University "Alexandru Ioan Cuza" of Iaşi
Faculty of Computer Science

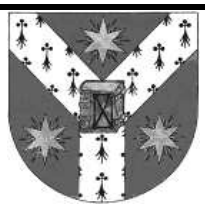
TECHNICAL REPORT

A Rewrite Stack Machine for ROC!

Georgiana Caltais, Eugen-Ioan Goriac
Dorel Lucanu, Gheorghe Grigoraş

TR 08-02, July 2008

ISSN 1224-9327



Universitatea "Alexandru Ioan Cuza" Iaşi
Facultatea de Informatică
Str. Berthelot 16, 6600-Iaşi, Romania
Tel. +40-32-201090, email: bibl@infoiasi.ro

A Rewrite Stack Machine for ROC! *

Georgiana Caltais
gcaltais@info.uaic.ro

Eugen-Ioan Goriac
egoriac@info.uaic.ro

Dorel Lucanu
dlucanu@info.uaic.ro

Gheorghe Grigoraş
grigoras@info.uaic.ro

Faculty of Computer Science
Alexandru Ioan Cuza University
Iaşi, Romania

July 4, 2008

Abstract

ROC! is a deterministic rewrite strategy language which includes the rewrite rules as basic operators, and the deterministic choice and the repetition as high-level strategy operators. In this paper we present a method which, for a given term rewriting system (TRS) R, constructs a new TRS $\overline{R}$ such that $\overline{R}$-rewriting is equivalent (sound and complete) with R-rewriting constrained by ROC! . Since $\overline{R}$ uses a stack, it is called a rewrite stack machine.

1 Introduction

Rewriting strategies are expressions (terms) built over a strategy language used for controlling the rewrite rule application. Roughly speaking, a strategy expression describes which computations (rewrite sequences), among all the possible ones, are appropriate for a given purpose. A rewriting strategy language consists of expressions built using rewrite rules and strategy operators. Various approaches have been followed, yielding slightly different strategy languages such as ELAN [5, 2], Stratego [10], TOM [1], or Maude [8]. All these provide flexible and expressive strategy languages where high-level strategies are defined by combining low level primitives, and they all share the concern to provide abstract ways to express control of rule applications, by using reflexivity and the metalevel for Maude, or the notion of rewriting strategies for ELAN or Stratego. Strategies such as bottom-up, top-down or leftmost-innermost are higher-order features that describe how rewrite rules should be applied.

In this paper we consider a simple strategy language, called ROC! , where the primitives are given by the rewrite rules applied at top, and the high-level strategy operators are the deterministic choice and the repetition. In spite of its simplicity, this language is powerful enough and very efficient for metalanguage applications built using the patterns

*This work is partially supported by the PNII grant IDEI 393

presented in [4]. A particular application of this kind which uses ROC! is CIRC [6]. The strategy language ROC! is parametric in the term rewriting system (TRS) over which it is defined; let ROC! (R) denote the particular language built over R, i.e., the primitives are given by the rewrite rules in R. The idea of implementing a rewriting strategy language is different from the classical ones. A pair $(R, \text{ROC! } (R))$ is "compiled" into a new TRS $\overline{R}$, such that $\overline{R}$-rewriting is equivalent with R-rewriting constrained by ROC! (R). The equivalence is showed by proving that $\overline{R}$ is sound and complete with respect to the semantics of $(R, \text{ROC! } (R))$.

The organization of the paper is as follows. Section 2 presents a situation where using a strategy language is suited. Section 3 defines the concept of *rewriting stack machine*. A strategy language is introduced in Section 4. Section 5 presents the specification of a stack machine that evolves according to the strategy language semantics. The soundness and completeness of the machine are proved in Section 6.

2 Motivating example

CIRC [6] is a Maude metalanguage application which implements the circular coinduction algorithm in order to prove *properties* (goals expressed as equations) for a given *equational behavioral specification*. CIRC is built using the patterns described in [4]. The procedure for proving a property uses a set of *rewrite rules* that can be applied according to some *proof strategy*. Three of the rules implemented for the CIRC prover are:

- $[comm]$ processes a goal expressing the commutativity of an operator by adding a special operator and some equations in their frozen form to the specification. The rule fails when the current property to be proved is not a commutativity goal.
- $[eqRed]$ removes an equational goal whenever it can be proved using ordinary equational reduction. The rule fails when the left hand side of an equation is different from the right hand side.
- $[ccstep]$ implements the circularity principle: an equation's frozen normalized form is added to the specification and its derivatives are added to the set of goals. This rule fails whenever it finds a visible goal which cannot be proved using ordinary equational reduction.

The following steps describe a possible strategy for the prover:

1. Try to apply $[comm]$. If it fails, leave the system state unchanged.
2. Check whether the first goal is proved using $[eqRed]$. If the check fails try to apply $[ccstep]$.
3. Repeat step 2 for as many times as possible.

The first problem is how to specify this kind of strategies. There are many ways to define rewriting strategies [1, 2, 3, 5, 10]. A previous version of CIRC uses regular strategies [7]. However, the union, concatenation and iteration operators of the regular expressions language are not proper for specifying a behavior like "apply a rule for as many times as possible" or "apply a rule only if some other rule fails".

The solution is to use strategy operators appropriate for specifying these kinds of behavior. We prefer to call the elements of this strategy language *actions*. A basic action

is a rewrite rule. Generally, actions are combined by means of several operators: $\triangleright$ (orelse), $\circ$ (composition) and ! (repeat) resulting in other actions. For the given example, the action described by the steps 1 - 3 is:

$$([comm] \triangleright [id]) \circ ([eqRed] \triangleright [ccstep])!$$

where

- $[id]$ leaves the list of goals unchanged. This rule never fails.
- $\triangleright$ has the semantics of an $orelse$-like operator. The system tries to apply $[comm]$ in the first place, and only if the action does not succeed should it apply $[id]$. The same strategy is followed for the rules $[eqRed]$ and $[ccstep]$.
- $\circ$ has the semantics of a sequential composition operator. For the example above, the full action is successfully applied if both $([comm] \triangleright [id])$ and $([eqRed] \triangleright [ccstep])!$ are evaluated exactly in this order with success.
- ! imposes that an action is applied for as many times as possible. In our case, the system stops following the strategy only when both $[eqRed]$ and $[ccstep]$ fail.

It is obvious that once the set of rules is chosen and the semantics of the strategy operators is provided, complex proof strategies can easily be specified in the same manner previously described.

The second problem is how to implement a system able to execute a strategy specified using the new language. A solution like the one described in [3] is not suitable because CIRC is a metalanguage application and the tool described in the paper is used at object level.

A solution is to use a rewriting stack machine that gives the operational semantics of the language.

3 Rewriting stack machines

In this section we introduce a new way to implement strategy languages. The general idea is as follows: given a TRS R and a language L of strategies over the rules from R, a new TRS $\overline{R}$ is defined. The purpose of $\overline{R}$ is to apply the rewrite rules R according to the semantics of the strategy language L. Since $\overline{R}$ is defined using a stack it is called a *rewriting stack machine*.

We assume that the stack data type is specified by the sorts `Stack` for stacks and `Elt` for stack elements, a constant `empty` for the stack with zero elements and an operation `_;_ : Stack Stack -> Stack`, which is associative and has the identity `empty`. We also consider the two basic operators for handling the stack, `push` and `pop`.

Let R be a TRS. Consider a sort `State` for the system states, so that if $r : u \rightarrow v$ `if` $c \in R$ then u and v are of sort `State`. Also let L be a strategy language like ROC! whose elements are called *actions*. We follow the line from [9] and we specify the semantics of an action a by a relation : $(t_{0..n}, r_{1..n}) \models a$, meaning that the rewriting $t_0 \xrightarrow{r_1} t_1 \xrightarrow{r_2} \ldots \xrightarrow{r_n} t_n$ is accomplished according to action a.

Given a term t_0 and an action a, the property $(\nexists\, t_{1..j}, r_{1..j})\ (j \geq 1) \wedge (t_{0..j}, r_{1..j}) \models a$ will be abbreviated under the form $t_0\uparrow \not\models a$. In plain English, this means that there no rewriting sequence starting from t_0 can be applied according to action a.

A *rewriting stack machine* for (R, L) is a TRS $\overline{R}$ with rewriting rules of the form $\lambda : (t, s) \rightarrow (t', s')$ `if` c, where t, t' are of sort `State` and s, s' are of sort `Stack`. The stack behavior is a list of transitions $s \xrightarrow{*} s'$, each transition being accomplished using the `push` and `pop` basic operators.

For each action $a \in L$ there is defined an initial term $\texttt{init}(a)$ of sort $\texttt{Stack}^{\texttt{init}}$ and there is a predicate which can be used to identify stacks representing final states (of sort $\texttt{Stack}^{\texttt{final}}$). A pair (t, s) with s of sort $\texttt{Stack}^{\texttt{final}}$ cannot be rewritten according to $\overline{R}$. Both $\texttt{Stack}^{\texttt{init}}$ and $\texttt{Stack}^{\texttt{final}}$ are subsorts of `Stack`.

The relationship between (R, L) and $\overline{R}$ is expressed by the following properties:

1. *soundness*:
   if $(t, \texttt{init}(a)) \xrightarrow{*} (t', s')$ with s' of sort $\texttt{Stack}^{\texttt{final}}$ then $(\exists\ t_{0..n}, r_{1..n})(n \geq 1)$ so that $t_0 = t$, $t_n = t'$ and $(t_{0..n}, r_{1..n}) \models a$

2. *completeness*:
   if $(t_{0..n}, r_{1..n}) \models a$ then there is a stack $s' : \texttt{Stack}^{\texttt{final}}$ so that $(t_0, \texttt{init}(a)) \xrightarrow{*} (t_n, s')$

4 The language ROC!

We present the formal description of the strategy language used in the current implementation of CIRC, called ROC!. The basic actions are Rewriting rules and the strategy operators are Orelse, Composition and !.

The grammar giving the syntax of the ROC! language is expressed as follows:

$$act ::= r \mid act \triangleright act \mid act \circ act \mid act\,!$$

where r is the label of a rule from the initial TRS R.

The semantics of ROC! is given by the next definitions:

1. $sem_r : (t_{0..n}, r_{1..n}) \models r \overset{def}{\Leftrightarrow} n = 1 \wedge r_1 = r \wedge t_0 \xrightarrow{r_1} t_1$
2. $sem_{\triangleright} : (t_{0..n}, r_{1..n}) \models a_1 \triangleright a_2 \overset{def}{\Leftrightarrow} (t_{0..n}, r_{1..n}) \models a_1 \vee (t_0\uparrow \not\models a_1 \wedge (t_{0..n}, r_{1..n}) \models a_2)$
3. $sem_{\circ} : (t_{0..i..n}, r_{1..i..n}) \models a_1 \circ a_2 \overset{def}{\Leftrightarrow} (t_{0..i}, r_{1..i}) \models a_1 \wedge (t_{i..n}, r_{i+1..n}) \models a_2$
4. $sem^{ind}_{!} : (t_{0..i..n}, r_{1..i..n}) \models a! \overset{def}{\Leftrightarrow} (t_{0..i}, r_{1..i}) \models a \wedge (t_{i..n}, r_{i+1..n}) \models a!$
5. $sem^{base}_{!} : (t, \phi) \models a! \overset{def}{\Leftrightarrow} t\uparrow \not\models a$

Note that ϕ from the last definition denotes an empty list of rules and is used to express that $(t_0, \phi) \models a!$ when there are no rules that can be applied to t_0 according to action a. (t, ϕ) is used in definition 4 when $i = n$.

