrac{nil}{nil} \rangle \rightarrowtriangle fail\ \mathsf{fail}\ \langle \tfrac{nil}{nil} \rangle,\ \text{by}\ (S_1\text{:fail}) \tag{7}$$

$$call\ T_1{=}T_2\ \langle \tfrac{nil}{nil} \rangle \rightarrowtriangle fail\ T_1{=}T_2\ \langle \tfrac{nil}{nil} \rangle,\ \text{by}\ (S_1\text{:unif:1}),\ \text{if no mgu} \tag{8}$$

$$call\ G_A\ \langle \tfrac{nil}{nil} \rangle \rightarrowtriangle fail\ G_A\ \langle \tfrac{nil}{nil} \rangle,\ \text{by}\ (S_1\text{:atom:1}),\ \text{if no resolvent} \tag{9}$$

With Lemma 4.1(6) we have a harder time since no derivation of length $n \leq 2$ can end up in a redo, so with the number of derivations getting out of hand, we turn to reconstructing a minimal legal derivation of a redo. A legal redo event stems from a redo or from a fail, so for a minimal derivation it had to be a fail: $fail\ B\ \langle \frac{\Sigma}{2/A,B \bullet V} \rangle \rightarrowtriangle redo\ A\ \langle \frac{\Sigma}{1/A,B \bullet V} \rangle$. Now we are looking for a minimal derivation of this fail event, which may not include any redo events. The predecessor may be a fail (pushing the A-stack) or a call (not affecting the stacks). In case of a call, we get $exit\ A\ \langle \frac{\Sigma}{1/A,B \bullet V} \rangle \rightarrowtriangle call\ B\ \langle \frac{\Sigma}{2/A,B \bullet V} \rangle \rightarrowtriangle fail\ B\ \langle \frac{\Sigma}{2/A,B \bullet V} \rangle$. In this manner we eventually reconstruct the minimal derivations for redo:

$$call\ A,B\ \langle \tfrac{nil}{nil} \rangle \rightarrowtriangle call\ A\ \langle \tfrac{nil}{1/A,B \bullet nil} \rangle \rightarrowtriangle exit\ A\ \langle \tfrac{nil}{1/A,B \bullet nil} \rangle \rightarrowtriangle call\ B\ \langle \tfrac{nil}{2/A,B \bullet nil} \rangle \rightarrowtriangle$$
$$\rightarrowtriangle fail\ B\ \langle \tfrac{nil}{2/A,B \bullet nil} \rangle \rightarrowtriangle redo\ A\ \langle \tfrac{nil}{1/A,B \bullet nil} \rangle$$

and they satisfy Lemma 4.1(6).

Assume Lemma 4.1 holds for derivations of length $1 \leq n < k$ and consider a derivation of length k. We shall use the following simple observation: If $call\ G\ \langle \frac{nil}{nil} \rangle \rightarrowtriangle^*\ \Gamma\ A\ \langle \frac{\Theta}{W} \rangle$, and $W \neq nil$ or $\Gamma \neq call$, then also

$$call\ G\ \langle \tfrac{nil}{nil} \rangle \rightarrowtriangle^+\ \Gamma\ A\ \langle \tfrac{\Theta}{W} \rangle \tag{10}$$

First we discuss the cases for a legal fail event. There are eight possibilities for the last step of its derivation: (S_1:fail), (S_1:unif:1), (S_1:atom:1), (S_1:conj:3), (S_1:disj:3), (S_1:atom:3), (S_1:true:2) and (S_1:unif:2). The first three cases are again directly

satisfied. Case (S_1:conj:3):

$$call\ G\ \langle \tfrac{nil}{nil} \rangle \rightarrowtriangle^{+} fail\ A\ \langle \tfrac{\Theta}{1/A,B \bullet W} \rangle \rightarrowtriangle fail\ A, B\ \langle \tfrac{\Theta}{W} \rangle,\ \text{the last step} \tag{11}$$

$$call\ G\ \langle \tfrac{nil}{nil} \rangle \rightarrowtriangle^{*} call\ A\ \langle \tfrac{\Theta}{1/A,B \bullet W} \rangle \rightarrowtriangle^{*} fail\ A\ \langle \tfrac{\Theta}{1/A,B \bullet W} \rangle,\ \text{by IND(FAIL)} \tag{12}$$