The following propositions are used for proving properties in Section 6.

Proposition 1: $t_0\uparrow \not\models a$ *iff* $t_0\uparrow \not\models a!$

Proof

We use the *proof by contradiction* method for both implications:

1. "$\Rightarrow$" Assume that $t_0\uparrow \not\models a$. If $(t_{0..n}, r_{1..n}) \models a!$ then there is an i so that $(1 \leq i \leq n)$ and $(t_{0..i}, r_{1..i}) \models a$ according to $sem^{ind}_{!}$, which contradicts the hypothesis.

2. "$\Leftarrow$" Assume that $t_0\uparrow \not\models a!$. If $(\exists i \geq 1)$ so that $(t_{0..i}, r_{1..i}) \models a$ then there must be an $n \geq i$ so that $(t_{i..n}, r_{i+1..n}) \models a!$ holds. Hence $(\exists n \geq 1)(t_{0..n}, r_{1..n}) \models a!$ holds according to $sem^{ind}_{!}$, which contradicts the hypothesis. $\square$

Proposition 2: $(t_{0..n}, r_{1..n}) \models a!$ *iff* $(t_{0..n}, r_{1..n}) \models a!!$

Proof

If $n = 0$ the proof is trivial:

$$(t_0, \phi) \models a! \text{ iff } (t_0\uparrow \not\models a) \text{ iff } (t_0\uparrow \not\models a!) \text{ iff } (t_0, \phi) \models a!!$$

If $n \geq 0$:

1. "$\Rightarrow$" Assume that $(t_{0..n}, r_{1..n}) \models a!$

   It follows from the hypothesis that $t_n\uparrow \not\models a$. *Proposition 1* implies that $t_n\uparrow \not\models a!$, hence $(t_n, \phi) \models a!!$. It follows from the definition $sem_!^{ind}$ that $(t_{0..n}, r_{1..n}) \models a!!$.

2. "$\Leftarrow$" Assume that $(t_{0..n}, r_{1..n}) \models a!!$

   There is an i so that $(1 \leq i \leq n)$ and $(t_{0..i}, r_{1..i}) \models a!$. It follows that $t_i\uparrow \not\models a$, which implies $t_i\uparrow \not\models a!$. Because $t_n\uparrow \not\models a!$ (from the hypothesis), we obtain $i = n$, *i.e.* $(t_{0..n}, r_{1..n}) \models a!$.

□

5 A stack machine for ROC!

The syntactical tree for an action can be specified by a list of equations. For example, the action exemplified in Section 2, $a = ([comm] \triangleright [id]) \circ ([eqRed] \triangleright [ccstep])!$, is specified as follows:

$a = a_1 \circ a_2$
$a_1 = comm \triangleright id$
$a_2 = a_3!$
$a_3 = eqRed \triangleright ccstep$

Let *def* denote the operator defined as follows: given an action a, *def*(a) returns the right hand side of the equation which defines a.

We use a stack whose elements are of the form $\langle term, action, remainder, trail \rangle$, where:

- $term$ is either the empty term $null$ or the term of the system before starting the execution of the action
- $action$ is the label of the action being processed
- $remainder$ is the part of the action that remains to be processed or ε when there is nothing left to be processed
- $trail$ is the list of rules that have successfully been applied so far from the current action (the empty list is represented by the symbol ϕ)

A stack is of sort $\texttt{Stack}^{\texttt{init}}$ if it has only one element which has the form $\texttt{init}(a)=\langle null, a, \textit{def}(a), \phi \rangle$ and of sort $\texttt{Stack}^{\texttt{final}}$ if it has only one element which can be represented as $\langle null, a, \varepsilon, trl \rangle$. From a final stack we are able to conclude if the action a has been successfully executed based on the value trl. As trl is the list of rules that have been successfully applied according to a, if trl is empty (ϕ) then the action has not been executed, and if it is not empty then the action has been successfully executed.

The behavior of the rewriting stack machine for ROC! is given by two classes of rules. The first class contains rules compiled from the initial TRS R. Given a rule $r : u \rightarrow v \texttt{ if } c \in R$, we denote a condition c^{ko} which is semantically equivalent to the negation of c. For the uniformity of the notation, c will be referred to as c^{ok}. The next two rules belong to $\overline{R}$:

1. $\overline{r^{ok}} : (u, [\langle \tau, a, r, \phi \rangle; \mathsf{ST}]) \rightarrow (v, [\langle \tau, a, \varepsilon, r \rangle; \mathsf{ST}])$ `if` c^{ok}
2. $\overline{r^{ko}} : (u, [\langle \tau, a, r, \phi \rangle; \mathsf{ST}]) \rightarrow (v, [\langle \tau, a, \varepsilon, \phi \rangle; \mathsf{ST}])$ `if` c^{ko}

The second class of rules from $\overline{R}$ give the behavior of the stack machine according to the action currently being executed. For the uniformity of the notation, the rules from this class will be overlined:

1. the start of an *orelse* action
   $\overline{rl_{\triangleright}^{start}} : (t, [\langle \tau, a, a_1 \triangleright a_2, trl \rangle; \mathsf{ST}]) \rightarrow$
   $(t, [\langle \tau, a_1, \mathit{def}(a_1), \phi \rangle; \langle \tau, a, a_1 \triangleright a_2, trl \rangle; \mathsf{ST}])$
2. success for the first choice of an *orelse* action
   $\overline{rl_{\triangleright}^{ok}} : (t, [\langle \tau, a_1, \varepsilon, trl' \rangle; \langle \tau, a, a_1 \triangleright a_2, trl \rangle; \mathsf{ST}]) \rightarrow$
   $(t, [\langle \tau, a, \varepsilon, trl\, trl' \rangle; \mathsf{ST}])$
3. failure for the first choice starts the second choice
   $\overline{rl_{\triangleright}^{ko}} : (t, [\langle \tau, a_1, \varepsilon, \phi \rangle; \langle \tau, a, a_1 \triangleright a_2, trl \rangle; \mathsf{ST}]) \rightarrow$
   $(t, [\langle \tau, a_2, \mathit{def}(a_2), \phi \rangle; \langle \tau, a, \varepsilon, trl \rangle; \mathsf{ST}])$
4. the end of the second choice (success or failure)
   $\overline{rl_{\triangleright}^{merge}} : (t, [\langle \tau, a', \varepsilon, trl' \rangle; \langle \tau, a, \varepsilon, trl \rangle; \mathsf{ST}]) \rightarrow$
   $(t, [\langle \tau, a, \varepsilon, trl\, trl' \rangle; \mathsf{ST}])$
5. the start of a *sequential composition* action
   $\overline{rl_{\circ}^{start}} : (t, [\langle \tau, a, a_1 \circ a_2, trl \rangle; \mathsf{ST}]) \rightarrow$
   $(t, [\langle t, a_1, \mathit{def}(a_1), \phi \rangle; \langle \tau, a, a_1 \circ a_2, trl \rangle; \mathsf{ST}])$
6. success for the first part of a *sequential composition*
   $\overline{rl_{\circ}^{ok_1}} : (t', [\langle \tau', a_1, \varepsilon, trl' \rangle; \langle \tau, a, a_1 \circ a_2, trl \rangle; \mathsf{ST}]) \rightarrow$
   $(t', [\langle \tau', a_2, \mathit{def}(a_2), \phi \rangle; \langle \tau, a, a_1 \circ a_2, trl\, trl' \rangle; \mathsf{ST}])$
7. failure for the first part of a *sequential composition*
   $\overline{rl_{\circ}^{ko_1}} : (t', [\langle \tau', a_1, \varepsilon, \phi \rangle; \langle \tau, a, a_1 \circ a_2, trl \rangle; \mathsf{ST}]) \rightarrow$
   $(\tau', [\langle \tau, a, \varepsilon, trl \rangle; \mathsf{ST}])$
8. success for the second part of a *sequential composition*
   $\overline{rl_{\circ}^{ok2}} : (t', [\langle \tau', a_2, \varepsilon, trl' \rangle; \langle \tau, a, a_1 \circ a_2, trl \rangle; \mathsf{ST}]) \rightarrow$
   $(t', [\tau, a, \varepsilon, trl\, trl' \rangle; \mathsf{ST}]); j \geq 1$
9. failure for the second part of a *sequential composition*
   $\overline{rl_{\circ}^{ko2}} : (t', [\langle \tau', a_2, \varepsilon, \phi \rangle; \langle \tau, a, a_1 \circ a_2, trl \rangle; \mathsf{ST}]) \rightarrow$
   $(\tau', [\langle \tau, a, \varepsilon, \phi \rangle; \mathsf{ST}])$
10. the start of a *repeat* action
    $\overline{rl_{!}^{start}} : (t, [\langle \tau, a, a_1!, trl \rangle; \mathsf{ST}]) \rightarrow$
    $(t, [\langle \tau, a_1, \mathit{def}(a_1), \phi \rangle; \langle \tau, a, a_1!, trl \rangle; \mathsf{ST}])$
11. success for the *repeat* action
    $\overline{rl_{!}^{ok}} : (t, [\langle \tau, a_1, \varepsilon, trl' \rangle; \langle \tau, a, a_1!, trl \rangle; \mathsf{ST}]) \rightarrow$
    $(t, [\langle \tau, a, a_1!, trl\, trl' \rangle; \mathsf{ST}])$
12. failure for the *repeat* action
    $\overline{rl_{!}^{ko}} : (t, [\langle \tau, a_1, \varepsilon, \phi \rangle; \langle \tau, a, a_1!, trl \rangle; \mathsf{ST}]) \rightarrow$
    $(t, [\langle \tau, a, \varepsilon, trl \rangle; \mathsf{ST}])$

6 Soundness and Completeness

Generally, a system is *sound* if when proving that something is true, it really is true and is *complete* if when something is true, the system is capable of proving it.

For the ROC! stack machine, the soundness and completeness properties are expressed by the following equivalence:

$(\forall n \geq 1)\ (t_0, [\langle \tau, a, \mathit{def}(a), \phi \rangle; \mathsf{ST}]) \xrightarrow{*} (t_n, [\langle \tau, a, \varepsilon, r_{1..n} \rangle; \mathsf{ST}])$ *iff*
$(t_{0..n}, r_{1..n}) \models a$

6.1 Soundness

In order to prove that our stack machine system is sound, we have to prove that the following theorem holds:

Theorem 1:
if $(\forall n \geq 1)(t_0, [\langle \tau, a, \mathit{def}(a), \phi \rangle; \mathsf{ST}]) \xrightarrow{*} (t_n, [\langle \tau, a, \varepsilon, r_{1..n} \rangle; \mathsf{ST}])$
then $(t_{0..n}, r_{1..n}) \models a$

where a is an action from the ROC! strategy language.

Recall from Section 4 that an action is inductively defined, therefore we prove the theorem above by structural induction. For a given action a, the implication within the theorem is referenced by $P(def(a))$.

In order to prove that *Theorem 1* holds, we have to show that P holds for all the actions a from the ROC! language. This requires to prove that P holds for basic actions (rules) and then, assuming that $P(def(a_1))$ and $P(def(a_2))$ hold, to show that $P(a_1 \triangleright a_2)$, $P(a_1 \circ a_2)$ and $P(a_1!)$ hold.

We introduce a lemma that will be used in our proof:

Lemma 1:
if $(t, [\langle \tau, a, \mathit{def}(a), \phi \rangle; \mathsf{ST}]) \xrightarrow{*} (t', [\langle \tau, a, \varepsilon, \phi \rangle; \mathsf{ST}])$
then $t = t'$ *and* $t\uparrow \not\models a$

This lemma states that if the trail resulted after the execution of an action a is empty, then there is no rewrite that modifies the initial term t and the sequences $t_{1..i}, r_{1..i}$ cannot be identified, such that for $t = t_0$ the statement $(t_{0..i}, r_{1..i}) \models a$ holds. Recall that the stack machine behavior is deterministic so given the initial configuration, the evolution above is the only one possible using the rules from $\overline{R}$. The lemma is also proved by structural induction, following the same reasoning used for proving *Theorem 1*.

We come back for proving *Theorem 1*. We choose obnly to prove that $P(a_1 \triangleright a_2)$ holds. For the other cases we follow the same idea.

$P(a_1 \triangleright a_2)$:
if $(\forall n \geq 1)(t_0, [\langle \tau, a, a_1 \triangleright a_2, \phi \rangle; \mathsf{ST}]) \xrightarrow{*} (t_n, [\langle \tau, a, \varepsilon, r_{1..n} \rangle; \mathsf{ST}])$
then $(t_{0..n}, r_{1..n}) \models a$

Proof

Actions a_1 and a_2 have simpler structures than a so both $P(\mathit{def}(a_1))$ and $P(\mathit{def}(a_2))$ hold by the induction hypothesis. From the initial configuration, the first transition in the stack evolution is:

$(t_0, [\langle \tau, a, a_1 \triangleright a_2, \phi \rangle; \mathsf{ST}]) \xrightarrow{\overline{rl_{\triangleright}^{start}}}$
$(t_0, [\langle \tau, a_1, \mathit{def}(a_1), \phi \rangle; \langle \tau, a, a_1 \triangleright a_2, \phi \rangle; \mathsf{ST}])$

From the hypothesis, we know that the final stack is $[\langle \tau, a, \varepsilon, r_{1..n} \rangle; \mathsf{ST}]$ so the stack evolution process is finite. We conclude that there are a finite number of transitions that can accomplish a rewriting starting from t_0, according to action a_1. In this case the stack evolution is:

$(t_0, [\langle \tau, a_1, \textit{def}(a_1), \phi \rangle; \langle \tau, a, a_1 \triangleright a_2, \phi \rangle; \mathsf{ST}]) \xrightarrow{*}$
$(t', [\langle \tau, a_1, \varepsilon, trl' \rangle; \langle \tau, a, a_1 \triangleright a_2, \phi \rangle; \mathsf{ST}])$

We identify two situations:

1. $trl' = \phi$

   From *Lemma 1* it follows that $t' = t_0$ and $t_0\uparrow \not\models a_1$ hold. Given the current stack configuration, the only possible transition is the one corresponding to $\overline{rl_{\triangleright}^{ko}}$:

   $(t_0, [\langle \tau, a_1, \varepsilon, \phi \rangle; \langle \tau, a, a_1 \triangleright a_2, \phi \rangle; \mathsf{ST}]) \xrightarrow{\overline{rl_{\triangleright}^{ko}}}$
   $(t_0, [\langle \tau, a_2, \textit{def}(a_2), \phi \rangle; \langle \tau, a, \varepsilon, \phi \rangle; \mathsf{ST}])$

   The final stack configuration is $[\langle \tau, a, \varepsilon, r_{1..n} \rangle; \mathsf{ST}]$, so following a reasoning similar to the one used in a_1's case we conclude that the next transition takes place:

   $(t_0, [\langle \tau, a_2, \textit{def}(a_2), \phi \rangle; \langle \tau, a, \varepsilon, \phi \rangle; \mathsf{ST}]) \xrightarrow{*}$
   $(t'', [\langle \tau, a_1, \varepsilon, trl'' \rangle; \langle \tau, a, \varepsilon, \phi \rangle; \mathsf{ST}])$

   The only transition that can follow is the one corresponding to $\overline{rl_{\triangleright}^{merge}}$:

   $(t'', [\langle \tau, a_1, \varepsilon, trl'' \rangle; \langle \tau, a, \varepsilon, \phi \rangle; \mathsf{ST}]) \xrightarrow{\overline{rl_{\triangleright}^{merge}}}$
   $(t'', [\langle \tau, a, \varepsilon, trl'' \rangle; \mathsf{ST}])$

   From the hypothesis it is obvious that $t'' = t_n$ and $trl'' = r_{1..n}$, so during the stack evolution we are able to identify the following transition:

   $(t_0, [\langle \tau, a_2, \textit{def}(a_2), \phi \rangle; \mathsf{ST'}]) \xrightarrow{*}$
   $(t_n, [\langle \tau, a_2, \varepsilon, r_{1..n} \rangle; \mathsf{ST'}])$

   According to the property $P(\textit{def}(a_2))$, we deduce that $(t_{0..n}, r_{1..n}) \models a_2$ holds. This property, along with the true statement $t_0\uparrow \not\models a_1$ leads to $(t_{0..n}, r_{1..n}) \models a_1 \triangleright a_2$ being satisfied.

2. $trl' = r_{1..i}$

   The only transition that can be applied is the one corresponding to $\overline{rl_{\triangleright}^{merge}}$:

   $(t', [\langle \tau, a_1, \varepsilon, r_{1..i} \rangle; \langle \tau, a, \varepsilon, \phi \rangle; \mathsf{ST}]) \xrightarrow{\overline{rl_{\triangleright}^{merge}}}$
   $(t', [\langle \tau, a, \varepsilon, r_{1..i} \rangle; \mathsf{ST}])$

   The induction hypothesis forces us to consider the equalities $t' = t_n$ and $i = n$. Therefore we have managed to identify the transition:

   $(t_0, [\langle \tau, a_1, \textit{def}(a_1), \phi \rangle; \mathsf{ST'}]) \xrightarrow{*}$
   $(t_n, [\langle \tau, a_1, \varepsilon, r_{1..n} \rangle; \mathsf{ST'}])$

   By the induction hypothesis $(t_{0..n}, r_{1..n}) \models a_1$ holds, hence $(t_{0..n}, r_{1..n}) \models a_1 \triangleright a_2$ holds.

The conclusion is that $P(a_1 \triangleright a_2)$ holds.

□

Using a similar reasoning it is proven that statements $P(r)$, $P(a_1 \circ a_2)$ and $P(a_1!)$ hold, so *Theorem 1* holds, meaning that our stack machine is sound.

6.2 Completeness

In order to prove the completeness of the stack machine system we have to prove that the following theorem holds:

Theorem 2:
if $(\forall n \geq 1)(t_{0..n}, r_{1..n}) \models a$
then $(t_0, [\langle \tau, a, \mathit{def}(a), \phi \rangle; \mathsf{ST}]) \xrightarrow{*} (t_n, [\langle \tau, a, \varepsilon, r_{1..n} \rangle; \mathsf{ST}])$

The proof is also by structural induction. For a given action a, the implication from the theorem above is referenced by $Q(def(a))$.

In order to prove that *Theorem 2* holds, we use a reasoning similar to the one described in Section 6.1. We prove that Q holds for basic actions and then, assuming that $Q(def(a_1))$ and $Q(def(a_2))$ hold, we show that $Q(a_1 \triangleright a_2)$, $Q(a_1 \circ a_2)$ and $Q(a_1!)$ hold.

We introduce a new lemma that will be used for proving the theorem above:

Lemma 2:
if $t\uparrow \not\models a$
then $(t, [\langle \tau, a, \mathit{def}(a), \phi \rangle; \mathsf{ST}]) \xrightarrow{*} (t, [\langle \tau, a, \varepsilon, \phi \rangle; \mathsf{ST}])$

The lemma states that if we cannot identify the sequences $t_{0..i}, r_{1..i}$, $t_0 = t$ such that $(t_{0..i}, r_{1..i}) \models a$ then there is no rewrite in $\overline{R}$ that modifies t and the trail resulted after trying to execute a is empty. In this case, considering the stack machine's deterministic behavior, the only transition that can be applied for the initial configuration $(t, [\langle \tau, a, \mathit{def}(a), \phi \rangle; \mathsf{ST}])$ leads to $(t, [\langle \tau, a, \varepsilon, \phi \rangle; \mathsf{ST}])$. *Lemma 2* is proved by structural induction, following the same reasoning used for proving *Theorem 2*.

We return for the proof of *Theorem 2*. We choose only to prove that $Q(a_1 \triangleright a_2)$ holds. For the other cases we use the same technique.

$Q(a_1 \triangleright a_2)$:
if $(\forall n \geq 1)(t_{0..n}, r_{1..n}) \models a$
then $(t_0, [\langle \tau, a, a_1 \triangleright a_2, \phi \rangle; \mathsf{ST}]) \xrightarrow{*} (t_n, [\langle \tau, a, \varepsilon, r_{1..n} \rangle; \mathsf{ST}])$

Proof

Actions a_1 and a_2 have simpler structures than a, so both $Q(\mathit{def}(a_1))$ and $Q(\mathit{def}(a_2))$ hold by the induction hypothesis. The first step in the evolution of the stack for action a can only be:

$(t_0, [\langle \tau, a, a_1 \triangleright a_2, \phi \rangle; \mathsf{ST}]) \xrightarrow{\overline{rl_{\triangleright}^{start}}}$
$(t_0, [\langle \tau, a_1, \mathit{def}(a_1), \phi \rangle; \langle \tau, a, a_1 \triangleright a_2, \phi \rangle; \mathsf{ST}])$

Further, if we consider the semantics of $\triangleright$, two situations are identified:

1. $(t_{0..n}, r_{1..n}) \models a_1$

   $Q(\mathit{def}(a_1))$ holds, so the stack evolution becomes:

   $(t_0, [\langle \tau, a_1, \mathit{def}(a_1), \phi \rangle; \langle \tau, a, a_1 \triangleright a_2, \phi \rangle; \mathsf{ST}]) \xrightarrow{*}$
   $(t_n, [\langle \tau, a_1, \varepsilon, r_{1..n} \rangle; \langle \tau, a, a_1 \triangleright a_2, \phi \rangle; \mathsf{ST}])$

The only possible transition is the one corresponding to $\overline{rl^{ok}_{\triangleright}}$:

$(t_n, [\langle \tau, a_1, \varepsilon, r_{1..n} \rangle; \langle \tau, a, a_1 \triangleright a_2, \phi \rangle; \mathsf{ST}]) \xrightarrow{\overline{rl^{ok}_{\triangleright}}}$
$(t_n, \langle \tau, a, \varepsilon, r_{1..n} \rangle; \mathsf{ST}])$

2. $t_0\uparrow \not\models a_1$ and $(t_{0..n}, r_{1..n}) \models a_2$

   By *Lemma 2* it follows that $(t_0, [\langle a_1, \mathit{def}(a_1), \phi \rangle; \mathsf{ST}'])$ evolves by a finite number of transitions to $(t_0, [\langle a_1, \varepsilon, \phi \rangle; \mathsf{ST}'])$. The stack evolution is:

   $(t_0, [\langle \tau, a_1, \mathit{def}(a_1), \phi \rangle; \langle \tau, a, a_1 \triangleright a_2, \phi \rangle; \mathsf{ST}]) \xrightarrow{*}$
   $(t_0, [\langle \tau, a_1, \varepsilon, \phi \rangle; \langle \tau, a, a_1 \triangleright a_2, \phi \rangle; \mathsf{ST}])$

   The transition that follows is:

   $(t_0, [\langle \tau, a_1, \varepsilon, \phi \rangle; \langle \tau, a, a_1 \triangleright a_2, \phi \rangle; \mathsf{ST}]) \xrightarrow{\overline{rl^{ko}_{\triangleright}}}$
   $(t_0, [\langle \tau, a_2, \mathit{def}(a_2), \phi \rangle; \langle \tau, a, \varepsilon, \phi \rangle; \mathsf{ST}])$

   $Q(\mathit{def}(a_2))$ holds so the stack evolves according to the transition:

   $(t_0, [\langle \tau, a_2, \mathit{def}(a_2), \phi \rangle; \langle \tau, a, \varepsilon, \phi \rangle; \mathsf{ST}]) \xrightarrow{*}$
   $(t_n, [\langle \tau, a_2, \varepsilon, r_{1..n} \rangle; \langle \tau, a, \varepsilon, \phi \rangle; \mathsf{ST}])$

   Now, the only possible transition is:

   $(t_n, [\langle \tau, a_2, \varepsilon, r_{1..n} \rangle; \langle \tau, a, \varepsilon, \phi \rangle; \mathsf{ST}]) \xrightarrow{\overline{rl^{merge}_{\triangleright}}}$
   $(t_n, [\langle \tau, a, \varepsilon, r_{1..n} \rangle; \mathsf{ST}])$

The conclusion is that $(t_0, [\langle \tau, a, a_1 \triangleright a_2, \phi \rangle; \mathsf{ST}])$ evolves to $(t_n, [\langle \tau, a, \varepsilon, r_{1..n} \rangle; \mathsf{ST}])$, which proves that $Q(a_1 \triangleright a_2)$ holds.
□

In the same manner it can be shown that $Q(r)$, $Q(a_1 \circ a_2)$ and $Q(a_1!)$ hold, proving that *Theorem 2* holds, which means that our stack machine is complete.

7 Conclusion

This paper introduces a strategy language for specifying three kinds of behavior for a TRS:

- perform an action only if some other action fails
- perform two actions sequentially
- perform an action for as many times as possible

where an atomic action is the application of a rule at the top. The language is implemented using a stack machine system which is proved to be both sound and complete.

We have successfully managed to specify proof strategies in this new manner for CIRC after adding the stack machine implementation to the prover source code.

It is worth mentioning that if the sequential composition operator is not required, the rewriting stack machine is simplified. In this case, the structure of the stack elements is simplified from a quadruple to a triple, by eliminating the $term$ component. This component is no longer required because during the evolution of the system there is no need to turn back to a previous term (this is needed only when trying to execute an action of the form $a_1 \circ a_2$). An example emphasizing this situation is provided in Appendix A.

References

[1] E. Balland, P. Brauner, R. Kopetz, P.-E. Moreau, and A. Reilles. Tom: Piggybacking Rewriting on Java. In F. Baader, editor, *RTA*, volume 4533 of *Lecture Notes in Computer Science*, pages 36–47. Springer-Verlag, 2007.

[2] P. Borovanský, C. Kirchner, H. Kirchner, and C. Ringeissen. Rewriting with Strategies in ELAN: A Functional Semantics. *Int. J. Found. Comput. Sci.*, 12(1):69–95, 2001.

[3] S. Eker, N. Martí-Oliet, J. Meseguer, and A. Verdejo. Deduction, strategies, and rewriting. *Electron. Notes Theor. Comput. Sci.*, 174(11):3–25, 2007.

[4] E. Goriac, G. Caltais, D. Lucanu, O. Andrei, and G. Grigoraş. Patterns for Maude Metalanguage Applications. In *Proceedings of WRLA*, 2008.

[5] C. Kirchner, H. Kirchner, and M. Vittek. Designing Constraint Logic Programming Languages using Computational Systems. In P. Van Hentenryck and V. Saraswat, editors, *Principles and Practice of Constraint Programming. The Newport Papers.*, chapter 8, pages 131–158. MIT Press, 1995.

[6] D. Lucanu and G. Roşu. Circ: A Circular Coinductive Prover. In T. Mossakowski and et al., editors, *2nd Conference on Algebra and Coalgebra in Computer Science (CALCO 2007)*, volume 4624 of *Lecture Notes in Computer Science*, pages 372–378. Springer, 2007.

[7] D. Lucanu, G. Roşu, and G. Grigoraş. Regular strategies as proof tactics for circ. *Electron. Notes Theor. Comput. Sci.*, 204:83–98, 2008.

[8] N. Martí-Oliet, J. Meseguer, and A. Verdejo. Towards a Strategy Language for Maude. *Electronic Notes in Theoretical Computer Science*, 117:417–441, 2005.

[9] J. Meseguer. The temporal logic of rewriting: A gentle introduction. In *Concurrency, Graphs and Models*, pages 354–382, 2008.

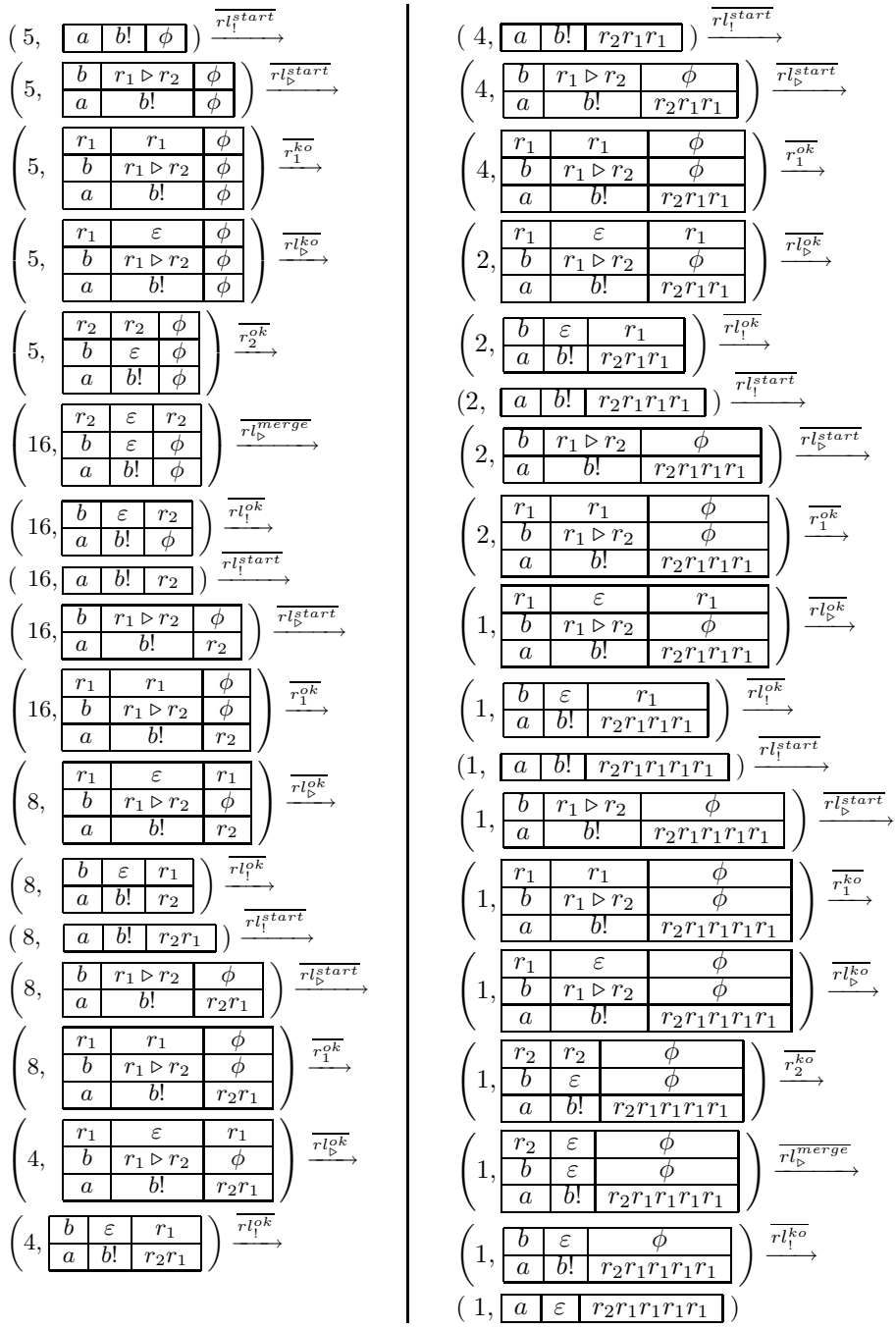
[10] E. Visser. Stratego: A Language for Program Transformation based on Rewriting Strategies. System Description of Stratego 0.5. In A. Middeldorp, editor, *RTA*, volume 2051 of *Lecture Notes in Computer Science*, pages 357–361. Springer-Verlag, 2001.

A Example

We consider a term rewriting system inspired from Collatz's function:

$r_1 : n \to n/2 \quad \texttt{if}\ \ n \equiv 0\ (\mathit{mod}\ 2)\ \wedge\ n \neq 1$
$r_2 : n \to 3n+1\ \ \texttt{if}\ \ n \neq 1$

Let us analize how the stack machine evolves for the term $t_0 = 5$ and the strategy $(r_1 \triangleright r_2)!$. The actions specifying the strategy are: $a = b!$ and $b = r_1 \triangleright r_2$.



According to the machine evolution we have: $(5\ 16\ 8\ 4\ 2\ 1, r_2r_1r_1r_1r_1) \models (r_1 \triangleright r_2)!$

B Soundness and Completeness

In this section we show the full proof for the lemmas and theorems presented in Section 6.

B.1 Soundness

We introduce the following lemma that will be used during the soundness proof:

Lemma 1:
if $(t, [\langle \tau, a, \mathit{def}(a), \phi \rangle; \mathsf{ST}]) \xrightarrow{*} (t', [\langle \tau, a, \varepsilon, \phi \rangle; \mathsf{ST}])$
then $t = t'$ *and* $t\uparrow \not\models a$

In order to prove the lemma above, we consider the property
$L_1(def(a))$:

if $(t, [\langle \tau, a, \mathit{def}(a), \phi \rangle; \mathsf{ST}]) \xrightarrow{*} (t', [\langle \tau, a, \varepsilon, \phi \rangle; \mathsf{ST}])$
then $t = t'$ *and* $t\uparrow \not\models a$

and apply the structural induction method.

1. Case $def(a) = r$.

   Consider the configuration $(t, [\langle \tau, r, r, \phi \rangle; \mathsf{ST}])$. In order to satisfy the hypothesis, the only possible transition is:

   $$(t, [\langle \tau, r, r, \phi \rangle; \mathsf{ST}]) \xrightarrow{\overline{r^{ko}}} (t, [\langle \tau, r, \varepsilon, \phi \rangle; \mathsf{ST}])$$

   It is obvious that $t' = t$ and that we cannot find a transition $t \xrightarrow{r} t'$. According to sem_r, $t\uparrow \not\models r$ holds, hence $L_1(r)$ holds.