$$call\ G\ \langle \tfrac{nil}{nil} \rangle \rightarrowtriangle^{*} call\ A, B\ \langle \tfrac{\Theta}{W} \rangle \rightarrowtriangle call\ A\ \langle \tfrac{\Theta}{1/A,B \bullet W} \rangle,\ \text{predecessor \& (10)} \tag{13}$$

$$call\ G\ \langle \tfrac{nil}{nil} \rangle \rightarrowtriangle^{*} call\ A, B\ \langle \tfrac{\Theta}{W} \rangle \rightarrowtriangle^{*} fail\ A, B\ \langle \tfrac{\Theta}{W} \rangle,\ \text{by (11)-(13)},\quad q.e.d.$$

Case (S_1:atom:3) is similar to the previous. Case (S_1:disj:3) can be handled by double use of IND(FAIL):

$$call\ G\ \langle \tfrac{nil}{nil} \rangle \rightarrowtriangle^{+} fail\ B\ \langle \tfrac{\Theta}{2/A;B \bullet W} \rangle \rightarrowtriangle fail\ A; B\ \langle \tfrac{\Theta}{W} \rangle,\ \text{the last step} \tag{14}$$

$$call\ G\ \langle \tfrac{nil}{nil} \rangle \rightarrowtriangle^{*} call\ B\ \langle \tfrac{\Theta}{2/A;B \bullet W} \rangle \rightarrowtriangle^{*} fail\ B\ \langle \tfrac{\Theta}{2/A;B \bullet W} \rangle,\ \text{by IND(FAIL)} \tag{15}$$

$$call\ G\ \langle \tfrac{nil}{nil} \rangle \rightarrowtriangle^{*} fail\ A\ \langle \tfrac{\Theta}{1/A;B \bullet W} \rangle \rightarrowtriangle call\ B\ \langle \tfrac{\Theta}{2/A;B \bullet W} \rangle,\ \text{predecessor \& (10)} \tag{16}$$

$$call\ G\ \langle \tfrac{nil}{nil} \rangle \rightarrowtriangle^{*} call\ A\ \langle \tfrac{\Theta}{1/A;B \bullet W} \rangle \rightarrowtriangle^{*} fail\ A\ \langle \tfrac{\Theta}{1/A;B \bullet W} \rangle,\ \text{by IND(FAIL)} \tag{17}$$

$$call\ G\ \langle \tfrac{nil}{nil} \rangle \rightarrowtriangle^{*} call\ A; B\ \langle \tfrac{\Theta}{W} \rangle \rightarrowtriangle call\ A\ \langle \tfrac{\Theta}{1/A;B \bullet W} \rangle,\ \text{predecessor \& (10)} \tag{18}$$

$$call\ G\ \langle \tfrac{nil}{nil} \rangle \rightarrowtriangle^{*} call\ A; B\ \langle \tfrac{\Theta}{W} \rangle \rightarrowtriangle^{*} fail\ A; B\ \langle \tfrac{\Theta}{W} \rangle,\ \text{by (14)-(18)},\quad q.e.d.$$

For the last two cases (redo-fail transitions) we shall use IND(REDO). Case (S_1:true:2):

$$call\ G\ \langle \tfrac{nil}{nil} \rangle \rightarrowtriangle^{+} redo\ \mathsf{true}\ \langle \tfrac{\Theta}{W} \rangle \rightarrowtriangle fail\ \mathsf{true}\ \langle \tfrac{\Theta}{W} \rangle,\ \text{the last step} \tag{19}$$

$$call\ G\ \langle \tfrac{nil}{nil} \rangle \rightarrowtriangle^{*} exit\ \mathsf{true}\ \langle \tfrac{\Theta}{W} \rangle \rightarrowtriangle^{*} redo\ \mathsf{true}\ \langle \tfrac{\Theta}{W} \rangle,\ \text{by IND(REDO)} \tag{20}$$