2. Case $def(a) = a_1 \triangleright a_2$.

   Given the initial configuration, the only possible way for the stack to evolve is:

   $$(t, [\langle \tau, a, a_1 \triangleright a_2, \phi \rangle; \mathsf{ST}]) \xrightarrow{\overline{rl_{\triangleright}^{start}}}$$
   $$(t, [\langle \tau, a_1, def(a_1), \phi \rangle; \langle \tau, a, a_1 \triangleright a_2, \phi \rangle; \mathsf{ST}])$$

   By the hypothesis, the stack evolution is finite so the following transition is mandatory:

   $$(t, [\langle \tau, a_1, def(a_1), \phi \rangle; \langle \tau, a, a_1 \triangleright a_2, \phi \rangle; \mathsf{ST}]) \xrightarrow{*}$$
   $$(t'_i, [\langle \tau, a_1, \varepsilon, trl'_i \rangle; \langle \tau, a, a_1 \triangleright a_2, \phi \rangle; \mathsf{ST}])$$

   Action a leaves no trail so the equality $trl'_i = \phi$ must hold. The next transition is:

   $$(t'_i, [\langle \tau, a_1, \varepsilon, \phi \rangle; \langle \tau, a, a_1 \triangleright a_2, \phi \rangle; \mathsf{ST}]) \xrightarrow{\overline{rl_{\triangleright}^{ko}}}$$
   $$(t'_i, [\langle \tau, a_2, def(a_2), \phi \rangle; \langle \tau, a, \varepsilon, \phi \rangle; \mathsf{ST}])$$

   Using a reasoning that resembles the one for a_1's case, we identify the transition:

   $$(t'_i, [\langle \tau, a_2, def(a_2), \phi \rangle; \langle \tau, a, \varepsilon, \phi \rangle; \mathsf{ST}]) \xrightarrow{*}$$
   $$(t'_j, [\langle \tau, a_2, \varepsilon, trl'_j \rangle; \langle \tau, a, \varepsilon, \phi \rangle; \mathsf{ST}])$$

   The stack can only evolve according to $\overline{rl_{\triangleright}^{merge}}$:

$(t'_j, [\langle \tau, a_2, \varepsilon, trl'_j \rangle; \langle \tau, a, \varepsilon, \phi \rangle; \mathsf{ST}]) \xrightarrow{\overline{rl_{\triangleright}^{merge}}}$
$(t'_j, [\langle \tau, a, \varepsilon, trl'_j \rangle; \mathsf{ST}])$

In order to satisfy the hypothesis, both the equalities $t'_j = t'$ and $trl'_j = \phi$ hold. During the stack evolution two transitions are identified:

$(t, [\langle \tau, a_1, \mathit{def}(a_1), \phi \rangle; \mathsf{ST}']) \xrightarrow{*} (t'_i, [\langle \tau, a_1, \varepsilon, \phi \rangle; \mathsf{ST}'])$
$(t'_i, [\langle \tau, a_2, \mathit{def}(a_2), \phi \rangle; \mathsf{ST}'']) \xrightarrow{*} (t', [\langle \tau, a_2, \varepsilon, \phi \rangle; \mathsf{ST}''])$

Both a_1 and a_2 have simpler structures than a, so $L_1(def(a_1))$ and $L_1(def(a_2))$ hold. In this case we state that $t = t'_i = t'$, $t\uparrow \not\models a_1$ and $t\uparrow \not\models a_2$. In conclusion, according to $sem_{\circ}$, $t\uparrow \not\models a$ holds, therefore $L_1(a_1 \triangleright a_2)$ holds.

3. Case $def(a) = a_1 \circ a_2$

   This proof is very similar to the one corresponding to $a_1 \triangleright a_2$ case. The first transition is:

   $(t, [\langle \tau, a, a_1 \circ a_2, \phi \rangle; \mathsf{ST}]) \xrightarrow{\overline{rl_{\circ}^{start}}}$
   $(t, [\langle t, a_1, def(a_1), \phi \rangle; \langle \tau, a, a_1 \circ a_2, \phi \rangle; \mathsf{ST}])$

   By the hypothesis, the number of stack evolution steps is finite, therefore we identify the transition:

   $(t, [\langle t, a_1, def(a_1), \phi \rangle; \langle \tau, a, a_1 \circ a_2, \phi \rangle; \mathsf{ST}]) \xrightarrow{*}$
   $(t'_i, [\langle t, a_1, \varepsilon, trl'_i \rangle; \langle \tau, a, a_1 \circ a_2, \phi \rangle; \mathsf{ST}])$

   Two situations are handled:

   (a) $trl'_i = \phi$

   According to $\overline{rl_{\circ}^{ko1}}$ the stack evolution is:

   $(t'_i, [\langle t, a_1, \varepsilon, \phi \rangle; \langle \tau, a, a_1 \circ a_2, \phi \rangle; \mathsf{ST}]) \xrightarrow{\overline{rl_{\circ}^{ko1}}}$
   $(t, [\langle \tau, a, \varepsilon, \phi \rangle; \mathsf{ST}])$

   If we analyze the stack evolution, the following transition is identified:

   $(t, [\langle t, a_1, \mathit{def}(a_1), \phi \rangle; \mathsf{ST}']) \xrightarrow{*}$
   $(t'_i, [\langle t, a_1, \varepsilon, \phi \rangle; \mathsf{ST}'])$

   a_1 has a simpler structure than a and by the induction hypothesis, we conclude that $t = t'_i$ and $t\uparrow \not\models a_1$, therefore $t\uparrow \not\models a$.

   (b) $trl'_i \neq \phi$

   According to $\overline{rl_{\circ}^{ok1}}$ the stack evolution is:

   $(t'_i, [\langle t, a_1, \varepsilon, trl'_i \rangle; \langle \tau, a, a_1 \circ a_2, \phi \rangle; \mathsf{ST}]) \xrightarrow{\overline{rl_{\circ}^{ok1}}}$
   $(t'_i, [\langle t, a_2, def(a_2), \phi \rangle; \langle \tau, a, a_1 \circ a_2, trl'_i \rangle; \mathsf{ST}])$

   By the hypothesis, the stack evolution process is finite, so the following transition is mandatory:

   $(t'_i, [\langle t, a_2, def(a_2), \phi \rangle; \langle \tau, a, a_1 \circ a_2, trl'_i \rangle; \mathsf{ST}]) \xrightarrow{*}$
   $(t'_j, [\langle t, a_2, \varepsilon, trl'_j \rangle; \langle \tau, a, a_1 \circ a_2, trl'_i \rangle; \mathsf{ST}])$

In order to satisfy the hypothesis we have to consider the equality $trl'_j = \phi$. a_2 has a simpler structure than a and by the transition above, we conclude that $t'_i = t'_j$ and $t'_i\uparrow \not\models a_2$. The next possible transition is:

$(t'_i, [\langle t, a_2, \varepsilon, \phi\rangle; \langle \tau, a, a_1 \circ a_2, trl'_i\rangle; \mathsf{ST}]) \xrightarrow{\overline{rl^{ko2}_{\circ}}}$
$(t, [\langle \tau, a, \varepsilon, \phi\rangle; \mathsf{ST}])$

By the hypothesis, the equality $t' = t$ holds. We have already shown that $t'_i\uparrow \not\models a_2$, so even if the sequences $t'_{1..i}, r'_{1..i}, i \geq 1$ exist and $t'_0 = t$ such that $(t'_{0..i}, r'_{1..i}) \models a_1$, according to $sem_{\circ}$ we conclude that $t\uparrow \not\models a$.

It is obvious that no matter the path we choose for our stack evolution - (a) or (b), the result is that $L_1(a_1 \circ a_2)$ holds.

4. Case $def(a) = a_1!$.

   Similar to the reasoning used for the cases above, if we consider the initial configuration, the only transition that can be applied is:

   $(t, [\langle \tau, a, a_1!, \phi\rangle; \mathsf{ST}]) \xrightarrow{\overline{rl^{start}_{!}}}$
   $(t, [\langle \tau, a_1, def(a_1), \phi\rangle; \langle \tau, a, a_1!, \phi\rangle; \mathsf{ST}])$

   Assume that $a_1 \neq a'_1!$. From the hypothesis, the stack evolution process is finite and action a leaves no trail, so the next transition takes place:

   $(t, [\langle \tau, a_1, def(a_1), \phi\rangle; \langle \tau, a, a_1!, \phi\rangle; \mathsf{ST}]) \xrightarrow{*}$
   $(t'_i, [\langle \tau, a_1, \varepsilon, \phi\rangle; \langle \tau, a, a_1!, \phi\rangle; \mathsf{ST}])$

   We know that a_1's structure is simpler than a's so $L_1(def(a_1))$ holds. We deduce that both $t = t'_i$ and $t\uparrow \not\models a_1$ hold. The only transition that can follow and that does not violate the hypothesis is:

   $(t, [\langle \tau, a_1, \varepsilon, \phi\rangle; \langle \tau, a, a_1!, \phi\rangle; \mathsf{ST}]) \xrightarrow{\overline{rl^{ko}_{!}}}$
   $(t, [\langle \tau, a, \varepsilon, \phi\rangle; \mathsf{ST}])$

   It follows that $t = t'$ and $t\uparrow \not\models a_1$ hold. By *Proposition 1* this implies that $t\uparrow \not\models a_1!$, hence $t\uparrow \not\models a$.

   Assume that $a_1 = a'_1!$. According to *Proposition 1* if we want to check whether $t\uparrow \not\models a'_1!!$ holds then we have to check whether $t\uparrow \not\models a'_1!$ holds using the same reasoning described above.

   We have shown that $L_1(a_1!)$ holds.

We have proven that $L_1(def(a))$ holds for all the four cases, hence *Lemma 1* holds.

In order to prove that the stack machine system is sound, it is necessary to demonstrate that the following implication holds:

$P(\mathit{def}(a))$:
if $(\forall n \geq 1)(t_0, [\langle \tau, a, \mathit{def}(a), \phi\rangle; \mathsf{ST}]) \xrightarrow{*}$
$(t_n, [\langle \tau, a, \varepsilon, r_{1..n}\rangle; \mathsf{ST}])$
then $(t_{0..n}, r_{1..n}) \models a$

The proof is realized by structural induction.

1. Case $\mathit{def}(a) = r$.

   The only rule that can be applied for the initial configuration is $\overline{r^{ok}}$. The configuration that follows after $(t_0, [\langle \tau, r, r, \phi \rangle; \mathsf{ST}])$ is $(t_1, [\langle \tau, r, \varepsilon, r \rangle; \mathsf{ST}])$. The latter is the final configuration $(n = 1)$. Therefore $t_0 \xrightarrow{r} t_1$, which means that $(t_0 t_1, r) \models r$. Basically we have shown that $P(r)$ holds.