$$call\ G\ \langle \tfrac{nil}{nil} \rangle \rightarrowtriangle^{*} call\ \mathsf{true}\ \langle \tfrac{\Theta}{W} \rangle \rightarrowtriangle exit\ \mathsf{true}\ \langle \tfrac{\Theta}{W} \rangle,\ \text{predecessor \& (10)} \tag{21}$$

$$call\ G\ \langle \tfrac{nil}{nil} \rangle \rightarrowtriangle^{*} call\ \mathsf{true}\ \langle \tfrac{\Theta}{W} \rangle \rightarrowtriangle^{*} fail\ \mathsf{true}\ \langle \tfrac{\Theta}{W} \rangle,\ \text{by (19)-(21)},\quad q.e.d.$$

The last case (S_1:unif:2) can be handled in the same manner.

It remains to discuss the cases for a legal redo event. There are four possibilities for the last step of its derivation: (S_1:conj:5), (S_1:conj:6), (S_1:disj:5) and (S_1:atom:4). Here a symmetry between 'entering redo' and 'leaving exit' takes over. Namely, if we take the rules for entering redo, turn the arrow around and replace exit for redo and call for fail, then we obtain the rules for leaving exit. Due to determinacy of such transitions, each 'entering redo' can be simulated with an appropriate 'leaving exit', which reconstructs the stacks in exactly the same manner. The case (S_1:conj:5) is special because it uses IND(FAIL):

$$call\ G\ \langle \tfrac{nil}{nil} \rangle \rightarrowtriangle^{+} fail\ B\ \langle \tfrac{\Theta}{2/A,B \bullet W} \rangle \rightarrowtriangle redo\ A\ \langle \tfrac{\Theta}{1/A,B \bullet W} \rangle,\ \text{the last step} \tag{22}$$

$$call\ G\ \langle \tfrac{nil}{nil} \rangle \rightarrowtriangle^{*} call\ B\ \langle \tfrac{\Theta}{2/A,B \bullet W} \rangle \rightarrowtriangle^{*} fail\ B\ \langle \tfrac{\Theta}{2/A,B \bullet W} \rangle,\ \text{by IND(FAIL)} \tag{23}$$

$$call\ G\ \langle \tfrac{nil}{nil} \rangle \rightarrowtriangle^{*} exit\ A\ \langle \tfrac{\Theta}{1/A,B \bullet W} \rangle \rightarrowtriangle call\ B\ \langle \tfrac{\Theta}{2/A,B \bullet W} \rangle,\ \text{predecessor \& (10)} \tag{24}$$

$$call\ G\ \langle \tfrac{nil}{nil} \rangle \rightarrowtriangle^{*} exit\ A\ \langle \tfrac{\Theta}{1/A,B \bullet W} \rangle \rightarrowtriangle^{*} redo\ A\ \langle \tfrac{\Theta}{1/A,B \bullet W} \rangle,\ \text{by (22)-(24)},\quad q.e.d.$$

Case (S_1:conj:6):

$$call\ G\ \langle \tfrac{nil}{nil} \rangle \rightarrowtriangle^{+} redo\ A, B\ \langle \tfrac{\Theta}{W} \rangle \rightarrowtriangle redo\ B\ \langle \tfrac{\Theta}{2/A,B \bullet W} \rangle \tag{25}$$

$$call\ G\ \langle \tfrac{nil}{nil} \rangle \rightarrowtriangle^{*} exit\ A, B\ \langle \tfrac{\Theta}{W} \rangle \rightarrowtriangle^{*} redo\ A, B\ \langle \tfrac{\Theta}{W} \rangle,\ \text{by IND(REDO)} \tag{26}$$

$$call\ G\ \langle \tfrac{nil}{nil} \rangle \rightarrowtriangle^{*} exit\ B\ \langle \tfrac{\Theta}{2/A,B \bullet W} \rangle \rightarrowtriangle exit\ A, B\ \langle \tfrac{\Theta}{W} \rangle,\ \text{predecessor \& (10)} \tag{27}$$

$$call\ G\ \langle \tfrac{nil}{nil} \rangle \rightarrowtriangle^{*} exit\ B\ \langle \tfrac{\Theta}{2/A,B \bullet W} \rangle \rightarrowtriangle redo\ B\ \langle \tfrac{\Theta}{2/A,B \bullet W} \rangle,\ \text{by (25)-(27)},\quad q.e.d.$$