2. Case $\mathit{def}(a) = a_1 \triangleright a_2$.

   Actions a_1 and a_2 have simpler structures than a so $P(\mathit{def}(a_1))$ and $P(\mathit{def}(a_2))$ hold by the induction hypothesis. From the initial configuration, the first transition in the stack evolution is:

   $(t_0, [\langle \tau, a, a_1 \triangleright a_2, \phi \rangle; \mathsf{ST}]) \xrightarrow{\overline{rl_{\triangleright}^{start}}}$
   $(t_0, [\langle \tau, a_1, \mathit{def}(a_1), \phi \rangle; \langle \tau, a, a_1 \triangleright a_2, \phi \rangle; \mathsf{ST}])$

   From the hypothesis, we know that the final stack is $[\langle \tau, a, \varepsilon, r_{1..n} \rangle; \mathsf{ST}]$ so the stack evolution process is finite. We conclude that there are a finite number of transitions that can accomplish a rewriting starting from t_0, according to action a_1. In this case the stack evolution is:

   $(t_0, [\langle \tau, a_1, \mathit{def}(a_1), \phi \rangle; \langle \tau, a, a_1 \triangleright a_2, \phi \rangle; \mathsf{ST}]) \xrightarrow{*}$
   $(t', [\langle \tau, a_1, \varepsilon, trl' \rangle; \langle \tau, a, a_1 \triangleright a_2, \phi \rangle; \mathsf{ST}])$

   We identify two situations:

   (a) $trl' = \phi$

   From *Lemma 1* it follows that $t' = t_0$ and $t_0 \uparrow \not\models a_1$ hold. Given the current stack configuration, the only possible transition is the one corresponding to $\overline{rl_{\triangleright}^{ko}}$:

   $(t_0, [\langle \tau, a_1, \varepsilon, \phi \rangle; \langle \tau, a, a_1 \triangleright a_2, \phi \rangle; \mathsf{ST}]) \xrightarrow{\overline{rl_{\triangleright}^{ko}}}$
   $(t_0, [\langle \tau, a_2, \mathit{def}(a_2), \phi \rangle; \langle \tau, a, \varepsilon, \phi \rangle; \mathsf{ST}])$

   The final stack is $[\langle \tau, a, \varepsilon, r_{1..n} \rangle; \mathsf{ST}]$, so following a reasoning similar to the one used in a_1's case we conclude that the next transition takes place:

   $(t_0, [\langle \tau, a_2, \mathit{def}(a_2), \phi \rangle; \langle \tau, a, \varepsilon, \phi \rangle; \mathsf{ST}]) \xrightarrow{*}$
   $(t'', [\langle \tau, a_1, \varepsilon, trl'' \rangle; \langle \tau, a, \varepsilon, \phi \rangle; \mathsf{ST}])$

   The only transition that can follow is the one corresponding to $\overline{rl_{\triangleright}^{merge}}$:

   $(t'', [\langle \tau, a_1, \varepsilon, trl'' \rangle; \langle \tau, a, \varepsilon, \phi \rangle; \mathsf{ST}]) \xrightarrow{\overline{rl_{\triangleright}^{merge}}}$
   $(t'', [\langle \tau, a, \varepsilon, trl'' \rangle; \mathsf{ST}])$

   From the hypothesis it is obvious that $t'' = t_n$ and $trl'' = r_{1..n}$, so during the stack evolution we are able to identify the following transition:

   $(t_0, [\langle \tau, a_2, \mathit{def}(a_2), \phi \rangle; \mathsf{ST}']) \xrightarrow{*}$
   $(t_n, [\langle \tau, a_2, \varepsilon, r_{1..n} \rangle; \mathsf{ST}'])$

   According to the property $P(\mathit{def}(a_2))$, we deduce that $(t_{0..n}, r_{1..n}) \models a_2$ holds. This property, along with the true statement $t_0 \uparrow \not\models a_1$ leads to $(t_{0..n}, r_{1..n}) \models a_1 \triangleright a_2$ being satisfied.

(b) $trl' = r_{1..i}$

The only transition that can be applied is the one corresponding to $\overline{rl_{\triangleright}^{merge}}$:

$(t', [\langle \tau, a_1, \varepsilon, r_{1..i} \rangle; \langle \tau, a, \varepsilon, \phi \rangle; \mathsf{ST}]) \xrightarrow{\overline{rl_{\triangleright}^{merge}}}$
$(t', [\langle \tau, a, \varepsilon, r_{1..i} \rangle; \mathsf{ST}])$

The induction hypothesis forces us to consider the equalities $t' = t_n$ and $i = n$. Therefore we have managed to identify the transition:

$(t_0, [\langle \tau, a_1, \mathit{def}(a_1), \phi \rangle; \mathsf{ST'}]) \xrightarrow{*}$
$(t_n, [\langle \tau, a_1, \varepsilon, r_{1..n} \rangle; \mathsf{ST'}])$

By the induction hypothesis $(t_{0..n}, r_{1..n}) \models a_1$ holds, hence $(t_{0..n}, r_{1..n}) \models a_1 \triangleright a_2$ holds.

The conclusion is that $P(a_1 \triangleright a_2)$ holds.

3. Case $\mathit{def}(a) = a_1 \circ a_2$

Actions a_1 and a_2 have simpler structures than a, so $P(\mathit{def}(a_1))$ and $P(\mathit{def}(a_2))$ hold. Consider the initial configuration. The first transition is:

$(t_0, [\langle \tau, a, a_1 \circ a_2, \phi \rangle; \mathsf{ST}]) \xrightarrow{\overline{rl_{\circ}^{start}}}$
$(t_0, [\langle t_0, a_1, \mathit{def}(a_1), \phi \rangle; \langle \tau, a, a_1 \circ a_2, \phi \rangle; \mathsf{ST}])$

By the induction hypothesis, the final stack configuration is $[\langle \tau, a, \varepsilon, r_{1..n} \rangle; \mathsf{ST}])$ so using a reasoning similar to the one described for $P(a_1 \triangleright a_2)$, we identify the transition:

$(t_0, [\langle t_0, a_1, \mathit{def}(a_1), \phi \rangle; \langle \tau, a, a_1 \circ a_2, \phi \rangle; \mathsf{ST}]) \xrightarrow{*}$
$(t', [\langle t_0, a_1, \varepsilon, trl' \rangle; \langle \tau, a, a_1 \circ a_2, \phi \rangle; \mathsf{ST}])$

At this point we have to consider the two possible values for trl':

(a) $trl' = \phi$

By *Lemma 1*, the equality $t' = t_0$ holds. The only transition that follows is:

$(t_0, [\langle t_0, a_1, \varepsilon, \phi \rangle; \langle \tau, a, a_1 \circ a_2, \phi \rangle; \mathsf{ST}]) \xrightarrow{\overline{rl_{\circ}^{ko1}}}$
$(t_0, [\langle \tau, a, \varepsilon, \phi \rangle; \mathsf{ST}])$

This situation contradicts the hypothesis which assumes that at least one rewrite rule must be applied successfully.

(b) $trl' = r_{1..i}$

In this case, the stack evolution is:

$(t', [\langle t_0, a_1, \varepsilon, r_{1..i} \rangle; \langle \tau, a, a_1 \circ a_2, \phi \rangle; \mathsf{ST}]) \xrightarrow{\overline{rl_{\circ}^{ok1}}}$
$(t', [\langle t_0, a_2, \mathit{def}(a_2), \phi \rangle; \langle \tau, a, a_1 \circ a_2, r_{1..i} \rangle; \mathsf{ST}])$

Recall that the evolution process is finite so we identify the transition:

$(t', [\langle t_0, a_2, \mathit{def}(a_2), \phi \rangle; \langle a, a_1 \circ a_2, r_{1..i} \rangle; \mathsf{ST}]) \xrightarrow{*}$
$(t'', [\langle t_0, a_2, \varepsilon, trl'' \rangle; \langle \tau, a, a_1 \circ a_2, r_{1..i} \rangle; \mathsf{ST}])$

There are two possible values for trl'':

i. $trl'' = \phi$
By *Lemma 1*, the equality $t' = t''$ is satisfied. In this case, the next transition is:

$(t', [\langle t_0, a_2, \varepsilon, \phi\rangle; \langle \tau, a, a_1 \circ a_2, r_{1..i}\rangle; \mathsf{ST}]) \xrightarrow{\overline{rl_{\circ}^{ko2}}}$
$(t_0, [\langle \tau, a, \varepsilon, \phi\rangle; \mathsf{ST}])$

It is obvious that the final stack configuration contradicts the hypothesis.

ii. $trl'' = r_{i+1..i+j}$
In this case, the only possible transition is:

$(t'', [\langle t_0, a_2, \varepsilon, r_{i+1..i+j}\rangle; \langle \tau, a, a_1 \circ a_2, r_{1..i}\rangle; \mathsf{ST}]) \xrightarrow{\overline{rl_{\circ}^{ok2}}}$
$(t'', [\langle \tau, a, \varepsilon, r_{1..i+j}\rangle; \mathsf{ST}])$

In order to satisfy the hypothesis, $t'' = t_n$ and $i + j = n$ must hold. Given the stack evolution we identify the following two transitions:

$(t_0, [\langle t_0, a_1, \mathit{def}(a_1), \phi\rangle; \mathsf{ST}']) \xrightarrow{*}$
$(t', [\langle t_0, a_1, \varepsilon, r_{1..i}\rangle; \mathsf{ST}'])$

and

$(t', [\langle t_0, a_2, \mathit{def}(a_2), \phi\rangle; \mathsf{ST}'']) \xrightarrow{*}$
$(t_n, [\langle t_0, a_2, \varepsilon, r_{i+1..i+j}\rangle; \mathsf{ST}''])$

The first transition rewrites t_0 to t' using the rules $r_1, ..., r_i$. Both statements $P(\mathit{def}(a_1))$ and $P(\mathit{def}(a_2))$ hold, therefore $(t_{0..i}, r_{1..i}) \models a_1$ and $(t_{i..n}, r_{i+1..n}) \models a_2$ hold.

The conclusion is that the property $P(a_1 \circ a_2)$ holds.

4. Case $a = a_1!$.

Recall that $P(def(a_1))$ holds. Consider the initial configuration. The only possible transition is:

$(t_0, [\langle \tau, a, a_1!, \phi\rangle; \mathsf{ST}]) \xrightarrow{\overline{rl_{!}^{start}}}$
$(t_0, [\langle \tau, a_1, \mathit{def}(a_1), \phi\rangle; \langle \tau, a, a_1!, \phi\rangle; \mathsf{ST}])$

According to the hypothesis there are a finite number of transitions in the stack evolution process. This means that there are also a finite number of such transitions for action a_1. Note that the trail $r_{1..n}$ resulting after a's execution is not empty, therefore the trail that results after a_1's first execution is also not empty. It is obvious that in order to satisfy the hypothesis, the following transition is mandatory:

$(t_0, [\langle \tau, a_1, \mathit{def}(a_1), \phi\rangle; \langle \tau, a, a_1!, \phi\rangle; \mathsf{ST}]) \xrightarrow{*}$
$(t_i, [\langle \tau, a_1, \varepsilon, r_{1..i}\rangle; \langle \tau, a, a_1!, \phi\rangle; \mathsf{ST}])$

The only transition that follows is:

$(t_i, [\langle \tau, a_1, \varepsilon, r_{1..i}\rangle; \langle \tau, a, a_1!, \phi\rangle; \mathsf{ST}]) \xrightarrow{\overline{rl_{!}^{ok}}}$
$(t_i, [\langle \tau, a, a_1!, r_{1..i}\rangle; \mathsf{ST}])$

Using the inductive hypothesis we deduce the statement $(t_{0..i}, r_{1..i}) \models a_1$. From this point forward, the evolution process continues according to transitions similar to the ones described above. Therefore we identify sequences of the form

$t_{i'..j'}$, $r_{i'+1..j'}$ ($i' \geq i, i' \leq j'$ and $j' \leq n$) such that $(t_{i'..j'}, r_{i'+1..j'}) \models a_1$ holds. Recall that this process is finite so we can only identify a finite number of such transitions. Consider the last transition:

$(t_{i'}, [\langle \tau, a, a_1!, r_{1..i'} \rangle; \mathsf{ST}]) \xrightarrow{\overline{rl_!^{start}}}$
$(t_{i'}, [\langle \tau, a_1, \mathit{def}(a_1), \phi \rangle; \langle \tau, a, a_1!, r_{1..i'} \rangle; \mathsf{ST}]) \xrightarrow{*}$
$(t_{j'}, [\langle \tau, a_1, \varepsilon, r_{i'+1..j'} \rangle; \langle \tau, a, a_1!, r_{1..i'} \rangle; \mathsf{ST}])$

We conclude that $r_{i'+1..j'} = \phi$ (otherwise the evolution process continues). By *Lemma 2* the equality $t'_i = t'_j$ holds. The next transition is:

$(t_{i'}, [\langle \tau, a_1, \varepsilon, \phi \rangle; \langle \tau, a, a_1!, r_{1..i'} \rangle; \mathsf{ST}]) \xrightarrow{rl_!^{ko}}$
$(t_{i'}, [\langle \tau, a, \varepsilon, r_{1..i'} \rangle; \mathsf{ST}])$

In order to satisfy the hypothesis, the equality $t'_i = t_n$ holds. Recall the transition:

$(t_n, [\langle \tau, a_1, def(a_1), \phi \rangle; \mathsf{ST}']) \xrightarrow{*}$
$(t_n, [\langle \tau, a_1, \varepsilon, \phi \rangle; \mathsf{ST}'])$

By *Lemma 2* the statement $t_n\uparrow \not\models a_1$ holds so we can deduce that $(t_n, \phi) \models a_1!$ holds.