The last two cases (S_1:disj:5) and (S_1:atom:4) can be handled in the same manner. This concludes the proof of Lemma 4.1. □

Theorem 4.2 (legal transitions are unique) *If E is a legal event, then E can have only one legal predecessor, and only one successor. In case E is non-initial, there is exactly one legal predecessor. In case E is non-final, there is exactly one successor.*

Proof. The successor part follows from the functionality of $\rightarrowtriangle$. Looking at the rules, we note that only two kinds of events may have more than one predecessor: *fail* $G_A\,\langle\frac{\Sigma}{U}\rangle$ and *fail* $T_1{=}T_2\,\langle\frac{\Sigma}{U}\rangle$. Let *fail* $T_1{=}T_2\,\langle\frac{\Sigma}{U}\rangle$ be a legal event. Then it has at least one derivation from an initial event, say *call* $G\,\langle\frac{nil}{nil}\rangle$. Its predecessor in this derivation may have been *call* $T_1{=}T_2\,\langle\frac{\Sigma}{U}\rangle$, on the condition that $\Sigma(T_1)$ and $\Sigma(T_2)$ have no mgu, or it may have been *redo* $T_1{=}T_2\,\langle\frac{\sigma\bullet\Sigma}{U}\rangle$. In the latter case we obtain:

$$\textit{call}\ G\,\langle\tfrac{nil}{nil}\rangle \rightarrowtriangle^*\ \textit{redo}\ T_1{=}T_2\,\langle\tfrac{\sigma\bullet\Sigma}{U}\rangle \rightarrowtriangle\ \textit{fail}\ T_1{=}T_2\,\langle\tfrac{\Sigma}{U}\rangle,\ \text{assumption} \tag{28}$$

$$\textit{call}\ G\,\langle\tfrac{nil}{nil}\rangle \rightarrowtriangle^*\ \textit{exit}\ T_1{=}T_2\,\langle\tfrac{\sigma\bullet\Sigma}{U}\rangle \rightarrowtriangle^*\ \textit{redo}\ T_1{=}T_2\,\langle\tfrac{\sigma\bullet\Sigma}{U}\rangle,\ \text{Lemma 4.1(6)} \tag{29}$$

$$\textit{call}\ G\,\langle\tfrac{nil}{nil}\rangle \rightarrowtriangle^*\ \textit{call}\ T_1{=}T_2\,\langle\tfrac{\Sigma}{U}\rangle \rightarrowtriangle\ \textit{exit}\ T_1{=}T_2\,\langle\tfrac{\sigma\bullet\Sigma}{U}\rangle,\ \text{by } (S_1\text{:unif:1}) \tag{30}$$

Let us comment a bit on (29)-(30). From (29) we know that *exit* $T_1{=}T_2\,\langle\frac{\sigma\bullet\Sigma}{U}\rangle$ is a legal event, i. e. it is reachable from an initial event. But there is only one possibility to enter *exit* $T_1{=}T_2\,\langle\frac{\sigma\bullet\Sigma}{U}\rangle$, namely via ($S_1$:unif:1), under the condition $mgu(\Sigma(T_1),\Sigma(T_2)) = \sigma$. To sum up: If *redo* $T_1{=}T_2\,\langle\frac{\sigma\bullet\Sigma}{U}\rangle$ is a legal predecessor of *fail* $T_1{=}T_2\,\langle\frac{\Sigma}{U}\rangle$, then $mgu(\Sigma(T_1),\Sigma(T_2)) = \sigma$.

So depending on the existence of this particular mgu, either *redo* $T_1{=}T_2\,\langle\frac{\sigma\bullet\Sigma}{U}\rangle$ or *call* $T_1{=}T_2\,\langle\frac{\Sigma}{U}\rangle$ is a legal predecessor of *fail* $T_1{=}T_2\,\langle\frac{\Sigma}{U}\rangle$, but never both.

By a similar argument, this time using Lemma 4.1(5), we can prove that *fail* $G_A\,\langle\frac{\Sigma}{U}\rangle$ can have only one legal predecessor. □