From the stack evolution process we identify the statements:

$(t_{0..i}, r_{1..i}) \models a_1,\ 1 \leq i$
$(t_{i..i+j}, r_{i+1..i+j}) \models a_1,\ i < j$
$\ldots$
$(t_{i+k..n}, r_{i+k+1..n}) \models a_1, i + k < n$
$(t_n, \phi) \models a_1!$

From the semantics of ! these statements recursively lead us to a property that holds: $(t_{0..n}, r_{1..n}) \models a_1!$.

We have shown that $P(a_1!)$ holds.

In conclusion, we have proven by structural induction that $P(def(a))$ holds, so the stack machine is sound.

B.2 Completeness

We introduce another lemma used for proving the completeness of the stack machine:

Lemma 2:
if $t\uparrow \not\models a$
then $(t, [\langle \tau, a, \mathit{def}(a), \phi\rangle; \mathsf{ST}]) \xrightarrow{*} (t, [\langle \tau, a, \varepsilon, \phi\rangle; \mathsf{ST}])$

In order to prove *Lemma 2* consider the property
$L_2(def(a))$:

if $t\uparrow \not\models a$
then $(t, [\langle \tau, a, \mathit{def}(a), \phi\rangle; \mathsf{ST}]) \xrightarrow{*} (t, [\langle \tau, a, \varepsilon, \phi\rangle; \mathsf{ST}])$

and use the structural induction method.

1. Case $def(a) = r$.

   If $t\uparrow \not\models r$ then we cannot identify t' such that $(tt', r) \models r$ therefore a transition $t \xrightarrow{r} t'$ is never possible. Consider the initial configuration. The only feasible transition is:

   $$(t, [\langle \tau, r, r, \phi\rangle; \mathsf{ST}]) \xrightarrow{\overline{r^{ko}}} (t, [\langle \tau, r, \varepsilon, \phi\rangle; \mathsf{ST}])$$

   Therefore $L_2(r)$ holds.

2. Case $def(a) = a_1 \triangleright a_2$.

   By the hypothesis $t\uparrow \not\models a_1 \triangleright a_2$ holds, meaning that both $t\uparrow \not\models a_1$ and $t\uparrow \not\models a_2$ hold. Given the initial configuration, the first transition can only be:

   $$(t, [\langle \tau, a, a_1 \triangleright a_2, \phi\rangle; \mathsf{ST}]) \xrightarrow{\overline{rl_{\triangleright}^{start}}} (t, [\langle \tau, a_1, def(a_1), \phi\rangle; \langle \tau, a, a_1 \triangleright a_2, \phi\rangle; \mathsf{ST}])$$

   Action a_1 has a simpler structure than a, so by the induction hypothesis the following transition is mandatory:

   $$(t, [\langle \tau, a_1, def(a_1), \phi\rangle; \langle \tau, a, a_1 \triangleright a_2, \phi\rangle; \mathsf{ST}]) \xrightarrow{*} (t, [\langle \tau, a_1, \varepsilon, \phi\rangle; \langle \tau, a, a_1 \triangleright a_2, \phi\rangle; \mathsf{ST}])$$

   The next transition is:

   $$(t, [\langle \tau, a_1, \varepsilon, \phi\rangle; \langle \tau, a, a_1 \triangleright a_2, \phi\rangle; \mathsf{ST}]) \xrightarrow{\overline{rl_{\triangleright}^{ko}}} (t, [\langle \tau, a_2, def(a_2), \phi\rangle; \langle \tau, a, \varepsilon, \phi\rangle; \mathsf{ST}])$$

   The structure of a_2 is simpler than a so again we use the induction hypothesis again:

   $$(t, [\langle \tau, a_2, def(a_2), \phi\rangle; \langle \tau, a, \varepsilon, \phi\rangle; \mathsf{ST}]) \xrightarrow{*} (t, [\langle \tau, a_2, \varepsilon, \phi\rangle; \langle \tau, a, \varepsilon, \phi\rangle; \mathsf{ST}])$$

   The only possible transition is:

   $$(t, [\langle \tau, a_2, \varepsilon, \phi\rangle; \langle \tau, a, \varepsilon, \phi\rangle; \mathsf{ST}]) \xrightarrow{\overline{rl_{\triangleright}^{merge}}} (t, [\langle \tau, a, \varepsilon, \phi\rangle; \mathsf{ST}])$$

   Therefore $L_2(a_1 \triangleright a_2)$ holds.

3. Case $def(a) = a_1 \circ a_2$.

   $t\uparrow \not\models a_1 \circ a_2$ implies either that $t\uparrow \not\models a_1$ or the sequences $t_{0..i}\,, r_{1..i}$ exist and $t_0 = t$ such that $(t_{0..i}, r_{1..i}) \models a_1$ and $t_i\uparrow \not\models a_2$ are satisfied.

   An initial transition is valid for both situations above:

   $$(t, [\langle \tau, a, a_1 \circ a_2, \phi\rangle; \mathsf{ST}]) \xrightarrow{\overline{rl_{\circ}^{start}}} \\ (t, [\langle t, a_1, def(a_1), \phi\rangle; \langle \tau, a, a_1 \circ a_2, \phi\rangle; \mathsf{ST}])$$

   Consider the first case. a_1' structure is simpler than a's structure so, by the induction transition, the following transition is identified:

   $$(t, [\langle t, a_1, def(a_1), \phi\rangle; \langle \tau, a, a_1 \circ a_2, \phi\rangle; \mathsf{ST}]) \xrightarrow{*} \\ (t, [\langle t, a_1, \varepsilon, \phi\rangle; \langle \tau, a, a_1 \circ a_2, \phi\rangle; \mathsf{ST}])$$

   The next transition can only be:

   $$(t, [\langle t, a_1, \varepsilon, \phi\rangle; \langle \tau, a, a_1 \circ a_2, \phi\rangle; \mathsf{ST}]) \xrightarrow{\overline{rl_{\circ}^{ko1}}} \\ (t, [\langle \tau, a, \varepsilon, \phi\rangle; \mathsf{ST}])$$

   Consider the second case. The stack evolution is finite, so the following transition is mandatory:

   $$(t, [\langle t, a_1, def(a_1), \phi\rangle; \langle \tau, a, a_1 \circ a_2, \phi\rangle; \mathsf{ST}]) \xrightarrow{*} \\ (t', [\langle t, a_1, \varepsilon, trl'\rangle; \langle \tau, a, a_1 \circ a_2, \phi\rangle; \mathsf{ST}])$$

   If $trl' = \phi$ then the situation is identical to the one above, otherwise the next transition is:

   $$(t', [\langle t, a_1, \varepsilon, trl'\rangle; \langle \tau, a, a_1 \circ a_2, \phi\rangle; \mathsf{ST}]) \xrightarrow{\overline{rl_{\circ}^{ok1}}} \\ (t', [\langle t, a_2, def(a_2), \phi\rangle; \langle \tau, a, a_1 \circ a_2, trl'\rangle; \mathsf{ST}])$$

   Action a_2 has a simpler structure than a so according to the induction hypothesis, we identify the transition:

   $$(t', [\langle t, a_2, def(a_2), \phi\rangle; \langle \tau, a, a_1 \circ a_2, trl'\rangle; \mathsf{ST}]) \xrightarrow{*} \\ (t', [\langle t, a_2, \varepsilon, \phi\rangle; \langle \tau, a, a_1 \circ a_2, trl'\rangle; \mathsf{ST}])$$

   The only possible transition is:

   $$(t', [\langle t, a_2, \varepsilon, \phi\rangle; \langle \tau, a, a_1 \circ a_2, trl'\rangle; \mathsf{ST}]) \xrightarrow{\overline{rl_{\circ}^{ko2}}} \\ (t, [\langle \tau, a, \varepsilon, \phi\rangle; \mathsf{ST}])$$

   Therefore $L_2(a_1 \circ a_2)$ holds.

4. Case $def(a) = a_1!$.

   Assume that $a_1 \neq a_1'!$. By the hypothesis we know that $t\uparrow \not\models a_1!$. According to *Proposition 1* we deduce that $t\uparrow \not\models a_1$. Consider the initial configuration. The only possible transition is:

   $$(t, [\langle \tau, a, a_1!, \phi\rangle; \mathsf{ST}]) \xrightarrow{\overline{rl_{!}^{start}}} \\ (t, [\langle \tau, a_1, def(a_1), \phi\rangle; \langle \tau, a, a_1!, \phi\rangle; \mathsf{ST}])$$

Action a_1 has simpler structure than a so we use the inductive hypothesis and identify the transition:

$(t, [\langle \tau, a_1, def(a_1), \phi\rangle; \langle \tau, a, a_1!, \phi\rangle; \mathsf{ST}]) \xrightarrow{*}$
$(t, [\langle \tau, a_1, \varepsilon, \phi\rangle; \langle \tau, a, a_1!, \phi\rangle; \mathsf{ST}])$

The next transition is:

$(t, [\langle \tau, a_1, \varepsilon, \phi\rangle; \langle \tau, a, a_1!, \phi\rangle; \mathsf{ST}]) \xrightarrow{\overline{rl_!^{ko}}}$
$(t, [\langle \tau, a, \varepsilon, \phi\rangle; \mathsf{ST}])$

Therefore $L_2(a_1!)$ holds for $a_1 \neq a_1'!$. Assume that $a_1 = a_1'!$. According to *Proposition 1* the property $t\uparrow \not\models a_1'!!$ reduces to $t\uparrow \not\models a_1'!$. We only need to follow the reasoning described above for a_1' .

In conclusion, no matter the assumption we make, $L_2(a_1!)$ holds.

We have shown that $L_2(def(a))$ holds for all the four cases so *Lemma 2* holds.

In order to prove the completeness for the stack machine system we have to prove that the following implication holds:

$Q(\textit{def}(a))$:
if $(\forall n \geq 1)(t_{0..n}, r_{1..n}) \models a$
then $(t_0, [\langle \tau, a, \textit{def}(a), \phi\rangle; \mathsf{ST}]) \xrightarrow{*}$
$(t_n, [\langle \tau, a, \varepsilon, r_{1..n}\rangle; \mathsf{ST}])$

The proof is also by structural induction.

1. Case $\textit{def}(a) = r$.

   According to sem_r it is obvious that $n = 1$ and $r_1 = r$. The statement $(t_0 t_1) \models r$ holds so the transition step $t_0 \xrightarrow{r} t_1$ is possible. In other words, the next evolution of the stack machine is valid:

   $(t_0, [\langle \tau, r, r, \phi\rangle; \mathsf{ST}]) \xrightarrow{\overline{rl_r^{ok}}} (t_1, [\langle \tau, r, \varepsilon, r\rangle; \mathsf{ST}])$

   We have proven that $Q(r)$ holds.

2. Case $\textit{def}(a) = a_1 \triangleright a_2$.

   Actions a_1 and a_2 have simpler structures than a, so $Q(\textit{def}(a_1))$ and $Q(\textit{def}(a_2))$ hold by the induction hypothesis. The first step in the evolution of the stack for action a can only be:

   $(t_0, [\langle \tau, a, a_1 \triangleright a_2, \phi\rangle; \mathsf{ST}]) \xrightarrow{\overline{rl_\triangleright^{start}}}$
   $(t_0, [\langle \tau, a_1, \textit{def}(a_1), \phi\rangle; \langle \tau, a, a_1 \triangleright a_2, \phi\rangle; \mathsf{ST}])$

   Further, if we consider the semantics of $\triangleright$, two situations are identified:

   (a) $(t_{0..n}, r_{1..n}) \models a_1$
   The property $Q(\textit{def}(a_1))$ holds, so the stack evolution becomes:

   $(t_0, [\langle \tau, a_1, \textit{def}(a_1), \phi\rangle; \langle \tau, a, a_1 \triangleright a_2, \phi\rangle; \mathsf{ST}]) \xrightarrow{*}$
   $(t_n, [\langle \tau, a_1, \varepsilon, r_{1..n}\rangle; \langle \tau, a, a_1 \triangleright a_2, \phi\rangle; \mathsf{ST}])$

The only possible transition is the one corresponding to $\overline{rl_{\triangleright}^{ok}}$:

$(t_n, [\langle \tau, a_1, \varepsilon, r_{1..n} \rangle; \langle \tau, a, a_1 \triangleright a_2, \phi \rangle; \mathsf{ST}]) \xrightarrow{\overline{rl_{\triangleright}^{ok}}}$
$(t_n, \langle \tau, a, \varepsilon, r_{1..n} \rangle; \mathsf{ST}])$

(b) $t_0\uparrow \not\models a_1$ and $(t_{0..n}, r_{1..n}) \models a_2$
By *Lemma 2*, $(t_0, [\langle a_1, \mathit{def}(a_1), \phi \rangle; \mathsf{ST}'])$ evolves to $(t_0, [\langle a_1, \varepsilon, \phi \rangle; \mathsf{ST}'])$. The stack evolution is:

$(t_0, [\langle \tau, a_1, \mathit{def}(a_1), \phi \rangle; \langle \tau, a, a_1 \triangleright a_2, \phi \rangle; \mathsf{ST}]) \xrightarrow{*}$
$(t_0, [\langle \tau, a_1, \varepsilon, \phi \rangle; \langle \tau, a, a_1 \triangleright a_2, \phi \rangle; \mathsf{ST}])$

The transition that follows is:

$(t_0, [\langle \tau, a_1, \varepsilon, \phi \rangle; \langle \tau, a, a_1 \triangleright a_2, \phi \rangle; \mathsf{ST}]) \xrightarrow{\overline{rl_{\triangleright}^{ko}}}$
$(t_0, [\langle \tau, a_2, \mathit{def}(a_2), \phi \rangle; \langle \tau, a, \varepsilon, \phi \rangle; \mathsf{ST}])$

$Q(\mathit{def}(a_2))$ holds so the stack evolves according to the transition:

$(t_0, [\langle \tau, a_2, \mathit{def}(a_2), \phi \rangle; \langle \tau, a, \varepsilon, \phi \rangle; \mathsf{ST}]) \xrightarrow{*}$
$(t_n, [\langle \tau, a_2, \varepsilon, r_{1..n} \rangle; \langle \tau, a, \varepsilon, \phi \rangle; \mathsf{ST}])$

Now, the only possible transition is:

$(t_n, [\langle \tau, a_2, \varepsilon, r_{1..n} \rangle; \langle \tau, a, \varepsilon, \phi \rangle; \mathsf{ST}]) \xrightarrow{\overline{rl_{\triangleright}^{merge}}}$
$(t_n, [\langle \tau, a, \varepsilon, r_{1..n} \rangle; \mathsf{ST}])$

In conclusion, $(t_0, [\langle \tau, a, a_1 \triangleright a_2, \phi \rangle; \mathsf{ST}])$ evolves to $(t_n, [\langle \tau, a, \varepsilon, r_{1..n} \rangle; \mathsf{ST}])$. This proves that $Q(a_1 \triangleright a_2)$ holds.

3. Case $\mathit{def}(a) = a_1 \circ a_2$.

Recall that both $Q(\mathit{def}(a_1))$ and $Q(\mathit{def}(a_2))$ hold. Given the semantics of the $\circ$ operator, we can find a natural number $i \geq 1$ such that $(t_{0..i}, r_{1..i}) \models a_1$ and $(t_{i..n}, r_{i+1..n}) \models a_2$. The first step in the stack evolution is:

$(t_0, [\langle \tau, a, a_1 \circ a_2, \phi \rangle; \mathsf{ST}]) \xrightarrow{\overline{rl_{\circ}^{start}}}$
$(t_0, [\langle t_0, a_1, \mathit{def}(a_1), \phi \rangle; \langle \tau, a, a_1 \circ a_2, \phi \rangle; \mathsf{ST}])$

According to $Q(\mathit{def}(a_1))$ the stack evolution is:

$(t_0, [\langle t_0, a_1, \mathit{def}(a_1), \phi \rangle; \langle \tau, a, a_1 \circ a_2, \phi \rangle; \mathsf{ST}]) \xrightarrow{*}$
$(t_i, [\langle t_0, a_1, \varepsilon, r_{1..i} \rangle; \langle \tau, a, a_1 \circ a_2, \phi \rangle; \mathsf{ST}])$

The following transition can only be:

$(t_i, [\langle t_0, a_1, \varepsilon, r_{1..i} \rangle; \langle \tau, a, a_1 \circ a_2, \phi \rangle; \mathsf{ST}]) \xrightarrow{\overline{rl_{\circ}^{ok1}}}$
$(t_i, [\langle t_0, a_2, \mathit{def}(a_2), \phi \rangle; \langle \tau, a, a_1 \circ a_2, r_{1..i} \rangle; \mathsf{ST}])$

It is obvious that according to $Q(\mathit{def}(a_2))$ the next transition is:

$(t_i, [\langle t_0, a_2, \mathit{def}(a_2), \phi \rangle; \langle \tau, a, a_1 \circ a_2, r_{1..i} \rangle; \mathsf{ST}]) \xrightarrow{*}$
$(t_n, [\langle t_0, a_2, \varepsilon, r_{i+1..n} \rangle; \langle \tau, a, a_1 \circ a_2, r_{1..i} \rangle; \mathsf{ST}])$

In the end, the transition that is applied is:

$(t_n, [\langle t_0, a_2, \varepsilon, r_{i+1..n}\rangle; \langle \tau, a, a_1 \circ a_2, r_{1..i}\rangle; \mathsf{ST}]) \xrightarrow{\overline{rl_{\circ}^{ok2}}}$
$(t_n, [\langle \tau, a, \varepsilon, r_{1..n}\rangle; \mathsf{ST}])$

We conclude that $Q(a_1 \circ a_2)$ holds.

4. Case $a = a_1!$.

   According to $sem_!^{ind}$, we can identify a number $i < n$ such that $(t_{0..i}, r_{1..i}) \models a_1$ and $(t_{i..n}, r_{i+1..n}) \models a_1!$ hold. Both sequences $t_{i..n}, r_{i+1..n}$ are finite so there are a finite number of relations of the form $(t_{j..n}, r_{j+1..n}) \models a_1!$, $j \geq i$, $j < n$ that hold.

   Assume that $(t_{k..n}, r_{k+1..n}) \models a_1!$ is the last relation from the sequence. Considering $sem_!^{ind}$, the statements $(t_{k..n}, r_{k+1..n}) \models a_1$ and $(t_n, \phi) \models a_1!$ hold.

   Given the initial configuration, the first transition that applies is the one corresponding to $\overline{rl_!^{start}}$:

$(t_0, [\langle \tau, a, a_1!, \phi\rangle; \mathsf{ST}]) \xrightarrow{\overline{rl_!^{start}}}$
$(t_0, [\langle \tau, a_1, \mathit{def}(a_1), \phi\rangle; \langle \tau, a, a_1!, \phi\rangle; \mathsf{ST}])$

   a_1 has a simpler structure than a, so the following transition takes place:

$(t_0, [\langle \tau, a_1, \mathit{def}(a_1), \phi\rangle; \langle \tau, a, a_1!, \phi\rangle; \mathsf{ST}]) \xrightarrow{*}$
$(t_i, [\langle \tau, a_1, \varepsilon, r_{0..i}\rangle; \langle \tau, a, a_1!, \phi\rangle; \mathsf{ST}])$

   Recall that $i \geq 1$ so $r_{0..i} \neq \phi$. The next transition is:

$(t_i, [\langle \tau, a_1, \varepsilon, r_{0..i}\rangle; \langle \tau, a, a_1!, \phi\rangle; \mathsf{ST}]) \xrightarrow{\overline{rl_!^{ok}}}$
$(t_i, [\langle \tau, a, a_1!, r_{0..i}\rangle; \mathsf{ST}])$

   It is obvious that the stack evolution process is a finite number of transitions resembling the ones described above. Let us consider the last one:

$(t_k, [\langle \tau, a, a_1!, r_{1..k}\rangle; \mathsf{ST}]) \xrightarrow{\overline{rl_!^{start}}}$
$(t_k, [\langle \tau, a_1, \mathit{def}(a_1), \phi\rangle; \langle \tau, a, a_1!, r_{1..k}\rangle; \mathsf{ST}]) \xrightarrow{*}$
$(t_n, [\langle \tau, a_1, \varepsilon, r_{k+1..n}\rangle; \langle \tau, a, a_1!, r_{1..k}\rangle; \mathsf{ST}]) \xrightarrow{\overline{rl_!^{ok}}}$
$(t_n, [\langle \tau, a, a_1!, r_{1..n}\rangle; \mathsf{ST}])$

   Again we apply the transition corresponding to $\overline{rl_!^{start}}$:

$(t_n, [\langle \tau, a, a_1!, r_{1..n}\rangle; \mathsf{ST}]) \xrightarrow{\overline{rl_!^{start}}}$
$(t_n, [\langle \tau, a_1, \mathit{def}(a_1), \phi\rangle; \langle \tau, a, a_1!, r_{1..n}\rangle; \mathsf{ST}])$

   The property $(t_n, \phi) \models a_1!$ holds so we deduce that $t_n{\uparrow} \not\models a_1$ holds. By *Lemma 2* the next transition is:

$(t_n, [\langle \tau, a_1, \mathit{def}(a_1), \phi\rangle; \langle \tau, a, a_1!, r_{1..n}\rangle; \mathsf{ST}]) \xrightarrow{*}$
$(t_n, [\langle \tau, a_1, \varepsilon, \phi\rangle; \langle \tau, a, a_1!, r_{1..n}\rangle; \mathsf{ST}])$

The only possible transition is:

$$(t_n, [\langle \tau, a_1, \varepsilon, \phi \rangle; \langle \tau, a, a_1!, r_{1..n} \rangle; \mathsf{ST}]) \xrightarrow{\overline{rl_!^{ko}}} (t_n, [\langle \tau, a, \varepsilon, r_{1..n} \rangle; \mathsf{ST}]),$$

proving that $Q(a_1!)$ holds.

In conclusion, we have proven by structural induction that $Q(def(a))$ holds, so the stack machine is complete.